



realtimepublishers.com<sup>tm</sup>

*Tips and Tricks*  
*Guide<sup>tm</sup> To*

**Active Directory  
Troubleshooting**

  
**NETPRO**  
The Directory Experts

*Don Jones*

**Note to Reader:** This book presents tips and tricks for Active Directory troubleshooting topics. For ease of use and for cross referencing, the questions are numbered.

Q.21: How does DNS work? .....	1
The Process .....	1
DNS Replies.....	2
Record Keeping .....	3
Sharing Zones .....	3
IP Address to Name .....	3
Communications .....	4
Data Storage.....	4
Q.22: How do trusts work under AD? .....	4
Win2K Intra-Forest Trusts.....	4
Foreign Trusts .....	5
Trusting Trees .....	7
Inter-Forest Trusts.....	8
Q.23: I've heard the term <i>split brain DNS</i> ; what does it mean?.....	9
Splitting the Brain.....	9
How it Works.....	10
Split-Brain Pitfalls .....	10
Same-Site Split-Brain .....	10
Q.24: Why won't Active Directory let me create new objects? .....	11
Troubleshooting the Problem.....	12
Preventing the Problem.....	12
RID Threshold Changes.....	13
Q.25: How can I see why some DNS queries are failing?' .....	13
Nslookup.....	15

---

## Copyright Statement

© 2003 Realtimerepublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimerepublishers.com, Inc. (the "Materials") and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimerepublishers.com, Inc or its web site sponsors. In no event shall Realtimerepublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtimerepublishers.com and the Realtimerepublishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimerepublishers.com, please contact us via e-mail at [info@realtimerepublishers.com](mailto:info@realtimerepublishers.com)

## Q.21: How does DNS work?

**A:** The Domain Name System (or Service, depending on who you listen to—DNS) is one of the most important components of modern networks, including the global Internet. Without it, you couldn't type [www.Microsoft.com](http://www.Microsoft.com) into your Web browser; you'd have to type a difficult-to-remember IP address instead. DNS saves the day by translating human-friendly names into computer-friendly IP addresses. It actually does much more than that—providing computers with critical information about network services such as the locations of mail servers, domain controllers, and more.

### The Process

In any DNS transaction, at least two computers are involved: the DNS server and a DNS client. In most networks, there is often also a second DNS server known as the *forwarder* as well as additional *authoritative* DNS servers. Consider an example: You're on your company's internal network, which is equipped with a DNS server. That server is set up to forward to your ISP's DNS server, and you're trying to surf to [www.Microsoft.com](http://www.Microsoft.com). Here's what happens:

1. Your client sends a DNS request packet, usually via User Datagram Protocol (UDP), to the local DNS server.
2. That DNS server has no clue what the IP address for [www.Microsoft.com](http://www.Microsoft.com) is, so it sends the entire request to your ISP's DNS server—a process known as forwarding.
3. The ISP's DNS server also has no clue. However, it is configured with root hints, meaning it knows the IP address of a DNS server that is authoritative for the ".com" top-level domain (TLD).
4. The ISP's DNS server contacts the .com TLD server and asks for a reference for the Microsoft.com domain. This action is called a recursive query. Your own DNS server could have performed this type of query, but it's more common for businesses to forward requests to their ISP's DNS server.
5. The TLD server, being authoritative, has records for every .com domain in existence (actually, there are a bunch of load-balanced DNS servers handling that TLD, but let's pretend it is one big server for now). So the TLD server sends back the IP address of a DNS server that's authoritative for the Microsoft.com domain. Now the DNS server knows how to contact Microsoft.com, so it just needs to figure out the "www" part.
6. The ISP's DNS server then contacts the Microsoft.com DNS server. In all likelihood, that server is owned by Microsoft and contained in one of Microsoft's data centers (it's also probably part of another load balanced set of DNS servers because Microsoft.com gets so much traffic). The ISP's DNS server asks for the address of [www.Microsoft.com](http://www.Microsoft.com).
7. As an authoritative server for the Microsoft.com domain, the Microsoft.com DNS server has a record for the host named www. So it sends that host's IP address back to the ISP's DNS server.

 If the Microsoft.com server didn't have a record for a host named www, it would send back a negative reply. As an authoritative server, it *by definition* has records for all hosts in the domain; in other words, if that DNS server doesn't know about the host, the host doesn't exist.

8. The ISP's DNS server sends the IP address for www.Microsoft.com to your local DNS server.
9. The local DNS server sends the IP address to your client.
10. You client now has the IP address, and can contact www.Microsoft.com directly.

Now, there's a bit of extra subtlety in there. Each DNS server along the way will try to cache previous lookups. With a popular site such as www.Microsoft.com, it's likely that your local DNS server, or certainly your ISP's DNS server, will have already been through this process once. DNS servers will use their cached results from earlier attempts whenever possible to avoid having to go through this entire process every single time a request comes through.

## DNS Replies

DNS replies can often contain multiple responses. For example, if your client queries my.yahoo.com, you'll get back a response with at least two IP addresses. Your client is free to try either one, although it'll usually try the first one. A DNS technique called *round robin* can be implemented on the server; when more than one record exists for a single host, the server will rotate the order of the IP addresses in outgoing responses. This behavior helps direct clients to different servers in a Web farm, for example, more evenly spreading the workload. Figure 21.1 shows a Network Monitor capture of a DNS response in which you can clearly see two IP addresses for my.yahoo.com: 216.115.105.17 and 216.115.105.16.

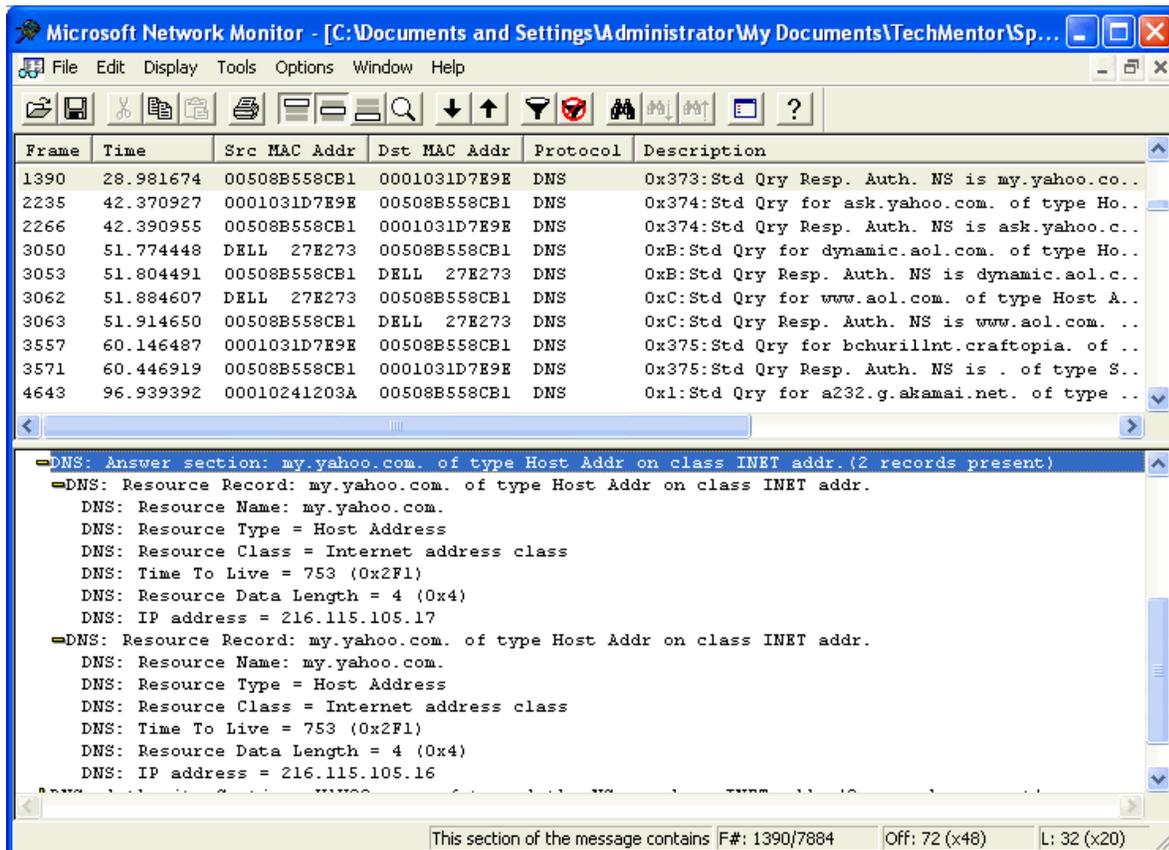


Figure 21.1: Multiple replies for my.yahoo.com.

## Record Keeping

DNS keeps track of information in *zones*. Essentially, a zone is a flat-file database for a particular domain, such as `www.Microsoft.com`. The zone can contain different record types, all of which can be queried by clients:

- **A**, which is a Host Address record—This resolves a single host name, such as `www`, to an IP address.
- **CNAME**, or Alias—This resolves a name such as `www` to an actual host name, such as `www1`. Think of it as a nickname for a computer—“`www`,” for example, is easier to remember and more standardized than a computer name like “`w4laswin`,” which is what a Web server’s real name might be.
- **MX**, or Mail Exchanger—This provides the name of the mail server for a domain. Multiple MX records can be provided for fault tolerance or load balancing and a priority assigned to each. Clients, such as sending mail servers, will attempt to contact the server in the MX record with the lowest-numbered priority.
- **AAAA**—This maps an IPv6 IP address to a host name.
- **SRV**, or Service—This provides the IP address of one or more servers providing a particular service. AD uses SRV records to allow clients to locate domain controllers, among other things.
- **SOA**, or Start of Authority—This special record indicates that the DNS server hosting the zone is authoritative for the zone and is the primary source of name resolution for hosts within that domain.

## Sharing Zones

It's common for companies to have more than one DNS server for their domains. In fact, the rules for DNS say that each domain must be hosted by at least two DNS servers for fault tolerance. However, having multiple DNS servers can be a configuration nightmare, because you would have to update records in multiple locations.

Rather than creating this maintenance nightmare, DNS servers can be configured to host *secondary zones*, which are essentially read-only copies of a *primary zone*. Configuration changes can be made in the primary zone, then transferred, or replicated, to the secondary zones through a process known as a *zone transfer*.

## IP Address to Name

Sometimes computers need to resolve an IP address to a computer name. For example, when two SMTP email servers communicate, the receiving server might want to verify that the sending server is, in fact, really coming from the domain it says it is. The server can use DNS to perform a *reverse lookup*, attempting to translate the sending server's IP address into a host name and domain name.

Reverse lookups are provided through special, independent *reverse lookup zones*, which contain *PTR records*. PTR, or pointer records, resolve an IP address into a computer name. Like regular *forward* lookup zones, you can create primary and secondary reverse lookup zones for load balancing and redundancy.

## Communications

Most DNS communications—queries and replies, at least—are conducted over connectionless UDP transmissions on UDP port 53. UDP does not provide packet confirmation, so if a request gets dropped in transit, the client will never know. Clients therefore wait for a fixed period of time—called a *timeout* period—before assuming the packet was lost and retransmitting it. Most clients will try three times, then assume that their DNS server has failed.

Zone transfers, which are usually large and involve hundreds or thousands of packets in sequence, are conducted over TCP port 53. TCP requires a bit more processing time to set up, but provides packet confirmation and allows for single-packet retransmissions if necessary.

The DNS specification makes it clear that UDP communications are intended to be single packet, and DNS requests are therefore limited to 256 bytes over TCP. This limitation is normally sufficient for any query. However, some replies with multiple hosts can exceed 256 characters. Although the DNS specification is unclear how these should be handled, in practice, the server will set up a TCP communications channel to send the response. In these cases, TCP port 53 is used. Clients listen for replies on both UDP and TCP port 53, ensuring clear communications.

## Data Storage

Most DNS servers, including nearly all UNIX-based servers, store their DNS records in *zone files*, which are essentially text files. In fact, many UNIX DNS servers are simply background daemons (services), with no UI whatsoever. Changing DNS records requires you to edit the text file, then restart (or *hup* in UNIX parlance) the service, which rereads the updated file.

Microsoft DNS running on a Windows 2000 (Win2K) domain controller can convert a standard, file-based zone to an AD -integrated zone. In this type of zone, records are stored in the AD database and replicated to all domain controllers. Any domain controller running the Microsoft DNS service is therefore a DNS server. Technically, because all domain controllers have full read/write access to the AD database, all DNS servers in an AD -integrated zone are said to host a *primary zone* because they can all make changes to the zone. Changes made on one server are replicated through AD to the others.

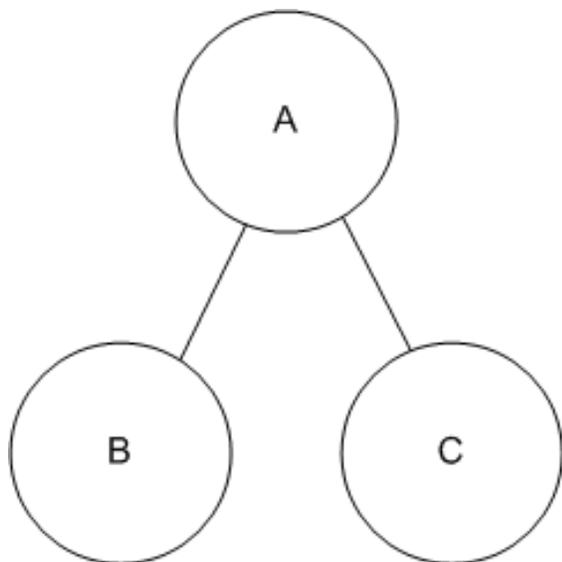
 For more information about AD replication, see Question 19.

## Q.22: How do trusts work under AD?

**A:** Trusts didn't go away when Windows 2000 (Win2K) came onto the scene—they just changed a lot. Actually, Active Directory (AD) continues to support Windows NT-style trusts, and for good reason.

### Win2K Intra-Forest Trusts

First, a quick review of the primary type of trust—the one that exists between parent and child domains in the same forest. As Figure 22.1 shows, every child domain has a two-way, automatic, nonconfigurable trust with its parent. This trust is transitive, meaning that, in this example, domains B and C trust one another through their trust with their parent, domain A.

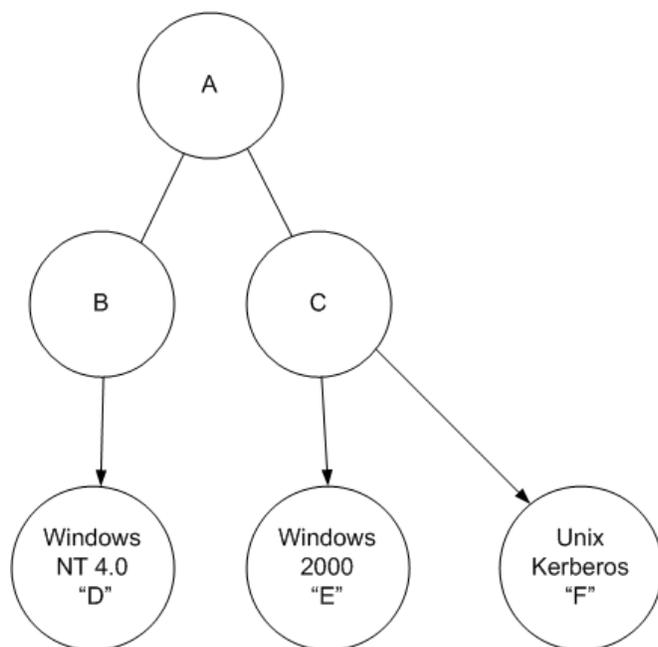


**Figure 22.1: Native Win2K AD trusts.**

It's important to remember that trusts don't necessarily confer any special privileges to any users; it simply makes it possible for user accounts in one domain to be granted permissions to resources in another domain. The exception is the Enterprise Admins group, which has special, full administrative control over every domain in the forest.

### **Foreign Trusts**

An AD domain, not a forest, can establish a manual, one-way, non-transitive trust with another AD domain, a Kerberos domain, or an NT 4.0 domain. Figure 22.2 shows an example.



**Figure 22.2: Foreign AD trusts.**

In this example, Domain B trusts Domain D, meaning Domain B can assign permissions on resources to user accounts contained within Domain D. The relationship is not reciprocal; if Domain D wants to utilize accounts from Domain B, a separate, independent D-trusts-B trust will have to be created.

In addition, these trusts are not transitive. Although Domain C trusts Domain E, it does not follow that Domain B also trusts Domain E even though Domain B trusts Domain C. If Domain B wants to utilize user accounts that exist in Domain E, a separate trust will have to be established in which Domain B trusts Domain E.

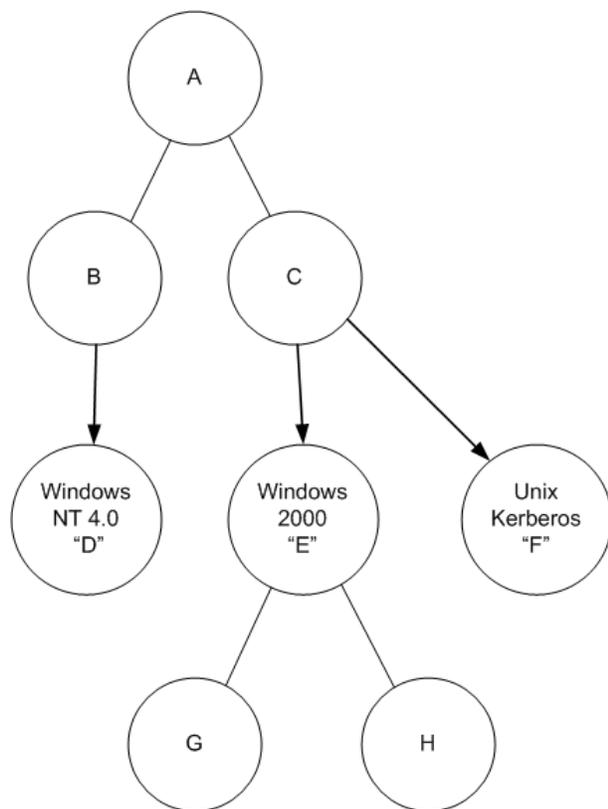
The trust relationship, especially the ability to obtain security IDs (SIDs) for user accounts in the trusted domain, lies with the domain controller holding the PDC emulator role in AD domains, with the PDC in NT domains, and with an individual Kerberos server for Kerberos realms. The inter-domain communications are critical and are encrypted. The encryption key is a shared secret, or password, which you create when you establish the trust. Actually, as soon as the trust is established, the two domains negotiate a new password, which they then change on a periodic basis. One problem in trust relationships is when the two domains lose sync of the trust password. In that case, you'll have to delete and re-create the trust. Doing so will not generally affect the permissions assigned to users from the trusted domain, provided you repair the trust quickly.

 You can use the Netdom command-line tool from the Support Tools to verify, delete, and create trusts in a Win2K domain.

Note that the trust relationship between Domain C and Domain E in this example does *not* necessarily make Domain E a member of the forest containing Domain A, Domain B, and Domain C. New domain trees can be added to an existing forest, but you make this decision when you install the domain controller. Domain C and Domain E, in this example, do not necessarily share a common AD schema because they are separate forests.

## Trusting Trees

It's possible for a domain in one tree to trust a domain in another tree, as Figure 22.3 illustrates.



**Figure 22.3: Two trees with inter-domain trusts.**

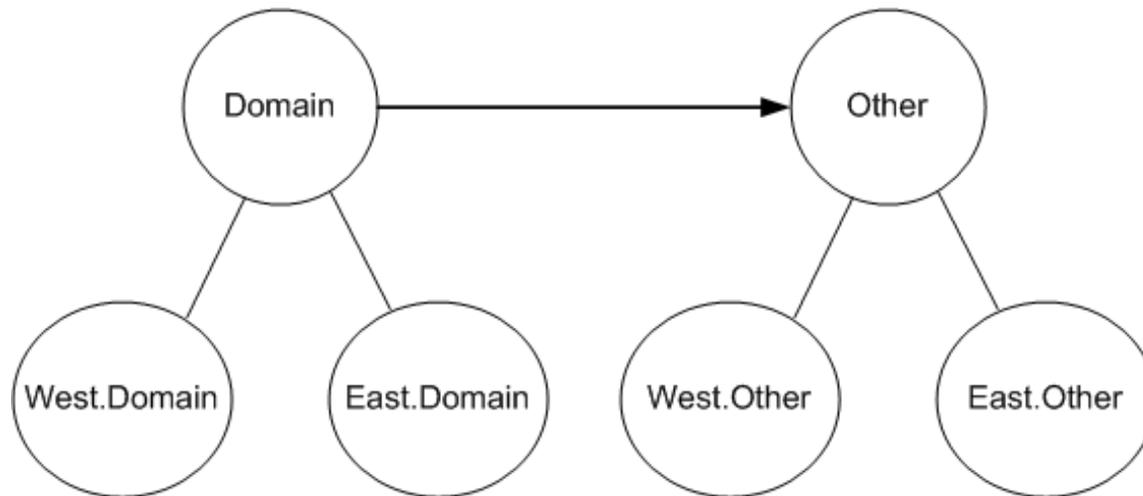
In this example:

- Domain A trusts Domain B, and Domain B trusts Domain A
- Domain A trusts Domain C, and Domain C trusts Domain A
- Domain B trusts Domain C, and Domain C trusts Domain B
- Domain C trusts Domain E
- Domain E trusts Domain G, and Domain G trusts Domain E
- Domain E trusts Domain H, and Domain H trusts Domain E.
- Domain G trusts Domain H, and Domain H trusts Domain G.

In other words, the normal two-way transitive trusts operate within each tree, and one domain from each tree trusts a foreign domain. The manual trust between Domain C and Domain E is non-transitive and one-way. Again, this configuration has not formed a forest of six domains: there are two forests, with three domains, and one domain tree apiece, and the two forests don't trust one another. Password synchronization occurs between the PDC emulators in Domain C and Domain E.

## Inter-Forest Trusts

New in Windows Server 2003, inter-forest trusts allow two forests to trust one another. These trusts are one-way or two-way, and they take advantage of the transitive nature of intra-tree trusts. For example, Figure 22.4 shows two forests, each with three domains in a single domain tree. The root domains in each forest trust one another.



**Figure 22.4: Inter-forest trust.**

This feature is only available in AD domains in which all domain controllers are running Windows Server 2003 and the forest functional level has been raised to its highest level, Windows Server 2003 Native.

In this example, Domain, West.Domain, and East.Domain all trust Other, West.Other, and East.Other. The trust, then, is transitive. You can also configure a two-way trust, which in this example, would result in all six domains trusting one another. Forest trusts can use *SID filtering*, which prevents specific SIDs in one domain from being used in another domain.



New in Windows Server 2003: Forest trusts were created to acknowledge the power of the Enterprise Admins group. Prior to forest trusts, Microsoft often stated that domains were the primary security boundary. In fact, as a result of the forest-wide rights of Enterprise Admins, the forest was the only true security boundary. Many enterprises created separate forests to resolve political issues about who controlled which resources; forest trusts give these organizations a way to provide easier cross-forest access to their users, while restricting control of the Enterprise Admins group in each forest. SID filtering can be used to ensure that foreign Enterprise Admins have no control in a forest, and they have no rights in a foreign forest by default.

Forest trusts are considered to be an external trust, and as with any other external trust, there's no implicit synchronization of data. For example, the Domain and Other domain trees do not have to share a common AD schema, as they would if both trees belonged to the same forest. There's also no synchronization of address lists or directory objects of any kind, and Global Catalog (GC) servers in each forest maintain independent catalogs. If synchronization of some kind is desired, check out Microsoft Metadirectory Services (MMS), which can be used to synchronize objects across multiple directory services or domains.

## Q.23: I've heard the term *split brain DNS*; what does it mean?

**A:** Most companies these days have a public Internet domain name, such as `www.Microsoft.com`. When companies begin implementing Active Directory (AD), which requires a DNS name for the domain name, they have to make a decision: use a private domain name (such as `Microsoft.pri`) for the AD domain, or keep using the public name (`Microsoft.com`) for both AD and public use.

The benefit of using a private domain name is that your domain is safer from harm. You have two domains, a public one and a private one. The downside is that users might think it's weird. For example, in AD, users can log on using their fully qualified user name; `don@braincore.net`, for example. That's easy for users because it looks like their email address. However, if you select a private domain name, users will receive email at `don@braincore.net`, but they'll log on with `don@braincore.pri`, which is confusing.

So-called *split brain DNS* offers an elegant solution that combines the benefits of a private domain name with the benefits of a public domain name—with almost none of the downsides of either.

### Splitting the Brain

Figure 23.1 shows how split DNS can be configured so that both your internal and public domains use the same domain name, `company.com`.

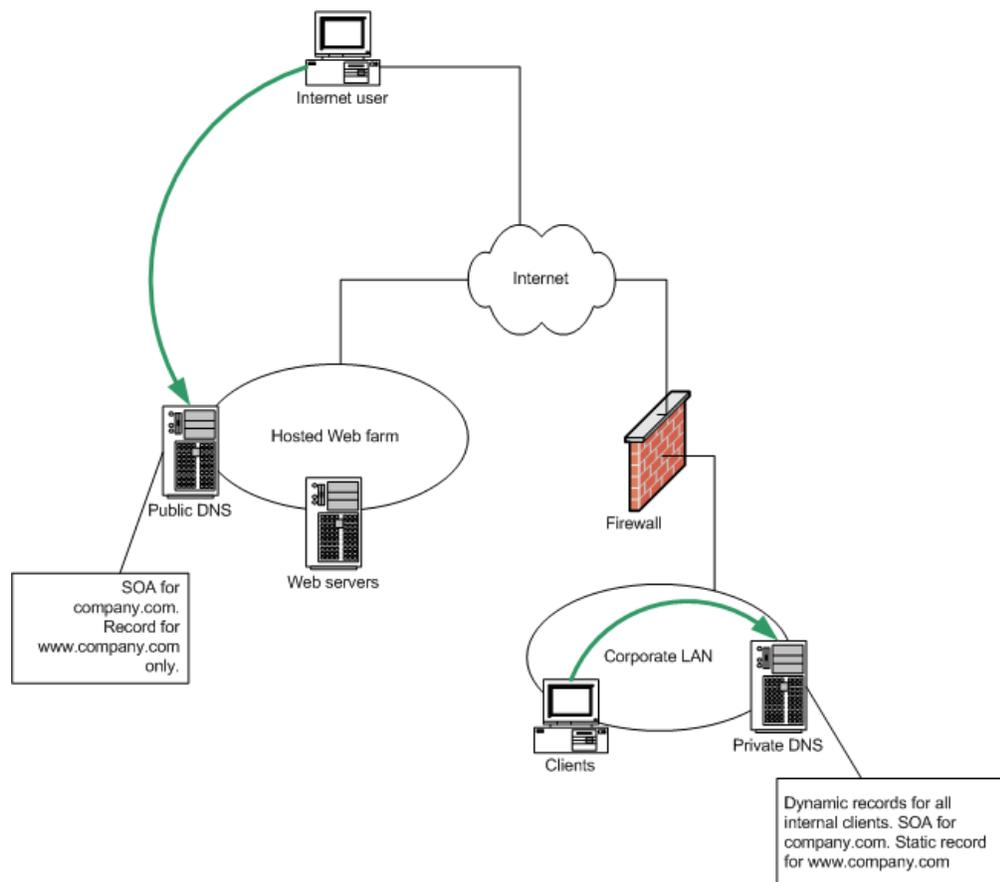


Figure 23.1: An example split DNS configuration.

In this example, you have two DNS servers: one on your internal network, and one that's accessible to the public (you'd likely have at least two in each location, but that's not relevant to the basic configuration). Both DNS servers believe they're authoritative for your domain, meaning that they are the final source for all records related to the domain.

 For more information about authoritative servers, see Question 21.

### **How it Works**

Your public domain record lists your public DNS server(s), meaning that the top-level domain (TLD) domain name servers will also list your public DNS server(s). When Internet users attempt to look up any host in your domain, such as `www.company.com`, they'll use your public DNS server. That server is configured with static records, including any Web servers, mail servers, and other publicly accessible servers.

Internally, your client computers are configured to use your internal DNS server. It might have dynamic DNS configured and believe it is authoritative for `company.com`. It might also be configured to forward requests to your ISP's DNS server so that your client computers can access Internet Web servers. Hence, the trick—because your Web server isn't connected to your internal DNS server, it won't be able to dynamically register its name. Thus, your internal DNS server won't contain a record for, say, `www.company.com`.

No problem, right? Isn't that what forwarding solves? Wrong. Because your internal DNS server is authoritative for `company.com`, it will never forward requests for anything involving `company.com`. "If I don't have it, nobody else will" is your DNS server's motto in this case. So you will need to create static records for all publicly accessible servers in the `company.com` domain. Doing so will permit internal users to access these external assets.

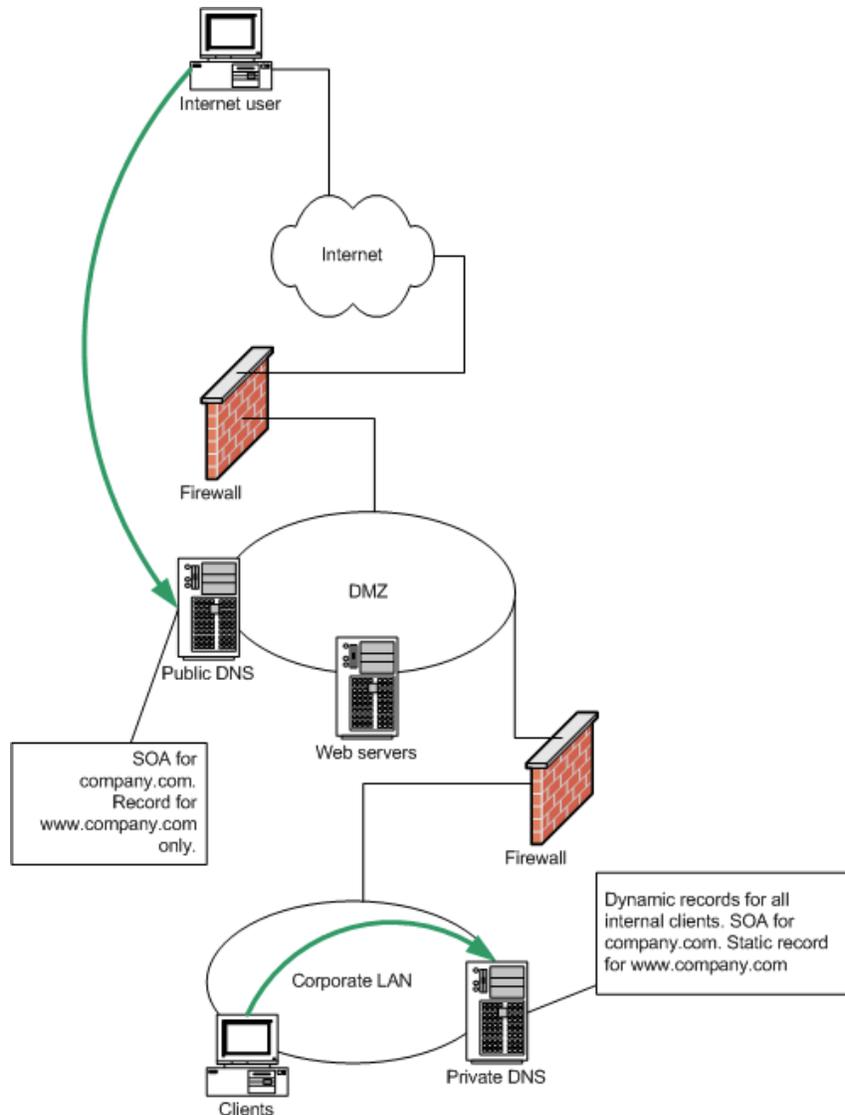
### **Split-Brain Pitfalls**

Split-brain DNS only has a couple of minor drawbacks. Primarily, you'll need to manually keep both DNS servers up to date with your publicly accessible resources. Technically, the public DNS server could use dynamic DNS, but the security implications of doing so aren't worth the convenience.

You can't perform zone transfers between the servers because they're both hosting a primary zone for `company.com`, and they can only transfer to another *secondary* zone of the same name. You wouldn't necessarily configure them as forwarders to one another, either; remember, authoritative servers never forward requests for their own domains.

### **Same-Site Split-Brain**

In the previous example, the external resources were hosted on a different network. So what if you host your own Web servers? The picture looks remarkably similar, as Figure 23.2 shows.



**Figure 23.3: Split DNS in a single location.**

In this example, a DMZ or perimeter network contains the publicly accessible resources, including the Web server and the public DNS server. The internal LAN contains the internal resources. Close physical proximity or network connectivity doesn't matter for split DNS; the two DNS servers never talk to one another, and each is unaware that the other even exists.

## Q.24: Why won't Active Directory let me create new objects?

**A:** Active Directory (AD) objects must all have an identifier that is unique within the domain. This identifier, called a security ID (SID), is a combination of a domain-wide identification number and a unique, per-object *relative identifier* (RID). Because every domain controller in a domain can create new objects, the possibility exists for duplicate RIDs, which would be a problem. To prevent that, each domain controller is only permitted to issue RIDs from a pool, and that pool is assigned by the RID master, a special Flexible Single Master Operations (FSMOs) role held by one domain controller in each domain.

 For more information about the RID master role, see Question 1.

Normally, when a domain controller runs out of RIDs, it contacts the RID master for a new pool. The RID master ensures that each domain controller receives a unique pool, preserving the uniqueness of the object SIDs in the domain. However, sometimes a domain controller uses up RIDs faster than it can get them. This situation can happen more frequently in a domain that is receiving migrated user accounts, each of which needs a RID. Automated migration tools can create new user accounts more quickly than a domain controller can get more RIDs.

You'll see evidence of this in the Directory Services event log on the affected domain controller. The event ID to look for is 16645, which reads:

The maximum account identifier allocated to this domain controller has been assigned. The domain controller has failed to obtain a new identifier pool. A possible reason for this is that the domain controller has been unable to contact the master domain controller. Account creation on this domain controller will fail until a new pool has been allocated. There may be network or connectivity problems in the domain, or the master domain controller may be offline or missing from the domain. Verify that the master domain controller is running and connected to the domain.

### ***Troubleshooting the Problem***

The event itself gives good advice: Make sure that the RID master is online, and if it isn't consider transferring the role to another domain controller.

 For instructions on transferring the RID master role, see Question 8.

You can verify that the RID pool is the problem by connecting directly to another domain controller and attempting to create a new user or group. If that domain controller is able to create the object, the problem is confined to the domain controller on which the event appeared. Once the RID master is back online, the affected domain controller will eventually request a new pool (there's no way to force the process), and start working again.

### ***Preventing the Problem***

If you know you're going to be using up RIDs quickly, consider expanding the size of the RID pool allocated to each domain controller. Doing so requires a registry hack on the domain controller that's playing RID master: Locate the key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\RID Values`. Modify the RID Block Size value from its default of 0 to a higher number. The default value of 0 is treated internally as 500, which is the default RID pool size.

 Prior to Service Pack 4 (SP4), values beyond 500 are treated as 500, meaning there's effectively no way to increase the pool size. SP4 corrects this problem. However, 500 is still the minimum size you can configure; setting the value to 400 will still be treated as 500.

Be sure to remember your change; if you ever decommission that domain controller or move the RID master role to another one, you will have to modify the new RID master's registry. You can safely modify this value on all domain controllers, if you like, to make this possibility less of a concern.

 For more details, refer to the Microsoft article "RID Pool Allocation and Sizing Changes in Windows 2000 SP4."

 Worried about using up all the RIDs? Don't. It's possible, but there are about  $2^{30}$  RIDs available in a domain, which represents millions and millions of objects.

### ***RID Threshold Changes***

Domain controllers are designed to request a new RID pool when they've exhausted 80 percent of their current pool. At a fast rate of consumption, especially when the domain's RID master is across a WAN link (as might be the case when migrating a field office), the remaining 20 percent of the RID pool is used up before the domain controller can snag a new pool.

Under Win2K Server SP4 and later, this threshold was changed to 50 percent. This decreases the chances of RID pool exhaustion becoming a problem, unless you're creating new objects at a truly breakneck pace.

### **Q.25: How can I see why some DNS queries are failing?'**

**A:** DNS is a reasonably simple system, although it does have some specific quirks that can make troubleshooting difficult.

 For more information about how DNS works, see Question 21.

One way to troubleshoot DNS problems is to use Network Monitor or a similar packet-capture tool to capture the client's DNS queries. Also look for the DNS server's replies. If your network has a firewall, attempt to capture the requests and replies on *both* sides of the firewall. Figure 25.1 shows a sample DNS communication in Network Monitor; the top pane shows the flow of responses and queries, and the bottom pane shows the detail of one response.

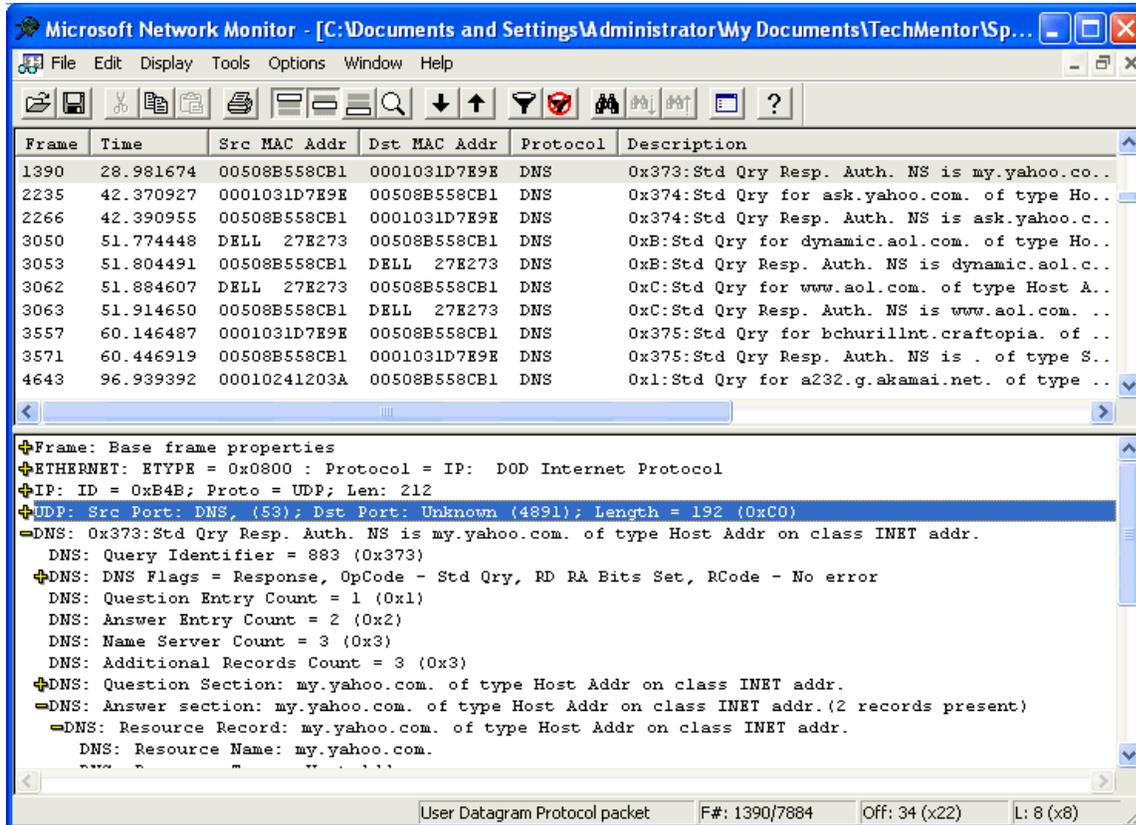


Figure 25.1: Sample DNS query response.

Fortunately, there's not much that can go wrong with DNS, and examining these packets will allow you to quickly narrow the source of the problem. Check the following:

- Is the packet making it through the firewall? Check captures from both sides for both the query and response. Queries will usually, although not always, be sent on UDP port 53; replies may come back on either UDP or TCP port 53. The DNS specification talks mostly about UDP for queries and responses, and so some firewalls might only allow UDP responses to come in. In the sample packet that Figure A shows, the **IP** line shows that this was sent via **Proto=UDP**, not TCP.
- Is any response coming back? If not, there's obviously a DNS configuration problem. If negative (empty) responses are coming back, it's possible that one or more servers in a chain of forwarders isn't working. Try to capture packets from your internal DNS server to wherever it forwards packets, and check for responses.
- Is the request properly formed? It's rare for it not to be, but compare suspect requests to those from computers that are working. Request packets are usually quite short.
- Is the response properly formed, and does it contain accurate data? You can use the response that Figure A shows as a sample. The response should contain the original query (**DNS Question Section**) and a reply (**DNS Answer section**). The reply should contain at least one entry with a valid IP address; you can use ping or another utility to verify the IP address returned.

## Nslookup

Nslookup is a command-line tool that is included with every Windows NT-based Windows operating system (OS) from NT 4.0 on up, and can be used to verify the workings of DNS. As Figure 25.2 shows, you can simply type any host or domain name to receive a list of records through DNS. This query shows a *non-authoritative answer*, meaning that the answer was provided from a DNS server other than Microsoft.com's own authoritative one. The server delivering the answer was SERVER1.mshome.net, which looks suspiciously like a gateway. The gateway probably forwarded the request to an ISP, which returned the reply. The actual record came from www.Microsoft.akadns.net, which is a DNS hosting service that probably helps Microsoft load-balance its DNS requests. This illustration shows how most replies on most corporate networks are received.



```

C:\WINDOWS\System32\cmd.exe - nslookup
>
> www.microsoft.com
Server: SERVER1.mshome.net
Address: 192.168.0.1

Non-authoritative answer:
Name: www.microsoft.akadns.net
Addresses: 207.46.249.27, 207.46.249.190, 207.46.249.222, 207.46.134.190
           207.46.134.222
Aliases: www.microsoft.com
>

```

**Figure 25.2: Nslookup results.**

Notice that the reply returned five unique IP addresses. It's likely that the DNS server is using round-robin to reorder these addresses on each query, helping to balance incoming traffic to this busy Web site. A second query, which Figure 25.3 shows, confirms this—notice that the first IP address in the second query was the second address in the first query.



```

C:\WINDOWS\System32\cmd.exe - nslookup
>
> www.microsoft.com
Server: SERVER1.mshome.net
Address: 192.168.0.1

Non-authoritative answer:
Name: www.microsoft.akadns.net
Addresses: 207.46.249.27, 207.46.249.190, 207.46.249.222, 207.46.134.190
           207.46.134.222
Aliases: www.microsoft.com
>
> www.microsoft.com
Server: SERVER1.mshome.net
Address: 192.168.0.1

Non-authoritative answer:
Name: www.microsoft.akadns.net
Addresses: 207.46.249.190, 207.46.249.222, 207.46.134.190, 207.46.134.222
           207.46.249.27
Aliases: www.microsoft.com
>

```

**Figure 25.3: Seeing round robin in action.**

Nslookup can be a valuable tool for seeing the details of how DNS is working. You can type ? at any Nslookup prompt (>) for a list of available commands.