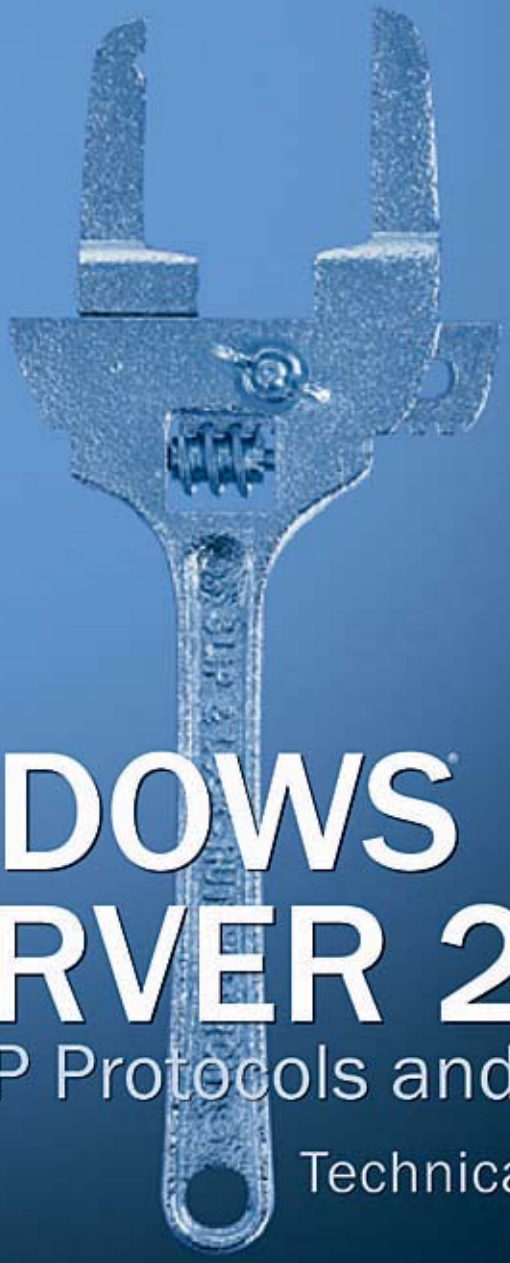
Microsoft

Technical Reference



Microsoft® WINDOWS® SERVER 2003

TCP/IP Protocols and Services

Technical Reference

Joseph Davies
Thomas Lee

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

Library of Congress Cataloging-in-Publication Data
Davies, Joseph.
Understanding IPv6 / Joseph Davies.
p. cm.
Includes index.
ISBN 0-7356-1245-5
1. TCP/IP (Computer network protocol) 2. Internet. I. Title.

TK5105.585 .D38 2002
004.62--dc21 2002029525

Printed and bound in the United States of America.

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

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

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

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

Acquisitions Editors: Juliana Aldous Atkinson, Jeff Koch
Project Editor: Maureen Williams Zimmerman
Technical Editor: Jim Johnson

Body Part No. X08-04478

Contents

PART III Transport Layer Protocols

Chapter 6
Internet Protocol (IP) Addressing

To successfully administer and troubleshoot IP internetworks, it is important to understand all aspects of IP addressing. One of the most important aspects of TCP/IP network administration is the assignment of unique and proper IP addresses to all the nodes of an IP internetwork. Although the concept of IP address assignment is simple, the actual mechanics of efficient IP address allocation using subnetting techniques are somewhat complicated. Additionally, it is important to understand the role of IP broadcast and multicast traffic and how these addresses map to Network Interface Layer addresses such as Ethernet and Token Ring media access control (MAC) addresses.

Types of IP Addresses

An IP address is a 32-bit logical address that can be one of the following types:

- **Unicast** A unicast IP address is assigned to a single network interface attached to an IP internetwork. Unicast IP addresses are used in one-to-one communications.
- **Broadcast** A broadcast IP address is designed to be processed by every IP node on the same network segment. Broadcast IP addresses are used in one-to-everyone communications.
- **Multicast** An IP multicast address is an address on which one or multiple nodes can be listening on the same or different network segments. IP multicast addresses are used in one-to-many communications.

Expressing IP Addresses

The IP address is a 32-bit value that computers are adept at manipulating. Humans, however, do not think in binary mode, 32 bits at a time. Because most humans are trained in the use of decimal (base 10 numbering system) rather than binary (base 2 numbering system), it is common to express IP addresses in a decimal form.

The 32-bit IP address is divided from the high-order bit to the low-order bit into four 8-bit quantities called *octets*. IP addresses are normally written as four separate decimal octets delimited by a period (a dot). This is known as dotted decimal notation.

For example, the IP address 00001010000000001111100010100011 is subdivided into four octets:

00001010 00000001 11110001 01000011

Each octet is converted to a base 10 number and separated from the others by periods:

10.1.241.67

A generalized IP address is indicated with *w.x.y.z,* as Figure 6-1 shows.

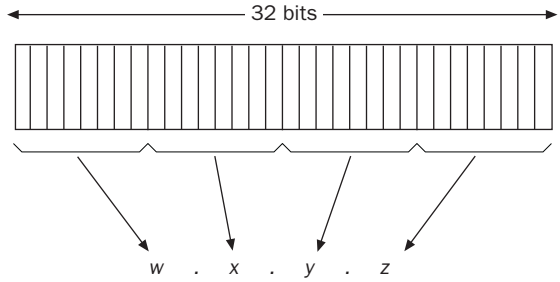


Figure 6-1. *The generalized IP address consisting of 32 bits expressed in dotted decimal notation.*

Converting from Binary to Decimal

To convert a binary number to its decimal equivalent, add the numbers represented by the bit positions that are set to 1. Figure 6-2 shows an 8-bit number and the decimal value of each position.

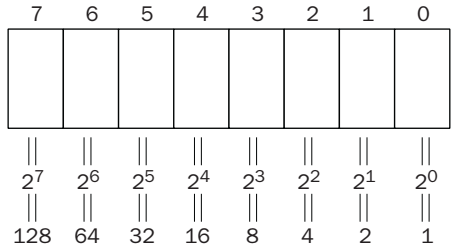


Figure 6-2. *An 8-bit number showing bit positions and their decimal equivalents.*

For example, the 8-bit binary number 01000011 is 67 (64 + 2 + 1). The maximum number that can be expressed with an 8-bit number (11111111) is 255 (128 + 64 + 32 + 16 + 8 + 4 + 2 + 1).

Converting from Decimal to Binary

To convert from decimal to binary, analyze the decimal number to see if it contains the quantities represented by the bit positions from the high-order bit to the low-order bit. Starting from the high-order bit quantity (128), if each quantity is present, the bit in that bit position is set to 1. For example, the decimal number 211 contains 128, 64, 16, 2, and 1. Therefore, 211 is 11010011 in binary notation.

IP Addresses in the IP Header

IP addresses are used in the IP header's Source Address and Destination Address fields.

- The IP header's Source Address field is always either a unicast address or the special address 0.0.0.0. The unspecified IP address, 0.0.0.0, is used only when the IP node is not configured with an IP address and the node is attempting to obtain an address through a configuration protocol such as Dynamic Host Configuration Protocol (DHCP).
- The IP header's Destination Address field is a unicast address, multicast address, or broadcast address.

Unicast IP Addresses

Each network interface on which TCP/IP is active must be identified by a unique, logical, unicast IP address. The unicast IP address is a logical address because it is an Internet Layer address that has no direct relation to the address being used at the Network Interface Layer. For example, the unicast IP address assigned to a host on an Ethernet network has no relation to the 48-bit MAC address used by the Ethernet network adapter.

The unicast IP address is an internetwork address for IP nodes that contains a network ID and a host ID.

- The network ID, or network address, identifies the nodes that are located on the same logical network. In most cases, a logical network is the same as a physical network segment with boundaries that are defined by IP routers. In some cases, multiple logical networks exist on the same physical network using a practice called *multinetting*. All nodes on the same logical network share the same network ID. If all nodes on the same logical network are not configured with the same network ID, routing or delivery problems occur. The network ID must be unique to the internetwork.
- The host ID, or host address, identifies a node within a network. A node is a router or host (a nonrouter interface such as a workstation, server, or other TCP/IP–based system). The host ID must be unique within each network segment.

Note The term *network ID* applies to class-based network IDs, subnetted network IDs, and classless network IDs.

Figure 6-3 is an example of a unicast IP address and its network ID and host ID portions.

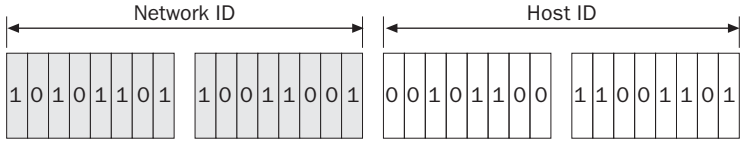


Figure 6-3. *The structure of an example IP address showing the network ID and host ID.*

A History Lesson: IP Address Classes

This section is called "A History Lesson" because modern networks are not based on the Internet address classes. Because of the Internet's recent rapid expansion, the Internet authorities saw clearly that the original class-based structure did not scale well to the size of a global internetwork. For example, if the Internet authorities were still handing out class-based addresses, there would be hundreds of thousands of routes in the routing tables of Internet backbone routers. To prevent this scaling problem, addressing on the modern Internet is classless. However, the understanding of Internet address classes is an important element in understanding IP addressing.

RFC 791 defined the unicast IP address in terms of address classes to create well-defined networks of various sizes. The design goal was to create the following:

- A small number of large networks (networks with a large amount of nodes)
- A moderate number of moderate-sized networks
- A large number of small networks

The result was the creation of address classes, subdivisions of the 32-bit IP address space defined by setting high-order bits and dividing the remaining bits into network ID and host ID.

Class A

Class A addresses are designed for networks with a large number of hosts. The high-order bit is set to 0. The first 8 bits (the first octet) are defined as the network ID; the last 24 bits (the last three octets) are defined as the host ID. Figure 6-4 illustrates the class A address.

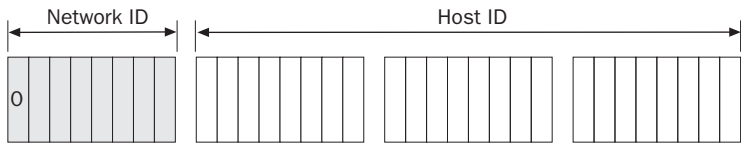


Figure 6-4. *The class A address showing the network ID and the host ID.*

Class B

Class B addresses are designed for moderate-sized networks with a moderate number of hosts. The two high-order bits are set to 10. The first 16 bits (the first two octets) are defined as the network ID; the last 16 bits (the last two octets) are defined as the host ID. Figure 6-5 illustrates the class B address.

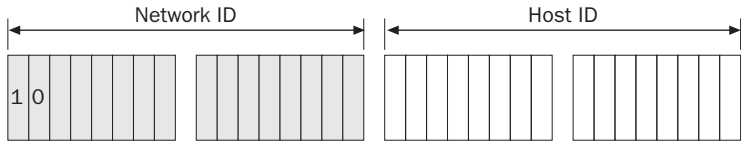


Figure 6-5. *The class B address showing the network ID and the host ID.*

Class C

Class C addresses are designed for small networks with a small number of hosts. The three high-order bits are set to 110. The first 24 bits (the first three octets) are defined as the network ID; the last 8 bits (the last three octets) are defined as the host ID. Figure 6-6 illustrates the class C address.

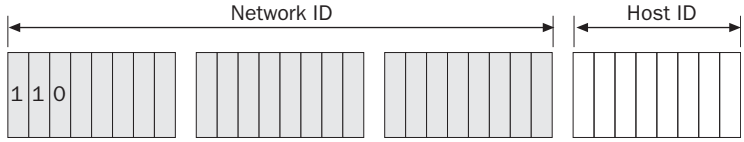


Figure 6-6. *The class C address showing the network ID and the host ID.*

Additional Address Classes

Class D and E addresses are defined, in addition to unicast address classes A, B, and C.

Class D

Class D addresses are for IP multicast addresses. The four high-order bits are set to binary 1110. The next 28 bits are used for individual IP multicast addresses. For more information on IP multicast addresses, see the section entitled "IP Multicast Addresses," later in this chapter. The Microsoft Windows Server 2003 family supports class D addresses for IP multicast traffic.

Class E

Class E addresses are experimental addresses reserved for future use. The five high-order bits in a class E address are set to 11110. The Windows Server 2003 family does not support the use of class E addresses.

Rules for Enumerating Network IDs

When enumerating IP network IDs, the following rules apply:

- **The network ID cannot begin with 127 as the first octet.** All 127.*x.y.z* addresses are reserved as loopback addresses.
- **All the bits in the network ID cannot be set to 1.** Network IDs set to all 1s are reserved for broadcast addresses.
- **All the bits in the network ID cannot be set to 0.** Network IDs set to all 0s are reserved for indicating a host on the local network.
- **The network ID must be unique to the IP internetwork.**

Table 6-1 lists the ranges of network IDs based on the IP address classes. Network IDs are expressed by setting all host bits to 0 and expressing the result in dotted decimal notation.

Table 6-1. Address Class Ranges of Network IDs

Address Class	First Network ID	Last Network ID	Number of Networks
Class A	1.0.0.0	126.0.0.0	126
Class B	128.0.0.0	191.255.0.0	16,384
Class C	192.0.0.0	223.255.255.0	2,097,152

Note IP network IDs, even though expressed in dotted decimal notation, are not IP addresses assigned to network interfaces. The IP network ID is the network address that is common for all network interfaces attached to the same logical network.

Rules for Enumerating Host IDs

When enumerating IP host IDs, the following rules apply:

- **All bits in the host ID cannot be set to 1.** Host IDs set to all 1s are reserved for broadcast addresses.
- **All the bits in the host ID cannot be set to 0.** Host IDs set to all 0s are reserved for the expression of IP network IDs.
- **The host ID must be unique to the network.**

Table 6-2 lists the ranges of host IDs based on the IP address classes.

Table 6-2. Address Class Ranges of Host IDs

Address Class	First Host ID	Last Host ID	Number of Hosts
Class A	*w*.0.0.1	*w*.255.255.254	16,777,214
Class B	*w.x*.0.1	*w.x*.255.254	65,534
Class C	*w.x.y*.1	*w.x.y*.254	254

Subnets and the Subnet Mask

Subnetting is designed to make more efficient use of a fixed address space, namely, an IP network ID. The network bits are fixed and the host bits are variable. Originally, the host bits were designed to indicate host IDs within an IP network ID. With subnetting, host ID bits can be used to express a combination of a subnetwork ID and a subnetwork host ID, thereby better utilizing the host bits.

Consider a class B network that has 65,534 possible hosts. A network segment of 65,534 hosts is technically possible but impractical because of the accumulation of broadcast traffic. All nodes on the same physical network segment belong to the same broadcast domain and share the same broadcast traffic. Because making all 65,534 hosts share the same broadcast traffic is not a practical configuration, most of the host IDs are not usable.

To create smaller broadcast domains and make better use of the host bits, RFC 950 defines a method of subdividing a network ID into subnetworks—subsets of the original class-based network—by using bits in the host ID portion of the original IP network ID. Each subnetwork, or subnet, is assigned a new subnetted network ID. Hosts on subnets are assigned host IDs from the remaining host bits in the subnetted network ID.

Although RFC 950 discusses subnetting in terms of class-based network IDs, subnetting is a general technique that can be used on classless network IDs or used recursively on subnetted network IDs. This is described in the section entitled "Variable-Length Subnetting," later in this chapter.

The proper subnetting of a network ID is transparent to the rest of the IP internetwork. For example, consider the class B network ID of 131.107.0.0 (shown in Figure 6-7), which is connected to the Internet. The class-based network ID was obtained from the InterNIC and is a fixed address space. Because this class B network ID represents an impractical broadcast domain, it is subnetted. However, in subnetting 131.107.0.0, we should not require any reconfiguration of the Internet routers.

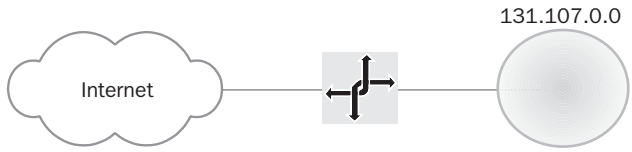


Figure 6-7. *The class B network 131.107.0.0 before subnetting.*

From an analysis of broadcast traffic, it is determined that there should be no more than 250 nodes on each broadcast domain. Therefore, network ID 131.107.0.0 is subnetted to look like a class C address by using the first 8 high-order host bits (the third octet represented by *y*) for the subnetted network ID. Note that before the subnetting, only the first two octets are considered the network ID. After the subnetting, the first three octets are considered the network ID. The new network IDs are 131.107.1.0, 131.107.2.0, and 131.107.3.0, as Figure 6-8 shows.

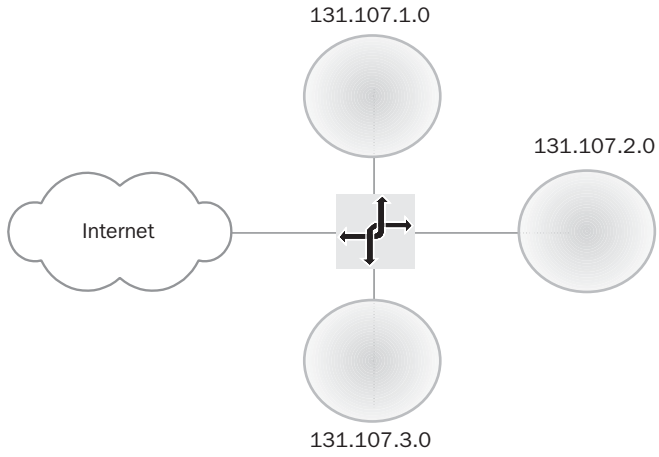


Figure 6-8. *The class B network 131.107.0.0 after subnetting.*

The IP router connected to the Internet has an interface on each of the subnets and is aware of the new subnetting scheme. The IP router forwards IP datagrams from the Internet to the host on the appropriate subnet. The Internet routers are completely unaware of the subnetting of 131.107.0.0. They still consider all IP addresses in the range of 131.107.0.0 through 131.107.255.255 to be reachable through the IP router's Internet interface.

The Subnet Mask

With subnetting, a host or router can no longer assume the network ID and host ID designations of the IP address classes. The node needs additional configuration to distinguish the network ID and host ID portions of an IP address, whether the network ID is class-based, classless, or a subnetted network ID.

RFC 950 defines the use of a bit mask to identify which bits in the IP address belong to the network ID and which belong to the host ID. This bit mask, called a *subnet mask* or *address mask*, is defined by the following:

- If the bit position corresponds to a bit in the network ID, it is set to 1.
- If the bit position corresponds to a bit in the host ID, it is set to 0.

Since the publication of RFC 950, TCP/IP nodes require a subnet mask to be configured for each IP address, even when class-based addressing is used. A default subnet mask corresponds to a class-based network ID. A custom subnet mask corresponds to either a classless network ID or a subnetted network ID. The subnet mask is the definitive piece of configuration information that allows the node to determine its own network ID.

Subnet Masks in Dotted Decimal Representation

Frequently, the subnet mask is expressed in dotted decimal notation. Although expressed in the same form as an IP address, the subnet mask is not an IP address. As an example of subnet masks in dotted decimal notation, default subnet masks are based on the IP address classes. Table 6-3 lists the default subnet masks for class A, B, and C network IDs in dotted decimal notation.

Table 6-3. Dotted Decimal Notation for Default Subnet Masks

Address Class	Bits for Subnet Mask	Subnet Mask
Class A	11111111 00000000 00000000 00000000	255.0.0.0
Class B	11111111 11111111 00000000 00000000	255.255.0.0
Class C	11111111 11111111 11111111 00000000	255.255.255.0

A custom subnet mask is used whenever you perform nonclassful addressing. In our earlier example, the classful network ID 131.107.0.0 is subnetted by using the third octet for subnets. The subnetted network ID 131.107.1.0 no longer uses the default subnet mask 255.255.0.0. To express the third octet as part of the network ID, the custom subnet mask 255.255.255.0 is used.

The subnetted network ID and its corresponding subnet mask are expressed in dotted decimal notation as 131.107.1.0, 255.255.255.0.

Network Prefix Length Representation of Subnet Masks

Although it is technically possible to subnet IP network IDs by choosing host bits in a noncontiguous fashion, it is impractical and mathematically challenging to enumerate the subnetted network IDs and the host IDs per subnet. For this reason, you must subnet by choosing host bits in a contiguous fashion from the high-order host bit.

Because the network ID bits are always contiguous starting from the highest order bit, an easier and more compact way of expressing the subnet mask is to indicate the number of network ID bits using network prefix notation, or Classless Inter-Domain Routing (CIDR) notation. Network prefix notation views the IP address in terms of the prefix (the network ID) and the suffix (the host ID). Network prefix notation is:

/# of bits in the network ID

Network prefix notation is commonly used with TCP/IP implementations other than the Windows Server 2003 family, and it is an important notation to understand looking forward to IP version 6 (IPv6).

Table 6-4 lists the equivalent subnet mask in network prefix notation for the IP address classes.

Table 6-4. Network Prefix Notation for Default Subnet Masks

Address Class	Bits for Subnet Mask	Network Prefix
Class A	11111111 00000000 00000000 00000000	/8
Class B	11111111 11111111 00000000 00000000	/16
Class C	11111111 11111111 11111111 00000000	/24

In our earlier example, the classful network ID 131.107.0.0, with the subnet mask of 255.255.0.0, is expressed in network prefix notation as 131.107.0.0/16. If 131.107.0.0 were subnetted by using the third octet to express subnets, a total of 24 contiguous bits would be used for the subnetted network ID. The subnetted network ID 131.107.1.0 and its corresponding subnet mask are expressed in network prefix notation as 131.107.1.0/24.

Expressing Network IDs

The fixed network ID bits and the subnet mask define the network ID. Therefore, network IDs must always be expressed by the combination of the network ID and a subnet mask. Expressing a network ID without its subnet mask is ambiguous. For example, for the network ID 10.16.0.0, which bits are used for the network ID? The first 16? The first 24? The first 12?

The following are examples of properly expressed network IDs:

- 192.168.45.0, 255.255.255.0
- 10.99.0.0/16

All hosts on the same logical network must be using the same network ID bits and the same subnet mask. For example, 131.107.0.0/16 is not the same as 131.107.0.0/24. For the network ID 131.107.0.0/16, the usable IP addresses range from 131.107.0.1 through 131.107.255.254. For the network ID 131.107.0.0/24, the usable IP addresses range from 131.107.0.1 through 131.107.0.254. Clearly, 131.107.0.0/16 and 131.107.0.0/24 do not represent the same group of hosts.

Determining the Network ID

In earlier examples, classful network IDs and subnetted network IDs all fell along octet boundaries where it was easy to determine the network ID and host ID portion of the IP address. However, real-world subnetting is not always done along octet boundaries. For example, some network administrators might determine that, for their situation, they need only three host bits for subnetting.

Because subnetting can occur along nonoctet boundaries, there must be a method of determining the network ID from an IP address with an arbitrary subnet mask. IP uses a method called a *bit-wise logical AND* to extract the network ID.

Recall how the subnet mask is defined: 1 is used to indicate a network ID bit and 0 is used to indicate a host ID bit. In a logical AND comparison, the result is 1 when the value of the two bits being compared is 1. Otherwise, the result is 0. This comparison is done for all 32 bits of the IP address and subnet mask. The result of the bit-wise logical AND of the IP address and the subnet mask is the network ID.

For example, what is the network ID of the IP node 131.107.164.26 with a subnet mask of 255.255.240.0? To obtain the result in binary notation, convert both the IP address and subnet mask to binary. Then perform the logical AND comparison for each bit.

IP address 10000011 01101011 10100100 00011010

Subnet mask 11111111 11111111 11110000 00000000

Network ID 10000011 01101011 10100000 00000000

The result of the bit-wise logical AND of the 32 bits of the IP address and the subnet mask is the network ID 131.107.160.0 with the subnet mask of 255.255.240.0.

Notice the following:

- The bits in the network ID portion of the IP address are copied directly to the result. A value of 1 in the network ID portion of the IP address becomes a 1 in the result. A value of 0 in the network ID portion of the IP address becomes a 0 in the result.
- All bits in the host ID portion of the IP address are set to 0. Because the subnet mask uses a 0 for host ID bit positions, the logical AND comparison always yields a 0.

Therefore, because the bits in the network ID are copied and the bits in the host ID are set to 0, the result must be the network ID.

How to Subnet

The act of subnetting a network ID is a relatively complex procedure; although there are numerous subnet calculators available, the ability to subnet is a vital skill for any TCP/IP network administrator.

Subnetting is done in two basic steps:

1. Based on your design requirements, decide how many host bits you need for the proper balance between number of subnets and number of hosts per subnet.
2. Based on the number of host bits chosen, enumerate the subnetted network IDs, including the ranges of usable IP addresses for each subnetted network ID. The actual mechanics of defining the subnetted network IDs can be done in binary or decimal notation.

There are two methods for the second step of subnetting, the enumeration of the subnetted network IDs:

- The binary method, in which the individual bits of the subnetted network IDs are manipulated and converted to dotted decimal notation, can be used to subnet. However, this method does not scale well to large numbers of subnets. It is described here primarily to illustrate the subnetting process in its most fundamental form.
- The decimal method, in which subnetted network IDs are derived from calculations on decimal numbers, scales well to large numbers of subnets and lends itself well to spreadsheets and programming code.

Step 1: Determining the Number of Host Bits

To determine the number of host bits required for subnetting, perform an analysis of your internetwork. You should determine the following:

- **The number of subnets needed both now and in the future** Be sure to plan for expansion. Subnetting an existing network requires reassigning IP addresses to IP interfaces. Although DHCP can ease this burden, routers and other fixed-address types of hosts might need to be manually reconfigured. Subnetting is not something you want to do often.
- **The maximum number of hosts needed on each subnet** This number depends on how many hosts you want sharing the same broadcast traffic. In most cases, when choosing between more subnets and more hosts per subnet, the practical choice is to choose more subnets.

There is an inverse relationship between the number of subnets and the number of hosts per subnet that can be supported by a given subnetting scheme. As Figure 6-9 illustrates, when you choose more host bits, the number of subnets goes up, but the number of hosts per subnet goes down by approximately a factor of 2.

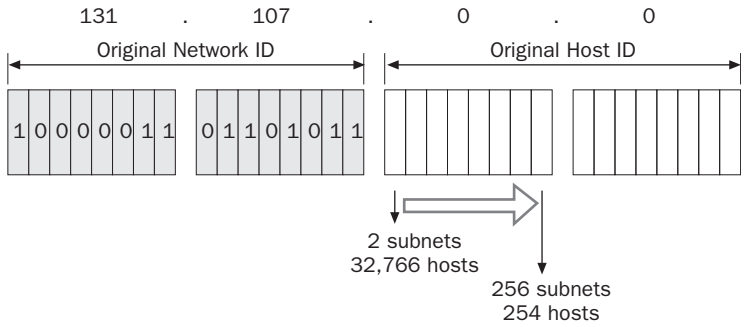


Figure 6-9. *The relationship between the number of subnets and hosts per subnet when subnetting the class B network ID 131.107.0.0.*

If we choose one host bit when subnetting the class B network ID 131.107.0.0, two subnets can be expressed, with 32,766 hosts per subnet. If we choose eight host bits, 256 subnets can be expressed with 254 hosts per subnet.

Determine how many subnets you need now, and plan for growth by estimating how many you will need in the next five years. Each physical network segment is a subnet. Point-to-point wide area network (WAN) connections such as leased lines might need subnetted network IDs, unless your routers support unnumbered connections. Non-broadcast multiple access (NBMA) WAN technologies such as frame relay need subnetted network IDs. Use additional host bits if the remaining host bits can express more hosts per subnet than you will need.

Subnetting always starts with a fixed address space in the form of a network ID. The network ID to be subnetted can be a classful network ID, a classless network ID (as allocated using CIDR), or a previously subnetted classful or classless network ID. The fixed address space contains a sequence of bits that are fixed (the network ID bits) and a sequence of bits that are variable (the host ID bits).

Based on your analysis of the desired number of subnets and number of hosts per subnet, a specific number of high-order host bits are converted from host bits into subnet bits. The combination of the original network ID bits and the converted host bits becomes the new subnetted network ID.

As you determine how many host bits you need, you determine the new subnet mask for your subnetted network IDs.

Tables 6-5, 6-6, and 6-7 list the subnetting of classful network IDs according to the requirement of a specific number of subnets. These tables can be useful when determining a subnetting scheme for a class-based network ID based on a required number of subnets and a desired number of hosts per subnet.

Table 6-5. Subnetting of a Class A Network ID

Required Number of Subnets	Number of Host Bits	Subnet Mask	Number of Hosts per Subnet
1–2	1	255.128.0.0 or /9	8,388,606
3–4	2	255.192.0.0 or /10	4,194,302
5–8	3	255.224.0.0 or /11	2,097,150
9–16	4	255.240.0.0 or /12	1,048,574
17–32	5	255.248.0.0 or /13	524,286
33–64	6	255.252.0.0 or /14	262,142
65–128	7	255.254.0.0 or /15	131,070
129–256	8	255.255.0.0 or /16	65,534
257–512	9	255.255.128.0 or /17	32,766

(continued)

Table 6-5. *(continued)*

Required Number of Subnets	Number of Host Bits	Subnet Mask	Number of Hosts per Subnet
513–1024	10	255.255.192.0 or /18	16,382
1025–2048	11	255.255.224.0 or /19	8190
2049–4096	12	255.255.240.0 or /20	4094
4097–8192	13	255.255.248.0 or /21	2046
8193–16,384	14	255.255.252.0 or /22	1022
16,385–32,768	15	255.255.254.0 or /23	510
32,769–65,536	16	255.255.255.0 or /24	254
65,537–131,072	17	255.255.255.128 or /25	126
131,073–262,144	18	255.255.255.192 or /26	62
262,145–524,288	19	255.255.255.224 or /27	30
524,289–1,048,576	20	255.255.255.240 or /28	14
1,048,577–2,097,152	21	255.255.255.248 or /29	6
2,097,153–4,194,304	22	255.255.255.252 or /30	2

Table 6-6. Subnetting of a Class B Network ID

Required Number of Subnets	Number of Host Bits	Subnet Mask	Number of Hosts per Subnet
1–2	1	255.255.128.0 or /17	32,766
3–4	2	255.255.192.0 or /18	16,382
5–8	3	255.255.224.0 or /19	8190
9–16	4	255.255.240.0 or /20	4094
17–32	5	255.255.248.0 or /21	2046
33–64	6	255.255.252.0 or /22	1022
65–128	7	255.255.254.0 or /23	510
129–256	8	255.255.255.0 or /24	254
257–512	9	255.255.255.128 or /25	126
513–1024	10	255.255.255.192 or /26	62
1025–2048	11	255.255.255.224 or /27	30
2049–4096	12	255.255.255.240 or /28	14
4097–8192	13	255.255.255.248 or /29	6
8193–16,384	14	255.255.255.252 or /30	2

Table 6-7. Subnetting of a Class C Network ID

Required Number of Subnets	Number of Host Bits	Subnet Mask	Number of Hosts per Subnet
1–2	1	255.255.255.128 or /25	126
3–4	2	255.255.255.192 or /26	62
5–8	3	255.255.255.224 or /27	30
9–16	4	255.255.255.240 or /28	14
17–32	5	255.255.255.248 or /29	6
33–64	6	255.255.255.252 or /30	2

Step 2: Defining the Subnetted Network IDs (Binary Method)

The technique presented here describes how to subnet an arbitrary network ID into subnets that yield both subnetted network IDs and their corresponding range of valid IP addresses using binary analysis. There are other techniques that might seem easier, but they are typically limited in scope. This technique works for any subnetting situation.

Step 2a: Enumerating the Subnetted Network IDs (Binary)

Create a three-column table with 2^n rows where n is the number of host bits chosen for the subnetting. The first column is used for the subnet number, the second column is for the binary representation of the subnetted network ID, and the third column is for the dotted decimal representation of the subnetted network ID.

For the binary representation for each entry in the table, the original network ID bits are fixed at their original values. The host bits chosen for subnetting, hereafter known as the subnet bits, are allowed to vary over all of their possible values, and the remaining host bits are set to 0.

The table's first entry is the subnet, defined by setting all the subnet bits to 0 (also called the all-zeros subnet). The result is converted to dotted decimal notation. This subnetted network ID does not appear to be different from the original network ID; but remember that a network ID is a combination of the dotted decimal notation and a subnet mask. With the new subnet mask, the subnetted network ID is clearly different from the original network ID.

In the following entries, treat the subnet bits as though they were distinct binary numbers. Increment the value within the subnet bits and convert the result of the entire 32-bit subnetted network ID to dotted decimal notation.

As an example of this technique, subnet the class B network ID 131.107.0.0 by using three bits of the classful host ID. The new subnet mask for the subnetted network IDs is 255.255.224.0, or /19. Based on using three host bits, create a table with eight entries ($8 = 2^3$). The first entry is the all-zeros subnet. The additional entries are increments of the binary number represented by the subnet bits (underlined). Table 6-8 lists the subnetted network IDs.

Table 6-8. A 3-Bit Subnetting of 131.107.0.0 (Binary)

Subnet	Binary Representation	Subnetted Network ID
1	10000011.01101011.000<u>00000</u>.00000000	131.107.0.0/19
2	10000011.01101011.001<u>00000</u>.00000000	131.107.32.0/19
3	10000011.01101011.010<u>00000</u>.00000000	131.107.64.0/19
4	10000011.01101011.011<u>00000</u>.00000000	131.107.96.0/19
5	10000011.01101011.100<u>00000</u>.00000000	131.107.128.0/19
6	10000011.01101011.101<u>00000</u>.00000000	131.107.160.0/19
7	10000011.01101011.110<u>00000</u>.00000000	131.107.192.0/19
8	10000011.01101011.111<u>00000</u>.00000000	131.107.224.0/19

Step 2b: Enumerating IP Address Ranges for Each Subnetted Network ID (Binary)

For each subnetted network ID, the range of valid IP addresses must be determined as follows:

1. Create a three-column table with 2^n entries where n is the number of host bits chosen for the subnetting. The first column is used for the subnet number, the second column is for the binary representation of the first and last IP address in the range, and the third column is for the dotted decimal representation of the first and last IP address in the range. Alternately, you can extend the table created for enumerating the subnetted network IDs by adding two columns.
2. Express the first and last IP address in the range in binary notation. The first IP address is defined by setting the remaining host bits to 0, except for the last host bit. The last IP address is defined by setting the remaining host bits to 1, except for the last host bit.
3. Convert the binary representation of the first and last IP address to dotted decimal notation.
4. Repeat steps 2 and 3 until the table is complete.

To continue our example, Table 6-9 lists the enumeration of the range of valid IP addresses for the 3-bit subnetting of 131.107.0.0. The host bits are underlined.

Table 6-9. Enumeration of IP Addresses for the 3-Bit Subnetting of 131.107.0.0 (Binary)

Subnet	Binary Representation	Range of IP Addresses
1	10000011.01101011.000<u>00000.00000001</u> – 10000011.01101011.000<u>11111.11111110</u>	131.107.0.1 – 131.107.31.254
2	10000011.01101011.001<u>00000.00000001</u> – 10000011.01101011.001<u>11111.11111110</u>	131.107.32.1 – 131.107.63.254
3	10000011.01101011.010<u>00000.00000001</u> – 10000011.01101011.010<u>11111.11111110</u>	131.107.64.1 – 131.107.95.254

(continued)

Table 6-9. *(continued)*

Subnet	Binary Representation	Range of IP Addresses
4	10000011.01101011.01100000.00000001 – 10000011.01101011.01111111.11111110	131.107.96.1 – 131.107.127.254
5	10000011.01101011.10000000.00000001 – 10000011.01101011.10011111.11111110	131.107.128.1 – 131.107.159.254
6	10000011.01101011.10100000.00000001 – 10000011.01101011.10111111.11111110	131.107.160.1 – 131.107.191.254
7	10000011.01101011.11000000.00000001 – 10000011.01101011.11011111.11111110	131.107.192.1 – 131.107.223.254
8	10000011.01101011.11100000.00000001 – 10000011.01101011.11111111.11111110	131.107.224.1 – 131.107.255.254

Step 2: Defining the Subnetted Network IDs (Decimal Method)

The previous technique describes a subnetting technique using binary. Although this method works for any valid subnetting scheme, it does not scale well. For example, if you were performing a 10-bit subnetting, you would have 1024 entries in the table. Whereas programmers are adept at binary manipulation and can create programs to automate this process, nonprogrammers find it easier to work with decimal numbers. Therefore, the following technique treats the 32-bit network ID and IP address as a single decimal number to enumerate the subnetted network ID and its corresponding range of IP addresses. Either technique—binary or decimal—yields the same result.

Step 2a: Enumerating the Subnetted Network IDs (Decimal)

1. Create a three-column table with 2^n entries where n is the number of host bits chosen for the subnetting. The first column is used for the subnet number the second column is for the decimal representation of the subnetted network ID, and the third column is for the dotted decimal representation of the subnetted network ID.
2. Convert the original network ID from dotted decimal notation ($w.x.y.z$) to N, its decimal representation:

   $N = w \times 16777216 + x \times 65536 + y \times 256 + z$
3. Compute I, the increment value, based on h, the number of host bits remaining:

   $I = 2^h$
4. For the first table entry, the all-zeros subnet, the decimal representation of the subnetted network ID is N, and the subnetted network ID is $w.x.y.z$, with its new subnet mask.
5. For the decimal representation of the next table entry, add the increment I to the previous entry.

6. Convert the decimal representation of the subnetted network ID to dotted decimal notation (*W.X.Y.Z*) using the following formulas (where *s* is the decimal representation of the subnetted network ID):

   W = int (s/16777216)

   X = int ((s mod 16777216)/65536)

   Y = int ((s mod 65536)/256)

   Z = s mod 256

   In the formulas, *int ( )* denotes integer division and yields the integer multiple, and *mod ( )* denotes the modulus operator and yields the remainder after division.

7. Repeat steps 5 and 6 until the table is complete.

To compare the two techniques and verify that they will both yield the same result, let us perform a decimal 3-bit subnetting of 131.107.0.0.

Based on n = 3, we create a table with eight entries. The entry for Subnet 1 is the all-zeros subnet. *N*, the decimal representation of 131.107.0.0, is 2204827648 (131 × 16777216 + 107 × 65536). Because there are 13 remaining host bits, the increment value *I* is 2^{13}, or 8192. Entries for Subnets 2 through 8 are incremented by 8192.

Table 6-10 lists the subnetted network IDs of 131.107.0.0.

Table 6-10. A 3-Bit Subnetting of 131.107.0.0 (Decimal)

Subnet	Decimal Representation	Subnetted Network ID
1	2204827648	131.107.0.0/19
2	2204835840	131.107.32.0/19
3	2204844032	131.107.64.0/19
4	2204852224	131.107.96.0/19
5	2204860416	131.107.128.0/19
6	2204868608	131.107.160.0/19
7	2204876800	131.107.192.0/19
8	2204884992	131.107.224.0/19

Step 2b: Enumerating IP Address Ranges for Each Subnetted Network ID (Decimal)

For each subnetted network ID, the range of valid IP addresses must be determined as follows:

1. Create a three-column table with 2^n entries where n is the number of host bits chosen for the subnetting. The first column is used for the subnet number, the second column is for the decimal representation of the first and last IP address in the range, and the third column is for the dotted decimal representation of the first and last IP address in the range. Alternately, you can extend the table created for enumerating the subnetted network IDs by adding two columns.

2. Compute the increment value *J* based on *h*, the number of host bits remaining:
   $J = 2^h - 2$
3. The decimal representation of the first IP address is *N* + 1, where *N* is the decimal representation of the subnetted network ID. The decimal representation of the last IP address is *N* + *J*.
4. Convert the decimal representation of the first and last IP address to dotted decimal notation (*W.X.Y.Z*) using the following formulas (where *s* is the decimal representation of the first or last IP address):
   W = int (*s*/16777216)
   X = int ((*s* mod 16777216)/65536)
   Y = int ((*s* mod 65536)/256)
   Z = *s* mod 256
   In the formulas, *int ( )* denotes integer division and yields the integer multiple, and *mod ( )* denotes the modulus operator and yields the remainder after division.
5. Repeat steps 3 and 4 until the table is complete.

To continue with our example, we enumerate the range of valid IP addresses for the 3-bit subnetting of 131.107.0.0. Compute the increment value $J = 2^{13} - 2 = 8190$. Table 6-11 lists the ranges of IP addresses for the eight subnetted network IDs.

Table 6-11. Enumeration of IP Addresses for the 3-Bit Subnetting of 131.107.0.0 (Decimal)

Subnet	Decimal Representation	Range of IP Addresses
1	2204827649 – 2204835838	131.107.0.1 – 131.107.31.254
2	2204835841 – 2204844030	131.107.32.1 – 131.107.63.254
3	2204844033 – 2204852222	131.107.64.1 – 131.107.95.254
4	2204852225 – 2204860414	131.107.96.1 – 131.107.127.254
5	2204860417 – 2204868606	131.107.128.1 – 131.107.159.254
6	2204868609 – 2204876798	131.107.160.1 – 131.107.131.107
7	2204876801 – 2204884990	131.107.192.1 – 131.107.223.254
8	2204884993 – 2204893182	131.107.224.1 – 131.107.255.254

All-Zeros and All-Ones Subnets

In the previous discussion's examples, we used the subnet where all the host bits were set to 0 (the all-zeros subnet) and the subnet where all the host bits were set to 1 (the all-ones subnet). The use of these subnets is somewhat controversial.

Originally, RFC 950 forbade the use of these subnets as valid subnets because:

- The all-zeros subnet caused problems for early routing protocols that did not use a subnet mask to distinguish a network ID. Therefore, 131.107.0.0/16 was the same network to the router as 131.107.0.0/19.

- The subnet broadcast address for the all-ones subnet uses the same address as a special broadcast address, called the *all-subnets-directed broadcast address*. An IP datagram for the all-subnets-directed broadcast was designed to be forwarded by routers to all classful network ID subnets. For more information on the all-subnets-directed broadcast address, see the section entitled "IP Broadcast Addresses," later in this chapter.

The restriction on the use of the all-zeros and all-ones subnets is part of the legacy of classful networks. The result of this restriction is that substantial portions of a fixed address space are unusable and wasted. For example, when performing a 3-bit subnetting of 131.107.0.0 and excluding the all-zeros and all-ones subnets, only six subnets are available. The range of IP addresses 131.107.0.1 through 131.107.31.254 for the all-zeros subnet and 131.107.224.1 through 131.107.255.254 for the all-ones subnet are unusable.

RFC 1812 now allows the use of all-zeros and all-ones subnets for classless environments for the following reasons:

- Classless environments use routing protocols that advertise the subnet mask with the network ID. Therefore, 131.107.0.0/16 is distinguishable from 131.107.0.0/19.
- The all-subnets-directed broadcast has no meaning in a classless environment.

Even though RFC 1812 now allows the use of these special subnets, there is no guarantee that all of your routers and hosts support them. It is a common default configuration for routers not to support one or the other special subnet and they must be instructed to do so. Verify that your routers and hosts support the all-zeros and all-ones subnets before using them. Hosts and routers running a member of the Windows Server 2003 family support the use of the all-zeros and all-ones subnets without additional configuration.

Variable-Length Subnetting

The preceding discussion illustrates how a fixed network ID can be subdivided into equally sized subnets. The 3-bit subnetting of the classful network ID 131.107.0.0/16 produced eight equally sized subnets, each containing 8190 possible IP addresses. However, in the real world, network segments are not of equal sizes. Some network segments require more IP addresses than others. For example, a network segment containing hosts requires more IP addresses than a backbone network segment containing just a few routers. Point-to-point WAN connections require only two IP addresses.

If equally sized subnetting were done, it would have to be done based on the network segment that required the largest amount of hosts. All other network segments would have the same amount of IP addresses, some of which are unassigned or unusable.

To maximize the use of the fixed address space, subnetting is applied recursively to produce subnets of different sizes all derived from the same original network ID. This is

known as variable-length subnetting. Differently sized subnets use different subnet masks, or variable-length subnet masks (VLSM).

Because all of the subnets are derived from the same network ID, if the subnets are contiguous, the routes for all the subnets can be summarized by advertising the original network ID. Contiguous subnets are subnets of the same network ID that are connected to each other.

When performing variable-length subnetting, care must be taken so that each subnet is unique, and with its subnet mask, can be distinguished from all other subnets of the original network ID. Variable-length subnetting requires a careful analysis of your network segments to determine how many of each sized network you require. Then, starting from your network ID, subnetting is performed as many times as needed to express as many subnets as desired with the proper sizes.

With variable-length subnetting, the subnetting technique is applied recursively: You subnet a previously subnetted network ID. When subnetting a previously subnetted network ID, the subnetted network ID bits are fixed and an appropriate number of remaining host bits is chosen for subnetting.

Example of Variable-Length Subnetting

To expand on our earlier example, let us continue subnetting the classful network ID of 131.107.0.0/16. After the 3-bit subnetting has been performed, the remaining addresses must be divided such that:

- Half of the addresses are reserved for future use
- Three subnets are allocated with up to 8190 IP addresses
- 31 subnets are allocated with up to 254 IP addresses
- 64 subnets are allocated with only 2 IP addresses

Recall that the 3-bit subnetting of 131.107.0.0/16 produced the eight subnets listed in Table 6-12.

Table 6-12. The Eight Subnets for the 3-Bit Subnetting of 131.107.0.0/16

Subnet	Subnetted Network ID
1	131.107.0.0/19
2	131.107.32.0/19
3	131.107.64.0/19
4	131.107.96.0/19
5	131.107.128.0/19
6	131.107.160.0/19
7	131.107.192.0/19
8	131.107.224.0/19

Reserve Half of the IP Addresses for Future Use

To reserve half of the addresses for future use, set aside the first four subnets (131.107.0.0/19, 131.107.32.0/19, 131.107.64.0/19, 131.107.96.0/19).

Obtain Three Subnets with up to 8190 IP Addresses

To obtain three subnets with up to 8190 IP addresses per subnet, choose the next three subnets (131.107.128.0/19, 131.107.160.0/19, 131.107.192.0/19). Each subnet has 13 host bits, for a total of 8190 IP addresses per subnet.

Obtain 31 Subnets with up to 254 IP Addresses

To obtain 31 subnets, each with up to 254 IP addresses, perform a 5-bit subnetting of 131.107.224.0/19. The result is 32 subnets (131.107.224.0/24, 131.107.225.0/24, 131.107.226.0/24 . . . 131.107.253.0/24, 131.107.254.0/24, 131.107.255.0/24). To fulfill the requirement, choose the first 31 subnets (131.107.224.0/24 to 131.107.254.0/24).

Obtain 64 Subnets with only 2 IP Addresses

To obtain 64 subnets with only 2 usable IP addresses, perform a 6-bit subnetting of 131.107.255.0/24. The result is 64 subnets (131.107.255.4/30, 131.107.255.8/30, 131.107.255.12/30 . . . 131.107.255.244/30, 131.107.255.248/30, 131.107.255.252/30).

Figure 6-10 shows the variable-length subnetting of 131.107.0.0/16.

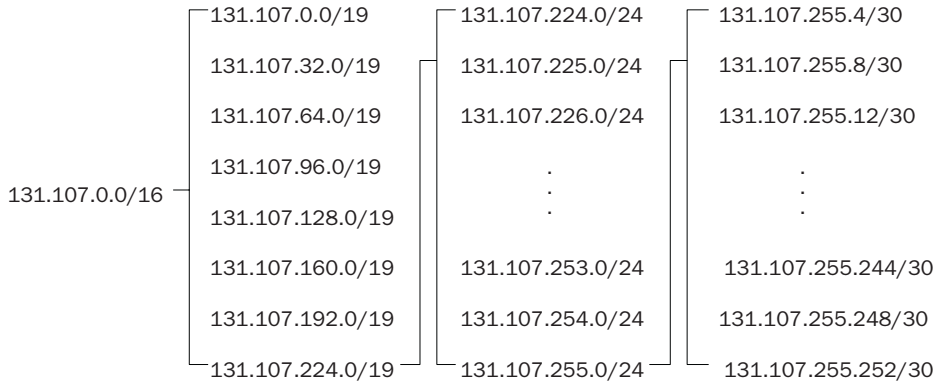


Figure 6-10. *The variable-length subnetting of 131.107.0.0/16 into subnets of different sizes.*

Variable-Length Subnetting and Routing

Variable-length subnetting requires routing protocols to advertise the subnet mask with the network ID. Routing Information Protocol (RIP) version 2, Open Shortest Path First (OSPF), and Border Gateway Protocol version 4 (BGP-v4) support variable-length subnetting environments, but RIP version 1 does not.

Supernetting and CIDR

As the Internet grew suddenly from a collection of educational institutions and government agencies to a business-oriented, pervasive, global internetwork, great stress was placed on the IP address space. Assigning classful network IDs to organizations meant a quick, wasteful depletion of the Internet address space.

For example, numerous organizations worldwide require more than 254 IP addresses. Therefore, a single class C network ID is insufficient. A single class B network ID, however, provides sufficient IP addresses and enough host bits to implement subnetting within the organization's internal network. Although this is good for the organization, it is bad for the Internet IP address space. Consider the smaller organization that needs only 4000 IP addresses. Assigning a class B network with 65,534 possible IP addresses means that 61,534 IP addresses are unassigned and wasted.

Now, instead of an entire class B network ID, the InterNIC assigns a range of class C network IDs. For example, InterNIC assigns 16 class C network IDs to an organization needing 4000 IP addresses. Each class C network ID allows for 254 IP addresses. Therefore, 16 class C network IDs allow for 4064 IP addresses.

This technique minimizes the wasting of Internet IP addresses, but it introduces a new problem. If a single class B network ID is assigned, that single class B network ID becomes a single route in the routing tables of the Internet backbone routers. If 16 class C network IDs are assigned, 16 class C network IDs become 16 routes in the routing tables of the Internet backbone routers.

Extending this example to its ultimate limits, there are more than 2 million class C network IDs. After assigning them all, it is possible to have more than 2 million routes in the routing tables of the Internet backbone routers. Even with today's technology, it is difficult to build an IP router that can have a routing table with millions of entries, and forward IP datagrams at megabit or gigabit per second speeds.

To prevent this scaling problem from overwhelming Internet routers, a route aggregation technique called CIDR is used to express a range of class C network IDs as a single route. This is the method of address allocation that the modern Internet uses. CIDR solves the scaling problem by minimizing the total number of routes that must be stored in the routing tables of Internet routers.

CIDR uses a supernetted subnet mask to express the range of class C network IDs. A supernetted subnet mask is less specific or contains fewer network ID bits than a classful subnet mask. In contrast, a subnetted subnet mask is more specific, or contains more network ID bits, than a classful subnet mask.

Views on CIDR Allocation

The CIDR method of address allocation can be viewed in two different ways:

1. A range of class C network IDs
2. An address space in which multiple classful networks are combined into a single classless network

The latter perspective is more appropriate for today's Internet and for looking forward to IPv6.

A Range of Class C Network IDs

Viewed as a range of class C network IDs, our requirement is based on the number of class C network segments needed in our organization. The following requirements are for a range of class C network IDs to be expressible as a single route using a network ID and a subnet mask:

- The class C network IDs must be sequential.
- The number of allocated class C network IDs must be expressed as a power of 2.

For example, Table 6-13 lists the range (or block) of eight class C network IDs, starting with network ID 223.1.184.0.

Table 6-13. A Block of Eight Class C Network IDs Starting with 223.1.184.0

Starting Network ID	223.1.184.0	11011111 00000001 10111000 00000000
Ending Network ID	223.1.191.0	11011111 00000001 10111111 00000000

Notice that the first 21 bits (underlined) of the range of class C network IDs are the same. The last 3 bits of the third octet vary over all possible values from 000 through 111. This range of class C network IDs can be aggregated with the network ID and subnet mask listed in Table 6-14.

Table 6-14. The Aggregated Block of Class C Network IDs

Network ID	223.1.184.0
Subnet Mask (binary)	11111111 11111111 11111000 00000000
Subnet Mask	255.255.248.0
Network Prefix Length	/21

A block of class-based network IDs, as allocated in this example, is known as a *CIDR block*.

Table 6-15 lists the number of class C network IDs and the supernetted subnet mask for a required number of hosts.

Table 6-15. Supernetting and Class C Addresses

Required Hosts	Number of Class C Network IDs	Supernetted Subnet Mask
2–254	1	255.255.255.0 or /24
255–508	2	255.255.254.0 or /23
509–1016	4	255.255.252.0 or /22
1017–2032	8	255.255.248.0 or /21
2033–4064	16	255.255.240.0 or /20
4065–8128	32	255.255.224.0 or /19
8129–16,256	64	255.255.192.0 or /18
16,257–32,512	128	255.255.128.0 or /17
32,513–65,024	256	255.255.0.0 or /16

An Address Space

From the perspective of an address space, CIDR blocks are no longer viewed as a range of class C network IDs. Even though the CIDR block is obtained from the class-defined range of class C network IDs, it does not necessarily represent a range of class C network IDs. Viewing the CIDR block as a range of class C network IDs implies that we will assign each class C network ID within the block to each of our networks.

In reality, we typically want to assign network IDs of various sizes to the networks of our intranet in a variable-length subnetting scheme. Now our requirement is based on the number of IP addresses required, rather than the number of class C networks in our organization.

For example, to assign 4000 IP addresses to an organization, determine the number of bits required to express 4000 IP addresses. Using powers of 2, 12 bits are needed to express 4094 IP addresses. Therefore, 12 bits are used for the host ID portion, and 20 bits for the network ID portion. The subnet mask indicates 20 bits of network ID. For example, starting from an unassigned portion of the IP address space, the InterNIC allocates the 223.1.176.0 network with the subnet mask of 255.255.240.0 (or 223.1.176.0/20) address space to the organization.

The allocated address space allows the assignment of the range of IP addresses from 223.1.176.1 through 223.1.191.254. However, it is unlikely that the organization will use all 4094 IP addresses on the same network segment. Rather, the organization can use variable-length subnetting and the 12 host bits to create a series of subnets containing the suitable number of appropriately sized subnets.

With CIDR, IP network IDs lose their classful heritage and become address spaces where certain bits are fixed (the network ID bits), and certain bits are variable (the host ID bits). Using variable-length subnetting techniques, the organization's needs should determine how to best utilize the host bits.

CIDR and Routing

CIDR, like variable-length subnetting, requires routing protocols to advertise the subnet mask with the network ID. RIP version 2, OSPF, and BGP-v4 support CIDR environments, but RIP version 1 does not.

Public and Private Addresses

When deploying an IP addressing scheme in your organization, the main consideration is whether your intranet is connected to the Internet:

- If your organization is not connected to the Internet, it is technically possible to choose any IP network IDs—classful or classless—without concern for using overlapping addresses being used on the Internet. However, it is highly recommended that you choose a private address range.
- If your organization is connected to the Internet, it can be connected in one of two ways. If your organization uses a direct-routed connection using a router or firewall, you must use InterNIC-compliant addresses as allocated by the InterNIC or an Internet service provider (ISP). If your organization uses an indirect connection using a proxy server or a Network Address Translator (NAT), you must use addresses that do not overlap with addresses that do, or might, exist on the Internet.

Organizations connected to the Internet must choose between the use of public or private addresses.

Public Addresses

The InterNIC assigns public addresses that are within the public address space consisting of all of the possible unicast addresses on the Internet worldwide. Historically, the InterNIC assigned classful network IDs to organizations connecting to the Internet without regard to geographical location. Today, the InterNIC assigns CIDR blocks to ISPs based on geographical location; the ISPs then subdivide their assigned CIDR blocks to customers. Subdivision of the remaining class C address space based on geographical location was done to provide hierarchical routing and to minimize the number of routes in Internet backbone routers. Public addresses are guaranteed to be globally unique.

When an organization or an ISP is assigned a block of addresses in the public address space, a route exists in the Internet routers' routing tables so that the assigned public addresses are reachable through the ISP. Historically, a classful network ID was added to all of the Internet routers. Today, a route consisting of the range of assigned addresses is added to the routing tables of regional and ISP Internet routers.

One or more (network ID, mask) pairs summarize the range of public IP addresses assigned to an organization. These pairs become the routes in the ISP and Internet routers so that the IP addresses of the organization can be reached.

Illegal or Overlapping Addresses

Organizations that are not connected to the Internet either directly or indirectly are free to choose any addressing scheme without regard to whether the addresses have been assigned to another ISP or organization. However, if that organization later decides to connect to the Internet, a new addressing scheme might be required.

The addresses assigned when the organization was not connected to the Internet might include public addresses that have been assigned to other organizations or ISPs by the InterNIC. If that is the case, these addresses are duplicates that conflict with assigned addresses. This is known as illegal, or overlapping, addressing. Internet traffic from hosts using illegal addresses is forwarded to the routers of the organization that was originally assigned those addresses. Therefore, organizations using illegal addressing are unreachable on the Internet.

For example, an organization that is not connected to the Internet decides to use the address space 207.46.130.0/24 for its intranet. As long as the organization does not connect to the Internet, the use of 207.46.130.0/24 is not an issue. If the organization then connects to the Internet using a direct routed connection, the use of 207.46.130.0/24 is illegal and no responses from hosts on the 207.46.130.0/24 network segment are received.

In this configuration, when a host sends traffic to an Internet location, it sends the traffic with the source IP address within the address space of 207.46.130.0/24. When the Internet host sends a response, it sends the response to the destination IP address within the address space of 207.46.130.0/24. InterNIC assigned Microsoft Corporation the address space 207.46.130.0/24, and a route exists in Internet routers to forward traffic with the destination IP address in this range to Microsoft Corporation's routers. Therefore, the responses to traffic sent by the hosts on the illegal address space 207.46.130.0/24 are forwarded to Microsoft Corporation's routers, and not to the routers of the organization using the illegal addresses.

Note It is common practice among ISPs to discard IP packets sent from a customer site when the source IP address field is not set to a valid public address assigned to the customer. This is known as *ingress filtering,* which attempts to prevent the sending of traffic from hosts using illegal addresses and address spoofing (the sending of IP traffic from a source IP address that is not assigned to a host).

Private Addresses

As the Internet experienced exponential growth, the demand for public IP addresses increased commensurately. Because each node on an organization's intranet required a globally unique public IP address, organizations requested enough IP addresses from the InterNIC to assign unique IP addresses to all of the nodes within their organizations.

However, when an analysis of IP addressing within organizations was done, the Internet authorities noticed that most organizations actually needed very few public addresses. The only hosts that required public IP addresses were those that communicated directly with systems on the Internet, such as Web servers, File Transfer Protocol (FTP) servers, e-mail servers, proxy servers, and firewalls. Most of the hosts within an organization's intranet obtained access to Internet resources through Application Layer gateways such as proxy servers and e-mail servers.

For hosts within the organization's intranet that do not require direct access to the Internet, a legal IP address space must be used. For this purpose, Internet authorities created the private address space, a subset of the Internet IP address space that can be used without conflict within an organization, for hosts that do not require a direct connection to the Internet.

The private and public address spaces are separate and do not overlap. The InterNIC never assigns private addresses—IP addresses within the private address space—to an organization or ISP. This also means that private IP addresses are not reachable on the Internet.

Because private addresses are not reachable on the Internet, hosts on an intranet with private addressing cannot be directly connected to the Internet. Rather, they must be indirectly connected to the Internet using a NAT or an Application Layer gateway such as a proxy server.

A NAT is a router that translates between private addresses and public addresses for Internet traffic. The proxy server receives a request from a host on the intranet for Internet resources. The proxy server then sends the request to the Internet resource and the response traffic is forwarded back to the requesting host. When the proxy server sends the request to the Internet resource, it uses public addressing. Both proxy servers and NATs have private addresses on their intranet interface and public addresses on their Internet interface.

More Info For more information on network address translation, see RFC 3022, which can be found in the \Rfc folder on the companion CD-ROM.

The following three address blocks define the private address space:

- **10.0.0.0/8** The 10.0.0.0/8 private network is an address space with 24 host bits that can be used for any subnetting scheme within the private organization.
- **172.16.0.0/12** The 172.16.0.0/12 private network is an address space with 20 host bits that can be used for any subnetting scheme within the private organization. From a classful perspective, the 172.16.0.0/12 private network ID is the range of 16 class B network IDs from 172.16.0.0/16 through 172.31.0.0/16.
- **192.168.0.0/16** The 192.168.0.0/16 private network is an address space with 16 host bits that can be used for any subnetting scheme within the private organization. From a classful perspective, the 192.168.0.0/16 private network ID is the range of 256 class C network IDs from 192.168.0.0/24 through 192.168.255.0/24.

More Info For more information on the public address space, see RFC 1918, which can be found in the \Rfc folder on the companion CD-ROM.

Automatic Private IP Addressing

When you configure a computer running a member of the Windows Server 2003 family to obtain its IP address automatically and a DHCP server does not respond to the DHCPREQUEST and DHCPDISCOVER messages, TCP/IP for the Windows Server 2003 family configures itself using the Automatic Private IP Addressing (APIPA) feature (provided TCP/IP is not configured to use an alternate static address configuration). Using APIPA, TCP/IP for the Windows Server 2003 family randomly picks an IP address in the address space of 169.254.0.0/16. This address space has been reserved by the Internet Assigned Numbers Authority (IANA) and is not reachable on the Internet.

After choosing an IP address, TCP/IP for the Windows Server 2003 family sends a gratuitous Address Resolution Protocol (ARP) to check for IP address uniqueness. After receiving no response to the gratuitous ARP, TCP/IP for the Windows Server 2003 family is configured for the randomly chosen IP address and the subnet mask of 255.255.0.0. If a response to the gratuitous ARP is received, TCP/IP for the Windows Server 2003 family randomly chooses a new address in the 169.254.0.0/16 address space. After APIPA configuration, TCP/IP for the Windows Server 2003 family continues to send DHCPDISCOVER messages every five minutes. If a DHCP server responds, TCP/IP for the Windows Server 2003 family abandons the APIPA configuration and the DHCP-allocated address takes effect. For more information on gratuitous ARP, see Chapter 3, "Address Resolution Protocol (ARP)."

APIPA was designed to simplify the configuration of a single subnet small office/home office (SOHO) network that is not connected to the Internet or any other IP internetwork. With APIPA, all the computers on a single-subnet SOHO network configure themselves and are able to communicate without manually configuring TCP/IP or setting up a DHCP server.

APIPA does not provide automatic configuration of a default gateway, the IP address of a Domain Name System (DNS) server, a DNS domain name, the IP address of a Windows Internet Name Service (WINS) server, or NetBIOS node type. A single-subnet SOHO network does not need a default gateway, and broadcast NetBIOS name queries resolve names for communication between computers.

TCP/IP for the Windows Server 2003 family APIPA behavior is controlled by the following registry settings:

IPAutoconfigurationEnabled

Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters and HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces*InterfaceGUID*
Value type: REG_DWORD
Valid range: 0 - 1
Default: 1
Present by default: No

IPAutoconfigurationEnabled either enables (when set to 1) or disables (when set to 0) APIPA-based IP address configuration either globally or per interface. The default is enabled both globally and per interface, and the setting for an interface overrides the global setting.

IPAutoconfigurationSubnet

```
Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters and
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\InterfaceGUID
Value type: REG_SZ (String)
Valid range: A valid IP network ID expressed in dotted decimal notation.
Default: 169.254.0.0
Present by default: No
```

IPAutoconfigurationSubnet specifies the IP network ID for the network prefix of APIPA-configured addresses. The default value is 169.254.0.0. IPAutoconfigurationSubnet can be specified globally or per interface, and the setting for an interface overrides the global setting.

IPAutoconfigurationMask

```
Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters and
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\InterfaceGUID
Value type: REG_SZ (String)
Valid range: A valid subnet mask expressed in dotted decimal notation.
Default: 255.255.0.0
Present by default: No
```

IPAutoconfigurationMask specifies the subnet mask for the network prefix of APIPA-configured addresses. The default value is 255.255.0.0. IPAutoconfigurationMask can be specified globally or per interface and the setting for an interface overrides the global setting.

Note The network ID specified for the IPAutoconfigurationSubnet cannot be more specific than the subnet mask specified for the IPAutoconfigurationMask. In other words, the network ID cannot contain bits set to 1 when the corresponding bit in the mask is set to 0. An example of an incorrect network ID and subnet mask combination is the network ID 169.254.47.0 with the subnet mask of 255.255.0.0. The correct subnet mask for this network ID is 255.255.255.0.

IP Broadcast Addresses

IP broadcast addresses are used for single-packet one-to-everyone delivery. A sending host addresses the IP packet using a broadcast address and every node on the sending node's network segment receives and processes the packet. IP broadcast addresses can be used only as the destination IP address.

There are four different types of IP broadcast addresses. For each type, the broadcast IP packet is addressed at the Network Interface Layer using the network technology's broadcast address. For example, for Ethernet and Token Ring networks, all IP broadcasts are sent using the Ethernet and Token Ring broadcast address 0xFF-FF-FF-FF-FF-FF.

Network Broadcast

The IP network broadcast address is the address formed by setting all the host bits to 1 for a classful address. An example of a network broadcast address for the classful network ID 131.107.0.0/16 is 131.107.255.255. Network broadcasts are used to send packets to all hosts of a classful network, which listen for and process packets addressed to the network broadcast address. IP routers do not forward network broadcast packets.

Subnet Broadcast

The IP subnet broadcast address is the address formed by setting all the host bits to 1 for a nonclassful address. An example of a network broadcast address for the nonclassful network ID 131.107.26.0/24 is 131.107.26.255. Subnet broadcasts are used to send packets to all hosts of a subnetted, supernetted, or otherwise nonclassful network. All hosts of a nonclassful network listen for and process packets addressed to the subnet broadcast address. IP routers do not forward subnet broadcast packets.

For a classful network, there is no subnet broadcast address, only a network broadcast address. For a nonclassful network, there is no network broadcast address, only a subnet broadcast address.

All-Subnets-Directed Broadcast

The IP all-subnets-directed broadcast address is the address formed by setting all the original classful network ID host bits to 1 for a nonclassful network. A packet addressed to the all-subnets-directed broadcast is intended to reach all hosts on all of the subnets of a subnetted class-based network ID. An example of an all-subnets-directed broadcast address for the subnetted network ID 131.107.26.0/24 is 131.107.255.255. The all-subnets-directed broadcast is the network broadcast address of the original classful network ID.

All hosts of a nonclassful network listen for and process packets addressed to the all-subnets-directed broadcast address. RFC 922 required IP routers to forward all-subnets-directed broadcast packets to all subnets of the original classful network ID implied in the address. However, this forwarding was not widely implemented.

With the advent of classless network IDs, the all-subnets-directed broadcast address is no longer relevant. According to RFC 1812, the use of the all-subnets-directed broadcast has been deprecated.

Notice how the all-subnets-directed address is the same as the subnet broadcast for the all-ones subnet. For example, the 8-bit subnetting of the class B network ID 157.54.0.0 produces the subnets {157.54.0.0/24, 157.54.1.0/24 . . . 157.54.254.0/24, 157.54.255.0/24}.

For the last subnet, 157.54.255.0/24, the subnet broadcast is 157.54.255.255, which is the same as the all-subnets-directed broadcast address of 157.54.255.255. This address conflict is not an issue for routers that do not forward all-subnets-directed broadcast traffic.

Limited Broadcast

The limited broadcast address is the address formed by setting all 32 bits of the IP address to 1 (255.255.255.255). The limited broadcast address is used when an IP node must perform a one-to-everyone delivery on the local network but the network ID is unknown.

The limited broadcast address is typically used only by nodes during an automated configuration process such as Boot Protocol (BOOTP) or DHCP. For example, with DHCP, a DHCP client must use the limited broadcast address for all traffic sent until the DHCP server acknowledges the IP address lease.

All hosts, classful or nonclassful, listen for and process packets addressed to the limited broadcast address. Although it appears that the limited broadcast address is addressed to all nodes on all networks, it appears only on the local network and is never forwarded by routers. The limited broadcast packet is limited to the local network segment.

The following registry setting controls the address of the limited broadcast address:

UseZeroBroadcast

```
Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
\InterfaceGUID
Value type: REG_DWORD
Valid range: 0 - 1
Default: 0
Present by default: Yes
```

UseZeroBroadcast determines whether the limited broadcast is 0.0.0.0 (when set to 1) or 255.255.255.255 (when set to 0). By default, UseZeroBroadcast is set to 0. Some implementations of TCP/IP, such as those derived from UNIX, use 0.0.0.0 as their limited broadcast address. On the same subnet, all nodes should be using the same limited broadcast address.

IP Multicast Addresses

IP multicast addresses are used for single-packet one-to-many delivery. A sending host addresses the IP packet using an IP multicast address; every node on the sending node's internetwork that is listening for the multicast traffic receives and processes the packet. Unlike broadcast packets, routers forward IP multicast packets and only the hosts listening for the IP multicast traffic are disturbed. IP multicast addresses can be used only as the destination IP address.

As RFC 1112 describes, the set of hosts listening for the traffic of a specific IP multicast address is called a *host group*. Host group members can be located anywhere on the IP internetwork. They also can join and leave the host group at any time. For routers to forward IP multicast traffic to host group members, the routers must be aware of where the members of a multicast group are located. For more information on how hosts and routers facilitate the forwarding of IP multicast traffic, see Chapter 9, "Internet Group Management Protocol (IGMP)."

Multicast IP addresses are in the class D range. Multicast IP addresses range from 224.0.0.0 through 239.255.255.255 (224.0.0.0/4). Multicast IP addresses in the range 224.0.0.0 through 224.0.0.255 (224.0.0.0/24) are reserved for local subnet traffic. Table 6-16 lists some of the reserved IP addresses in this range used by the Windows Server 2003 family. For a complete list, see *http://www.iana.org/assignments/multicast-addresses*.

Table 6-16. Reserved Local Subnet IP Multicast Addresses

Multicast IP Address	Purpose
224.0.0.1	The all-hosts multicast address, designed to reach all hosts on a subnet
224.0.0.2	The all-routers multicast address, designed to reach all routers on a subnet
224.0.0.5	The AllSPFRouters address, designed to reach all OSPF routers on a subnet
224.0.0.6	The AllDRRouters address, designed to reach all OSPF designated routers on a subnet
224.0.0.9	The RIP version 2 multicast address, designed to reach all RIP version 2 routers on a subnet

Mapping IP Multicast Addresses to MAC Addresses

To fulfill the promise of IP multicast traffic—where a single IP datagram is processed only by the host group members—IP multicast traffic must be mapped to a corresponding MAC-level multicast address. The corresponding MAC-level multicast becomes an interesting address to the network interface card (NIC), and all traffic addressed to that interesting address with a valid frame check sequence is passed up through a hardware interrupt to the operating system.

Ethernet and Fiber Distributed Data Interface

To denote a MAC-level multicast address, Ethernet and Fiber Distributed Data Interface (FDDI) network adapters set the Individual/Group (I/G) bit, the low-order bit of the first byte of the destination MAC address, to 1. For IP multicast addressing, the range of multicast MAC addresses is 0x01-00-5E-00-00-00 to 0x01-00-5E-7F-FF-FF. The high-order 25 bits are set to 0000001 00000000 01011110 0. The low-order 23 bits are available for use by IP multicast addresses.

To map an IP multicast address to an Ethernet or FDDI MAC-level multicast address, the low-order 23 bits of the IP multicast address are copied to the low-order 23 bits in the Ethernet multicast address as Figure 6-11 shows.

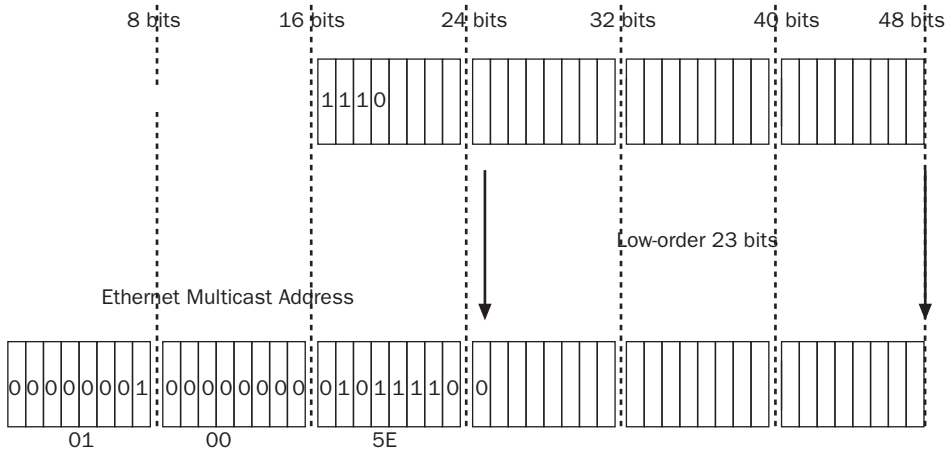


Figure 6-11. *The mapping of IP multicast addresses to Ethernet and FDDI MAC addresses.*

In the high-order 9 bits of the IP multicast address, the first 4 bits are set to 1110; the next 5 bits are variable. These 5 bits do not map to the corresponding Ethernet and FDDI multicast address. Therefore, up to 32 different IP multicast addresses can map to the same Ethernet and FDDI MAC-level multicast address. IP multicast packets received that do not correspond to a multicast address registered by an application or another protocol are silently discarded.

A node registers interest in a specific multicast group by informing the NIC to listen for another interesting destination address for incoming frames. In the Windows Server 2003 family, this is done through the *NDISRequest( )* function. For example, by default TCP/IP for the Windows Server 2003 family listens for all multicast traffic sent to the all-hosts multicast address 224.0.0.1. Therefore, TCP/IP informs the NIC through NDIS to pass up frames with the destination MAC address of 0x01-00-5E-00-00-01.

Token Ring

As RFC 1469 describes, Token Ring can support the same type of multicast IP-address-to-MAC mapping as Ethernet and FDDI. However, because of the hardware limitations of most Token Ring network adapters, typically all IP multicast addresses are mapped to the single Token Ring functional address of 0xC0-00-00-04-00-00.

TCP/IP for the Windows Server 2003 family multicast behavior for Token Ring is controlled by the following registry setting:

TrFunctionalMcastAddress

```
Key:  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
Value type: REG_DWORD
Valid range: 0 - 1
Default: 1
Present by default: No
```

TrFunctionalMcastAddress either enables (when set to 1) or disables (when set to 0) the use of the Token Ring functional address of 0xC0-00-00-04-00-00 for all IP multicast traffic sent on Token Ring adapters. If disabled, IP multicast traffic is sent using the MAC-level broadcast address 0xFF-FF-FF-FF-FF-FF. This setting is enabled by default.

Summary

IP addresses can be unicast, broadcast, or multicast. For unicast addresses, subnetting techniques allow a network ID to be allocated, in an efficient manner, to the subnets of an IP internetwork. Internet authorities have defined public addresses that are reachable on the Internet and private addresses that are designed for use on private intranets not directly connected to the Internet. IP broadcast addresses are used to send IP datagrams to all the nodes on a physical or logical subnet. IP multicast addresses are used to send IP datagrams to all members of a multicast host group.

Chapter 17
Domain Name System (DNS)

Every host that runs TCP/IP must have a unique IP address that is used when communicating with other computers in a network. Computers operate easily with IP addresses, but people do not; users would rather identify systems by a name. To facilitate effective and efficient communication, users need to be able to refer to computers by name and still have their computer use IP addresses transparently.

In the early days of ARPANET, the forerunner to today's Internet, there were only a small number of computers attached to the network. The Network Information Center (NIC), located at the Stanford Research Institute (SRI), was responsible for compiling a single file named HOSTS.TXT that contained the names and addresses of every computer. Administrators would e-mail updates to SRI, which would then update the HOSTS.TXT file. ARPANET users would then download the new version of HOSTS.TXT using File Transfer Protocol (FTP) and convert it for local use (for example, copy HOSTS.TXT /Etc/Hosts on UNIX systems).

As ARPANET grew, it became obvious that this approach would not scale for the following three key reasons:

- The bandwidth consumed in transmitting updated versions of an ARPANET-wide host file was proportional to the square of the number of hosts in the ARPANET. With the number of hosts growing at an exponential rate, the long-term impact was likely to be a load that no one host would be able to sustain.
- The static flat host file also meant that no two computers on the entire ARPANET could have the same name. As the number of hosts grew, the risk of adding a duplicate name grew, as did the difficulty of trying to control this centrally.
- The nature of the underlying network was changing—the large, time-sharing computers that had once made up the ARPANET were being superseded by networks of workstations, each of which needed to have a unique host name. This would be difficult, if not impossible, to control centrally.

As the ARPANET continued to grow, it became clear that a better solution was required. Several proposals were generated based on the concept of a distributed naming service, based on a hierarchical namespace. RFCs 882 and 883 emerged, which describe the design for a domain name system based on a distributed database containing generalized resource information. This design is described in RFCs 1034 and 1035. The service that is used in today's Internet has evolved beyond these early RFCs, and updates and refinements continue.

Overview of DNS

To facilitate communications between computers, computers can be given names within a namespace. The specific namespace defines the rules for naming a computer and for how names are resolved into IP addresses. When one computer communicates with other computers, it must resolve, or convert, a computer name into an IP address based on the rules of the namespace being used. This resolution is done by a name-resolution service.

There are two main namespaces, and name-resolution methods, used within Microsoft Windows Server 2003: NetBIOS, implemented by Windows Internet Naming Service (WINS; described in Chapter 18), and the Domain Name System (DNS), described in this chapter. Windows Server 2003 also provides support for other namespaces, such as Novell NetWare and Banyan Vines, although discussion of these is outside the scope of this book.

This section describes DNS and the DNS protocol that is used to provide name resolution.

What Is DNS?

DNS is an Internet Engineering Task Force (IETF) standard name service. The DNS service enables client computers on your network to register and resolve DNS domain names. These names are used to find and access resources offered by other computers on your network or other networks, such as the Internet. The following are the three main components of DNS:

- **Domain namespace and associated resource records (RRs)** A distributed database of name-related information.
- **DNS name servers** Servers that store the domain namespace and RRs and answer queries from DNS clients.
- **DNS resolvers** The facility within a DNS client that contacts DNS name servers and issues name queries to obtain resource record information.

Key DNS Terms

This section describes the key components of DNS and defines important DNS terms.

Domain Namespace

The *domain namespace* is a hierarchical, tree-structured namespace, starting at an unnamed root used for all DNS operations. In the DNS namespace, each node and leaf in the domain namespace tree represents a named domain. Each domain can have additional child domains. Figure 17-1 illustrates the structure of an Internet domain namespace.

Domain Names

Each node in the DNS tree, as illustrated in Figure 17-1, has a separate name, referred to in RFC 1034 as a label. Each DNS label can be from 1 through 63 characters in length, with the root domain having a length of zero characters.

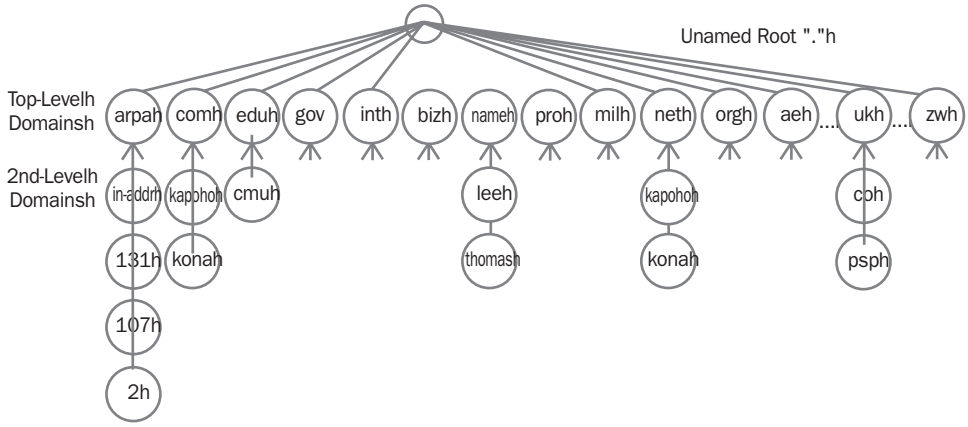


Figure 17-1. *Domain namespace for the Internet.*

A specific node's *domain name* is the list of the labels in the path from the node being named to the DNS namespace root. DNS convention is that the labels that compose a domain name are read left to right—from the most specific to the root (for example, www.kapoho.com). This full name is also known as the *fully qualified domain name* (FQDN).

Domain names can be stored as uppercase or lowercase, but all domain comparisons and domain functions are defined by RFC 1034 to be case-insensitive. Thus, www.kapoho.com is identical to WWW.KAPOHO.COM for domain naming operations.

Top-Level Domains

The domains directly below the root are called *top-level domains* (TLDs). Figure 17-1 illustrates some of the TLDs that have been defined for the Internet. Additional names (at least for the Internet) are difficult to create, although in 2000, the Internet Corporation for Assigned Names and Numbers (ICANN) did create an additional seven TLDs.

There are three categories of TLDs:

- **ARPA** This is a special domain—it is only used today for reverse-name lookups.
- **Generic TLDs** There are 14 TLDs, noted in Table 17-1. At one time the generic TLDs were all three characters long, but in November 2000, ICANN accepted seven new TLDs, some of which were longer (for example, .aero and .info).
- **Two-letter country-based domain names** These country code domains are based on the International Organization for Standardization (ISO) country name, and are used principally by companies and organizations outside the United States. The exception is the United Kingdom, which uses .uk as the TLD, even though its ISO country code is GB.

Table 17-1 shows the TLDs in use today, as defined by RFC 1591.

Table 17-1. Top-Level Domains Used on the Internet

Domain Name	Use
aero	Exclusively reserved for the aviation community
biz	A TLD that is aimed at large and small companies around the world
com	Commercial organizations, such as microsoft.com for the Microsoft Corporation
coop	A TLD for cooperatives
edu	Educational institutions, now mainly four-year colleges and universities, such as cmu.edu for Carnegie Mellon University
gov	Agencies of the U.S. Federal Government, such as fbi.gov for the U.S. Federal Bureau of Investigation
info	An unrestricted domain aimed at providing information for worldwide consumption
int	Organizations established by international treaties, such as nato.int for NATO
mil	U.S. military, such as af.mil for the U.S. Air Force
museum	A domain restricted to museums and related organizations and individuals
name	A global domain for use by individuals, possibly developing into a global digital identity for users
net	Computers of network providers, organizations dedicated to the Internet, Internet service providers (ISPs), and so forth, such as internic.net for the Internet Network Information Center (InterNIC)
org	A TLD for groups that do not fit anywhere else, such as nongovernment or nonprofit organizations (for example, reiki.org for information about Reiki)
pro	A TLD for professionals such as doctors, lawyers, and accountants

Resource Records

A resource record (RR) is a record containing information relating to a domain that the DNS database can hold and that a DNS client can retrieve and use. For example, the host RR for a specific domain holds the IP address of that domain (host); a DNS client uses this RR to obtain the IP address for the domain (host).

Each DNS server contains the RRs relating to those portions of the DNS namespace for which it is authoritative (or for which it can answer queries sent by a host). When a DNS server is authoritative for a portion of the DNS namespace, those systems' administrators are responsible for ensuring that the information about that DNS namespace portion is correct. To increase efficiency, a given DNS server can cache the RRs relating to a domain in any part of the domain tree.

There are numerous RR types defined in RFCs 1034 and 1035, and in later RFCs. Most of the RR types are no longer needed or used, although all are fully supported by Windows Server 2003. Table 17-2 lists the key RRs that might be used in a

Windows Server 2003 network. (For more detail on the contents of specific RRs, see the section entitled "DNS Resource Records," later in this chapter.)

Table 17-2. Key Resource Records as Used by a Windows Server 2003 Network

Resource Record Type	Contents	Use
A	Host Address	Used to hold a specific host's IP address.
CNAME	Canonical Name (alias)	Used to make an alias name for a host.
MX	Mail Exchanger	Provides message routing to a mail server, plus backup server(s) in case the target server is not active.
NS	Name Server	Provides a list of authoritative servers for a domain or indicates authoritative DNS servers for any delegated subdomains.
PTR	Pointer	Used for reverse lookup—resolving an IP address into a domain name using the in-addr.arpa domain.
SOA	Start of Authority	Used to determine the DNS server that is the primary server for a DNS zone and to store other zone property information.
SRV	Service Locator	Provides the ability to find the server providing a specific service. Active Directory service uses SRV records to locate domain controllers, global catalog servers, and Lightweight Directory Access Protocol (LDAP) servers.

RRs can be attached to any node in the DNS tree, although RRs are not provided in some domains. For example, Pointer (PTR) RRs are found only in domains below the in-addr.arpa domain. Thus, higher-level domains, such as microsoft.com, can have individual RRs (for example, Mail Exchanger [MX] records for mail to be sent to the Microsoft Corporation) as well as having subdomains that also might have individual RRs (for instance, eu.microsoft.com, which has a host record www.eu.microsoft.com).

Canonical Names

The Canonical Name (CNAME) RR enables the administrator to create an alias to another domain name. The use of CNAME RRs is recommended in the following scenarios:

- When a host specified in an A RR in the same zone needs to be renamed. For example, if you need to rename kona.kapoho.com to hilo.kapoho.com, you could create a CNAME entry for kona.kapoho.com to point to hilo.kapoho.com.
- When a generic name for a well-known service, such as ftp or www, needs to resolve to a group of individual computers (each with an individual A RR). For example, you might want www.kapoho.com to be an alias for kona.kapoho.com and hilo.kapoho.com. A user accessing www.kapoho.com is generally unaware of which computer is actually servicing the request.

DNS Query Operation

A DNS client issues a *query operation* against a DNS server to obtain some or all of the RR information relating to a specific domain, for instance, to determine which host A record or records are held for the domain named kapoho.com. If the domain exists and the requested RRs exist, the DNS server returns the requested information in a *query reply message*. The query reply message returns both the initial query and a reply containing the relevant records, assuming the DNS server can obtain the required RRs.

A DNS query, referred to in RFC 1034 as a standard query, contains a target domain name, a query type, and a query class. The query contains a request for the specific RR(s) that the resolver wishes to obtain (or a request to return all RRs relating to the domain).

DNS Update Operation

A DNS *update operation* is issued by a DNS client against a DNS server to update, add, or delete some or all of the RR information relating to a specific domain. A DNS client kona.kapoho.com could, for instance, update the host record for kona.kapoho.com to point to 10.10.1.100. The update operation is also referred to as a dynamic update.

DNS Zones

A DNS server that has complete information for part of the DNS namespace is said to be the *authority* for that part of the namespace. This authoritative information is organized into units called *zones*, which are the main units of replication in DNS. A zone contains one or more RRs for one or more related DNS domains.

The following are the four DNS zone types implemented in the DNS Server service for Windows Server 2003:

- **Standard primary** Holds the master copy of a zone and can replicate it to secondary zones. All changes to a zone are made on the standard primary.
- **Standard secondary** Contains a read-only copy of zone information that can provide increased performance and resilience. Information in a primary zone is replicated to the secondary zone by use of the zone transfer mechanism.
- **Active Directory–integrated** A Microsoft proprietary zone type, where the zone information is held in the Windows Active Directory and replicated using Active Directory replication. Active Directory–integrated zones provide the administrator with considerable flexibility in terms of replication.
- **Stub** A stub zone contains only the RRs needed to identify the authoritative DNS servers for the actual zone (SOA, NS, and Glue A records).

With traditional zones, the master copy of a zone is held in a primary zone on a single DNS server. On that server, the zone has a Start of Authority (SOA) RR that specifies it to be the primary zone. To improve performance and redundancy, the RRs contained in a primary zone can be automatically replicated to one or more secondary zones held on other (secondary) DNS servers. When changes are made to the zone, for instance, to add

an A record, the changes are made to the primary zone and are transferred to the secondary zone. The transfer of zone information is handled by the zone replication process, which is described later in the section entitled "Zone Transfer."

When a zone is first created in Windows Server 2003, two RRs are created:

- An SOA record, which provides some basic parameters for the zone, such as its name (for example, kapoho.com), as well as the IP address of the primary server, and so on.
- An NS record, which identifies the server as one that is authoritative for this zone.

After the zone is created, the administrator can then add RRs to the zone, or can set the domain to be dynamically updated so that suitably configured hosts can automatically update this zone. For example, the administrator could add A records (host records) for hosts in the domain, and a computer capable of performing its own DNS dynamic updates—such as a computer running Windows 2000, Windows XP, or Windows Server 2003—can then directly update the A and PTR records on the DNS server. If the DNS client is also a Dynamic Host Configuration Protocol (DHCP) client, the administrator can configure a DHCP server to send the updates.

Once administrators have created the zone, they can add additional subdomains to the zone (for example, jh.kapoho.com). These might be added to delegate DNS services to a new building that is managed separately from the parent domain. This subdomain, which might reside in a separate zone on a separate DNS server, would have RRs added (for example, a host record for rebecca.jh.kapoho.com).

As Figure 17-2 illustrates, if you create subdomains, they can either be part of the same zone or belong to another zone on the same or a different DNS server. For example, the subdomain jh.kapoho.com, which is subordinate to kapoho.com, could be held in the same zone as kapoho.com or in a separate zone. This allows the subdomain to be managed and included as part of the original zone records, or to be delegated away to another zone created to support that subdomain.

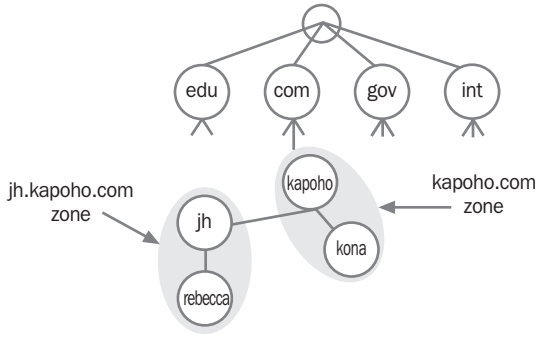


Figure 17-2. *Zones versus domains.*

In this example, the domain kapoho.com has a subdomain of jh.kapoho.com. In addition, both domains contain a single host record. In this example, the domains jh.kapoho.com and kapoho.com are held in separate zones on different DNS servers. The kapoho.com zone holds one host record for kona.kapoho.com and the jh.kapoho.com domain holds the host record for the host rebecca.jh.kapoho.com.

Active Directory–Integrated Zones

An important feature of the DNS Server service for Windows Server 2003 is the ability to store DNS zones within Active Directory. An *Active Directory–integrated zone* is a primary DNS zone that is held within Active Directory and replicated to other domain controllers using Active Directory replication (and not traditional zone transfer). Although this method of storing zones is a Microsoft proprietary approach, it can provide some useful benefits.

The main advantage of Active Directory–integrated zones is that the zones become, in effect, multimaster, with the capability of updates being made to any DNS server. This can increase the fault tolerance of the DNS service. In addition, replication of zone information occurs using Active Directory replication, which can be more efficient across slow links because of the way that Active Directory compresses replication data between sites.

In Microsoft Windows 2000, all Active Directory–integrated zones were held in the domain name context of the Active Directory. This meant that any updates to an Active Directory zone would automatically be replicated to all domain controllers in that domain, whether or not they were DNS servers. For enterprises that had a large number of domain controllers in a domain, but not many DNS servers, this approach generated unnecessary replication. In addition, it meant some extra DNS replication planning for those organizations with multiple zones. This extra planning was needed to ensure, at a minimum, that all domain controllers were able to resolve the forest side locator records kept in the _msdcs subdomain of the organization's forest root domain name (for example, the _msdcs.kapoho.com domain for the organization's kapoho.com forest root domain).

With Windows Server 2003, the Active Directory–integrated zones can be held in one of three places:

- All domain controllers in the domain (the Windows 2000 behavior).
- All domain controllers that are DNS servers in the current domain; this eliminates DNS information being replicated to domain controllers that are not also DNS servers.
- All domain controllers that are DNS servers in the entire forest; this simplifies DNS replication for larger organizations with multiple domains.

Reverse-Lookup Zones

Most queries sent to a DNS server involve a search based on the DNS name of another computer as stored in an address A RR. This type of query expects an IP address as the

resource data for the answered response and it is generally referred to as a *forward query*. DNS also provides a reverse-lookup process, which enables a host to determine another host's name based on its IP address. For example, "What is the DNS domain name of the host at IP address 10.10.1.100?"

To allow for reverse queries, a special domain, in-addr.arpa, was defined and reserved in the Internet DNS namespace. Subdomains within the in-addr.arpa domain are named using the reverse ordering of the numbers in the dotted-decimal notation of IP addresses. The reverse ordering of the domain name is needed because, unlike DNS names, IP addresses are read from left to right, but are interpreted in the opposite manner (that is, the left-most part is more generalized than the right-most part). For this reason, the order of IP address octets is reversed when building the in-addr.arpa domain tree; for example, the reverse-lookup zone for the subnet 192.168.100.0 is 100.168.192.in-addr.arpa.

This approach enables the administration of lower limbs of the DNS in-addr.arpa tree to be delegated to an organization when it obtains a set of IP addresses from an IP registry.

The in-addr.arpa domain tree makes use of the PTR RR, which is used to associate the IP address with the owning domain name. This lookup should correspond to an A RR for the host in a forward-lookup zone.

Note The in-addr.arpa domain is used only for Internet Protocol version 4 (IPv4)–based networks. In the DNS Microsoft Management Console (MMC) snap-in for Windows Server 2003, the DNS server's New Zone Wizard uses this domain when it creates a new reverse-lookup zone. Internet Protocol version 6 (IPv6)–based reverse-lookup zones are based on the domain ip6.arpa.

Reverse DNS zones have the same SOA and NS records as forward lookup zones. You can also configure reverse lookup zones to be traditional primary or secondary zones or to be Active Directory–integrated. Active Directory–integrated reverse lookup zones replicate in the same manner as Active Directory–integrated forward lookup zones.

Reverse Queries

A reverse query is one in which the DNS server is requested to return the DNS domain name for a host at a particular IP address. Reverse-Lookup Query messages are, in effect, standard queries, but relating to the reverse-lookup zone. The reverse-lookup zone is based on the in-addr.arpa domain name and mainly holds PTR RRs.

Note The creation of reverse-lookup zones and the use of PTR RRs for identifying hosts are optional parts of the DNS standard. Reverse-lookup zones are not required to use Windows Server 2003, although some networked applications can be configured to use the reverse-lookup zones as a form of additional security. In addition, the Nslookup.exe tool displays an error if the reverse-lookup zone for a DNS server is not available.

Inverse Queries

Inverse queries, originally described in RFC 1034, look up a host name based on its IP address. Inverse queries use a nonstandard DNS query operation. Inverse queries are now outdated and are generally not used by modern applications and operating systems.

Some early versions of Nslookup.exe, a utility used to test and troubleshoot a DNS service, did attempt to issue inverse queries. A Windows Server 2003 DNS server recognizes and accepts inverse query messages and answers them with a "fake" inverse query response. The version of Nslookup.exe that ships with Windows Server 2003 does not support inverse queries.

DNS Query Classes

DNS queries fall into one of two classes: recursive queries and iterative queries.

A *recursive query* is a DNS query sent to a DNS server in which the querying host asks the DNS server to provide a complete answer to the query, even if that means contacting other servers to provide the answer. When sent a recursive query, the DNS server issues separate iterative queries to other DNS servers on behalf of the querying host to obtain an answer for the query.

An *iterative query* is a DNS query sent to a DNS server in which the querying host requests it to return the best answer it can provide without seeking further assistance from other DNS servers.

In general, host computers issue recursive queries against DNS servers. The host assumes that the DNS server either knows the answer to the query or can find the answer. On the other hand, a DNS server issues iterative queries against other DNS servers if it is unable to answer a recursive query from cached or locally stored information.

DNS Resolver

In Windows Server 2003, the DNS *resolver* is a system component on a DNS client that performs DNS queries against a DNS server (or servers). The administrator configures the Windows Server 2003 TCP/IP protocol with the IP address of at least one DNS server to which the resolver sends one or more queries for DNS information.

In Windows Server 2003, the resolver is part of the DNS Client service. This service is installed automatically when Windows Server 2003 is installed, and runs within one of the many SVCHOST processes. Like most Windows Server 2003 services, the DNS Client service logs on using the System account. You can see the DNS Client service named Dnscache in the SVCHOST process with PID 812, in the following output:

```
C:\Documents and Settings\Administrator>tasklist /svc
Image Name                     PID Services
========================= ====== ============================================
System Idle Process              0 N/A
System                           4 N/A
```

```
smss.exe                      288 N/A
csrss.exe                     356 N/A
winlogon.exe                  380 N/A
services.exe                  424 Eventlog, PlugPlay
lsass.exe                     436 HTTPFilter, kdc, Netlogon, NtLmSsp,
                                  PolicyAgent, ProtectedStorage, SamSs
svchost.exe                   572 RpcSs
svchost.exe                   608 TermService
svchost.exe                   812 Dhcp, Dnscache
svchost.exe                   876 LmHosts
svchost.exe                   888 AppMgmt, BITS, Browser, CryptSvc, dmserver,
                                  ERSvc, EventSystem, helpsvc, lanmanserver,
                                  lanmanworkstation, Netman, Nla, RasMan,
                                  Schedule, seclogon, SENS, ShellHWDetection,
                                  uploadmgr, W32Time, winmgmt, wuauserv, WZCSV
spoolsv.exe                  1132 Spooler
msdtc.exe                    1212 MSDTC
certsrv.exe                  1304 CertSvc
dfssvc.exe                   1332 Dfs
dns.exe                      1364 DNS
inetinfo.exe                 1468 IISADMIN, NntpSvc, SMTPSVC
InoRpc.exe                   1504 InoRPC
InoRT.exe                    1676 InoRT
InoTask.exe                  1696 InoTask
ismserv.exe                  1784 IsmServ
llssrv.exe                   1812 LicenseService
ntfrs.exe                    1864 NtFrs
svchost.exe                   216 RemoteRegistry
locator.exe                   308 RpcLocator
wins.exe                      784 WINS
svchost.exe                  1360 W3SVC
explorer.exe                 3108 N/A
svchost.exe                  3220 TapiSrv
Realmon.exe                  3228 N/A
cmd.exe                      3240 N/A
cmd.exe                      3248 N/A
mmc.exe                      2520 N/A
mmc.exe                      3636 N/A
notepad.exe                  2604 N/A
setiathome-3.03.i386-winn    3196 N/A
setiathome-3.03.i386-winn    2004 N/A
cmd.exe                      3016 N/A
wmiprvse.exe                 2760 N/A
tasklist.exe                 3020 N/A
```

DNS Resolver Cache

An IP host that needs to contact another host on a regular basis needs to resolve a particular DNS name many times (for example, the name of the mail server). To avoid having to send queries to a DNS server each time the host wants to resolve the name, the DNS Client service on Windows Server 2003 hosts implements a local cache of DNS information.

The DNS Client service caches RRs from query responses that the DNS Client service receives. The information is held for a set Time-To-Live (TTL) and, when present, is used to answer subsequent queries and avoid sending the query to a DNS server. By default, the TTL used for the local DNS cache is the TTL value received in the DNS query response. When a query is resolved, the authoritative DNS server for the resolved domain defines the TTL for a given RR.

You can use the IPCONFIG command with the /DISPLAYDNS option to display the current DNS resolver cache contents. The following output shows an example:

```
C:\>ipconfig /displaydns
Windows IP Configuration
    1.0.0.127.in-addr.arpa
    ----------------------------------------
    Record Name . . . . . : 1.0.0.127.in-addr.arpa.
    Record Type . . . . . : 12
    Time To Live  . . . . : 0
    Data Length . . . . . : 4
    Section . . . . . . . : Answer
    PTR Record  . . . . . : localhost

    7c5fa925-791e-4aa3-b8d0-ea70ed74adc4._msdcs.kapoho.com
    ----------------------------------------
    Record Name . . . . . : 7c5fa925-791e-4aa3-b8d0-
ea70ed74adc4._msdcs.kapoho.com
    Record Type . . . . . : 5
    Time To Live  . . . . : 595
    Data Length . . . . . : 4
    Section . . . . . . . : Answer
    CNAME Record  . . . . : kapoho10.kapoho.com

    maui
    ----------------------------------------
    Record Name . . . . . : maui.kapoho.com
    Record Type . . . . . : 1
    Time To Live  . . . . : 1195
    Data Length . . . . . : 4
    Section . . . . . . . : Answer
    A (Host) Record . . . : 10.10.1.62
```

```
localhost
----------------------------------------
Record Name . . . . . : localhost
Record Type . . . . . : 1
Time To Live  . . . . : 0
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 127.0.0.11
```

Negative Caching

The DNS Client service for Windows Server 2003 provides negative caching support. Negative caching occurs when an RR for a queried domain name does not exist or when the domain name itself does not exist, in which case, the lack of resolution is stored. Negative caching prevents the repetition of additional queries for RRs or domains that do not exist.

If a query is made to a DNS server and the response is negative, subsequent queries for the same domain name are answered negatively, through the cache, for a default time of 300 seconds (5 minutes). To avoid the continued negative caching of stale information, any query information negatively cached is held for a shorter period than positive query responses. However, this negative caching time-out value can be changed in the registry using the following NegativeCacheTime registry value:

NegativeCacheTime

```
Location: HKEY_LOCAL_MACHINE
\System\CurrentControlSet\Services\Dnscache\Parameters
Data type: REG_DWORD
Default value: 0x12c (300 seconds, or 5 minutes)
Valid range: 0-0xFFFFFFFF
Present by default: No
```

Negative Cache Time sets the number of seconds for the negative caching time-out. The recommended time is less than 1 day, which prevents very stale records. Negative caching reduces the load on DNS servers, but should the relevant RRs become available, later queries can be issued to obtain the information.

Zone Transfer

To improve the resilience and performance of the DNS resolution, it is normal to have at least one standard secondary zone for each standard primary zone, where the secondary zone is held on another DNS server. Depending on the exact nature of the organization, multiple standard secondary zones might be appropriate. When changes are made to the primary zone, it is important that the zone information is promptly replicated to all secondary zones. The process of transferring zone information from a primary to a secondary zone is called a *zone transfer*.

Zone transfers usually occur automatically at intervals defined in the zone's SOA record. Zone transfers can also be performed manually using the DNS MMC snap-in, which might be done if the administrator suspects the secondary zone has not been properly updated.

When a standard secondary zone is created on a Windows Server 2003 DNS server, the DNS server transfers all RRs from the standard primary zone to the new standard secondary zone. The DNS server does this to obtain a full copy of all RRs for the zone. In many of the early DNS server implementations, this same method of full transfer for a zone is used also when the secondary zone requires updating to align it with the primary zone, after changes are made to the primary zone. For large zones, this can be very time-consuming and wasteful of network resources. This can be an issue because a zone transfer is needed each time the primary zone is updated, such as when a new host is added to the domain or the IP address for a host is changed.

After the RRs have been replicated, the server on which the new standard secondary zone resides checks at regular intervals with the server on which the primary zone resides to determine if there are any changes to the primary zone. This is determined by polling the primary zone on a regular basis, defined by the network administrator in the Zone SOA record in the standard primary zone, and checking if the zone's version number has changed. If the version number has been incremented, a zone transfer is necessary. This process is shown in the following Network Monitor trace (Capture 17-06, included in the \Captures folder on the companion CD-ROM):

```
1   21.108471  Kapoho10   Mahimahi   DNS    0x6000:Std Qry for kapoho.com. of
type SOA on class INET addr. 10.10.1.74  10.10.1.200
2   21.108471  Mahimahi   Kapoho10   DNS    0x6000:Std Qry Resp. for
kapoho.com. of type SOA on class INET addr. 10.10.1.200  10.10.1.74
3   21.108471  Kapoho10   Mahimahi   DNS    0x4000:Std Qry for kapoho.com. of
type Req for incrmntl zn Xfer on class INET addr. 10.10.1.74  10.10.1.200
4   21.108471  Mahimahi   Kapoho10   DNS    0x4000:Std Qry Resp. for
kapoho.com. of type SOA on class INET addr. 10.10.1.200 10.10.1.74
```

In this trace, the DNS server holding the secondary zone (Kapoho10) queries the primary zone to obtain the primary's SOA record, which will tell the secondary zone the version number of the zone at the primary server. The secondary server discovers that the version number on the primary DNS server is higher and requests a zone transfer.

For manually maintained DNS servers, updating (or not) the SOA records version number traditionally has been a key troubleshooting issue—changes are made to a primary zone, but the version number is unchanged, and thus the changes are not replicated to the secondary zone. With Windows Server 2003, changes made to the zone, either by manual update using the DNS snap-in or by dynamic registration, trigger an update to the version number, thus enabling the secondary server to carry out the server transfer at the next poll interval.

Incremental Zone Transfers

For large zones, zone transfers can consume a significant amount of bandwidth, especially when a zone transfer is carried across slow wide area network (WAN) links. To improve the efficiency of zone transfers, the DNS Server service for Windows Server 2003 implements *incremental zone transfer,* as defined in RFC 1995. With incremental zone transfers, only the changes to the zone are transferred, rather than the entire zone. This can significantly reduce the traffic needed to keep a secondary zone current.

With incremental zone transfers, the differences between the source and replicated versions of the zone are first determined. If the zones are identified as the same version—as indicated by the serial number field in the SOA RR of each zone—no transfer is made.

If the serial number for the zone at the source is greater than at the requesting secondary server, a transfer is made of only those changes to RRs for each incremental zone version. For an incremental zone transfer query to succeed and changes to be sent, the zone's source DNS server must keep a history of incremental zone changes to use when answering these queries. Windows Server 2003 stores the incremental zone transfer information for each zone in a text file in the %*Systemroot*%\System32\Dns folder with a name based on the name of the file holding the zone data (which was specified when the zone was defined). Thus, if the zone information for the kapoho.com zone is held in the file Kapoho.com.dns, the incremental update log is held in Kapoho.com.dns.log. A part of this log, which was used by the DNS server on kapoho10 to perform the preceding zone transfer, is shown here:

```
$SOURCE  ADMIN
$VERSION 84
$ADD
rebecca                 A      10.10.1.199

$SOURCE  ADMIN
$VERSION 85
$DELETE
@                       IN  SOA mahimahi.kapoho.net. hostmaster. (
  81            ; serial number
  15            ; refresh
  600           ; retry
  86400         ; expire
  3600        ) ; default TTL
$ADD
                        IN  SOA mahimahi.kapoho.net. hostmaster. (
  85            ; serial number
  15            ; refresh
  600           ; retry
  86400         ; expire
  3600        ) ; default TTL
```

Active Directory–Integrated Zone Replication

Standard zones use traditional zone replication mechanisms to transfer zone information. Active Directory–integrated zones, however, use Active Directory replication to replicate updates. This provides the following three key benefits:

- DNS servers become multimaster. With standard DNS zones, all updates need to be made to a single DNS server—in other words, to the server containing the primary zone. With Active Directory integration, any DNS server can accept the updates, which provide improved performance and scaling, as well as better fault tolerance.
- Active Directory replication is both more efficient and quicker. It transfers updated-only properties, not the entire zone, which means only the changes are transmitted across the network. In addition, replication between sites, typically involving slower links, can be highly compressed.
- The administrator only needs to plan and implement a single replication topology for Active Directory. This also replicates DNS changes but simplifies the planning and reduces bandwidth used.

For organizations using Active Directory, Active Directory–integrated zones are generally recommended. If the organization is, however, using third-party DNS servers, these servers are unlikely to support Active Directory–integrated zones.

Delegation of Domains

DNS is a distributed database of information designed specifically to overcome the limitations of the earlier HOSTS.TXT approach to name resolution. The key to scaling DNS to handle large namespaces or networks, such as the Internet, is the ability to delegate the administration of domains. *Domain delegation* occurs when the responsibility of the RRs of a subdomain is passed from the owner of the parent domain to the owner of the subdomain.

At the heart of the Internet are 13 root servers, named A.ROOT-SERVERS.NET through M.ROOT-SERVERS.NET. These root servers are widely distributed around the world, although most are located in the United States. The root servers hold data for all the generic TLDs, such as .com, .org, .net, and .name, as well as for geographical domains, such as .uk and .jp.

Below the root and TLDs are the domains and subdomains belonging to individual organizations. In some TLDs, additional hierarchy levels are provided. For example, in the .uk domain, there are subdomains co.uk for U.K.-based companies (for instance, psp.co.uk) and ac.uk for academic institutions (for instance, ic.ac.uk for Imperial College), and so forth.

As previously illustrated in Figure 17-2, there are boundaries between zones where responsibility for administration is delegated from zones closer to the TLDs to zones one

degree farther from the TLDs. Within the kapoho.com domain there is a subdomain jh.kapoho.com. Responsibility for the subordinate domain can be delegated to a different set of DNS servers.

To implement a delegation, the parent zone must have both an A RR and a Name Service (NS) RR. Both should point to the new delegated domain's DNS servers. In the kapoho.com zone, illustrated in Figure 17-2, there must be an A and an NS record in the parent domain that point to a server authoritative for jh.kapoho.com. Windows Server 2003 has a Delegation Wizard to simplify the task of implementing a delegation.

Forwarder and Slave DNS Servers

When a resolver contacts a DNS server to resolve a domain name and return relevant RRs, the contacted DNS server first attempts to perform the lookup using zones defined on the server or information contained in its own cache. If this fails, by default the DNS server then starts to issue iterative queries to resolve the domain. This starts at the namespace root. If the DNS server is one of several at a site connected to the outside world by slow links, this default behavior might not be desirable.

As illustrated by Figure 17-3, a *forwarder* is a DNS server that other DNS servers contact before attempting to perform the necessary name resolution.

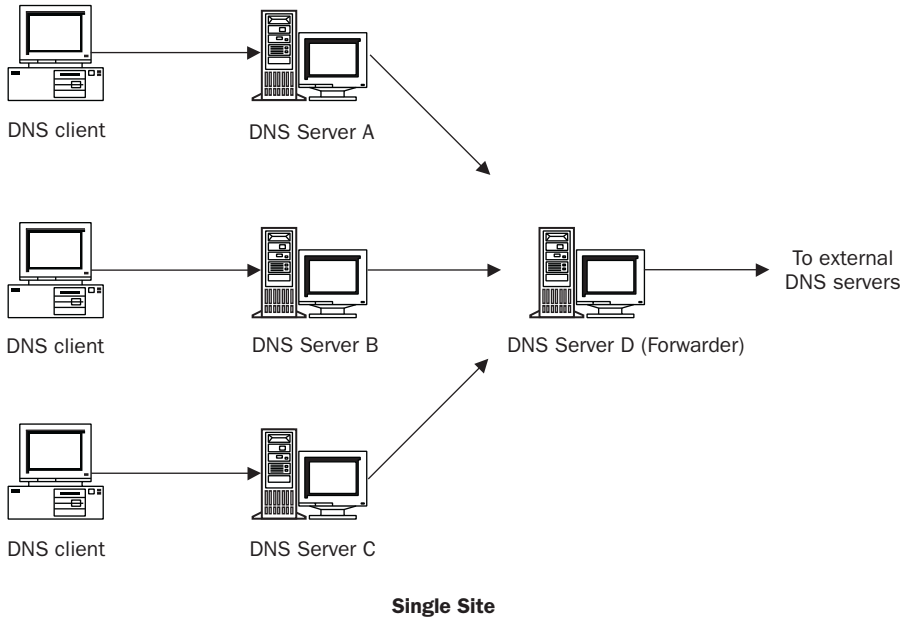


Figure 17-3. *DNS forwarder.*

In this example, when any of the DNS clients send recursive queries to DNS servers A, B, and C, each DNS server attempts to answer the query from locally held zones or from its local cache. If this is unsuccessful, these servers issue a recursive query to DNS server D, which has a better chance of answering the query from its own cache. This arrangement can reduce the external traffic needed to resolve host queries.

If the forwarder (server D in the example) is unable to answer the queries sent by DNS servers A, B, or C, these servers attempt to resolve the queries themselves by issuing iterative queries, which, again, might not be desirable.

A slave server is a DNS server that attempts to resolve name queries by checking its cache and local zone files, and then it forwards the query to a DNS forwarder. A slave server does not attempt to perform iterative queries if the DNS forwarder is unable to answer the query.

Round Robin Load Balancing

Round robin is an approach for performing DNS load balancing. It is used to share and distribute the network resource load. With round robin, the answers contained in a query, for which multiple RRs exist, are rotated each time the query is answered. Round robin is a very simple method for load balancing a client's use of Web servers and other frequently queried multihomed services.

For round robin to work, multiple A RRs for the queried name must exist in the zone being queried. For example, suppose there were three physical Web servers servicing www.kapoho.com, with the IP addresses of 10.1.1.151, 10.1.152, and 10.1.1.153 with three A records for www.kapoho.com pointing to the different servers. Round robin would need to be configured at the server. Round robin is enabled by default in Windows Server 2003. The first query for this domain would be returned in the order 10.1.1.151, 10.1.1.152, and 10.1.1.153. The following query would return 10.1.1.152, 10.1.1.153, and 10.1.1.151, and so on. Because the client usually takes the first IP address, the first query would use the IP address 10.1.1.151, and the second would use 10.1.1.152.

By default, the DNS Server service for Windows Server 2003 performs round robin for all RR types. You can specify that certain RR types do not have round robin rotation applied by setting the following registry entry:

DoNotRoundRobinTypes

```
Location:  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\DNS\Parameters
Data type: REG_SZ
Default value: none
Present by default: No
```

The string for DoNotRoundRobinTypes is a comma-separated list of valid RR types. If you wanted to not round robin PTR, SRV, and NS records, the DoNotRoundRobinTypes value would be "ptr,srv,ns."

You can also turn off round robin entirely by using the DNS snap-in or by setting the following registry entry:

RoundRobin

```
Location:  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\DNS\Parameters
Data type: REG_DWORD
Default value: 1
Valid range: 0-1 (1 - Enable round robin, 0 - Disable round robin)
Present by default: Yes
```

DNS Dynamic Updates

For large networks, getting all the necessary RR information into the DNS and keeping it current can be a significant task. To simplify the task, Windows Server 2003 includes support for dynamic updates to DNS, as described in RFC 2136.

With DNS dynamic updates, the client sends a DNS registration message to the DNS server, instructing the server to update the A record for the dynamic update host. Additionally, if the client is also a DHCP client of a Windows Server 2003 DHCP server, every time there is an address event (for instance, a new address or address renewal), as part of the DHCP lease-management process, the Windows DHCP client sends DHCP Option 81 to the DHCP server along with its fully qualified name. Option 81 instructs the DHCP server to register a PTR RR on its behalf. Windows Server 2003 computers that are statically configured register both the A RR and the PTR RR with the DNS server themselves.

If a Windows DHCP client talks to a lower level DHCP server that does not handle Option 81, the client registers a PTR RR on its own. The DNS Server service for Windows Server 2003 is capable of handling dynamic updates.

This approach (client updating the A record, DHCP server updating the PTR record) is taken because only the client knows which IP addresses on the host map to a given host name. The DHCP server might not be able to properly do the A RR registration because it has incomplete knowledge. If appropriate, the DHCP server also can be configured to register both records with the DNS.

IPv6 Support

IPv6 is a new version of IP. The DNS Server service for Windows Server 2003 provides support for IPv6 by implementing several additional pieces of functionality, including the following:

- **AAAA RR** This new record type is defined to store a host's IPv6 address. A multihomed IPv6 host (for example, a host that has more than one IPv6 address) must have more than one AAAA record (known as a "quad A" record). The AAAA RR is similar to the A RR, using the larger IP address size. The 128-bit

IPv6 address is encoded in the data portion of an AAAA RR in network byte order (high-order byte first).

- **AAAA query** The AAAA query for a specified domain name in the Internet class returns all associated AAAA RRs in the answer section of a response. The AAAA query does not perform additional section processing.
- **IP6.ARPA domain** This domain is used to provide reverse-lookup faculties for IPv6 hosts (as the in-addr.arpa domain does for IPv4 addresses).

Similar to the in-addr.arpa domain for IPv4, an IPv6 address is represented as a name in the IP6.ARPA domain by a sequence of nibbles (hexadecimal digits) separated by periods with the suffix .IP6.ARPA. The sequence of nibbles is encoded in reverse order; for instance, the low-order nibble is encoded first, with the highest order nibble last. A hexadecimal digit represents each nibble. For example, the inverse-lookup domain name corresponding to the address 3ffe:ffff:1:2:3:4:567:89ab would be

b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.f.f.f.f.e.f.f.3.IP6.ARPA.

Finally, to support IPv6, all existing DNS query types that perform type A additional section processing—such as NS or MX query types—must support both A and AAAA records and must do any processing associated with both of these record types. This means the DNS server adds any relevant IPv4 addresses and any relevant IPv6 addresses available locally to the additional section of a response when processing any one of these queries.

DNS Extension Mechanism

The DNS protocol, on the wire, uses a number of fixed-length fields or fields with a set of possible values that is nearly used up. One example of this is the maximum allowable DNS packet size when using User Datagram Protocol (UDP), 512 bytes. To enable DNS clients and servers to use larger values, RFC 2671 defines a general extension model for DNS as well as the first extension (EDNS0). EDNS0 specifically provides a mechanism for clients and servers to use larger DNS messages over UDP. It is anticipated that further DNS extensions will be proposed and will be identified as EDNS1, EDNS2, and so forth.

EDNS0 enables DNS clients to advertise and support the transfer of DNS messages larger than 512 bytes. A DNS client that is capable of handling EDNS0 sends queries over UDP that contain a special pseudo-RR, the OPT resource record, to note the UDP packet size that the DNS client can support. This pseudo-RR enables servers that can also handle EDNS0 to increase the size of a DNS reply and send as many RRs as are allowed in the maximum UDP packet size (specified in the OPT RR).

The OPT RR is not an RR that you can add to the DNS database, but it is generated if and when a host can handle the DNS extensions. In addition, the OPT RR does not contain actual DNS data. Rather it contains the DNS client's maximum UDP payload size in its CLASS field, as well as the number of octets in the largest UDP payload that the client can deliver in the clients' network.

By default, the DNS server includes an OPT RR in the additional section of any query it sends if it is capable of supporting EDNS0. The OPT record gives the UDP maximum length of a DNS response. If a DNS server receives a DNS query that does not contain an OPT RR, the server assumes the querier's system does not support EDNS0 and all responses to this system are truncated if the maximum size is greater than 512 octets.

If the DNS Server service for Windows Server 2003 receives either a request or response from any host that has an OPT record, the DNS server caches this information, by default, for one week. In addition, if a request or a response does not contain an OPT record, this is also cached.

You can modify the EDNS cache time-out by modifying the following registry value:

EDNSCacheTimeout

```
Location:  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\DNS\Parameters
Data type: REG_DWORD
Default value: 25200 (7 days)
Valid range: 3600 (1 hour) to 15724800 (182 days)
Present by default: No
```

If a DNS server does not support ENDS0, then queries containing the OPT resource might not be handled and might return an error code. The error codes that could be encountered are as follows:

- **FORMERR (1)** The name server did not interpret the OPT RR.
- **SERVFAIL(2)** The name server did not process the query because of a problem with the name server.
- **NOTIMPL(4)** The name server does not support the kind of query requested.

DNS Security

RFC 2535 defines a set of facilities, known as DNS Security (DNSSEC), that provides authentication and data integrity of DNS data to resolvers. These extensions allow digital signatures to be encrypted using private keys. These signatures can then be sent as RRs from DNS servers hosting signed (DNSSEC-compliant) zones to resolvers. DNSSEC-aware revolvers can then authenticate these signatures using public keys. The digital signatures and public keys are added to a signed zone in the form of RRs.

With DNSSEC, a zone has a public and a private key. The private key, which is not loaded in the zone, is used to sign a zone or parts of a zone. The public key is used to validate this digital signature. The zone's private key can be used to sign each domain name in the zone (for example, rebecca.kapoho.com). These signatures can then be added to the zone and returned as part of a query to a DNSSEC-aware DNS resolver. Such a resolver can use the public key to authenticate the RRs.

With DNSSEC, a zone's public key is stored in a KEY RR. The digital signature is contained in a SIG RR. KEY RRs must be provided to the resolver before it can authenticate SIG RRs. DNSSEC introduces one further RR, the NXT record, which can be used to provide cryptographic assurance to a resolver that a particular RR does not exist.

DNSSEC is not fully supported in the DNS Server service for Windows Server 2003. The DNS Server service for Windows Server 2003 provides basic support of the DNSSEC protocol, as defined in RFC 2535. This would enable, for example, a Windows Server 2003 DNS server to operate as a secondary to a BIND server that fully supports DNSSEC. In particular, Windows Server 2003 provides no way to sign or verify the digital signatures. The Windows Server 2003 resolver does not validate any of the DNSSEC data returned as the result of a query.

How DNS Works

In this section, the DNS client and server functions are described in more detail.

Configuring DNS Client Functions

With Windows Server 2003, there is generally very little configuration to do for a client with respect to DNS. Generally, it is only necessary to configure the host with the IP address of a primary and secondary DNS server. This can be simplified by using DHCP to assign the IP address of the DNS servers.

Usually, DNS default client behavior is adequate. However, in certain cases, some change to the default behavior might be appropriate. The following registry keys can be used to change how the Windows Server 2003 family DNS client works.

Specifying a Default TTL

By default, the TTL for the A and PTR RR updates sent by a DNS client is 20 minutes. To change the TTL value, you can configure the following registry value:

DefaultRegistrationTTL

```
Key:  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
Value type: REG_DWORD
Default: 0x4B0 (1200 seconds, or 20 minutes)
Valid range: 0-0xffffffff
Present by default: No
```

After setting this registry key, you do not need to reboot your computer for the updated TTL to be used. All registration requests contain the new TTL set in the value entry.

Disabling Dynamic Updates

Although the automatic updating of DNS zones by a host can be useful, in some environments it might not be desirable. You can use the following registry key to disable DNS

dynamic updates either for a computer running Windows Server 2003 as a whole, or for just one or more interfaces on that computer. This entry disables the DNS client from registering the A and PTR records for some or all interfaces on a computer running Windows Server 2003.

DisableDynamicUpdate

```
Key: KEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
Or
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
Tcpip\Parameters\Interfaces\<InterfaceGUID>
Value type: REG_DWORD--Boolean
Valid range: 0, 1 (False, True)
Default: 0
Present by default: No
```

Resolving Names

DNS name resolution occurs when a DNS resolver sends a DNS server a query message containing a request to find the name and return certain RRs. The query message contains the domain name to search for, plus a code indicating the records that should be returned.

The following Network Monitor trace (Capture 17-01, included in the \Captures folder on the companion CD-ROM) shows the process of issuing and resolving a name query.

```
1  7.014382  0050564050E1  3COM  6B15C7  DNS  0x1C:Std Qry for
kona.kapoho.com. of type Host Addr on class INET addr. 10.10.1.68  10.10.1.200
+ Frame: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
+ IP: ID = 0xCC5; Proto = UDP; Len: 61
+ UDP: Src Port: Unknown, (1026); Dst Port: DNS (53); Length = 41 (0x29)
  DNS: 0x1C:Std Qry for kona.kapoho.com. of type Host Addr on class INET addr.
      DNS: Query Identifier = 28 (0x1C)
      DNS: DNS Flags = Query, OpCode - Std Qry, RD Bits Set, RCode - No error
          DNS: 0............... = Request
          DNS: .0000........... = Standard Query
          DNS: .....0.......... = Server not authority for domain
          DNS: ......0......... = Message complete
          DNS: .......1........ = Recursive query desired
          DNS: ........0....... = No recursive queries
          DNS: .........000.... = Reserved
          DNS: ............0000 = No error
      DNS: Question Entry Count = 1 (1)
      DNS: Answer Entry Count = 0 (0)
      DNS: Name Server Count = 0 (0)
      DNS: Additional Records Count = 0 (0)
```

```
      DNS: Question Section: kona.kapoho.com. of type Host Addr on class INET
      addr.
          DNS: Question Name: kona.kapoho.com.
          DNS: Question Type = Host Address
          DNS: Question Class = Internet address class

2  7.024396  3COM  6B15C7  0050564050E1  DNS  0x1C:Std Qry Resp. for
kona.kapoho.com. of type Host Addr on class INET addr. 10.10.1.200  10.10.1.68

+ Frame: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
+ IP: ID = 0x43AE; Proto = UDP; Len: 77
+ UDP: Src Port: DNS, (53); Dst Port: Unknown (1026); Length = 57 (0x39)
  DNS: 0x1C:Std Qry Resp. for kona.kapoho.com. of type Host Addr on class INET
  addr.
      DNS: Query Identifier = 28 (0x1C)
      DNS: DNS Flags = Response, OpCode - Std Qry, AA RD RA Bits Set, RCode -
      No error
          DNS: 1............... = Response
          DNS: .0000........... = Standard Query
          DNS: .....1.......... = Server authority for domain
          DNS: ......0......... = Message complete
          DNS: .......1........ = Recursive query desired
          DNS: ........1....... = Recursive queries supported by server
          DNS: .........000.... = Reserved
          DNS: ............0000 = No error
      DNS: Question Entry Count = 1 (1)
      DNS: Answer Entry Count = 1 (1)
      DNS: Name Server Count = 0 (0)
      DNS: Additional Records Count = 0 (0)
      DNS: Question Section: kona.kapoho.com. of type Host Addr on class INET
      addr.
          DNS: Question Name: kona.kapoho.com.
          DNS: Question Type = Host Address
          DNS: Question Class = Internet address class
      DNS: Answer section: kona.kapoho.com. of type Host Addr on class INET
      addr.
          DNS: Resource Name: kona.kapoho.com.
          DNS: Resource Type = Host Address
          DNS: Resource Class = Internet address class
          DNS: Time To Live = 1200 (0x4B0)
          DNS: Resource Data Length = 4 (0x4)
          DNS: IP address = 10.10.2.200
```

In this trace, a client sends a DNS query to request the DNS server to return all A RRs for kona.kapoho.com. The query response contains the question entry and the answer RRs. In this case, there is only one A record to return pointing to 10.10.2.200.

Network Monitor trace 17-2 (Capture 17-02, included in the \Captures folder on the companion CD-ROM) shows a Reverse-Lookup Query. In this trace, the querying host at 10.10.1.68 queries DNS for the host name for the host at 10.10.1.200. To determine this, the resolver queries for 200.1.10.10.in-addr.arpa and requests any PTR records. The DNS server then returns the PTR record, which shows the host to be mahimahi.kapoho.com.

Resolving Aliases

The CNAME RR allows you to create an alias for a host. For example, you could create an alias ns1.kapoho.com for the host mahimahi.kapoho.net. When a resolver performs forward name resolution, it does not know, in advance, whether the name relates to a Host Address (A) RR or to a CNAME. If it relates to the CNAME, the DNS server could just return the CNAME record. This would, however mean that the real host name still needs to be resolved. To avoid extra DNS traffic, when a DNS server returns a CNAME in response to an A RR lookup, a Windows Server 2003 DNS server also returns the A record relating to the CNAME.

The following Network Monitor trace (Capture 17-03, included in the \Captures folder on the companion CD-ROM) shows the process of issuing and resolving a canonical name.

```
1  0.000000  0050564050E1  3COM  6B15C7  DNS  0x57:Std Qry for ns1.kapoho.com.
of type Host Add 10.10.1.68  10.10.1.200
+ Frame: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
+ IP: ID = 0x9C66; Proto = UDP; Len: 60
+ UDP: Src Port: Unknown, (1026); Dst Port: DNS (53); Length = 40 (0x28)
  DNS: 0x57:Std Qry for ns1.kapoho.com. of type Host Addr on class INET addr.
      DNS: Query Identifier = 87 (0x57)
    + DNS: DNS Flags = Query, OpCode - Std Qry, RD Bits Set, RCode - No error
      DNS: Question Entry Count = 1 (1)
      DNS: Answer Entry Count = 0 (0)
      DNS: Name Server Count = 0 (0)
      DNS: Additional Records Count = 0 (0)
      DNS: Question Section: ns1.kapoho.com. of type Host Addr on class INET
      addr.
          DNS: Question Name: ns1.kapoho.com.
          DNS: Question Type = Host Address
          DNS: Question Class = Internet address class
```

```
2  0.000000  3COM  6B15C7  0050564050E1  DNS  0x57:Std Qry Resp. for
ns1.kapoho.com. of type Ca 10.10.1.200  10.10.1.68

+ Frame: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
+ IP: ID = 0x4D06; Proto = UDP; Len: 99
+ UDP: Src Port: DNS, (53); Dst Port: Unknown (1026); Length = 79 (0x4F)
  DNS: 0x57:Std Qry Resp. for ns1.kapoho.com. of type Canonical name on class
       INET addr.
      DNS: Query Identifier = 87 (0x57)
    + DNS: DNS Flags = Response, OpCode - Std Qry, AA RD RA Bits Set, RCode -
    No error
      DNS: Question Entry Count = 1 (1)
      DNS: Answer Entry Count = 2 (0x2)
      DNS: Name Server Count = 0 (0)
      DNS: Additional Records Count = 0 (0)
      DNS: Question Section: ns1.kapoho.com. of type Host Addr on class INET
      addr.
          DNS: Question Name: ns1.kapoho.com.
          DNS: Question Type = Host Address
          DNS: Question Class = Internet address class
      DNS: Answer section: ns1.kapoho.com. of type Canonical name on class
      INET addr.
           (2 records present)
          DNS: Resource Record: ns1.kapoho.com. of type Canonical name on
          class INET addr.
              DNS: Resource Name: ns1.kapoho.com.
              DNS: Resource Type = Canonical name for alias
              DNS: Resource Class = Internet address class
              DNS: Time To Live = 3600 (0xE10)
              DNS: Resource Data Length = 11 (0xB)
              DNS: Owner primary name: mahimahi.kapoho.com.
          DNS: Resource Record: mahimahi.kapoho.com. of type Host Addr on
          class INET addr.
              DNS: Resource Name: mahimahi.kapoho.com.
              DNS: Resource Type = Host Address
              DNS: Resource Class = Internet address class
              DNS: Time To Live = 3600 (0xE10)
              DNS: Resource Data Length = 4 (0x4)
              DNS: IP address = 10.10.1.200
```

In this trace, the DNS client sends a DNS query to the DNS server requesting the A (host) record for ns1.kapoho.com, which is actually an alias for mahimahi.kapoho.com. In the

DNS reply, there are two answer RRs. The first is the CNAME RR for ns1.kapoho.com, which contains the canonical name. The second answer RR is the A record for mahimahi.kapoho.com, which contains the IP address of this computer (10.10.1.200).

Dynamically Updating DNS

DNS dynamic updates, described in RFC 2136, are a mechanism that enables DNS clients to add or delete a specific RR or sets of RRs, known as RRSets, to any zone. Dynamic updates can simplify the process of managing the contents of a DNS zone.

Dynamic update requests can also state prerequisites specified separately from update operations. These can be tested before an update can occur. When prerequisites are used with dynamic updates, the updates are said to be *atomic*; that is, all prerequisites must be satisfied for the update operation to be carried out.

Windows Server 2003 fully supports DNS dynamic updates. From the client side, the Windows Server 2003 TCP/IP client and the DHCP server both issue DNS dynamic update requests to register A and PTR records. On the server side, the DNS Server service for Windows Server 2003 accepts DNS dynamic updates. DNS dynamic updates for AAAA records are also supported by Windows XP and Windows Server 2003 for both DNS clients and servers.

The following Network Monitor trace (Capture 17-04, included in the \Captures folder on the companion CD-ROM) shows the process of dynamically registering an A RR.

```
1  15.472248  KAPOHO11        MAHIMAHI        DNS  0x62:Dyn Upd PRE/UPD
records to kapoho11.kapoho.n KAPOHO11         MAHIMAHI          IP
+ Frame: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
+ IP: ID = 0xB1BD; Proto = UDP; Len: 115
+ UDP: Src Port: Unknown, (1026); Dst Port: DNS (53); Length = 95 (0x5F)
  DNS: 0x62:Dyn Upd PRE/UPD records to kapoho11.kapoho.net. of type
       Canonical name
     DNS: Query Identifier = 98 (0x62)
   + DNS: DNS Flags = Query, OpCode - Dyn Upd, RCode - No error
     DNS: Zone Count = 1 (1)
     DNS: Prerequisite Section Entry Count = 1 (1)
     DNS: Update Section Entry Count = 2 (0x2)
     DNS: Additional Records Count = 0 (0)
     DNS: Update Zone: kapoho.net. of type SOA on class INET addr.
        DNS: Update Zone Name: kapoho.net.
        DNS: Update Zone Type = Start of zone of authority
        DNS: Update Zone Class = Internet address class
     DNS: Prerequisite: kapoho11.kapoho.net. of type Canonical name on
          class Unknown Class
```

```
         DNS: Resource Name: kapoho11.kapoho.net.
         DNS: Resource Type = Canonical name for alias
         DNS: Resource Class = 0x00FE
         DNS: Time To Live = 0 (0)
         DNS: Resource Data Length = 0 (0)
      DNS: Update: kapoho11.kapoho.net. of type Host Addr on class Req. for
      any(2 records present)
       + DNS: Resource Record: kapoho11.kapoho.net. of type Host Addr on class
       Req. for any
       + DNS: Resource Record: kapoho11.kapoho.net. of type Host Addr on class
       INET addr.

2  15.482263  MAHIMAHI        KAPOHO11        DNS  0x62:Dyn Upd Resp. PRE/UPD
records to kapoho11.ka MAHIMAHI          KAPOHO11         IP

+ Frame: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
+ IP: ID = 0x942B; Proto = UDP; Len: 115
+ UDP: Src Port: DNS, (53); Dst Port: Unknown (1026); Length = 95 (0x5F)
  DNS: 0x62:Dyn Upd Resp. PRE/UPD records to kapoho11.kapoho.net. of type
  Canonical name
     DNS: Query Identifier = 98 (0x62)
   + DNS: DNS Flags = Response, OpCode - Dyn Upd, RCode - No error
     DNS: Zone Count = 1 (1)
     DNS: Prerequisite Section Entry Count = 1 (1)
     DNS: Update Section Entry Count = 2 (0x2)
     DNS: Additional Records Count = 0 (0)
     DNS: Update Zone: kapoho.net. of type SOA on class INET addr.
        DNS: Update Zone Name: kapoho.net.
        DNS: Update Zone Type = Start of zone of authority
        DNS: Update Zone Class = Internet address class
     DNS: Prerequisite: kapoho11.kapoho.net. of type Canonical name on class
     Unknown Class
        DNS: Resource Name: kapoho11.kapoho.net.
        DNS: Resource Type = Canonical name for alias
        DNS: Resource Class = 0x00FE
        DNS: Time To Live = 0 (0)
        DNS: Resource Data Length = 0 (0)
     DNS: Update: kapoho11.kapoho.net. of type Host Addr on class Req. for
     any(2 records present)
      + DNS: Resource Record: kapoho11.kapoho.net. of type Host Addr on class
      Req. for any
      + DNS: Resource Record: kapoho11.kapoho.net. of type Host Addr on class
      INET addr.
```

In this trace, the DNS dynamic update message is sent from the DNS client to the DNS server to update the A RR for the host kapoholt.kapoho.com, which is now at IP address 10.10.1.52.

Transferring Zone Information

There are three methods of performing zone transfer:

- **Traditional zone transfer** This approach, described in RFC 1034, involves the secondary server requesting a full copy of the zone from the primary server.
- **Incremental zone transfer** This approach, defined in RFC 1995, requires the DNS server hosting the primary zone to keep a record of the changes that are made between each increment of the zone's sequence number. The secondary zone can thus request only the changes that occurred since the last time the secondary zone was updated.
- **Active Directory zone transfer** Active Directory zones are replicated to all domain controllers in the Windows Server 2003 domain using Active Directory replication.

The traditional zone transfer mechanism can be wasteful of network resources if the change in the transferred RRs is small in relation to the overall zone. In such cases, incremental zone transfer is more efficient. The following Network Monitor trace (Capture 17-05, included in the \Captures folder on the companion CD-ROM) shows a traditional zone transfer:

```
1   78.603025  Secondary   Primary    TCP  ....S., len:    0, seq: 105565995-
105565995, ack
2   78.603025  Primary     Secondary  TCP  .A..S., len:    0, seq:3198736198-
3198736198, ack
3   78.603025  Secondary   Primary    TCP  .A...., len:    0, seq: 105565996-
105565996, ack
4   78.613040  Secondary   Primary    DNS  0x0:Std Qry for kapoho.com. of type
Req for zn Xfer on class INET addr.
5   78.613040  Primary     Secondary  DNS  0x0:Std Qry Resp. for kapoho.com.
of type SOA on class INET addr.
6   78.613040  Primary     Secondary  DNS  0x6574:Std Qry for ÌÍ_ of type
Unknown Type on class Unknown Class
7   78.613040  Secondary   Primary    TCP  .A...., len:    0, seq: 105566028-
105566028, ack:
8   78.613040  Primary     Secondary  DNS  0x7200:Std Qry for X of type
Unknown Type on class Unknown Class
9   78.643083  Secondary   Primary    TCP  .A...F, len:    0, seq: 105566028-
105566028, ack
10  78.643083  Primary     Secondary  TCP  .A...., len:    0, seq:3198739143-
3198739143, ack
11  78.643083  Primary     Secondary  TCP  .A...F, len:    0, seq:3198739143-
3198739143, ack
12  78.643083  Secondary   Primary    TCP  .A...., len:    0, seq: 105566029-
105566029, ack
```

This trace shows a zone transfer of the zone kapoho.com from the primary to a secondary server. In this trace, the secondary DNS server first initiates a TCP connection with the primary server and issues a zone transfer message. The primary zone's DNS server then transfers the zone RRs. In a standard zone transfer, the first and last record transferred is always the zone's SOA record. After all the records are transferred, the TCP connection is terminated.

Incremental zone transfers, described in RFC 1995, can be more efficient than traditional zone transfers for both large and dynamic zones. However, they place additional processing requirements on the DNS server, which needs to keep track of the zone differences and sends only the changed records. By default, the DNS Server service for Windows Server 2003 uses incremental transfers when possible.

The following Network Monitor trace (Capture 17-06, included in the \Captures folder on the companion CD-ROM) shows an incremental zone transfer:

```
1   21.108471  Secondary  Primary    DNS  0x6000:Std Qry for kapoho.com. of
type SOA on class INET addr.
2   21.108471  Primary    Secondary  DNS  0x6000:Std Qry Resp. for kapoho.com.
of type SOA on class INET addr.
3   21.108471  Secondary  Primary    DNS  0x4000:Std Qry for kapoho.com. of
type Req for incrmntl zn Xfer on class INET addr.
4   21.108471  Primary    Secondary  DNS  0x4000:Std Qry Resp. for kapoho.com.
of type SOA on class INET addr.
```

In this trace, the DNS server initiating the zone transfer first queries for the SOA record, then requests an incremental zone transfer. In this example, the reply, contained in the fourth packet, fully fits inside a single UDP datagram. Had this not been the case, the reply message would have indicated that the reply was truncated, and the requesting server would have created a TCP session to the other DNS server and requested the zone transfer using TCP.

Active Directory replication is a proprietary solution that can be used only with Microsoft Windows 2000 or Windows Server 2003 domain controllers. Normal DNS zone transfers are pull in nature—the secondary DNS servers pull the zone or zone changes from the primary server. Active Directory replication, on the other hand, is push in nature—the directory changes are pushed from the domain controller on which the change occurred to the other domain controllers. For zones that change little, Active Directory replication ensures that all DNS servers storing the zone are updated quickly. For more dynamic zones this tends to smooth the replication traffic. Active Directory replication is beyond the scope of this book.

DNS Resource Records

This section describes the contents of the DNS RRs.

What Are Resource Records?

An RR is information related to a DNS domain; for example, the host record defining a host IP address. Each RR contains a common set of information, as follows:

- **Owner** Indicates the DNS domain in which the RR is found.
- **TTL** The length of time used by other DNS servers to determine how long to cache information for a record before discarding it. For most RRs, this field is optional. The TTL value is measured in seconds, with a TTL value of 0 indicating that the RR contains volatile data that is not to be cached. As an example, SOA records have a default TTL of 3600 seconds (1 hour). This prevents these records from being cached by other DNS servers for a longer period, which would delay the propagation of changes.
- **Class** For most RRs, this field is optional. Where it is used, it contains standard mnemonic text indicating the class of an RR. For example, a class setting of IN indicates the record belongs to the Internet (IN) class. At one time there were multiple classes (such as CH for Chaos Net), but today, only the IN class is used.
- **Type** This required field holds a standard mnemonic text indicating the type for an RR. For example, a mnemonic of A indicates that the RR stores host address information.
- **Record-Specific Data** This is a variable-length field containing information describing the resource. This information's format varies according to the type and class of the RR.

Note With Windows Server 2003, most DNS information is either automatically added to the server or can be left to a default value. For most organizations running Windows 2000, Windows XP, and Windows Server 2003, DNS is self-maintaining once the DNS servers are installed and the relevant zones are created. However, the details on RR types can be useful for those integrating Windows Server 2003 with a non–Windows DNS server, or for troubleshooting.

Standard DNS zone files contain the set of RRs for that zone as a text file. In this text file, each RR is on a separate line and contains all the just-mentioned data items, as a set of text fields separated by white space. In the zone file, each RR consists of these data items, although different records contain slightly differently formatted record-specific data.

Sample Zone Data

The zone data for the kapoho.com zone noted earlier in this chapter is as follows:

```
;
;  Database file kapoho.com.dns for kapoho.com zone.
;      Zone version:  93
@                       IN  SOA mahimahi.kapoho.com. hostmaster. (
  93           ; serial number
  15           ; refresh
  600          ; retry
  86400        ; expire
  3600       ) ; default TTL
;  Zone NS records
;  There are 2 DNS servers for this zone
@                       NS     kapoho10.kapoho.com.
@                       NS     mahimahi.kapoho.com.
;  Zone records
@                       600    A       10.10.1.74
@                       600    A       10.10.1.200
kapoho10                35     A       10.10.1.74
kona                    1200   A       10.10.2.200
mahimahi                        A      10.10.1.200
maui                    1200   A       10.10.1.62
tallguy                 1200   A       10.10.2.100
                        1200   A       10.10.1.100
waimea                  1200   A       10.10.1.67
; Aliases
news                    CNAME  kona2.kapoho.com.
ns1                     CNAME  mahimahi.kapoho.com.
```

Zone data for Active Directory–integrated zones are held as a series of Active Directory objects representing this data. For more detail on how Active Directory holds DNS-integrated zones, see Windows Server 2003 Help And Support.

Where Are RRs Located?

RRs for standard zones are stored in the folder %*Systemroot*%\System32\Dns. The RRs for each zone are held in a separate text file, which is named after the zone with an extension of .dns; for example, kapoho.com.dns.

Active Directory–Integrated Zone RRs

RRs for Active Directory–integrated DNS zones are stored within Active Directory itself. Active Directory uses the following two main object classes to hold this DNS information:

- **dnsZone** Represents an Active Directory–integrated zone that contains dnsNode objects. This object class is the Active Directory equivalent of a standard zone held as a text file. The dnsZone objects have a dnsProperty attribute that defines key details about the zone, such as whether it can be dynamically updated.
- **dnsNode** Corresponds to the individual RRs in the zone. Each dnsNode object has a dnsRecord attribute containing the resource information.

RRs Supported by Windows Server 2003

The DNS Server service for Windows Server 2003 supports all RFC-compliant RRs. Many of these are not commonly, or ever, used. The following sections list the most commonly used RRs. For the RR syntax and an example, see the topic entitled "Resource Records Reference" in Windows Server 2003 Help And Support.

- **Host Address (A)** This RR contains a host address RR that maps a DNS domain name to an IPv4 32-bit address.
- **IPv6 Host Record (AAAA)** This RR contains a host address RR that maps a DNS domain name to an IPv6 128-bit address.
- **Canonical Name (CNAME)** This RR maps an alias or alternate DNS domain name in the Owner field to a canonical or actual DNS domain name. There must also be an A RR for the canonical DNS domain name, which must resolve to a valid DNS domain name in the namespace. The fully qualified canonical name should end with a full stop (".").
- **KEY (KEY)** The KEY record holds a public key for a DNSSEC-aware zone. KEY RRs are signed by the key from their parent zone.
- **Mail Exchanger (MX)** The MX record provides message routing to a mail-exchanger host for any mail that is to be sent to the target domain. This RR also contains a two-digit preference value to indicate the preferred ordering of hosts, if multiple exchanger hosts are specified. Each exchanger host specified in an MX record must have a corresponding host A address RR in the current zone.
- **Next (NXT)** This record is used as part of DNSSEC. The NXT record is used to indicate the next RR in the zone. In conjunction with a SIG record, this can provide cryptographic assurance of the nonexistence of a name in a zone. For a given domain name, such as puna.kapoho.com, the NXT record points to the

next record in the zone, (for example, waimea.kapoho.com) and states which RRs exist at the domain name (for example, NS, SOA, and so on). If the DNS server receives a request for, say, sunshine.kapoho.com, the negative reply would contain the reply code NXDOMAIN. The authority section of the reply would include the NXT and SIG records for the domain name after and before the name that did not exist.

- **Option (OPT)** This is a pseudo-RR, used to provide the extended DNS functionality provided by EDNS0. An OPT RR can be in any DNS query response. The OPT record relates to the underlying transport-level message (that is, the UDP datagram's payload) and not to the DNS data.

  In an OPT pseudo-RR, the Name field is always empty (that is, the root domain), and the Class value holds the sender's maximum UDP payload size. This value signifies the largest UDP payload that can be delivered and reassembled in the sender's network stack. RFC 2671 suggests for Ethernet that a value of 1280 is considered reasonable; this is the default value used by Windows Server 2003.
- **Pointer (PTR)** The PTR record is used to store the FQDN for an IP address for reverse name queries. The PTR record is used only within the in-addr.arpa domain.
- **Signature (SIG)** The SIG record is used by DNSSEC to hold the digital signature for an RR or set of RRs.
- **Service Locator (SRV)** The SRV RR enables a computer to locate a host providing a specific service, such as a Windows Server 2003 Active Directory domain controller. This enables the administrator to have multiple servers, each providing a similar TCP/IP-based service to be located using a single DNS query operation. This record is mainly used to support the Windows Server 2003 Active Directory, where all relevant SRV RRs are automatically populated into the DNS.

DNS Messages

DNS messages are sent between a DNS client and a DNS server or between two DNS servers. These messages are usually transmitted using UDP with the DNS server binding to UDP port 53. In some cases, the message length, particularly for responses, might exceed the maximum size of a UDP datagram. In such cases, an initial response is sent with as much data as fits into the UDP datagram. The DNS server sets a flag to indicate a truncated response. The client can then contact the server using TCP (port 53) and reissue the request—taking advantage of TCP's capability to reliably handle longer streams of data. This approach uses UDP's performance for most queries while providing a simple mechanism to handle longer queries. If the client and server support EDNS0, they can exchange larger UDP datagrams.

DNS Message Types

DNS originally provided dynamic name resolution for essentially static, manually updated data, such as host records manually added to a zone. The original DNS messages involved sending a query to a DNS server and getting a response. RFC 2136 defines the DNS dynamic update facility, which makes use of update messages, whose format is similar to and derived from query messages. Both message types are described next.

DNS Query Message Format

All DNS query messages share a common basic format, as Figure 17-4 illustrates.

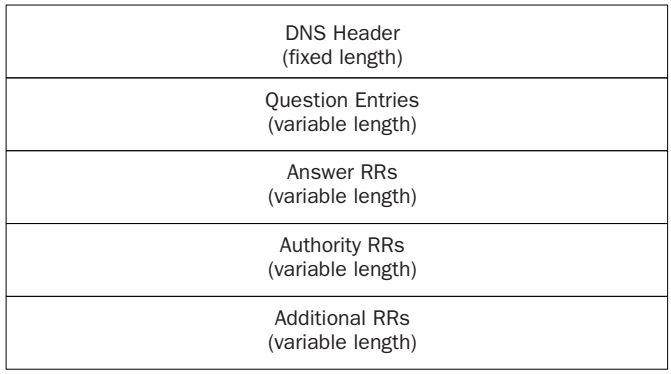


Figure 17-4. *Generic DNS query message format.*

As can be seen from Figure 17-4, the DNS query message consists of a fixed-length 12-byte header, plus a variable portion holding question, answer, authority, and additional DNS RRs.

DNS Query Message Header

The DNS message header consists of the following fields:

- **Transaction ID** A 16-bit field used to identify a specific DNS transaction. The originator creates the transaction ID and the responder copies it into a reply message. This enables the client to match responses received from a DNS server to the requests that were sent to the server.
- **Flags** A 16-bit field containing various service flags, described in more detail later in this chapter.
- **Question RR Count** A 16-bit field indicating the number of entries in the question section in the DNS message.
- **Answer RR Count** A 16-bit field indicating the number of entries in the answer RRs section of the DNS message.

- **Authority RR Count** A 16-bit field indicating the number of authority RRs in the DNS message.
- **Additional RR Count** A 16-bit field indicating the number of additional RRs in the DNS message.

The DNS header is shown in Figure 17-5.

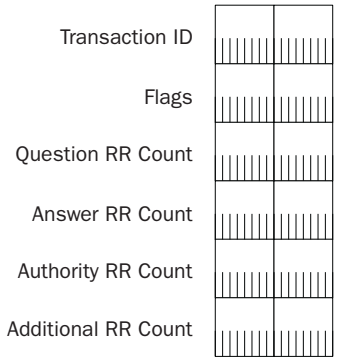


Figure 17-5. *DNS message header.*

The Flags field contains a number of status fields that are communicated between client and server. Figure 17-6 shows the format of the Flags field.

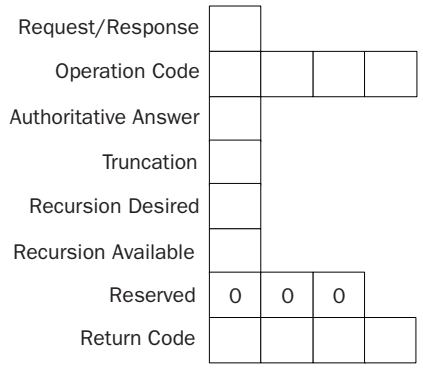


Figure 17-6. *DNS message Flags field.*

The individual fields in the Flags field are as follows:

- **Request/Response** This 1-bit field is set to 0 indicate a name service request and 1 to indicate a name service response.
- **Operation Code** This 4-bit field indicates the specific name service operation of the name service packet, as listed in Table 17-3.

Table 17-3. Operation Codes

Operation Code	Operation
0x0	Query
1	Inverse Query
0x2	Server Status Request

- **Authoritative Answer** This flag is set in a reply when the responder is authoritative for the domain name in the question section.
- **Truncation** This flag is set to 1 if the total number of responses could not fit into the UDP datagram (for instance, if the total number exceeds 512 bytes). In this case, only the first 512 bytes of the reply are returned unless EDNS0 and longer UDP datagrams are being used.
- **Recursion Desired** This flag is set to 1 to indicate a recursive query. For queries, if this bit is set to 0 and the name server contacted is not authoritative for the query, the DNS server returns a list of other name servers that can be contacted for the answer. This is how iterative queries are handled during name resolution.
- **Recursion Available** DNS servers set this flag to 1 to indicate that they can handle recursive queries.
- **Reserved** These 3 bits are reserved, and set to 0.
- **Return Code** A 4-bit field holding the return code. A value of 0 indicates a successful response (for instance, for name queries, this means the answer is in the reply message). A value of 0x3 indicates a name error, which is returned from an authoritative DNS server to indicate that the domain name being queried for does not exist.

DNS Query Question Entries

In a DNS query, the question entry contains the domain name being queried. Figure 17-7 displays the format of a Question entry.

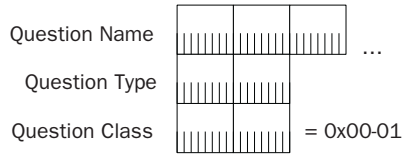


Figure 17-7. *Question entry format.*

The question entry is made up of the following three fields:

- **Question Name** The domain name being queried. The format of this field is discussed below.
- **Question Type** This field indicates the records that should be returned, expressed as a 16-bit integer, as shown in Table 17-4.

Table 17-4. Records Returned for Each Type Value

Type Value	Record(s) Returned
0x01	Host record
0x02	Name server (A) record
0x05	Alias (CNAME) record
0x0C (12)	Reverse-lookup (PTR) record
0x0F (15)	Mail exchanger (MX) record
0x21 (33)	Service (SRV) record
0xFB (251)	Incremental zone transfer (IXFR) record
0xFC (252)	Standard zone transfer (AXFR) record
0xFF (255)	All records

- **Question Class** Normally set to 0x00-01. This represents the IN (Internet) question class.

The Question Name field holds the name of the domain being queried. In DNS, these domain names are expressed as a sequence of labels using a length-value format. The domain kapoho.com, for example, consists of two labels (*kapoho* and *com*). In the Question Name field, the domain name is encoded as a series of length-value pairs consisting of a 1-byte field that indicates the length of the value, followed by the value (the label). The domain kapoho.com, therefore, would be expressed as 0x06kapoho0x03com0x00, where the hexadecimal digits represent the length of each label, the ASCII characters represent the individual labels, and the final 0 indicates the end of the name.

RRs

When a DNS server sends a query response back to a DNS client, the answer, authority, and additional information sections of the DNS message can contain RRs, which answer the question in the question section. Figure 17-8 illustrates the format of these RRs.

The fields in an RR are as follows:

- **RR Name** The DNS domain name held as a variable-length field. The format of this field is the same as the format of the Question Name field, described in the previous section.
- **Record Type** The RR type value, as previously noted.
- **Record Class** The RR class code; there is only one record class used currently: 0x00-01, Internet Class.

- **Time-To-Live** TTL, expressed in seconds held in a 32-bit unsigned field.
- **Resource Data Length** A 2-byte field that indicates the length of the resource data.
- **Resource Data** Variable-length data corresponding to the RR type.

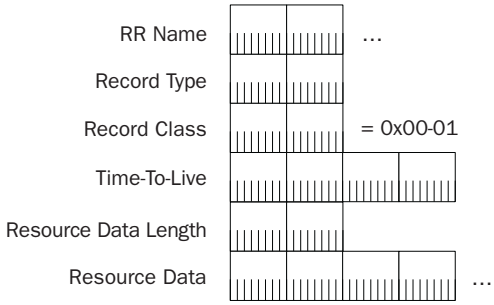


Figure 17-8. *DNS RR format.*

The RR Name field is encoded in the same way as the Question Name field described earlier. However, if the name is already present elsewhere in the DNS message, then a 2-byte field that acts as a pointer to the name already present is used in place of a length-value encoded name. A pointer, instead of an encoded name, is indicated by setting the two high-order bits in the first byte of the RR Name field to 11. The RR Name field is then interpreted as a 16-bit field: the first 2 bits are fixed at 11, and the last 14 bits are used as a byte offset pointer (starting at 0) indicating where the previously stored name is located in the DNS message.

For a simple DNS Query Response message, the RR Name for an Answer RR is the same as the Resource Name for the Question entry, which begins in the 12th byte position (starting at 0) from the beginning of the DNS message. Therefore, the RR Name field is a 2-byte field containing 0xC0-0C. Figure 17-9 shows this relationship.

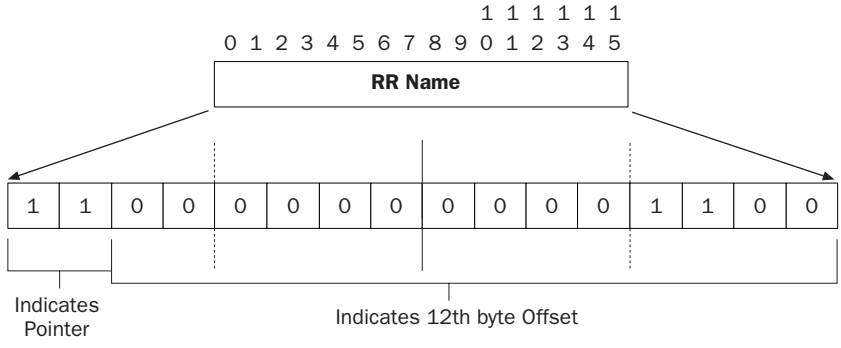


Figure 17-9. *The RR Name as a pointer to a name stored elsewhere in the DNS message.*

DNS Update Message Format

The format of a DNS Update message is very similar to DNS Query messages, and many of the fields are the same. The DNS Update message contains a header defining the update operation to be performed and a set of RRs that contain the update. Figure 17-10 displays the general format of the DNS Update message.

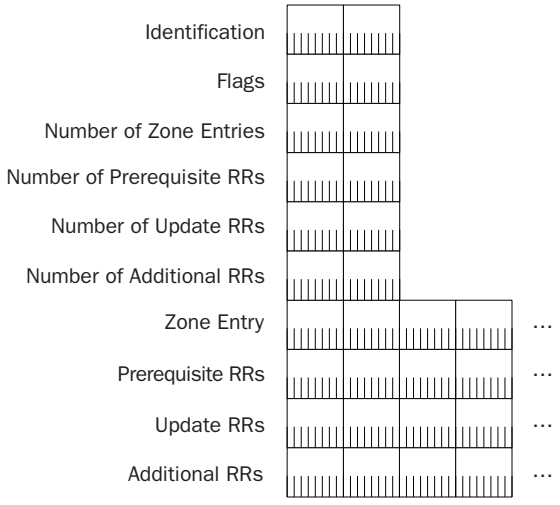


Figure 17-10. *General Update message format.*

DNS Update Message Flags

The DNS Update message has a Flags field, similar to DNS Query messages but with a slightly different format. Figure 17-11 shows the format of the Flags field for DNS Update messages.

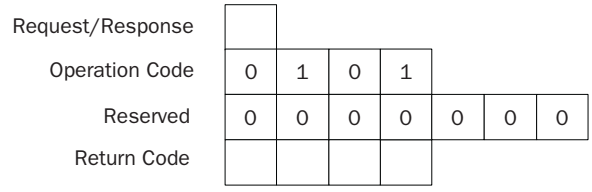


Figure 17-11. *The Flags field of a DNS Update message.*

The DNS Update message flags are used as follows:

- **Request/Response** A 1-bit field, set to 0 to indicate an update request and 1 to indicate an update response.
- **Operation Code** Set to 0x5 for DNS updates.
- **Reserved** These 7 bits are reserved and set to 0.
- **Return Code** For update responses, indicates the result of the query. The defined result codes are shown in Table 17-5.

Table 17-5. Defined Result Return Code Flags for Update Responses

Result Code Value	Meaning
0	No error; the update was successful.
1	Format error; the DNS server was unable to understand the update request.
0x2	The name server encountered an internal failure while processing this request.
0x3	Some name that should exist does not exist.
0x4	The operation code is not supported.
0x5	The name server refuses to perform the operation for policy or security reasons.
0x6	Some name that should not exist does exist.
0x7	Some RR set that should not exist does exist.
0x8	Some RR set that should exist does not exist.
0x9	The server is not authoritative for the zone named in the Zone section.
0xA	A name used in the Prerequisite or Update section is not within the zone denoted by the Zone section.

Name Query Message

A Name Query message uses the DNS message format defined in RFC 1034 and described in the section entitled "DNS Query Message Format," earlier in this chapter. Network Monitor trace 17-1 (Capture 17-01, included in the \Captures folder on the companion CD-ROM) shows an example name query. In this trace, the following fields are set:

- **Query Identifier** Set to a unique number so that the resolver can match the response to the query
- **Flags** Set to indicate a standard query, with recursion if necessary
- **Question Count** Set to 1
- **Question Entry** Set to the domain name to resolve (kona.kapoho.com) and the RR to return (the host A record)

Name Query Response Message

A Name Query Response message is sent in response to a Name Query and is sent using the same query message format as the query response. Network Monitor trace 17-1 (Capture 17-01, included in the \Captures folder on the companion CD-ROM) displays an example of a query response in which the following fields are set:

- **Query Identifier** Set to the unique number set in the query, to allow the resolver to match the response to the original query
- **Flags** Set to indicate a response and a successful lookup
- **Question Count** Set to 1
- **Answer Count** Set to 1
- **Question Entry** Set to the question contained in the query message
- **Answer Entry** The RR requested in the query (the host record, containing the IP address of the queried domain)

Network Monitor trace 17-3 (Capture 17-03, included in the \Captures folder on the companion CD-ROM) shows a slightly different response message. In this trace, a resolver is attempting to resolve the A RR for ns1.kapoho.com. There is no host A RR for this name, but there is an alias (CNAME). In the response, the replying DNS server returns two RRs: the CNAME RR, as well as the A RR for the canonical name (kona.kapoho.com).

Reverse Name Query Message

Reverse Name Query messages use the same message format as normal queries. The only differences in the contents are as follows:

- The domain name being queried is different. For a reverse lookup, the resolver constructs a domain name in the in-addr.arpa domain based on the IP address that is being queried.
- The queried record is a PTR record, rather than an A record.

The Reverse Name Query Response message is also the same as a Query Response message, except that a PTR record, rather than a host record, is returned. A reverse lookup can be seen in Network Monitor trace 17-2, shown earlier in this chapter. In the trace, the resolver is looking for the host name of the host at 10.10.1.52, and thus queries for the domain 52.1.10.10.in-addr.arpa. The Reverse Name Query Response returns the requested PTR record, which shows the host name at this IP address to be kapoholt.kapoho.com.

Name Update Message

Name Update messages use the Name Update message format defined in RFC 2136 and described in the section entitled "DNS Update Message Format," earlier in this chapter. Network Monitor trace 17-4 shows an example of a Name Update message. In this update, the key update message fields are set as follows:

- **Query Identifier** Like query messages, update messages contain an identifier to enable the sender to match the response to the original update.
- **Flags** Set to indicate a request and a dynamic update.
- **Update Zone** Set to 1, and the zone section contains the zone to be updated.
- **Prerequisites Zone** Set to 2, with two prerequisite records specified.
- **Update Zone** Contains the RR that is to be updated.

Name Update Response Message

A Name Update Response message is issued in response to a Name Update Request. Network Monitor trace 17-4 contains an example of this in which the response can be seen to be identical to the request, except that the DNS flags in the message header are set to indicate this is a successful response. If the response had been unsuccessful, the response message would have contained an error code.

Summary

DNS was once an option for most Microsoft Windows NT networks that used NetBIOS names and WINS for most domain operations, and as the basis of file and print sharing. DNS is now a key component in Windows Server 2003 networks and is required in those networks that deploy Windows Server 2003 Active Directory. In this chapter we have examined what DNS is and how it works, including DNS message formats and how the message appears on the wire. The chapter also has shown a number of Network Monitor captures, which are contained on the companion CD-ROM for further study.