

# TCP/IP Utilities Guide

**Microsoft® TCP/IP for  
Windows NT™**

**Microsoft Corporation**



# Contents

Before You Begin .....	v
How to Use This Manual .....	v
Documentation Conventions .....	vi
Finding Further Information .....	vi
<b>Chapter 1 Overview of Microsoft TCP/IP for Windows NT .....</b>	<b>1</b>
What Is TCP/IP for Windows NT? .....	1
How Does TCP/IP Work? .....	2
How TCP Works .....	2
Sequence Numbers, Checksum, and Port ID .....	2
TCP Headers .....	3
How IP Works .....	3
IP Addresses .....	3
Subnet Mask .....	3
Network and Host IDs .....	4
Default Gateway .....	4
Example .....	4
Overview of Windows Sockets .....	6
<b>Chapter 2 Installing and Configuring Microsoft TCP/IP .....</b>	<b>9</b>
Before Installing Microsoft TCP/IP .....	9
Installing and Configuring TCP/IP .....	9
Configuring TCP/IP .....	11
Configuring TCP/IP to Use DNS .....	13
Troubleshooting IP Connections .....	16
Ping .....	16
<b>Chapter 3 Understanding Name Resolution .....</b>	<b>17</b>
Dynamic Resolution .....	17
Host Files .....	17
HOSTS .....	18
LMHOSTS .....	18

<b>Chapter 4   Networking Concepts for TCP/IP</b>	<b>19</b>
Internet Protocol Suite	19
Transmission Control Protocol and Internet Protocol	20
User Datagram Protocol (UDP)	21
Address Resolution Protocol and Internet Control Message Protocol	21
IP Addressing	22
Subnet Masks	23
Routing	24
TCP/IP with Windows NT	25
NetBIOS over TCP/IP	25
NetBIOS Name Resolution	26
<b>Chapter 5   SNMP</b>	<b>27</b>
Overview	27
How the SNMP Service Works	27
SNMP Installation and Configuration	28
Performance Monitoring	31
LAN Manager MIB II Objects	31
Common Group	31
Server Group	32
Workstation Group	35
Domain Group	35
<b>Chapter 6   Utilities Reference</b>	<b>37</b>
arp	37
finger	38
ftp	38
Ftp Subcommands	39
hostname	43
nbtstat	43
netstat	46
ping	49
rcp	50
rexec	51
route	52
rsh	53
tftp	54
telnet	54

## Before You Begin

The *Microsoft® TCP/IP for Windows NT™ User's Guide* describes how to install, configure, and troubleshoot Microsoft TCP/IP on a computer running the Microsoft Windows NT operating system. It also provides a reference for the TCP/IP utilities.

This manual assumes that you are familiar with the Microsoft Windows NT operating system. If you are not familiar with this product, refer to your Microsoft Windows NT documentation set.

## How to Use This Manual

This manual contains the following chapters.

**Chapter 1, “Overview of Microsoft TCP/IP for Windows NT”**

Provides an overview of the Microsoft TCP/IP network protocol and IP addresses.

**Chapter 2, “Installing and Configuring Microsoft TCP/IP”**

Describes the process for installing and configuring Microsoft TCP/IP on a workstation running the Microsoft Windows NT operating system.

**Chapter 3, “Understanding Name Resolution”**

Describes how the HOSTS and LMHOSTS host table files enable your workstation to access resources on different TCP/IP networks.

**Chapter 4, “Networking Concepts for TCP/IP”**

Introduces key TCP/IP networking concepts for network administrators interested in a technical discussion of TCP/IP.

**Chapter 5, “SNMP”**

Describes the Simple Network Management Protocol (SNMP), how to configure SNMP, and lists objects.

**Chapter 6, “Utilities Reference”**

Describes the TCP/IP utilities and provides syntax, notes, and examples.

# Documentation Conventions

This manual uses several type styles and special characters.

Convention	Use
<b>bold</b>	Represents commands, command options, and file entries. Type bold words exactly as they appear (for example, <b>net use</b> ).
<i>italic</i>	Introduces new terms and represents variables. For example, the variable <i>computername</i> indicates that you type the name of a workstation or a server.
monospace	Represents examples, screen displays, and error messages.
ALL UPPERCASE	Represents filenames and paths. (You can, however, type entries in uppercase or lowercase letters, or a combination of the two.)
SMALL CAPITALS	Represent key names (for example, CTRL, ENTER, and F2).
[brackets]	Enclose optional items in syntax statements. For example, [ <i>password</i> ] indicates that you can choose to type a password with the command. Type only the information within the brackets, not the brackets themselves.
►	Indicates a procedure.

# Finding Further Information

For a more technical discussion of the topics mentioned in this manual, refer to the following texts and articles:

Calbaum, M., et al. “Untangling the Windows Sockets API,” *Dr. Dobb’s Journal*, February: 66–71, 1993.

Comer, D. *Internetworking with TCP/IP Volume I: Principles, Protocols, and Architecture*. Second edition. Englewood Cliffs, N.J.: Prentice Hall, 1991.

Comer, D. and D. Stevens. *Internetworking with TCP/IP Volume II: Design, Implementation, and Internals*. Englewood Cliffs, N.J.: Prentice Hall, 1991.

Comer, D. and D. Stevens. *Internetworking with TCP/IP Volume III: Client-Server Programming and Applications*, Englewood Cliffs, N.J.: Prentice Hall, 1991.

Hall, M., et al. *Windows Sockets: An Open Interface for Network Programming Under Microsoft Windows*, Version 1.1, Revision A, 1993.

Leffler, S. et al. *An Advanced 4.3 BSD Interprocess Communication Tutorial*.

Stevens, W.R. *UNIX Network Programming*, Englewood Cliffs, N.J.: Prentice Hall, 1990.

Volkman, Victor R. "Plug into TCP/IP with Windows Sockets." *Windows/DOS Developer's Journal*, December:6–18, 1992.





---

CHAPTER 1

# Overview of Microsoft TCP/IP for Windows NT

This chapter introduces Microsoft TCP/IP for Windows NT. It provides background material on basic TCP/IP concepts and gives you the information you need before installing Microsoft TCP/IP for Windows NT. (For more detailed information on TCP/IP and its integration with Microsoft Windows NT and other networking products, see Chapter 4, “Networking Concepts for TCP/IP.”)

*Transmission Control Protocol/Internet Protocol* (TCP/IP) is a networking protocol that provides communication across interconnected networks, between computers with diverse hardware architectures and various operating systems. TCP/IP can be used with Windows NT or to connect to other Windows networking products or non-Microsoft (for example, UNIX®) hosts.

## What Is TCP/IP for Windows NT?

The TCP/IP protocol family is a standard set of networking protocols, or rules, that govern how data is passed between systems on a network. Microsoft TCP/IP for Windows NT enables enterprise networking and connectivity on your Windows NT-based computer. Adding TCP/IP to a Windows NT configuration offers the following advantages:

- A standard, routable, enterprise networking protocol for Windows for Workgroup services. TCP/IP is viewed as the most complete and accepted networking protocol available. Virtually all modern operating systems offer TCP/IP support, and most large networks rely on TCP/IP for all their network traffic.
- An architecture that facilitates connection to foreign systems. Because most operating systems offer TCP/IP, many standard utilities have been designed to access and transfer data between heterogeneous environments. Examples of such standards include File Transfer Protocol (FTP) and Telnet (Terminal Emulation Protocol). The Windows Sockets interface offers compatibility with many foreign host connectivity products. Several applications vendors support this Application Programming Interface (API) standard.

- A robust, cross-platform client-server framework. TCP/IP for Windows NT offers the Windows Sockets interface, which is ideal for developing client-server applications. A Windows Sockets application developed to be used with Microsoft TCP/IP will be able to run other vendors' Windows Sockets-compliant stacks, as well.

## How Does TCP/IP Work?

The name TCP/IP is somewhat confusing because TCP (Transmission Control Protocol) and IP (Internet Protocol) are really only two protocols in the family of Internet protocols. Over time, however, TCP/IP has been used in industry to denote the family of common Internet protocols.

The following sections briefly explain how the TCP and IP protocols work. For a more detailed explanation, see Chapter 4, “Networking Concepts for TCP/IP.” For a full explanation, see *Internetworking with TCP/IP, Volume I* (listed in “Finding Further Information”).

## How TCP Works

TCP is a reliable, connection-oriented protocol. *Connection-oriented* implies that TCP first establishes a connection between the two systems that intend to exchange data. Since most networks are built on *shared media* (for example, several systems sharing the same cabling), it is necessary to break chunks of data into manageable pieces so that no two communicating systems monopolize the network. These pieces are called *packets*. When an application sends a message to TCP for transmission, TCP breaks the message into packets, sized appropriately for the network, and sends them over the network.

### Sequence Numbers, Checksum, and Port ID

Because a single message is often broken into many packets, TCP marks these packets with *sequence numbers* before sending them. The sequence numbers allow the receiving system to properly reassemble the packets into the original message. Being able to reassemble the original message is not enough —the accuracy of the data must also be verified. TCP does this by computing a *checksum*. A checksum is a simple mathematical computation applied, by the sender, to the data contained in the TCP packet. The recipient then does the same calculation on the received data and compares the result with the checksum that the sender computed. If the results match, the recipient sends an acknowledgment (ACK). If the results do *not* match, the recipient asks the sender to resend the packet. Finally, TCP uses *port IDs* to specify which application running on the system is sending or receiving the data.

## TCP Headers

The port ID, checksum, and sequence number are inserted into the TCP packet in a special section called the *header*. The header is at the beginning of the packet containing this and other “control” information for TCP.

## How IP Works

IP is the *messenger protocol* of TCP/IP. The IP protocol, much simpler than TCP, basically addresses and sends packets. IP relies on three pieces of information, which you provide, to receive and deliver packets successfully: *IP address*, *subnet mask*, and *default gateway*.

## IP Addresses

The *IP address* identifies your system on the TCP/IP network. IP addresses are 32-bit addresses that are globally unique on a network. They are generally represented in *dotted decimal notation*. Dotted decimal notation (explained in Chapter 4, “Networking Concepts for TCP/IP”) separates the 4 bytes of the address with periods. An IP address looks like this:

102.54.94.97

Although an IP address is a single value, it really contains two pieces of information:

- Your system’s network ID
- Your system’s host (or system) ID

## Subnet Mask

The *subnet mask*, also represented in dotted decimal notation, is used to extract these two values from your IP address. The value of the subnet mask is determined by setting the network ID bits of the IP address to 1’s and the host ID bits to 0’s. The result allows TCP/IP to determine the host and network IDs of the local workstation, as shown in Table 1.1.

**Table 1.1 Understanding an IP Address**

When the <i>IP address</i> is	102.54.94.97	(specified by the user)
And the <i>subnet mask</i> is	255.255.0.0	(specified by the user)
The <i>network ID</i> is	102.54	(IP address and subnet mask)
And the <i>host ID</i> is	94.97	(IP address and subnet mask)

## Network and Host IDs

The *network ID* identifies a group of systems that are all located on the same physical network. In *internetworks* (networks formed by a collection of networks), there are as many unique network IDs as there are networks. TCP/IP networks are connected by *routers* (or *gateways*), which have knowledge of the networks that are connected in the internet. The *host ID* identifies your system within a particular network ID.

## Default Gateway

The *default gateway* is needed only for systems that are part of an internet. When IP gets ready to send a packet on the wire, it inserts the local (source) IP address and destination address of the packet in the IP header, and verifies that the network ID of the destination matches the source. If they match, the packet is sent directly to the destination system on the local network. If the network IDs do not match, the packet is forwarded to the default gateway for delivery. Since the default gateway has knowledge of the network IDs of the other networks in the internet, it forwards the packet to other gateways until the packet is eventually delivered to a gateway connected to the specified destination. This process is known as *routing*.

## Example

This example shows how TCP/IP might deliver a message, using an analogy with the U.S. postal system to describe how these protocols work.

Steve (source host ID) in Seattle (source network ID) wants to send a two-page letter (message) to Dana (destination host ID) in Dartmouth (destination network ID). There is a limit to the length of the message that can be sent in a single letter (maximum transmission unit, or MTU).

First, Steve establishes the connection by writing a short note to Dana: “Dana, I am going to send you a two-page letter now via mail. Is this okay? -Steve” Steve then puts the letter in an envelope (IP packet) and addresses it to Dana in Dartmouth (destination IP address).

Steve’s mail carrier (Steve’s default gateway) picks up the letter, doesn’t know where Dartmouth is, and forwards it to the Seattle post office (gateway). From Seattle, the message goes (routes) to the Dartmouth post office (Dana’s default gateway). The Dartmouth mail carrier delivers it to Dana.

Dana writes back (ACKs) to Steve saying, “Okay, I’m ready and waiting for your letter.” She addresses the envelope using the return address from the envelope (IP packet) to Steve in Seattle. Dana’s mail carrier then carries her reply to the Dartmouth post office, and it travels back the same way Steve’s letter came.

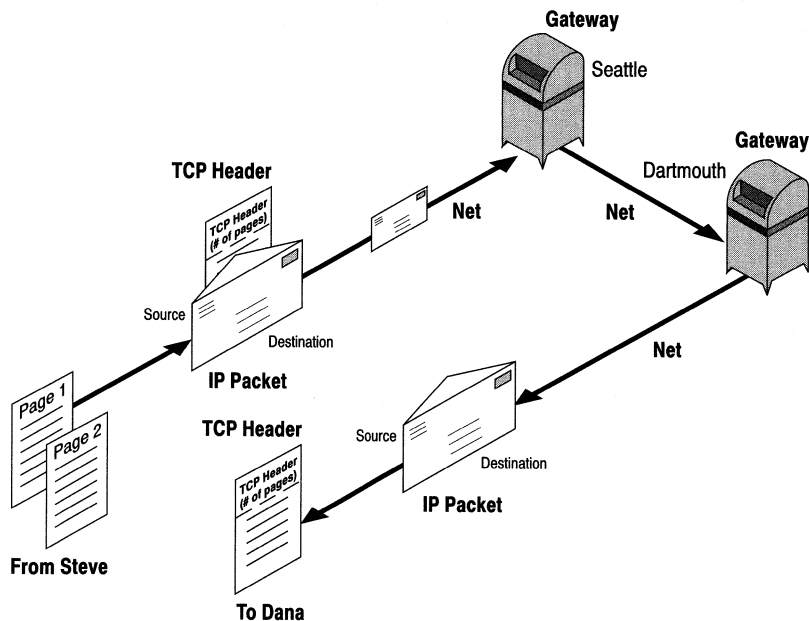
Steve realizes that his letter (message) is too long to fit in a single envelope (IP packet), tears off the first page (TCP packet) and writes “1 of 2” in the margin (TCP header), puts it into the envelope addressed to Dana (IP packet), and mails it.

The first page is successfully routed to Dana, who opens the envelope and reads the contents. But the letter is incomplete, so Dana mails a quick note (ACK) to Steve: “Steve, I received the first page of the message, but I haven’t seen any since. Are you okay?”

Steve reads Dana’s letter and immediately takes the second page (TCP packet), writes “2 of 2” in the margin (TCP header), puts it in an envelope addressed to Dana (IP packet), and mails it.

The second note is also successfully routed to Dana, so she replies (ACKs) with one final note: “Good letter, Steve. Thanks.”

The following figure shows how Steve’s message traveled.



# Overview of Windows Sockets

Microsoft TCP/IP for Windows NT includes support for Windows Sockets. A socket provides an end point to a connection; two sockets form a complete path. A socket works as a bidirectional pipe for incoming and outgoing data. The Windows Sockets API is a networking API tailored for use by programmers using the Microsoft Windows operating system. Windows Sockets is a public specification based on Berkeley UNIX sockets and aims to:

- Provide a familiar networking API to programmers using Windows or UNIX.
  - Offer binary compatibility between heterogeneous Windows-based TCP/IP stack and utilities vendors.
  - Support both connection-oriented and connectionless protocols.
- **To get a copy of the Windows Sockets specification via anonymous FTP:**
1. Type  
**ftp microdyne.com**
  2. Log on as **anonymous**.
  3. Type your electronic mail address for the *password*.
  4. Type  
**cd /pub/winsock/winsock-1.1**
  5. Choose the file with the format you want [ASCII (.TXT), PostScript® (.PS), or Microsoft Word for Windows (.DOC)], and then type  
**get winsock.ext**
- Or
1. Type  
**ftp vax.ftp.com**
  2. Log on as **anonymous**.
  3. Type your electronic mail address for the *password*.
  4. Type  
**cd /pub/winsockapi/winsock-1.1**
  5. Choose the file with the format you want [ASCII (.TXT), PostScript (.PS), or Microsoft Word for Windows (.DOC)], and then type  
**get winsock.ext**

- ▶ **To get a copy of the Windows Sockets specification from CompuServe®:**
  1. Type  
**go msl**
  2. Browse using the keywords **windows sockets**.
  3. Choose the file with the format you want [ASCII (.TXT), PostScript (.PS), or Microsoft Word for Windows (.DOC)], and then type  
**get winsock.ext**





---

C H A P T E R 2

# Installing and Configuring Microsoft TCP/IP

This chapter describes how to install TCP/IP for Windows NT and configure the protocol on your computer.

The TCP/IP protocol family is not installed as part of the Windows NT product. This chapter assumes that Windows NT has already been successfully installed on your system. You must be a member of the Administrator group to install and configure TCP/IP.

## Before Installing Microsoft TCP/IP

You need to know the following information before you install Microsoft TCP/IP on your computer:

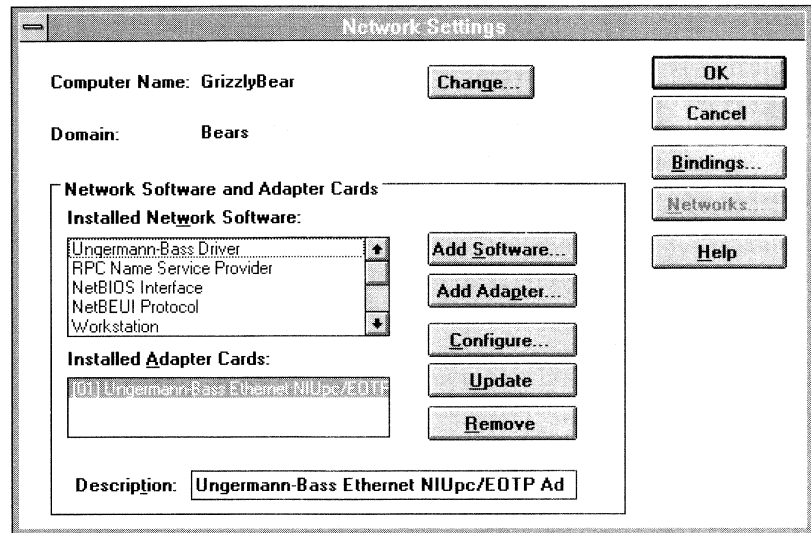
- Default gateway.
- IP address.
- Subnet mask.
- Whether to enable Domain Name Service (DNS). (See Chapter 4, “Networking Concepts for TCP/IP.”)

## Installing and Configuring TCP/IP

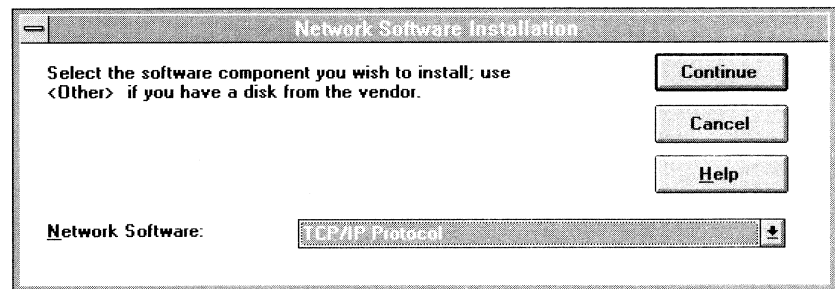
You must be a member of the Administrator group to install and configure TCP/IP.

► **To install Microsoft TCP/IP on a computer with Windows NT**

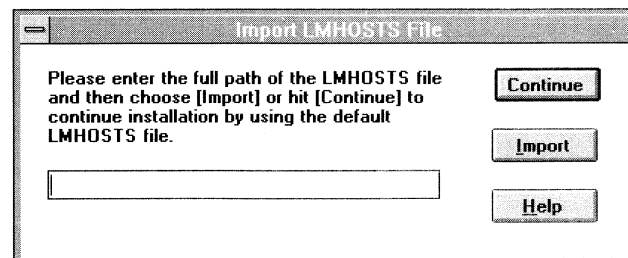
1. Start the Network utility in the Control Panel. The Network Settings dialog box appears.



2. Choose the Add Software button. The Network Software Installation dialog box appears.



3. Select TCP/IP Protocol in the Network Software list box, and choose the Continue button.
4. The Windows NT Setup dialog box appears and prompts for the full path to the Windows NT distribution files. Provide the appropriate location, and choose the Continue button. The Import LMHOSTS File dialog box appears.



5. Enter the path of the LMHOSTS file and choose the Import button, or choose the Continue button to use the default LMHOSTS file. The network administrator should provide this information, or see Chapter 4, “Networking Concepts for TCP/IP,” for more information about the LMHOSTS file.

The Microsoft TCP/IP for Windows NT software is copied to your computer, and the Network Setting dialog box reappears. Choose the OK button. The TCP/IP Configuration dialog box appears.

6. Continue with the configuration procedure, as described in the next section, “Configuring TCP/IP.” You can press the ESC key and configure TCP/IP later, but TCP/IP must be configured to operate.

## Configuring TCP/IP

After the Microsoft TCP/IP protocol software is installed on your computer, you must configure it with valid addressing information for TCP/IP to operate.

### To configure or reconfigure the TCP/IP protocol

1. The Configuration dialog box appears automatically after the TCP/IP software is installed on your computer.

If you did not configure TCP/IP when you installed it, or are reconfiguring TCP/IP, start the Network utility in the Control Panel. In the Installed Adapter Cards box, select the adapter card whose TCP/IP protocol you want to configure. In the Installed Network Software list box, select TCP/IP Protocol, and choose the Configure button. The TCP/IP Configuration dialog box appears.

**TCP/IP Configuration**

**General Information**

Default Gateway: 102 .54 .0 .1

**Adapter:** [01] Ungermann-Bass Ethernet NIUpc.

IP Address: 102 .54 .94 .97

Subnet Mask: 255 .255 .0 .0

**Descriptions:** Ungermann-Bass Ethernet

**Windows Networking on TCP/IP (NetBIOS)**

Windows Networking Adapter: [01] Ungermann-Bass Ethernet

Scope ID for Windows Networking:

OK Cancel Connectivity... Help

2. Type in the correct values in the text boxes for Default Gateway, IP Address, and Subnet Mask. These are generally the only values that must be configured. Descriptions for each field follow.

#### Default Gateway

Specifies the IP address of the default gateway used to forward packets to other networks or subnets. This parameter is required only for nodes on internetworks. If this parameter is not provided, IP functionality will be limited to the local subnet unless a route is specified with the **route** command. The network administrator should provide the correct value for this parameter.

#### IP Address

Specifies the IP address associated with your local computer, or if more than one network card is installed in the computer, the network adapter card listed in the Adapter box.

Duplicate IP addresses on a network might cause some systems on the network to “hang” or function unpredictably. The network administrator should provide the correct value for this parameter. (For more information about IP addresses, see Chapters 1 and 4.)

#### Subnet Mask

Specifies the subnet mask associated with the adapter to which TCP/IP is bound. Each interface used by TCP/IP must have a subnet mask configured. This allows the computer to separate the IP address into host and network IDs. The subnet mask defaults to the appropriate value shown in Table 4.2 (in Chapter 4). The network administrator should provide the correct value for this parameter.

3. If necessary, select a TCP/IP adapter, and provide a Scope ID in the Windows Networking on TCP/IP (NetBIOS) box. A description for this field follows.

#### Scope ID

Specifies the NetBIOS scope parameter for the NetBIOS over TCP/IP (NBT) module. To be able to communicate, all computers on a NetBIOS network must have the same scope ID. The network administrator should provide the correct value for this parameter, but you can generally leave this value blank.

4. To enable the Domain Name Service (DNS), choose the Connectivity button. Continue with the configuration procedure, as described in the next section.

If you do not want to use DNS and rely on the HOSTS file for name resolution, choose the OK button. The Network Settings dialog box returns. Choose the OK button. Microsoft TCP/IP has been configured. You must reboot the computer for the configuration to take effect.

# Configuring TCP/IP to Use DNS

Although TCP/IP uses IP addresses to identify and reach computers, users typically prefer using names. The Domain Name Service (DNS) is a naming service generally used in the UNIX networking community to provide standard naming conventions for IP workstations. (For details about DNS, see Chapter 3, “Understanding Name Resolution.”)

Microsoft Windows networking provides a dynamic naming system of its own using NBT; however, TCP/IP applications, such as **ftp** and **telnet** can use DNS to use names instead of IP addresses when connecting to foreign hosts.

You must configure how your computer will use DNS. TCP/IP must be installed and configured to set up the connectivity options.

## To configure or reconfigure TCP/IP connectivity

1. Start the Network utility in the Control Panel. In the Installed Adapter Cards box, select the adapter card whose TCP/IP NetBIOS protocol you want to configure. In the Installed Network Software list box, select TCP/IP Protocol and choose the Configure button. The TCP/IP Configuration dialog box appears.
2. Choose the Connectivity button. The TCP/IP Connectivity Configuration dialog box appears.

**TCP/IP - Connectivity Configuration**

Host Name:  TCP Domain Name:

**Name Resolution Search Order**

☐ DNS Only ☐ Host File First, Then DNS  
☒ DNS First, Then Hosts File ☐ Hosts File Only

**Domain Name Service (DNS) Search Order**

102.54.0.1  
102.54.0.3  
102.54.0.10

Order

**Domain Search Order**

bruins.com  
gts23.br.usw.com  
u.washington.edu

Order

3. Type in the Host Name (usually your computer name) and TCP Domain Name (usually an organization name, a period, and an extension that indicates the type of organization, such as .COM for commercial businesses or .EDU for educational institutions). Descriptions for each field follow.

#### Hostname

Specifies the host name for this computer. The host name is used to identify the local computer by name for authentication by utilities. Other TCP/IP-based utilities and applications can use this value to learn the name of the local computer. This value defaults to the Windows NT computer name, and it can be altered without affecting the computer name's value. The Hostenane is optional.

#### TCP Domain Name

Identifies your group in the DNS hierarchical naming convention, with descending levels of detail. The fully qualified domain name (FQDN) for the computer is the host name followed by a period (.) followed by the domain name, for example, **rhino.microsoft.com**, where **rhino** is the hostname and **microsoft.com** is the domain name. During DNS queries, the local domain name is appended to short names. Specifying a TCP Domain Name is optional. The DNS domain is not the same as a LAN Manager domain.

4. Select how you want your computer to search for name resolution in the Name Resolution Search Order dialog box.

#### Name Resolution Search Order

Determines when to use DNS host name resolution. The Domain Name Resolver (DNR) software is loaded at startup and is used to resolve host names, in conjunction with the local HOSTS file. This box allows you to specify the order and which method to use when resolving names.

5. Type the IP address of a server that will provide name resolution. Choose the Add button, and the IP address will move to the list on the right.

You can add up to three IP addresses. The servers at the IP addresses will be queried in the order listed. To change the order of the IP addresses, select an IP address to move, and use the up and down arrow buttons. To remove IP addresses, select the address to remove, and choose the Remove button.

#### Domain Name Service (DNS) Search Order

Determines the servers and the order that will provide name resolution.

6. Type the names of the domains in which to search for host name resolution.  
You can add up to six domain names. To change the order of the domains, select a domain to move, and use the up and down arrow buttons. To remove domain names, select the domain name to remove and choose the Remove button.

#### Domain Search Order

Determines the domains in which to search for host name resolution.

7. When you are done setting connectivity values, choose the OK button.  
The Microsoft TCP/IP-Connectivity Configuration dialog box closes, and the TCP/IP Configuration dialog box appears.
8. Choose the OK button. The Microsoft TCP/IP Configuration dialog box closes, and the Network Settings dialog box returns.
9. To complete TCP/IP Installation and Configuration, choose the OK button.  
DNS resolution has been configured.

A dialog box appears informing you that you must restart the computer so that the change you made will take effect. You can choose the Yes button to reboot the computer.

If you choose the No button, you can continue to work, but TCP/IP will not work as you just configured it until the computer is restarted.

# Troubleshooting IP Connections

If you have trouble installing Microsoft TCP/IP on your computer, follow the suggestions in the error messages. You can also use the **ping** utility, as described below.

## Ping

The **ping** utility can isolate network hardware problems and incompatible configurations by allowing you to verify a physical connection to a remote computer.

Use the **ping** utility to test both the host name and the IP address of the host. If the IP address is verified but the host name is not, you have a name resolution problem. In this case, be sure that the host name you are querying is in either the local HOSTS file or in the DNS database.

For the syntax and description of the **ping** command, see Chapter 6, “Utilities Reference.” For information about the HOSTS file, see Chapter 3, “Understanding Name Resolution.”



---

C H A P T E R 3

# Understanding Name Resolution

This chapter describes how the HOSTS and LMHOSTS files work, and how to use them to access resources on a different TCP/IP network.

## Dynamic Resolution

TCP/IP connections are established based on IP addresses, but because users generally prefer to use names, a mechanism for mapping names to IP addresses is useful.

The Domain Name Service (DNS) provides a way to look up name mappings when connecting your workstation to foreign hosts via applications such as FTP. To use the DNS, you must enable the domain name resolver (DNR) module on the Microsoft TCP/IP - Connectivity Configuration dialog box. (See Chapter 2, “Installing and Configuring Microsoft TCP/IP,” for details.)

NBT (NetBIOS over TCP/IP) provides a dynamic way for locating Windows NT-based hosts on the local network. (This mechanism is described in more detail in Chapter 4, “Networking Concepts for TCP/IP.”) For Windows NT-based systems located on remote subnets, the LMHOSTS file is needed to provide computername-to-IP address mappings.

## Host Files

The HOSTS and LMHOSTS files contain lists of known IP addresses. Each of these files is also known as a *host table*. The LMHOSTS and HOSTS files are located in the \WINNT\SYSTEM32\DRIVERS\ETC directory under your WINNT directory and can be edited using any ASCII editor (such as Notepad or Edit, which are part of Windows NT).

When you use a host table file, be sure to keep it up to date and organized. Follow these guidelines:

- Update the host table file whenever a workstation is changed, added to, or removed from the network.
- Because host table files are searched one line at a time from the beginning, list remote workstations in priority order, with the ones used most often at the top of the file. This arrangement increases the speed of searches for the most often used entries.

## HOSTS

If the Domain Name Resolver (DNR) is not used or fails to match a host name to an IP address, Microsoft TCP/IP searches the local host table file, HOSTS, for mappings of IP addresses to host names of remote computers. The HOSTS file is loaded into the workstation's memory when the workstation is started.

The HOSTS file format is the same as the format for host tables in the 4.3 BSD (Berkeley Software Distribution) UNIX */etc/hosts* file. For example, the entry for a node with an address of 192.102.73.6 and a host name of MIS.HOST.COM looks like this:

```
192.102.73.6      mis.host.com
```

A sample HOSTS file is created when you install Microsoft TCP/IP for Windows NT. Edit this file to include remote host names and their IP addresses for each computer with which you will communicate.

## LMHOSTS

The LMHOSTS file is a local text file that maps IP addresses to NetBIOS names of remote servers with which you want to communicate.

For example, the host table file entry for a node with an address of 192.45.36.5 and a host name of CPQ386 looks like this:

```
192.45.36.5      CPQ386
```

The LMHOSTS file format is the same as the format for host tables in the 4.2 BSD (Berkeley Software Distribution) UNIX */etc/hosts* file.

When using the Replication service, LMHOSTS entries are required on import and export servers for any computers on different subnets participating in the replication.

The LMHOSTS file is read when the workstation is started and stored in a system cache for later access. When a name must be resolved, NBT checks this cache before doing a b-node name discovery. (For details, see Chapter 4, "Networking Concepts for TCP/IP.")

To update the LMHOSTS file and cache, use an ASCII editor to edit the LMHOSTS file and issue the command **nbtstat -r**.

---

C H A P T E R 4

# Networking Concepts for TCP/IP

This chapter explains in some detail the various components of the Internet protocol suite, IP addressing, subnet masks, routing, and NetBIOS over TCP/IP. For additional information about any of the topics discussed here, see *Internetworking with TCP/IP, Volume I*, by Douglas E. Comer (Prentice Hall, 1991).

## Internet Protocol Suite

TCP/IP refers to the Internet suite of protocols. It includes a set of standards that specify how computers communicate and gives conventions for connecting networks and routing traffic through the connections. It is used to connect the Internet—a worldwide internetwork connecting universities, research labs, Department of Defense installations, and corporations. (According to convention, “Internet” is capitalized when referring to the worldwide, connected internet.)

The Internet protocols are a result of a Defense Advanced Research Projects Agency (DARPA) research project on network interconnection in the late 1970s. It was mandated on all United States defense long-haul networks in 1983 but was not widely accepted until the integration with 4.2 BSD (Berkeley Software Distribution) UNIX. The popularity of TCP/IP is based on:

- Robust client-server framework.

TCP/IP is an excellent client-server application platform, especially in wide-area network (WAN) environments.

- Information sharing

Thousands of academic, defense, scientific, and commercial organizations share data, electronic mail, and services on the connected Internet using TCP/IP.

- General availability

Implementations of TCP/IP are available on nearly every popular computer operating system. Source code is widely available for many implementations. Additionally, bridge, router, and network analyzer vendors all offer support for the TCP/IP protocol family within their products.

The following discussion introduces the components of the IP protocol suite. Although many of the details discussed are transparent to the user, knowledge of the architecture and interaction between the components is useful.

## Transmission Control Protocol and Internet Protocol

Transmission Control Protocol (TCP) and Internet Protocol (IP) are only two members of the IP protocol suite. IP is a protocol that provides packet delivery for all of the other protocols within the TCP/IP family. It provides a best-effort, connectionless delivery system for computer data. That is, IP packets are not guaranteed to arrive at their destination, nor are they guaranteed to be received in the sequence in which they were sent. The protocol's checksum feature confirms only the IP header's integrity. Thus, responsibility for the data contained within the IP packet (and the sequencing) is assured only by using higher-level protocols.

The most common higher-level IP protocol is TCP. TCP supplies a reliable, connection-based protocol over (or encapsulated within) IP. TCP guarantees the delivery of packets, ensures proper sequencing of the data, and provides a checksum feature that validates both the packet header and its data for accuracy. In the event that IP corrupts or loses a TCP/IP packet, TCP is responsible for retransmitting the faulty packet(s). This reliability defines TCP/IP as the protocol of choice for session-based data transmission, client-server applications, and critical services such as electronic mail.

This reliability does not come without a price. TCP headers require the use of additional bits to provide proper sequencing information, as well as a mandatory checksum to ensure reliability of both the TCP header and the packet data. To guarantee successful data delivery, the protocol also requires the recipient to acknowledge successful receipt of data. Such acknowledgments (or ACKs) generate additional network traffic, diminishing the level of data throughput in favor of reliability. To reduce the impact on performance, TCP implements a throttle mechanism that allows the required frequency of ACKs to vary with the reliability of the data link. This permits highly reliable connections to use fewer ACKs and less computing power.

## User Datagram Protocol (UDP)

If reliability is not essential, the TCP complement, *user datagram protocol* (UDP), offers a connectionless datagram service that guarantees neither delivery nor correct sequencing of delivered packets (much like IP). Higher-level protocols or applications provide many reliability mechanisms in addition to UDP/IP. UDP data checksums are optional, providing a manner to exchange data over highly reliable networks without unnecessarily consuming processing time or network resources. When UDP checksums are used, they validate both header and data. ACKs are also not enforced by the UDP protocol; this is left to higher-level protocols.

## Address Resolution Protocol and Internet Control Message Protocol

Although not directly related to the transport of user or application data, two other protocols in the IP suite perform important functions: *address resolution protocol* (ARP) and *internet control message protocol* (ICMP). These two protocols are maintenance protocols that support the IP framework and are generally invisible to users and to applications.

Although IP packets contain both source and destination IP addresses, the hardware address of the destination node must also be known. (This section assumes that the transmission type is Ethernet. Ethernet adapters contain a 48-bit, globally unique address in permanent memory.) IP can acquire a node's hardware address by broadcasting a special inquiry packet (an *ARP request packet*) containing the IP address of the node with which it is attempting to communicate. All of the ARP-enabled nodes on the IP network detect these broadcasts, and the node that owns the IP address in question replies by sending its hardware address to the requesting node in an ARP reply packet. The hardware/IP address mapping is then stored in the requesting node's ARP cache for subsequent use. Since the ARP reply can also be broadcast to the network, it is likely that other nodes on the network can use this information to update their own ARP caches.

ICMP allows two nodes on an IP network to share IP status and error information. This information can be used by higher-level protocols to recover from transmission problems or by network administrators to detect network trouble. Although ICMP packets are encapsulated within IP packets, they are not considered to be a higher-level protocol (ICMP is required in every TCP/IP implementation). The **ping** utility makes use of the ICMP *echo request* and *echo reply* packets to determine whether or not a particular IP node on a network is functional. This is useful for diagnosing IP network or gateway failures.

# IP Addressing

Every host interface on a TCP/IP network is identified by an IP address. This address is used to identify a node on a network; it also specifies routing information in an internetwork. IP addresses are 32-bit values typically represented in dotted decimal notation. Dotted decimal notation depicts each octet (or byte) of an IP address by its decimal value, separating each by a period, as in 102.54.94.97. IP addresses are used to provide nodes on a network with a unique address without relying on the underlying hardware to ensure unique addressing.

---

**Note** Since IP addresses identify nodes on an interconnected network, each node on the internet must be assigned a unique IP address.

---

Although represented as a single value, IP addresses provide two pieces of information: the network ID and the host ID for a node. The *network ID* (which must be unique among all networks within a connected internet) specifies the network to which a node is attached. The *host ID* (which is unique among nodes within a network) identifies the node within its network. Networks that connect to the public Internet must obtain an official network ID from Defense Data Network-Network Information Center (DDN-NIC) to protect IP network ID uniqueness. Once assigned a network ID, the local network administrator must assign unique host IDs for computers within the network. Although private networks that are not connected to the Internet can choose to use their own network identifier, obtaining a valid network ID from DDN-NIC allows a private network to connect to the Internet in the future without reassigning addresses.

The Internet community has defined address *classes* to accommodate networks of varying sizes. Each network class is easily derived by the first octet (byte) of its IP address. Table 4.1 summarizes the relationship between the first octet of a given address and its host and network ID fields. It also identifies the total number of network and host IDs for each address class that participates in the Internet addressing scheme. This sample uses IP address w.x.y.z.

**Table 4.1 IP Address Classes**

Class	w values (inclusive)	Net ID	Host ID	Available nets	Available hosts/net
A	1–126	w	x.y.z	126	16,777,214
B	128–191	w.x	y.z	16,384	65,534
C	192–223	w.x.y	z	2,097,151	254

---

**Note** The network address 127 is reserved for loopback testing and interprocess communication on the local computer; it is not a network address.

---

A node uses the network and host IDs to determine which packets it should receive or ignore and to determine the scope of the transmissions it produces (only nodes with the same network ID accept one another's IP-level broadcasts). Since the sender's IP address is included in every outgoing IP packet, it is useful for the receiving node to derive the originating network and host ID from the IP address field. This is accomplished using *subnet masks*.

## Subnet Masks

Subnet masks are 32-bit values that allow the recipient of IP packets to distinguish the network ID portion of the IP address from the host ID. Like an IP address, the value of a subnet mask is frequently represented in dotted decimal notation. Subnet masks are determined by assigning 1's to bits that belong to the network ID and 0's to the bits that belong to the host ID. Once the bits are in place, the 32-bit value is converted to dotted decimal notation (as shown in Table 4.2).

**Table 4.2 Examples of Subnet Masks for Standard IP Address Classes**

Class	Default subnet mask
A	255.0.0.0
B	255.255.0.0
C	255.255.255.0

Although configuring a host with a subnet mask might seem redundant after examining this table, subnet masks are also used to further segment an assigned network ID among several local networks. For example, suppose a network is assigned the Class-B network address 144.100. Table 4.1 shows that this is one of over 16,000 Class-B addresses capable of serving more than 65,000 nodes. However, the worldwide corporate network to which this ID is assigned is composed of 12 international LANs with 75 to 100 nodes each.

Instead of applying for 11 more network IDs, it is better to use subnetting to make more effective use of the assigned ID 144.100. The third octet of the IP address can be used as a *subnet ID*, to define the subnet mask 255.255.255.0. This effectively splits the Class-B address into 256 Class-C addresses: 144.100.0, 144.100.1, . . . , 144.100.255, each of which can have 254 nodes. (Host IDs 0 and 255 should not be assigned to a workstation; they are used as broadcast addresses, which are typically recognized by all workstations.) Any 12 of these network addresses could be assigned to the international LANs in this example. Within each LAN, each computer is assigned a unique host ID, and they all have the subnet mask 255.255.255.0.

The preceding example demonstrates a simple (and common) subnet scheme for Class-B addresses. Sometimes it is necessary to segment only portions of an octet, using only a few bits to specify subnet IDs (such as when subnets exceed 256 nodes). Be sure to check with your local network administrator to determine your network's subnet policy and your correct subnet mask.

---

**Note** It is important that all computers on a physical network use the same subnet mask and network ID; otherwise, addressing and routing problems can occur.

---

## Routing

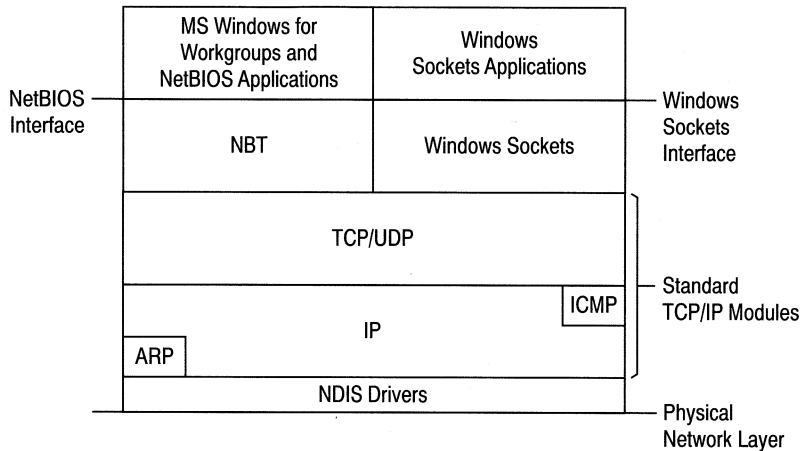
When individual IP subnets are connected to an internet, IP gateways or IP routers are used to provide *routing* (packet delivery) between the networks. When a TCP/IP node attempts to communicate with a different network (when source and destination network IDs differ), a gateway (or a series of gateways) must forward the packet to the appropriate destination network. Gateways maintain routing tables that specify the *direction* (address of the next gateway) a packet should take to reach its destination, as well as a table of local hosts on the networks it interconnects.

Typically, gateways are IP *routers*, or computers with two or more network adapters that are running some type of IP routing software; each adapter is connected to a different physical network. On networks that are not part of an internet, IP gateways are not required. If a network is part of an internet and a node does not specify a default gateway (or the gateway computer is not operating properly), only communication beyond the local subnet is impaired. Currently, Microsoft TCP/IP recognizes only a single default gateway per node. That is, each TCP/IP node must rely on a single gateway to deliver packets to other networks. The network administrator can provide the address of the local gateway. The Microsoft TCP/IP installation software checks to ensure that the network ID for the default gateway matches the network ID of the local IP address. (IP routing and intergateway protocols are beyond the scope of this discussion.)



# TCP/IP with Windows NT

The Microsoft Windows NT operating system with integrated networking is based on a protocol-independent architecture. This architecture, illustrated in the following figure, provides Microsoft Windows NT services over any network protocol that adheres to the *network basic input/output system* (NetBIOS) standard. The NetBIOS-compliant protocol(s) then package application network requests in their respective format(s) and send the requests to the appropriate network hardware via the *network device interface specification* (NDIS) interface. The NDIS specification allows multiple network protocols to reside over a wide variety of network adapters and media types.



Architectural Model of Windows NT and TCP/IP

## NetBIOS over TCP/IP

*NetBIOS over TCP/IP* (NBT) is a protocol module that provides NetBIOS naming services over TCP/IP. This layer allows NetBIOS applications, which establish connections and base data delivery on names, to function over TCP/IP, which relies on IP addresses. Using NBT, NetBIOS applications can locate other NetBIOS computers by name and simultaneously acquire the computers' IP addresses.

*Requests for comments* (RFCs), which are official documents that detail the IP protocol suite, provide a specification to do exactly this in RFC numbers 1001 and 1002. Microsoft TCP/IP complies with RFC 1001/1002. The current TCP/IP implementations are based on a *modified b-node* model, as described in the next section.

## NetBIOS Name Resolution

Two tasks are required of Windows NT nodes running TCP/IP:

- Registration and release of NetBIOS names
- Discovery of NetBIOS names

Configuring the Windows NT operating system with TCP/IP requires *computername* and *IP address* information. (For more information about configuration, see Chapter 2, “Installing and Configuring Microsoft TCP/IP.”) The *computername* is the permanent NetBIOS name to which the workstation responds. The *IP address* is the unique address by which all other TCP/IP nodes on the network recognize the workstation or server.

Since both name and address must be unique, the following process is used. When NBT is initialized, a NetBIOS *name registration request* packet is broadcast to the network stating the NetBIOS name and IP address for the node. Any node on the network that has previously claimed the NetBIOS name is required to challenge the name registration with a *negative name registration response*. Such challenges result in an error during initialization of the NBT module. If the registration request is not challenged within a specific time period, the node adopts that name and address and proceeds. Successive NetBIOS name registrations proceed in the same manner.

When a node has finished with a particular NetBIOS name (such as when the Server service is stopped), it no longer challenges other registration requests for the name. This mechanism is referred to as *releasing* a NetBIOS name.

Since TCP/IP recognizes IP addresses and the Windows NT operating system recognizes names, it is necessary for a Windows NT node running NBT to obtain the IP addresses of the computers with which it must communicate. This information is obtained during the *b-node NetBIOS name discovery process*. A packet containing the name of the destination computer (called the NetBIOS *name query request*, or discovery, packet) is broadcast over the network. The node that owns the name (assuming there is one) responds with a *positive name query response* packet containing its IP address, and the nodes are free to communicate via their IP addresses. This process is transparent to the user.

The discovery request is not foolproof. It is possible that b-node *name query request* packets will not be received by a functioning Windows NT node running TCP/IP. This generally occurs when the two nodes are located on different subnets bridged by an IP router. The discovery packet is implemented as an IP broadcast, which is generally not forwarded by IP routers. Since the discovery packet cannot be heard by the destination node, it would appear that communication would be impossible. This is not the case, however. The LMHOSTS file can be used to specify NetBIOS name/IP address mappings. (For information about the LMHOSTS file, see Chapter 3, “Understanding Name Resolution.”)

---

CHAPTER 5

# SNMP

## Overview

Network management protocols are used to communicate between a management program run by an administrator and the network management program running on a host or gateway. These protocols define the form and meaning of the messages exchanged, the representation of names and values in the messages, and administrative relationships among the gateways being managed.

*Simple Network Management Protocol* (SNMP) is a standard protocol used to monitor IP gateways and the networks to which they attach. SNMP defines a set of variables that the gateway must keep and specifies that all operations on the gateway are a side-effect of fetching or storing the data variables.

## How the SNMP Service Works

With the Microsoft SNMP service, a Windows NT machine can report its current status to an SNMP management system on a TCP/IP network. The service sends status information to a host in two instances:

- When a host requests such information
- When a significant event occurs on the server

The service works with any machine running Windows NT and the TCP/IP services.

The SNMP service can handle requests from one or more hosts. The SNMP service can also report network-management information, in discrete blocks of data called *traps*, to one or more hosts.

On TCP/IP networks, each device has a unique name and internet protocol (IP) address. The SNMP service uses host names and IP addresses to recognize the host or hosts to which it reports information and from which it receives requests.

This chapter assumes that you are familiar with network management, TCP/IP, and SNMP. It also assumes that you are familiar with the concept of a *management information base* (MIB). If you are not familiar with TCP/IP or the Internet MIB 2, see *Internetworking with TCP/IP* by Douglas E. Comer (Prentice Hall, 1991) and *The Simple Book* by Marshall T. Rose (Prentice Hall, 1991).

When a network manager requests information about a device on the network, SNMP management software can be used to determine object values that represent network status. MIB objects represent various types of information about the device. For example, the management station might request an object called **HardDiskBytesAvail**, which would be the free space left on the device's hard disk.

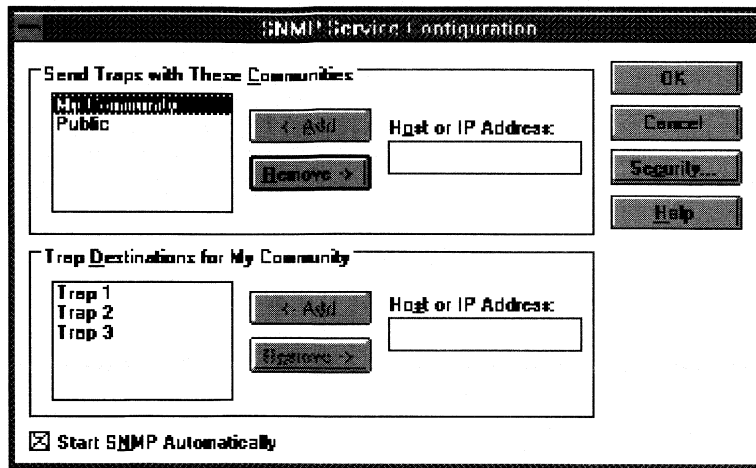
Because different network-management services are used for different types of devices or for different network-management protocols, each service has its own set of objects. The entire set of objects that any service or protocol uses is referred to as its management information base (MIB). The SNMP service for Windows NT supports multiple MIBs through an extension agent API interface. At SNMP service startup time, the SNMP service loads all of the extension-agent dynamic link libraries (DLLs) that are defined in the Windows NT registry.

There are two extension-agent DLLs that come with Windows NT; others may be user-developed and added. The SNMP service supports the Internet MIB II, the LAN Manager MIB II, and a set of LAN Manager alert objects. This section lists the objects in the LAN Manager MIB II and the LAN Manager alert objects that the SNMP service supports.

## SNMP Installation and Configuration

After the TCP/IP transport is installed, configure SNMP as follows:

1. From the Control Panel, select Networks (if it isn't already open).
2. Choose the Advanced button.
3. Select SNMP Service from the Network Software list.
4. Choose the Configure button. The SNMP Service Configuration dialog box appears, as shown in the following figure.



5. Change the configuration; the parameters are described in the following discussion. Note that changes don't take effect until the next time the SNMP service is started; the service reads these parameters only at startup time.

#### Send Traps with These Communities

A community is a group of hosts to which a server running the SNMP service belongs. You can specify one or more communities to which the Windows NT workstation you are installing SNMP service on will send traps. The community name is placed in the SNMP packet when the trap is sent.

When you install the SNMP service, this list contains the default name Public. If the service is installed and running, this list contains community names chosen previously.

To add an entry to the list, enter the community name in the box to the left side of the Add/Remove buttons. Entries in the list may be deleted by clicking on the entry to be deleted in the list and clicking the Remove button.

#### Trap Destinations for *selected* Community

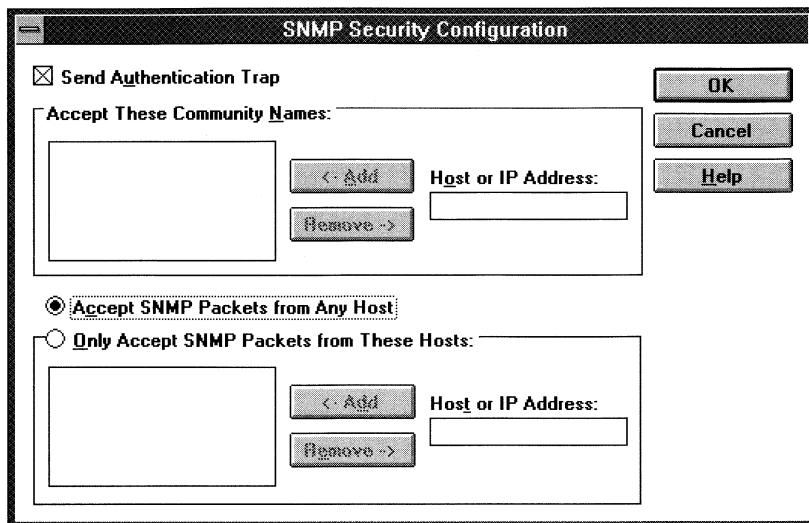
Trap destinations are the names or IP addresses of hosts to which you want the SNMP service to send traps with the selected community name. From the dialog box shown, you can add or delete trap destinations for each defined community.

#### Start SNMP Automatically

When the SNMP service is installed with this check box selected, the SNMP service starts automatically when the system boots. If not selected, the SNMP service starts only when you type **net start snmpsvc** at the command line prompt or by using the service manager tool that comes with Windows NT.

## Security

Selecting this button displays the SNMP Security Configuration dialog box, shown next.



The image shows a dialog box titled "SNMP Security Configuration". It has a checkbox labeled "Send Authentication Trap" which is checked. To the right of the checkbox are three buttons: "OK", "Cancel", and "Help". Below the checkbox is a section titled "Accept These Community Names:". It contains a large empty rectangular box on the left. To the right of this box are two buttons: "< Add" and "Remove ->". Further to the right is a text field labeled "Host or IP Address:". Below this section is a radio button labeled "Accept SNMP Packets from Any Host" which is selected. Below the radio button is another section titled "Only Accept SNMP Packets from These Hosts:". It contains a large empty rectangular box on the left. To the right of this box are two buttons: "< Add" and "Remove ->". Further to the right is a text field labeled "Host or IP Address:".

### Send Authentication Trap

When the SNMP service receives a request for information that does not contain the correct community name and doesn't match an accepted host name for the service, the SNMP service can send a trap to the trap destination(s), indicating that the request failed authentication. This check box is used to specify whether or not this authentication trap is sent.

### Accept These Community Names

A host must belong to a community that appears on this list for the SNMP service to accept requests from that host. Typically, all hosts belong to Public, which is the standard name for the common community of all hosts.

To add an entry to the list, enter the host name or IP address in the box to the left side of the Add/Remove buttons. Entries in the list may be deleted by clicking on the entry to be deleted in the list and clicking the Remove button.

### Accept SNMP Packets from Any Host

If this radio button is selected, no SNMP packets are rejected on the basis of source host ID, and the list of hosts under Only Accept SNMP Packets From These Hosts has no effect.

### Only Accept SNMP Packets from These Hosts

If this radio button is active, SNMP packets will be accepted only from the hosts listed.

To add an entry to the list, enter the host name or IP address in the box to the left side of the Add/Remove buttons. Entries in the list may be deleted by clicking on the entry to be deleted in the list and clicking the Remove button.

## Performance Monitoring

SNMP must be installed and started to monitor TCP/IP network components using Performance Monitor.

## LAN Manager MIB II Objects

The LAN Manager MIB II contains a set of objects specifically designed to support computers running LAN Manager. This section lists the objects in the MIB and provides a brief description of each. (Not all objects are supported in this beta release.)

### Common Group

The following objects apply to both LAN Manager servers and workstations.

#### **comVersionMaj {common 1}**

The major release version number of the LAN Manager software.

#### **comVersionMin {common 2}**

The minor release version number of the LAN Manager software.

#### **comType {common 3}**

The type of LAN Manager software this system is running.

#### **comStatStart {common 4}**

The time, in seconds, since January 1, 1970, at which time the LAN Manager statistics on this node were last cleared. The **comStatStart** object applies to the following statistical objects:

<b>comStatNumNetIOs</b>	<b>svStatErrorOuts</b>	<b>wkstaStatSessStarts</b>
<b>comStatFiNetIOs</b>	<b>svStatPwErrors</b>	<b>wkstaStatSessFails</b>
<b>comStatFcNetIOs</b>	<b>svStatPermErrors</b>	<b>wkstaStatUses</b>
<b>svStatOpens</b>	<b>svStatSysErrors</b>	<b>wkstaStatUseFails</b>
<b>svStatDevOpens</b>	<b>svStatSentBytes</b>	<b>wkstaStatAutoRecs</b>
<b>svStatJobsQueued</b>	<b>svStatRcvdBytes</b>	
<b>svStatSOPens</b>	<b>svStatAvResponse</b>	

#### **comStatNumNetIOs {common 5}**

The number of network I/O operations submitted on this node.

#### **comStatFiNetIOs {common 6}**

The number of network I/O operations on this node that failed issue.

**comStatFcNetIOs {common 7}**

The number of network I/O operations on this node that failed completion.

## Server Group

The following objects apply to LAN Manager servers. Any system that supports this group must also support the Common group.

**svDescription {server 1}**

A comment describing the server.

**svSvcNumber {server 2}**

The number of network services installed on the server.

**svSvcTable {server 3}**

A list of service entries describing the network service installed on the server.

**svSvcEntry {svSvcTable 1}**

The names of the network services installed on the server.

**svSvcName {svSvcEntry 1}**

The name of a LAN Manager network service.

**svSvcInstalledState {svSvcEntry 2}**

The installation status of a network.

**svSvcOperatingState {svSvcEntry 3}**

The operating status of a network service.

**svSvcCanBeUninstalled {svSvcEntry 4}**

Indicates whether the network service specified by this entry can be removed.

**svSvcCanBePaused {svSvcEntry 5}**

Indicates whether the network service specified by this entry can be paused.

**svStatsOpen {server 4}**

The total number of files that were opened on the server.

**svStatDevOpens {server 5}**

The total number of communication devices that were opened on the server.

**svStatQueuedJobs {server 6}**

The total number of print jobs that were spooled on the server.

**svStatSOpens {server 7}**

The number of sessions that were started on the server.

**svStatErrorOuts {server 8}**

The number of sessions disconnected because of an error on the server.

**svStatPwErrors {server 9}**

The number of password violations encountered on the server.

**svStatPermErrors {server 10}**

The number of access-permission violations encountered on the server.



**svStatSysErrors {server 11}**

The number of system errors encountered on the server.

**svStatSentBytes {server 12}**

The number of bytes sent by the server.

**svStatRcvdBytes {server 13}**

The number of bytes received by the server.

**svStatAvResponse {server 14}**

The mean number of milliseconds it took the server to process a workstation I/O request (for example, the average time an NCB sat at the server).

**svSecurityMode {server 15}**

The type of security running on the server.

**svUsers {server 16}**

The number of concurrent users the server can support.

**svStatReqBufsNeeded {server 17}**

The number of concurrent users the server can support.

**svStatBigBufsNeeded {server 18}**

The number of times the server needed but could not allocate a big buffer while processing a client request.

**svSessionNumber {server 19}**

The number of sessions on the server.

**svSessionTable {server 20}**

A list of session entries corresponding to the current sessions that clients have with the server.

**svSessionEntry {svSessionTable 1}**

A session that is currently established on the server.

**svSesClientName {svSessionEntry 1}**

The name of the remote computer that established the session.

**svSesUserName {svSessionEntry 2}**

The number of connections to server resources that are active in the current session.

**svSesNumConns {svSessionEntry 3}**

The number of connections to server resources that are active in the current session.

**svSesNumOpens {svSessionEntry 4}**

The number of files, devices, and pipes that are open in the current session.

**svSesTime {svSessionEntry 5}**

The length of time, in seconds, since the current session began.

**svSesIdleTime {svSessionEntry 6}**

The length of time, in seconds, that the session has been idle.

**svClientType {svSessionEntry 7}**

The type of client that established the session.

**svSesState {svSessionEntry 8}**

The state of the current session. (Setting the state of an active session to **deleted** with **netSessionDel** deletes the client session. The session state cannot be set to **active**.)

**svAutoDisconnects {server 21}**

The number of sessions that the server automatically disconnected because of inactivity.

**svDisConTime {server 22}**

The number of seconds the server waits before disconnecting an idle session.

**svAuditLogSize {server 23}**

The maximum size, in kilobytes, of the server's audit log.

**svUserNumber {server 24}**

The number of users who have accounts on the server.

**svUserTable {server 25}**

A table of active user accounts on the server.

**svUserEntry {svUserTable 1}**

A user account on the server.

**svUserName {svUserEntry 1}**

The name of a user account.

**svShareNumber {server 26}**

The number of shared resource on the server.

**svShareTable {server 27}**

A table of the shared resource on the server.

**svShareEntry {svShareTable 1}**

A table corresponding to a single shared resource on the server.

**svShareName {svShareEntry 1}**

The name of a shared resource.

**svSharePath {svShareEntry 2}**

The local name of a shared resource.

**svShareComment {svShareEntry 3}**

A comment associated with a shared resource.

**svPrintQNumber {server 28}**

The number of printer queues on the server.

**svPrintQTable {server 29}**

A table of the printer queues on the server.

**svPrintQEntry {svPrintQTable 1}**

A table entry corresponding to a single printer queue on the server.

**svPrintQName {svPrintQEntry 1}**

The name of a printer queue.

**svPrintQNumJobs {svPrintQEntry 2}**

The number of jobs currently in a printer.

## Workstation Group

The following objects apply to LAN Manager workstations. Any system that supports this group must also support the Common group.

**wkstaStatSessStarts {workstation 1}**

The number of sessions the workstation initiated.

**wkstaStatSessFails {workstation 2}**

The number of failed sessions the workstation had.

**wkstaStatUses {workstation 3}**

The number of connections the workstation initiated.

**wkstaStatUseFails {workstation 4}**

The number of failed connections the workstation had.

**wkstaStatAutoRecs {workstation 5}**

The number of sessions that were broken and then automatically reestablished.

**wkstaErrorLogSize {workstation 6}**

The maximum size, in kilobytes, of the workstation error log.

**wkstaUseNumber {workstation 7}**

The number of active uses currently on the workstation.

**wkstaUseTable {workstation 8}**

A table of the active uses made by this workstation.

**wkstaUseEntry {wkstaUseTable 1}**

A use of a remote network resource.

**useLocalName {wkstaUseEntry 1}**

The name of a local device assigned to a remote resource.

**useRemote {wkstaUseEntry 2}**

The name of a remote shared resource to which a local device name is assigned.

**useStatus {wkstaUseEntry 3}**

The status of the current connection.

## Domain Group

The following objects apply to both LAN Manager servers and workstations.

**domPrimaryDomain {domain 1}**

The name of the primary domain to which the computer belongs.

**domLogonDomain {domain 2}**

The name of the domain to which the computer is logged on.

**domOtherDomainNumber {domain 3}**

The number of entries in **domOtherDomainTable**.

**domOtherDomainTable {domain 4}**

The list of other domains that the computer is monitoring.

**domOtherDomainEntry {domOtherDomainTable 1}**

An entry in the table of other domains.

**domOtherDomainName {domOtherDomainEntry 1}**

The name of an additional domain the computer is monitoring.

**domServerNumber {domain 5}**

The number of entries in **domServerTable**.

**domServerTable {domain 6}**

The list of nonhidden servers that are in all of the domains that the computer is monitoring.

**domServerEntry {domServerTable 1}**

An entry in the domain server table.

**domServerName {domServerEntry 1}**

The name of a server in one of the domains that the computer is monitoring.

**domLogonNumber {domain 7}**

The number of entries in **domLogonTable**.

**domLogonTable {domain 8}**

The list of domain logons that the computer processed.

**domLogonEntry {domLogonTable 1}**

An entry in the logon table.

**domLogonUser {domLogonEntry 1}**

The name of a user who is logged on in this domain.

**domLogonMachine {domLogonEntry 2}**

The name of a computer where a user is logged on.

## CHAPTER 6

# Utilities Reference

This chapter contains a reference for using Microsoft TCP/IP for Windows NT utilities, which provide connectivity and network administration.

Connectivity commands allow you to interact with and use resources on non-Microsoft hosts such as UNIX workstations. Network administration commands help you maintain the network.

## arp

**Arp** displays and modifies the IP-to-Ethernet or token ring address translation tables used by ARP (address resolution protocol).

### Syntax

**arp -g** [*in\_addr*] [-N [*network*]]

**arp -d** *in\_addr*

**arp -a** *in\_addr ether\_addr*

### Parameters

**-g**

Displays all of the current ARP entries by interrogating the current protocol data. If an Internet address is specified, only that entry is displayed. If more than one network interface uses ARP, entries for each ARP table are displayed.

*in\_addr*

Specifies an Internet address.

**-N** [*network*]

Used with **-g** to display only the current ARP entries for the network interface with network number *network*.

**-d**

Used to delete an entry for the host with the Internet address *in\_addr*, specified as standard dotted-decimal notation.

**-a**

Creates an ARP entry associating the host with the Internet address *in\_addr* with the Ethernet address *ether\_addr*. The Ethernet address is given as 6 hexadecimal bytes separated by hyphens. The entry is permanent.

*ether\_addr*

Specifies an Ethernet address.

## finger

**Finger** displays information about a user on a specified system.

The output will vary according to the remote system, although it generally includes the logon name of the user, the user's full name, and the logon duration, among other statistics.

### Syntax

**finger** [*user@*]*host* [-l]

### Parameters

*user*

Specifies the user you want information about. Omit the *user* parameter to display information about all users on the specified host.

*host*

Specifies the server on the remote system whose users you want information about.

**-l**

Displays information in long list format.

## ftp

**Ftp** transfers files to and from a node running an FTP server program (sometimes called a daemon). See the next section for available **ftp** commands.

### Syntax

**ftp** [-v] [-d] [-i] [-n] [-g] [*host*]

### Parameters

**-v**

Suppresses display of remote server responses.

**-n**

Suppresses auto-login upon initial connection.

**-i**

Turns off interactive prompting during multiple file transfers.

**-d**

Enables debugging.

**-g**

Disables filename globbing (see **glob**, in the list of **ftp** subcommands).

*host*

Specifies the remote computer.

## Notes

### Connecting to hosts

The **ftp** command is the user interface to the standard File Transfer Protocol (FTP), described in RFC959.

The host with which **ftp** is to communicate may be specified on the command line. If this is done, **ftp** immediately attempts to establish a connection to an FTP server on that host; otherwise, **ftp** enters its command interpreter and awaits instructions from the user (see the list of **ftp** subcommands). A host can be specified either by name (**sallysm.bigco.com**) or IP addresses in dotted-decimal notation (**70.85.67.75**).

### Special Filenames

Files specified as arguments to **ftp** commands are processed according to the following rules.

1. If the filename - is specified, **stdin** (reading) or **stdout** (writing) is used.
2. If the first character of the filename is |, the remainder of the argument is interpreted as a shell command. The **ftp** command then starts a command.com shell, using the argument supplied, and reads (writes) from the **stdout** (**stdin**). A particularly useful example of this mechanism is **dir dirname | more**.
3. Failing the above and if globbing is enabled, local filenames are expanded.

### Comments on Subcommands

The **dir**, **ls**, **mget**, and **mdelete** commands do not act on nonexistent filenames.

The **mdir** and **mls** commands produce confusing output when used with wildcard filenames.

### Restrictions

Some FTP server implementations do not support experimental operations, such as print working directory.

## Ftp Subcommands

You can use the following commands at the **ftp** prompt.

**! *command***

Runs *command* on the local machine. If *command* is omitted, cmd.exe is started.

**< *cmd-file***

Reads **ftp** commands from *cmd-file*.

**append *local-file* [*remote-file*]**

Appends a local file to a file on the remote machine. If *remote-file* is left unspecified, the local filename is used in naming the remote file. File transfer uses the current setting for *type*.

**ascii**

Sets the file transfer *type* to network ASCII, the default.

In ASCII mode, any necessary character conversions to and from the network standard character set are performed, for example, the conversion of end-of-line characters between UNIX machines and PCs (UNIX uses CR and PCs use CR/LF). This mode should be used when transferring text files.

FTP supports ASCII and binary image file transfers. See also **binary**.

**bell**

Sounds a bell after each file transfer command is completed. By default, the bell is off.

**binary**

Sets the file transfer *type* to support binary image transfer.

In image mode the file is moved literally, byte by byte. This mode should be used when transferring binary files.

FTP supports ASCII and image file transfer types.

**bye**

Terminates the FTP session with the remote server and exits **ftp**.

**cd** *remote-directory*

Changes the working directory on the remote machine to *remote-directory*.

**close**

Terminates the FTP session with the remote server and returns to the command interpreter.

**delete** *remote-file*

Deletes the file *remote-file* on the remote machine.

**debug**

Toggles debugging mode. When debugging is on, each command sent to the remote machine is printed, preceded by the string --->. By default, debugging is off.

**dir** [*remote-directory*] [*local-file*]

Prints a listing of the directory contents in the directory *remote-directory* and, optionally, places the output in *local-file*. If no directory is specified, the current working directory on the remote machine is used. If no local file is specified, output comes to the terminal.

**get** *remote-file* [*local-file*]

Retrieves the *remote-file* and stores it on the local machine. If the local filename is not specified, it is given the same name it has on the remote machine. The current setting for *type* is used while transferring the file.



**glob**

Toggles filename globbing. With globbing enabled, each local file or path name is processed for wildcards. Remote files specified in multiple item commands, like **mput**, are globbed by the remote server. With globbing disabled, all files and path names are treated literally. By default, globbing is enabled.

**hash**

Toggles hash-sign (#) printing for each data block transferred. The size of a data block is 1024 bytes. By default, hash mark printing is off.

**help** [*command*]

Prints an informative message about the meaning of *command*. If no argument is given, **ftp** prints a list of all known commands.

**lcd** [*directory*]

Changes the working directory on the local machine. If no *directory* is specified, the user's home directory is used.

**literal** *arg1 arg2 . . .*

The arguments specified are sent, verbatim, to the remote FTP server. A single FTP reply code is expected in return.

**ls** [*remote-directory*] [*local-file*]

Prints an abbreviated listing of the contents of a directory on the remote machine. If *remote-directory* is not specified, the current working directory is used. If *local-file* is not specified, the output is sent to the terminal.

**mdelete** *remote-files*

Deletes the specified files on the remote machine. If globbing is enabled, the specification of remote files is first expanded using **ls**. If prompting is enabled, the user must respond to a prompt before each file is deleted.

**mdir** *remote-files local-file*

Obtains a directory listing of multiple files on the remote machine and places the result in *local-file*. If *local-file* is specified as -, the result is displayed on the terminal.

**mget** *remote-files*

Retrieves the specified files from the remote machine and places them in the current local directory. If globbing is enabled, the specification of remote files is first expanded using **ls**. If prompting is enabled, the user must respond to a prompt before each file is transferred.

**mkdir** *directory-name*

Makes a directory on the remote machine named *directory-name*.

**mls** *remote-files local-file*

Obtains an abbreviated listing of multiple files on the remote machine and places the result in *local-file*. If *local-file* is specified as -, the result is displayed on the terminal.

**mput** *local-files*

Transfers multiple local files from the current local directory to the current working directory on the remote machine. If globbing is enabled, the specification of local files is first expanded. If prompting is enabled, the user must respond to a prompt before each file is transferred.

**open** *host* [*port*]

Establishes a connection to the specified host FTP server. A port number may optionally be supplied, in which case FTP attempts to contact an FTP server at that port. If *auto-login* is on (default), FTP also attempts to automatically log the user on to the FTP server (see **-n** option).

**prompt**

Toggles interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files. If prompting is turned off, any **mget** or **mput** transfers all files. By default, prompting is on.

**put** *local-file* [*remote-file*]

Stores a local file on the remote machine. If *remote-file* is not specified, the local filename is used as the remote filename. File transfer uses the current setting for *type*.

**pwd**

Prints the name of the current working directory on the remote machine.

**quit**

A synonym for **bye**.

**recv** *remote-file* [*local-file*]

A synonym for **get**.

**remotehelp** [*command-name*]

Requests help from the remote FTP server. If a command-name is specified, it is supplied to the server as well.

**rename** *from to*

Renames the file *from* on the remote machine, to the file *to*.

**rmdir** *directory-name*

Deletes the directory *directory-name* on the remote machine.

**send** *local-file* [*remote-file*]

A synonym for **put**.

**status**

Shows the current status of FTP. For example:

Connected to rhino.microsoft.com.

Type: ascii; Verbose: on; Bell: off; Prompting: on; Globbing: on;

Debugging: off; Hash mark printing: off.

**type** [*type-name*]

Sets the file transfer *type* to *type-name*, which may take the value network ASCII or binary image. If no type is specified, the current type is printed. See also the **ascii** and **binary** commands. The default type is network ASCII.

**user** *user-name* [*password*] [*account*]

Identifies yourself to the remote FTP server. If the password is not specified and the server requires it, FTP prompts the user for it (after disabling local echo). If an account field is not specified and the FTP server requires it, the user is prompted for it. Unless FTP is invoked with auto-login disabled (with **-n** option), this process occurs automatically on initial connection to the FTP server.

**verbose**

Toggles verbose mode. In verbose mode, all responses from the FTP server are displayed to the user. In addition, if verbose is on, when a file transfer completes, statistics regarding the efficiency of the transfer are reported. By default, verbose is on for an interactive session; otherwise it is off.

**?** [*command*]

Provides help.

Command arguments that have embedded spaces may be quoted with quote (") marks.

FTP supports the ASCII and image types of file transfer.

## hostname

**Hostname** prints the name of the current host.

**Syntax**

**hostname**

To set the host name, use Networks in Control Panel.

## nbtstat

**Nbtstat** displays protocol statistics and the state of the current TCP/IP connections using NBT (NetBIOS over TCP/IP).

**Syntax**

**nbtstat** [-a] [-N] [-C] [*interval*]

**Parameters****-a**

Shows the server connections.

**-N**

Lists local NetBIOS names in the following format:

NetBIOS name info

Node name: <hostname> Scope Id: <scope>

Name	Type	Status
drs3	UNIQUE	REGISTERED
lasagne	GROUP	REGISTERED

Type may be Unique or Group, and status may be Registered, Conflict, Registering, or Deregistering.

**-C**

Lists the NetBIOS name cache in the following format:

NetBIOS name cache info

Node name: <hostname> Scope Id: <scope>

NetBIOS Name	Internet Host	Type	Life [sec]
fred			

*interval*

Redisplays selected statistics, pausing *interval* seconds between each display. Press CTRL+C to stop redisplaying statistics.

**Notes**

The **Nbtstat** utility provides statistics on the following network components.

**Type**

Indicates the type of connection. The possible types are:

<b>ssnnp</b>	A session endpoint
<b>genp</b>	A datagram endpoint
<b>nmsrv</b>	The name server
<b>sssrv</b>	The session server
<b>dgsrv</b>	The datagram server
<b>ssind</b>	A session indication

**Input**

Indicates the cumulative amount of data received.

**Output**

Indicates the cumulative amount of data sent.

**Local Name**

The local NetBIOS name associated with the connection.

**Foreign Name**

The remote NetBIOS name associated with the connection.

## State

The state of NetBIOS connections only. The possible states are:

### **DATA\_XFER**

The session has been established.

### **LINKED\_TCP**

A new TCP endpoint has been linked below ssind endpoint.

### **LISTEN**

Bound with `t_bind(qlen > 0)`, that is, a passive endpoint.

### **IDLE**

Bound with `t_bind(qlen = 0)`, that is, an active endpoint.

### **RETARGET**

An attempt has been made to connect to a remote endpoint and has been retargetted to another address by the remote session server. The connection is now being attempted to the retarget address.

### **UNBND**

Endpoint opened with `t_open()`.

### **UNLINK\_TCP**

The TCP connection has been disconnected, and a local unlink of the TCP from below the NBT driver has been requested. At completion of the unlink, the ssind is removed.

### **WACK\_BREQ\_TCP**

Awaits local ACK (from TCP) on bind request.

### **WACK\_CRES\_TCP**

Awaits local ACK (from TCP) of our connect response, that is, the incoming TCP connect indication has been accepted, and the full establishment of the TCP connection is being awaited.

### **WACK\_DREQ\_TCP**

The session indication was rejected, and a TCP disconnect request has been made. The ACK of the TCP disconnect is now expected.

### **WACK\_LREQ\_TCP**

Awaits local ACK (from NBTD) on request to link a TCP stream below the NBT driver.

### **WACK\_NREQ**

Awaits local ACK (from name server) on NetBIOS name request to get the internet address of the requested NetBIOS name.

### **WACK\_CREQ\_TCP**

Awaits local ACK (from TCP) on connect request.

### **WCON\_CREQ\_TCP**

Awaits a remote connect confirm on TCP connect request, that is, it awaits the other end accepting the TCP connection.



**-s**

Displays per-protocol statistics (error occurrences, and so on) instead of connection states, in the following format:

tcp:

```
0 incomplete headers
0 header checksum errors
```

udp:

```
0 incomplete headers
0 header checksum errors
0 bad data length fields
```

ip:

```
0 incomplete headers
0 header checksum errors
0 bad data length fields
```

There is a separate block of statistics for each protocol. By default, statistics are shown for TCP, UDP and IP; the **-p** option may be used to specify that a different set of statistics is to be displayed.

**-p *proto***

Shows connections for protocol *proto*; *proto* may be **tcp** or **udp**. If used with the **-s** option to display per-protocol statistics, *proto* may be **tcp**, **udp**, or **ip**.

**-r**

Displays the contents of the routing table, in the same format as *route*.

***interval***

Redisplays selected statistics, pausing *interval* seconds between each display. Press CTRL+C to stop redisplaying statistics.

**Notes**

By default, the state of all connections is displayed in the following format:

```
Active connections
Proto Recv-Q Send-Q Local Address Foreign Address (state)
tcp      0      0 local:telnet 89.0.4.1:5137 ESTABLISHED
```

The first line is a short header for the display. The second line consists of headers for each column of the display. The remaining lines contain the following columns:

**Proto**

The name of the protocol used by the connection.

**Recv-Q**

The amount of data (in bytes) received but not processed on the connection.

**Send-Q**

The amount of data (in bytes) waiting to be transmitted (including data sent but not yet acknowledged).

**Local Address**

The Internet address of the local host, as well as the port number the connection is using. The name corresponding to the Internet address is shown instead of the number if a name look-up succeeds, and likewise for the port. In cases where the port is not yet established, the port number is shown as an asterisk (\*).

**Foreign Address**

The Internet address and port number of the remote host to which the socket is connected.

**(state)**

Indicates the state of TCP connections only. The possible states are

CLOSED        LISTEN        SYN\_SEND  
SYN\_RECV    FIN\_WAIT\_1   FIN\_WAIT\_2  
ESTABLISHED   CLOSE\_WAIT

**Continuously Displaying Statistics**

If an *interval* is specified, **netstat** continuously redisplay the appropriate information, pausing *interval* seconds between each display.

**Restrictions**

The **-s** and **-e** options may be used together but not with the other options.

---

**Note** The **-a** option does not show Protocol Control Blocks, which contain all-zero local and foreign addresses/ports unless used in conjunction with the **-A** option or when TCP connection is in the CLOSED state.

---



# ping

**Ping** ascertains the net status of a remote host or hosts.

## Syntax

```
ping [-t] [-s] [-n number] [-l length] host [host . . .] [-f] [-p precedence]
[-i ttl] [-v tos] host [host . . .] [-r route-count] host [-j host [host] . . .] [-k host
[host] . . .]
```

## Parameters

**-t**

The destination is pinged forever.

**-s**

Information about returned packets is not displayed.

**-n** *number*

**Ping** transmits that *number* of ECHO packets (the default is 4).

**-l** *length*

**Ping** transmits ECHO packets containing the amount of data specified by *length* (the default is 64 bytes, and the maximum is 1024).

*host*

Specifies the remote host name or an Internet address.

**-f**

The *Don't Fragment* bit is set; the packet will not be fragmented by gateways on the route.

**-p** *precedence*

The *precedence* field is set to the following argument.

**-i** *ttl*

If specified, the *Time to Live* field is set to the *ttl* argument.

**-v** *tos*

The *Type of Service* field is set to the *tos* argument.

**-r** *route-count*

The route of the outgoing packet and that followed by the returning ICMP Echo Reply packet is stored in the *Record Route* field, up to a maximum number specified by the *route-count* argument.

**-j** *host* [*host*] . . .

The packet is routed via the list of hosts *host* [*host*] . . . Consecutive hosts may be separated by intermediate gateways (the packet is *Loose Source Routed*).

**-k** *host* [*host*] . . .

The packet is routed via the list of hosts *host* [*host*] . . . Consecutive hosts may not be separated by intermediate gateways (the packet is *Strict Source Routed*).

Only one of the last three options may be specified. Note that only up to nine intermediate hosts may be specified in a Loose or Strict Route, or recorded in a Record Route (this is a limitation of the IP protocol).

## Notes

The **ping** command ascertains the net status of a remote host or hosts, by sending ICMP ECHO packets to the host and listening for ECHO REPLY packets. Ping waits for up to 2 seconds for each packet sent and prints the number of packets transmitted and received. Each received packet is validated against the transmitted message. By default, four ECHO packets containing 64 bytes of data (a periodic uppercase sequence of alphabetic characters) are transmitted.

## rcp

**Rcp** copies files between machines. Each *file* or *directory* argument is either a remote filename of the form *rhost:path*, or a local filename, that is, one containing no : characters before the first backslash character.

## Syntax

**rcp** [-abh][*host1*.][*user1*:]*source* [*host2*.][*user2*:] *path\destination*

**rcp** [-r] *file* . . . *directory*

## Parameters

**-a**

Specifies the ASCII transfer mode (default). This mode converts between the EOL characters: CR for UNIX and CR/LF for PCs.

**-b**

Specifies binary image transfer mode.

**-h**

Used to transfer hidden files.

**-r**

If any of the source files are directories, **rcp** recursively copies each subtree rooted at that name; the destination must be a directory.

*path*

Is not a full path name; it is interpreted relative to your logon directory on the remote host. A *path* on a remote host may be quoted (using \, ", or ') so that the wildcard characters are interpreted remotely.

## Notes

The **rcp** command does not prompt for passwords; your current local user name must exist on the remote host and allow remote command execution privileges.

The **rcp** command handles third-party copies, where neither source nor target files are on the current machine. Host names may also take the form *rhost.rname* to use *rname* rather than the current user name on the remote host. If you use the *host.name* syntax, then *host.name* of the source should be included in the *.rhosts* of the target account, for example:

```
rcp rhino.userx:foo borisb.adminy:bar
```

The *.rhosts* file on **borisb** in **adminy** should have an entry for **rhino** on *userx*.

The remote processing is performed by a command run from the user's logon shell on most UNIX systems. The user's *.profile* or *.cshrc* is executed before parsing filenames, and exported shell variables may therefore be used, quoted, in remote filenames.

---

**Note** If you attempt to copy a number of files to a file rather than a directory, only the last file is copied. Also, the **rcp** command cannot copy a file onto itself.

---

If a host name is supplied as a full domain name containing dots, a user name must be appended to the host name, as previously described. This prevents the last element of the domain name from being interpreted as a user name, for example:

```
rcp domain-name1.user:foo domain-name2.user:foo
```

### Restriction

If you are logged onto the Windows NT machine on any domain, but the local machine and the domain controller are unavailable, the command will fail because **rcp** cannot determine the local logged user name. This information is not stored locally and must be obtained from the domain controller. The same restriction applies to **rsh**.

## rexec

**Rexec** provides remote execution facilities, similar to those of **rsh**, but with authentication based on user names and passwords.

### Syntax

```
rexec host [-l username] [-n] command
```

### Parameters

**-l**

Specifies the user name on the remote host when this is not the default.

**-n**

Redirects the input of **rexec** to NULL if you don't want input.

*command*

Specifies the command to execute.

### Notes

When the **rexec** command is executed, it prompts the user for a password on the remote host. Then it connects to the specified *host* and authenticates the given password. If the authentication passes, it then executes the specified *command*. **Rexec** copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit, and terminate signals are propagated to the remote command. **Rexec** normally terminates when the remote command does.

The remote user name used is the same as your local user name, unless you specify a different remote name with the **-l** option.

Shell metacharacters that are not quoted are interpreted on the local machine, while quoted metacharacters are interpreted on the remote machine. Thus the following command appends the remote file *remotefile* to the local file *localfile*:

```
rexec otherhost cat remotefile >> localfile
```

while the next command appends *remotefile* to *otherremotefile*:

```
rexec otherhost cat remotefile ">>" otherremotefile
```

## Restrictions

You cannot run most interactive commands. For example, vi or emacs cannot be run using **rexec**. To run these, use Windows Terminal.

# route

**Route** manually manipulates network routing tables.

## Syntax

**route** [-fs] [*command* [*destination*] [*gateway*]]

## Parameters

**-f**

**Route** clears the routing tables of all gateway entries. If this is used in conjunction with one of the commands, the tables are cleared prior to the command's application.

**-s**

The operation refers to the smart gateway. One smart gateway may be specified; packets for which no destination is found in the gateway table are routed to the smart gateway. The destination field is not specified when this option is in use.

## Notes

The **route** command accepts four commands:

### print

Prints a route

### add

Adds a route

### delete

Deletes a route

### change

Modifies an existing route

All symbolic names used for *destination* or *gateway* are looked up in the network and host name database files NETWORKS and HOSTS, respectively. If the command is **print** or **delete**, wildcards may be used for the destination and gateway or the gateway argument may be omitted.

# rsh

The **rsh** (remote shell) command connects to the specified host and executes the specified *command*.

**Syntax** **rsh** *host* [-**I** *username*] [-**n**] *command*

**Parameters** -**I** *username*  
Specifies the user name on the remote host when this is not the default.

-**n**  
Redirects the input of **rsh** to NULL if input isn't wanted.

**Notes** **Rsh** copies its standard input to the remote *command*, the standard output of the remote *command* to its standard output, and the standard error of the remote *command* to its standard error. **Rsh** normally terminates when the remote *command* does.

The remote *username* used is the same as your local user name, unless you specify a different remote name with the -**I** option. No provision is made for specifying a password with a command.

Shell wildcards that are not quoted are interpreted on the local machine, while quoted wildcards are interpreted on the remote machine. Thus the following command appends the remote file *remotefile* to the local file *localfile*:

```
rsh otherhost cat remotefile >> localfile
```

while the next command appends *remotefile* to *otherremotefile*:

```
rsh otherhost cat remotefile ">>" otherremotefile
```

## Restriction

If you are logged onto the NT machine on any domain, but the local machine and the domain controller are unavailable, the command will fail because **rsh** cannot determine the local logged user name. This information is not stored locally and must be obtained from the domain controller. The same restriction applies to **rcp**.

## tftp

**Tftp** provides the user interface to the Internet standard Trivial File Transfer Protocol (TFTP). The program allows a user to transfer files to and from a remote or a local workstation.

Since the TFTP protocol does not support any user authentication, files must be world-readable (writable) on the remote system.

### Syntax

**tftp** [-i] *host operation file-one [file-two]*

### Parameters

#### -i

Specifies binary image is the mode of transfer, which is also called octet (the default is ASCII). In image mode the file is moved literally, byte by byte. This mode should be used when transferring binary files.

In ASCII mode any necessary character conversions to and from the network standard character set are performed, for example, in the conversion of end-of-line characters between UNIX computers and personal computers (UNIX uses CR and personal computers use CR/LF). This mode should be used when transferring text files.

If a file transfer is successful, the data transfer rate is displayed.

#### *host*

Specifies the name or IP address of the remote host.

#### *operation file-one file-two*

Specify **put** if transferring file *file-two* on the local host to file *file-one* on remote host. Specify **get** if transferring file *file-two* on the remote host to file *file-one* on the remote host.

If the local file is -, the remote file is printed out on *stdout* (if getting), or is read from *stdin* (if putting). If *file-two* is omitted, it is assumed to have the same name as *file-one*.

## telnet

**Telnet** starts terminal emulation. The Windows Terminal program is started, the Telnet port is opened, and the Telnet prompt is displayed.

### Syntax

**telnet**

### Notes

**Telnet** simplifies TCP/IP terminal emulation with Windows NT.

First, **telnet** starts the **telnet** network service. Then **telnet** starts **TERMINAL.EXE**, instructing it to open the **telnet** port.

The Telnet network service can also be started with the Control Panel Services option or at the command prompt with the **net start telnet** command.

## Telnet Commands

The following commands can be typed at the Telnet prompt.

**open** *host* [*port*]

Connects to the specified *host*. *Host* can be a host name or Internet address. If *port* is not specified, **telnet** will attempt to contact a Telnet server at the default port.

**connect** *host* [*port*]

Same command as **open**.

**close**

Closes a **telnet** session and returns to command mode.

**quit**

Ends **telnet**.

**bye**

Ends **telnet**.

**escape**

Sets the escape character.

**status**

Displays the current status of **telnet**.

**options**

Toggles viewing of **telenet** options processing.

**crmod** [**on** | **off**]

Sets carriage return mode. Required only for hosts that require local echoing.

**debug**

Not currently supported.

**help** or **?**

Provides information about a command.