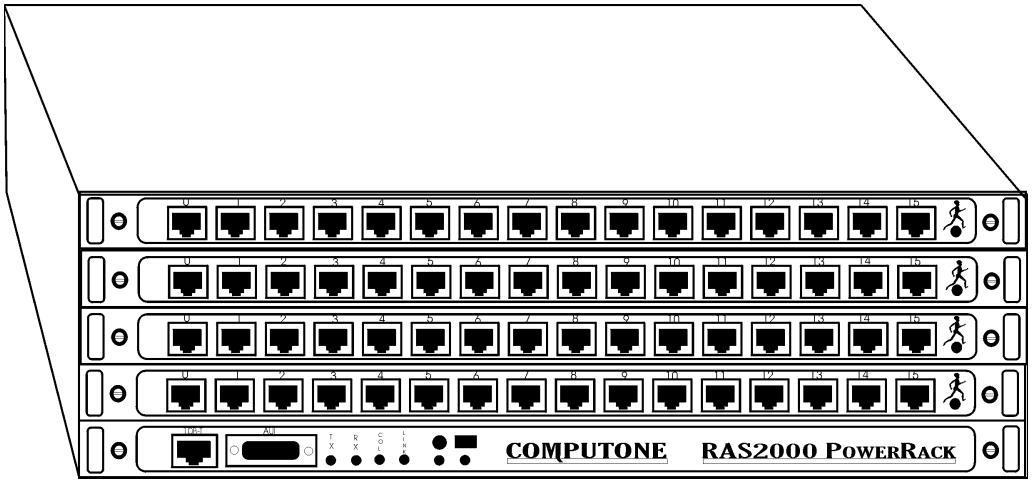
RAS 2000 PowerRack
Software Configuration Guide



COMPUTONE
Corporation

1060 Windward Parkway, Suite 100 Alpharetta, GA, 30005-3992 (USA)
(800) 241-3946 s Outside U.S./Canada: (770) 625-0000
FAX: (770) 625-0013 email: sales@computone.com
INTERNET World Wide Web - http://www.computone.com

Computone Corporation
1060 Windward Ridge Parkway
Alpharetta, GA 30005-3992
U.S.A.

Disclaimer: Computone Corporation ("Computone") makes no representations or warranties with respect to the contents hereof, and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Furthermore, Computone reserves the right to revise this publication and make changes from time to time to the contents hereof, without obligation of Computone to notify any person of such revisions or changes.

Note: This equipment has been tested and found to comply with the limits of a Class A device, pursuant to Part 15 of the United States FCC regulations. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy, and if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to cause harmful interference in which case the user will be required to correct the interference at his own expense.

There is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception (which can be determined by turning the offending equipment off and then on), you are encouraged to try to correct or remove the interference using one or more of the following methods: (a) reorient or relocate the receiving antenna; (b) increase the separation between the equipment and the receiver; (c) connect the equipment to an outlet on a circuit different from that of the receiver; (d) consult the dealer or an experienced radio/television technician for assistance.

This digital apparatus does not exceed the (Class A, Class B)* limits for radio noise emissions from digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications.

* Indicate only the class of digital apparatus which is appropriate for the specific application.

Le present appareil numerique n'emet pas de bruits radioelectriques depassant les limites applicables aux appareils numeriques (de la class A/de la class B)* prescrites dans le Reglement sur le brouillage radioelectrique edicte par le ministere des Communications du Canada.

* Indiquer seulement la classe d'appareils numeriques correspondant a l'application visee.

Support Information: If you require technical support, contact your Computone dealer or Computone Technical Support. The Computone Technical Support staff can be reached by phone at the following numbers, from 8:30 a.m. to 5:30 p.m. Eastern time, Monday through Friday:

(800) 241-3946 ext. 250
(770) 475-2725 ext. 250
(770) 664-1110 (FAX)
(770) 343-9737 (BBS)
(770) 664-1210 (BBS)

Technical Support can be contacted by email at the Internet address **support@computone.com**

Trademarks: Computone and IntelliServer are trademarks of Computone Corporation. All other brand names or product names are trademarks or registered trademarks of their respective corporations.

IntellIServer PowerRack Hardware Guide P/N: 0-22200

ISBN 0-000-000000-0

ABCDEFGHIJ-DO-89

Table of Contents

CHAPTER 1

Introduction

We are glad you have decided to purchase a Computone IntelliServer RAS 2000 PowerRack and we believe that your experience with this product will confirm that this was a wise decision.

This guide helps you understand the IntelliServer's great versatility and it is important that you are familiar with its contents. Otherwise, you will not be able to use the IntelliServer to its fullest potential. This guide is intended for the system administrator, VAR, or other technician who will be installing and configuring the IntelliServer for its particular application.

What Is an IntelliServer?

The RAS 2000 PowerRack (RAS 2000) comes with 16 serial ports and an Ethernet connection to your local network. It allows serial devices to access hosts on your local network and for hosts on that network to access the devices. You can:

You can connect terminals to the serial ports and use *telnet* or *rlogin* to log into hosts attached to your network. You can:

1. Attach serial printers directly to the IntelliServer's serial ports and configure a computer on your network to send printer output to them.
2. Attach printers to your terminal's *AUX* port and configure your computer to send printed output to them, apart from what is happening on the terminal.
3. Configure a host computer to access the RAS 2000's serial ports in much the same way as it would have accessed the ports on an internal multi-port serial card, providing a platform-independent multi-port solution for many applications.
4. Extend your network via PPP or SLIP connections running over one or more serial ports. Some customers support branch offices in this way. An IntelliServer in their remote branch uses one of its ports to run a PPP/SLIP connection back to their local network. Then, terminals are attached to remaining serial ports and can each log into hosts at the main site and only one modem connection is used.
5. Use the RAS 2000 to support PPP/SLIP Internet Access to dial-up customers for an Internet Service Providers (ISP's).

This is not a comprehensive list. As you understand what the RAS 2000 is capable of, your imagination will take over from there.

Ground Rules

Who is This Manual For

This manual is intended for the VAR, on-site system administrator, or installer who needs to set up the RAS 2000's software to support one or another applications. It is also intended for training support staff and for giving sales personnel an in-depth understanding of the RAS 2000's capabilities.

No advanced technical knowledge is required to understand this guide, but the less experienced may need to skip around and re-read certain sections for everything to be understood.

What Else do You Need to Read

The RAS 2000 is typically used in networks which contain a variety of host computers running different operating systems. This manual explains the *RAS 2000*. It does not try to explain everything you might need to know about how to configure your *other* computers so they can talk to the RAS 2000. From time to time you will need to consult your *other* computer's operating system manuals or online documentation.

This guide contains a basic introduction to networking concepts, but it is only a start. Your computer bookstore will have several good books on TCP/IP and other networking concepts. You might not need these books just to configure the RAS 2000, but you *probably* will need the background sooner or later.

What Products are Covered

This manual covers the RAS 2000 running software level 3.0 or later. If you are using an earlier software version, you may upgrade to a later version by loading the new version on a local network host and configure your RAS 2000 to boot over the network.

Table 1-1 lists the products that are covered by this guide. This list is subject to change as new products are added and older ones discontinued. Check with our sales department for the latest information.

TABLE 1-1. IntelliServer RAS 2000 PowerRack Products

Part Number	Description
RAS2000/16	RAS 2000 16-port PowerRack
RAS2000/32	RAS 2000 32-port PowerRack
RAS2000/48	RAS 2000 48-port PowerRack
RAS2000/64	RAS 2000 64-port PowerRack
REX-16RJ-232	16-port RJ-45 Expansion Module, RS-232 Support
VP-RJ-DB/T	RJ-45 (male) to DB-25 (male) 10-pin, 6 feet long cable
VP-RJ-DB/M	RJ-45 (male) to DB-25 (male) 10-pin, 6 feet long cable

Where do I Learn Pin-outs and Other Hardware Information

This guide contains everything you should need to know about the RAS 2000's software configuration, but it does not contain pin-outs and other hardware documentation. The *IntelliServer RAS 2000 PowerRack Hardware Guide* shipped with your unit contains pin-outs and other hardware information.

How this Manual is Organized

This manual is organized so that later chapters build on the earlier ones so you could read it from front to back without becoming hopelessly confused. The first thing explained is what happens when you first take the RAS 2000 PowerRack out of the box, and gives you a road map to different areas of interest.

In the next chapter, how to configure serial ports is explained. Then, modems are discussed, because they are closely related to serial ports. Finally, user configuration is discussed, since that is the next-most important concept.

In later chapters, you need to understand networking concepts, so a chapter is included to introduce the basics. Then, configuration that applies to your local network is covered, followed by a chapter on bringing up *remote* network links (i.e., PPP and SLIP connections). Once networking is covered, more complicated things like setting up printers and configuring RADIUS users is explained.

Don't expect to need to read this manual cover-to-cover. You probably will be looking things up, skipping around, trying to find answers to your important questions. A lot of illustrations and tables are provided to help you find information quickly, as well as numerous cross-references and an index.

CHAPTER 2

Out of the Box

In this chapter we will discuss things you will need to know when you first start using the IntelliServer

- How to attach a console terminal.
- Start-up console messages and what they mean.
- Configuration Road maps.

Hardware Installation

Your IntelliServer RAS 2000 PowerRack (RAS 2000) installation consists of the following:

- **The IntelliServer Main Unit**, with connections for power, network (Ethernet AUI or BNC connections) and 16 serial ports.
- **Power cord** - 110v or 220v.
- **Documentation Package**, including this manual, a hardware guide, release notes and other important information.

The RAS 2000 is available in different serial port configurations, some using RJ45 connections, some supporting RS232 and some supporting RS232/422.

Rules To Live By

1. Always turn the power switch off before disconnecting the power cord from the outlet.
2. It is o.k. to connect and disconnect *serial* cables from the RAS 2000 and its expansion boxes while power is applied, aside from the inevitable disruption of whatever you disconnected if you were using it at the time.
3. It is best to connect the RAS 2000 PowerRack to your network before you power it up.

Configuration — Getting Started

The RAS 2000 is designed to connect serial ports to your local Ethernet network. For configuration purposes, you can access it either from the network or through serial ports. Before you access it over the network, you have to configure network parameters. Before you access it through serial ports, *they* have to be configured. So, where do you start?

If you are like most users, you start by attaching a terminal to one of the enabled serial ports - port 0 for example (by default, ports 0, 7, 8, and 15 are enabled, other ports are disabled.) Then you configure the RAS 2000 using the menus and commands described later in this guide. Since you have not configured any of the ports yet, you need to set up your terminal to match the port's factory default settings of:

- 9600 baud
- 8-bit characters
- no parity

In chapter 14, *Saving and Restoring Configurations*, you are shown to use BOOTP protocol to supply the RAS 2000 with enough network settings to allow it to configure completely over the network. This is a bit complicated the first time you try it, so novices usually prefer to connect a terminal.

Refer to the *RAS 2000 PowerRack Hardware Guide* for the correct cabling to connect the RAS 2000 port to your terminal. Be sure to attach the cable to one of the four ports that are enabled by default. These ports are configured so that after the RAS 2000 starts up there is a command prompt. Why not enable all ports? To avoid sending data to any port until it is configured. It wouldn't cause a problem if nothing were attached to the ports, but perhaps you are replacing a unit and want to transfer all the cables at once. If the RAS 2000 had enabled all the ports, it would be sending data of an inappropriate nature at possibly incorrect baud rates to all the attached devices.

By enabling only ports 0, 7, 8, and 15, the other ports are all safe to connect to. Of these four, your terminal is connected to one of them; disconnect the cables from the other three ports until after they are configured. Why enable four ports and not just one? If you should have difficulty getting your terminal to work, you can try the other three ports to confirm that it is a terminal or cabling problem, not a dead port on the RAS 2000.

Attaching a Console Terminal

Although you could use any one of ports 0, 7, 8, or 15, port 0 is the best choice. When factory-default settings are in effect, this port is designated the *console* port. This means that power-on banners and messages are sent to this port, as well as any unexpected error messages and warnings. By attaching the console to this port, you are in a better position to see what is going on.

If you wish ultimately to connect a modem or non-terminal device to port 0, you can later reconfigure the RAS 2000 console to a different port or disable it altogether.

When Power is Applied

After you have attached your terminal to port 0 and connected your RAS 2000 to your network, power up the RAS 2000. Your RAS 2000 has two indicator lights or LED's; one is marked with a circle and one with a rectangle. When you apply power, they turn yellow, flashing in different combinations to indicate that the RAS 2000's power-on self test is progressing. If an error occurs during testing, these LED's display an error code as shown in Table 2-1 or Table 2-2.

Table 2-1: LED Codes

Circle	Rectangle	Description
Yellow	Yellow (*flashing)*	Power-On self test is proceeding.
Yellow	Off	
Yellow	Green	Running Technician's interface (only occurs during manufacturing).
Yellow	Red	PROM Checksum bad.
Off	Yellow	CPU hangs trying to access serial port (console) registers.
Off	Off	At start-up, would indicate that power is not present or that the RAS 2000's CPU is dead.
		During normal operation, indicates that the RAS 2000 is very busy.
Off	Green	During normal operation, indicates serial port access.
Off	Red	CPU hangs trying to access the Ethernet controller.
Green	Yellow	Not Used

Legend:

All LED colors are assumed to be steady unless indicated otherwise.

Table 2-1: LED Codes (Continued)

Circle	Rectangle	Description
Green	Off	During normal operation, indicates network access.
Green	Green	RAS 2000 is completely booted and is idle
Green *(flashing)*	Green *(flashing)*	During normal operation, the LED's flash off to indicate serial port and Ethernet access. As the RAS 2000 gets busier, the lights remain off for longer periods of time.
Green	Red	Timer test failed
Red	Yellow	Error reading CCR
Red	Off	CPU Test Failed
Red	Green	CPU hangs accessing CCR and Timer registers
Red	Red	CPU hangs trying to write an error message
Red	Flashing: Red, Yellow, Green	Fatal Error: There will be one Red flash, followed by some number of yellow and green flashes. The number of flashes of each color indicates the type of error. See Table 2-2.

Legend:

All LED colors are assumed to be steady unless indicated otherwise.

Table 2-2: LED Fatal Error Codes

Circle LED	Rectangle LED: Number of flashes of each color:			
(steady)	Red	Green	Yellow	Description
Red	1	1	0	Bad Data path to DRAM
Red	1	1	1	Bad DRAM
Red	1	1	3	Bad CPU (Data Cache)
Red	1	1	4	Bad CPU (Instruction Cache)
Red	1	1	5	Bad DRAM data interface
Red	1	1	6	Bad DRAM address interface
Red	1	2	0	DRAM Data bits stuck on
Red	1	2	1	DRAM Walking-bit test failed
Red	1	2	2	DRAM Data bits stuck off
Red	1	2	3	16-bit DRAM accesses bad

Table 2-2: LED Fatal Error Codes (Continued)

Circle LED (steady)	Rectangle LED: Number of flashes of each color: Red	Green	Yellow	Description
Red	1	2	4	8-bit DRAM accesses bad
Red	1	2	5	DRAM refresh bad
Red	1	2	6	Processor byte-ordering incorrect
Red	1	2	7	Bad configuration NVRAM (FLASH)
Red	1	3	0	CPU error (UTLB miss)
Red	1	3	1	CPU error: unexpected exception
Red	1	3	2	CPU error: TLB failure
Red	1	3	3	Runaway Interrupts detected during P.O.S.T.
Red	1	3	4	Missing/Extra Timer Interrupts
Red	1	3	5	Missing/Extra Local UART Interrupts
Red	1	3	6	Missing/Extra UART Interrupts (expansion box 1 or 3)
Red	1	3	7	Missing/Extra UART Interrupts (expansion box 2)
Red	1	3	8	Missing/Extra Ethernet Interrupts
Red	1	4	0	Bad O.S. Checksum
Red	1	4	1	Ethernet Slave Interface bad
Red	1	4	2	Panic Message is being written to the console port (normally port 0). Take down the information and have it available. when you contact Computone Technical Support.
Red	1	4	3	Wrong software
Red	1	4	4	Ethernet DMA bad
Red	1	4	5	Ethernet CAM load error
Red	1	4	6	Ethernet timer too slow (or main timer too fast)
Red	1	4	7	Ethernet timer too fast (or main timer too slow)
Red	1	5	0	Ethernet Loopback Failed (data error)
Red	1	5	1	Ethernet Loopback Failed (data late)
Red	1	5	2	Ethernet Loopback Failed (other)
Red	1	5	5	Serial Loopback Failed (data error)
Red	1	5	6	Serial Loopback Failed (data late)

Table 2-2: LED Fatal Error Codes (Continued)

Circle LED (steady)	Rectangle LED: Number of flashes of each color: Red	Green	Yellow	Description
Red	1	5	7	Serial Loopback Failed (DSS error)
Red	1	5	8	Serial Loopback Failed (other)
Red	1	6	?	Reserved for IntelliCluster errors (unused by RAS 2000)

If the LED's Report an Error

If the RAS 2000 does not boot up properly and the LED's are reporting an error condition, record what the LED's are doing (which one is what color, whether they are steady or flashing, and if flashing, how many times of what color). Have this information at hand when you call Computone Technical Support to report the problem.

Special Note about Panic Messages

If you ever see an LED error code in which the circle LED is steady red, and the rectangle LED flashes 1 red, 4 green, and 2 yellow, this indicates that the RAS 2000 has encountered an unexpected software condition and is unable to continue. The RAS 2000 prints a more explicit error message on the console port (usually port 0) and usually includes a register dump. If you have a terminal connected to that port, take down any messages that are present before you restart the RAS 2000. Have this information at hand (plus the RAS 2000's software version number) when you call Computone Technical Support. these messages and the register dump can be used by our engineers to determine the cause of the failure.

When Power-on Self-Test is Completed

As soon as the power-on self-test is complete, both LED's on the main unit will turn green. Then you will see messages on your console terminal as shown in Screen 2-1.

Screen 2-1: Console Messages at RAS 2000 Boot Time

`Boot Loader, Release 2.0 Version 951103`
 	
`CPU Speed  = 20 MHz`
`I/D Cache  = 4k/2k`
	The boot loader displays basic information like the version number and date.

`Memory     = 2048k`
`Switches   = 0000`
`Fast Reset = Y (DRAM tests omitted)`
`Image Size = 449k/1017k`
	This RAS 2000 has 2048K, or 2 Megabytes, of DRAM. Some have 4 Megabytes.
	
	The operating system is stored compressed in PROM. This shows both the compressed and uncompressed size. While it is uncompressing a *tumbling cursor* is displayed after the compressed size.

`**********`
`Computone IntelliServer`
`Release 1.3.0 Version 951103`
 	
`Kernel Text/Data/Heap = 366k/43k/193k`
`Directory = 562k`
`Memory Size/Available = 2048k/172k`
`Internet Address = 0.0.0.0`
`Ethernet Address = 00:80:69:80:09:97`
`Serial Ports = 16`
`**********`
	Boot loader is finished: this banner comes from the RAS 2000's Operating System.
	
	Note the IP address is 0.0.0.0, because network parameters have not been configured yet.
	
	There is an Ethernet Address: every RAS 2000 has a unique one.

`Network boot enabled`
`Sending bootp...`
`Sending rarp...`
`Sending bootp...`
`Sending rarp...`
`Sending bootp...`
`Sending rarp...`
`No reply.`
	See section "Booting a New IntelliServer" on page 366 to understand why Network boot is enabled.
	
	The *bootp* and *rarp* messages will be repeated a few times, assuming you have not configured a BOOTP or RARP server.

Screen 2-1: Console Messages at RAS 2000 Boot Time (Continued)

`NOTICE: Booting prom kernel.`
 	
`Boot Loader, Release 1.3.0 Version 951103`
 	
`CPU Speed = 20 MHz`
`I/D Cache = 4k/2k`
`Memory = 2048k`
`Switches = 0000`
`Soft Boot = Y (DRAM tests omitted)`
`Image Size = 449k/1017k`
	Because there was no reply containing net-boot information.
	
	Here is the boot loader again. Had we been booting a kernel from the network, this would be *that* kernel's boot loader. Here is the same one.

`**********`
`Computone IntelliServer`
`Release 1.3.0 Version 951103`
 	
`Kernel Text/Data/Heap = 366k/43k/461k`
`Directory = 562k`
`Memory Size/Available = 2048k/544k`
`Internet Address = 0.0.0.0`
`Ethernet Address = 00:80:69:80:09:97`
`Serial Ports = 16`
 	
`**********`
`Sending bootp...`
`Sending rarp...`
`Sending bootp...`
`Sending rarp...`
	Comparing this message to the first, notice that the Kernel Heap was 193K the first time, and is now 461K. The available memory (for applications) was originally 172K, but is now 544K.
	
	The first time, the software had configured its memory in preparation for net-booting. The second time, knowing there would be no net-booting, it configured itself for normal operation.

`Sending bootp...`
`Sending rarp...`
`No reply.`
 	
`init: need ip address to start network`
`508 KB available memory`
 	
`#`
	Because there is still no IP address, it's again with the *bootp, rarp*. See section chapter 14, *Saving and Restoring Configurations* to see why this happens.
	
	After a reminder that you still don't have an IP address, you get a command prompt.

Your messages will not look exactly like this: software versions later than this printing may have different sizes and release dates. The number of serial ports will vary from site to site, and so on. Still, this gives you an idea what to expect.

Now What?

Now you have a command prompt. Where do you go from here? Have your RAS 2000 handy to experiment with as your are reading this guide for the first time. Try things out and see what happens. Allow yourself some time to absorb the information.

Do I Have to Read All This?

Probably not. This guide contains everything you need to know about the RAS 2000, but most people only need to know a few things, and then use them over and over. For example, if you are using a RAS 2000 in a business office to provide terminal login access to a single networked UNIX host, you probably don't need to read chapter 11, *Remote Network Configuration*. If you are an Internet Service Provider (*ISP)* then you need this chapter, but probably won't need to read chapter 13, *IntelliFeatures*. If you are already a network expert, you can probably skip chapter 9, *Network Basics*, but you still need chapter 10, *Local Network Configuration*.

This guide is organized so that later chapters build on earlier material. This is good news if you want to read this cover-to-cover. There are times, however, when you won't really understand something in an early chapter until you have read through the later chapters. Don't be alarmed when this happens, just make a mental note and move on. There are a lot of cross-references between sections. Please use them.

Road Maps — Beginner

To help you use this manual more efficiently, road maps are provided: short lists of things you need to do or to study in order to do a task.

First Stop

The RAS 2000 allows you to access your network through serial ports. **If you do not know an Ethernet address from an Internet address, stop here and skim the chapter 9, *Network Basics*.** Then, start back here and continue to read.

Getting the Console Terminal Working

Most of this guide does not make sense unless you are able to connect a terminal to your console port. What if you can't? Suppose you connect a terminal to port0, power up the RAS 2000, and do not get anything like the messages shown at the beginning of this chapter, or you do not get the command prompt:

Table 2-3: Troubleshooting Chart

Action	Description
Check the LEDs	After a minute or so, they should both be green, but flash off briefly when there is network or serial port activity. If they are some other color, this may indicate that the RAS 2000 did not pass its power-on self test. Be sure to note what colors and patterns the LED's *are* displaying before you call technical support.
Check for output on the terminal	Assuming the LED's *are* green, do you see *any* output on your terminal? If not, check your cabling. Make sure it is wired correctly. Make sure the terminal itself is OK by trying it out with other equipment. If the terminal is not near the RAS 2000, bring it into the same room so you don't have to make assumptions about the wires in the wall. Try ports 7, 8, and 15 on the RAS 2000 (you will not get the console messages, but should get a command prompt when factory defaults are in effect). Recheck the line speed (baud rate) and other parameters on your terminal to make sure they are correct.
Check terminal output for improper characters.	Perhaps you are getting output but it doesn't look right (garbage characters). If about half of the characters look right and the other half are invalid, recheck the character size and parity. The factory defaults assume 9600 baud, 8-bits, no parity. You can change this later, after you have things working. (If this is an RAS 2000 that has already been installed, but you have just started working with it, the settings may have already been changed. If you do not know what they were changed to, you may need to try some trial-and-error to hunt for the correct speed.)
Check terminal output for missing characters.	Most of the terminal characters are OK, but letters are sometimes missing. Check your terminal's flow control. By default, the RAS 2000 expects you to be using XON/XOFF flow control on the terminal. While most terminals keep up at 9600 baud even without flow control, some may not.
Check terminal output when you type.	Perhaps all the output looks ok, but when you type a command at the prompt nothing is echoed back? Check the cabling. Try ports 7, 8, or 15. Make sure the terminal is not configured to transmit at a different rate than it is receiving.
Check software version.	Perhaps you are booting an earlier version of software.

Getting on the Network Really Fast

Assuming you already have the following:

- A local Ethernet network with some UNIX hosts
- A terminal connected to the RAS 2000, showing a command prompt

Try to log into one of your UNIX hosts from the terminal.

Step	Action	Description
1	Choose the Internet address, netmask, and broadcast address	You you need to assign an IP address that is a member of your local network. Suppose there is just two hosts on your network: with addresses 160.77.99.1 and 160.77.99.2. The IP address 160.77.99.3 is not in use so you could use that address, and it is an address from the same network. If your network is not using a subnet, you don't have to enter the broadcast address and netmask; defaults will be assigned.
2	Assign the Internet address, netmask and broadcast address, if applicable.	Once you have chosen an IP address, you can assign it using the *set server* command. See "IP Address, Subnet Mask, Broadcast Address" on page 204 for details on how to do this. If the RAS 2000 does not have an IP address when it boots, it will not bring up the network software. As soon as you assign an IP address, the RAS 2000 displays a message (on the console port only) saying it is bringing up the network.
3	Ping one of the hosts on the network.	One of the other hosts' IP addresses was 160.77.99.1. At the command prompt, type **`ping 160.77.99.1`** and **press enter**. You should see a message such as *160.77.99.1 is alive*. If you don't read, *"Network Administration" on page 293*, for ideas on what might be wrong. Make sure the RAS 2000 is physically connected to the network, of course.
4	Log into one of hosts using telnet.	Once you can *ping* hosts on your network, try to log into one using the *telnet* command. Type (for example) **`telnet 160.77.99.1`** and **press enter**. See if the host will send you a login prompt. If he doesn't, there may be good reasons: maybe he isn't configured to do telnet. You will have to check this out on your own time.
5	Log in as some user known to that host.	You haven't told the RAS 2000 what kind of terminal you have attached, so the host will probably ask you.

This isn't much, but it's a start and it didn't take long. There are some users who don't use their RAS 2000 for any more than this.

Logging Into the RAS 2000

When the RAS 2000's factory defaults are in effect, there is only one user defined (root) and ports 0, 7, 8, and 15 are configured for auto-login as that user. That is the reason you get a command prompt immediately and not a login prompt. If you want a port to issue a login prompt, configure it as **Login by Port**, **Login by Port/TCP**, or as **Login by Screen** (see *"Functional Characteristics" on page 60*, for details).

Now that you have a login prompt, you need to define users to log in. Read "Configuring Users" on page 115 or "Logging Into the RAS2000" on page 20 until you understand that **a user is what happens when he logs in**. If you want the RAS 2000 to automatically log into some network host as soon as a particular user logs into the RAS 2000, you need to set up that user's *connections* (see "Configuring Users" on page 115).

If you have more than a hundred users to configure, or if you have several RAS 2000s supporting a common user base, or if you are an Internet Service Provider, you should consider configuring your users using RADIUS rather than storing them in the RAS 2000's NVRAM. Read "User Authentication using RADIUS" on page 415 to get an idea how to do this.

Male the Terminal Log in Automatically

If you want a particular terminal to automatically access a network host as soon as it comes up, without requiring a user to log into the RAS 2000:

1. Configure an NVRAM user so that he would do what you want when he logs in.
2. Enter this user name in the port's configuration, and set the port to *Auto-Login*.

Road Maps - Advanced

Since these tasks are a little more complicated, it is assumed that you have mastered the basics. This section lists the major areas you need to be concerned about.

Connecting a Printer

Use the following procedure to connect a printer to the RAS 2000:

1. Read chapter 18, *Reverse TCP and Printing* and chapter 5, *Configuring Serial Ports*.
2. Learn whatever you need to about how your host computer handles printing.

Do you need to configure a print spooler? Do you need to set up your application? The RAS 2000 can support printing using several methods, but it can't choose which method would be most suitable for your system.

3. Configuring the port as a regular login port with a terminal connected.
4. Reconfigure the port for printing but leave the terminal connected. This way you know the terminal, port and cable are good.
5. Check the terminal screen for "printer output." Once you can see "printer" output on the terminal screen, it's time to attach the printer (remembering to change the port configuration and cabling if necessary).

■ End of Procedure

Setting Up an Outbound PPP link

Use the following procedure to set up an outbound PPP link:

1. Read chapter 10, *Local Network Configuration*, and chapter 11, *Remote Network Configuration*. If you are new to network concepts, also read chapter 9, *Network Basics*.
2. Make sure the RAS 2000's Internet address, broadcast address, and netmask have been set.
3. Set the RAS 2000's **Syslog Priority** to **`LOG_INFO`** (see page 207). Most problems bringing up a PPP link can be solved by using the syslog file.

Set the **Syslog Host** to the IP address of a network host capable of syslogging or set it to **`console`** and put a terminal on your console port. Syslogging to a host is generally a better choice because your host's syslog daemon can be configured to automatically store the output in a file.

4. Create a dial script (see "Dial Scripts" on page 255). The dial script tells the port how to dial whatever modem is attached to it.
5. Configure the port as **Outbound Connection** (see page 71). Set the port's **dial script** to the name of the dial script created above. Set up the port for RTS input flow control and CTS output flow control, modem enabled. Make sure the modem is cabled appropriately.
6. Create a login script (see "Login Scripts" on page 259). This tells the RAS 2000 how to log into the host that is providing your dial-up connection.
7. Consider whether you need to create a PPP Options profile, or whether you can use the **default** one provided in the factory defaults. See "Options Profiles" on page 263.
8. Create a remote profile.

For outbound connections, you must provide remote address (the IP address at the other end of the link), interface address (the IP address of this end of the link), interface type (must be outbound), protocol (must specify either PPP, SLIP, or CSLIP), and login script (supply the name of the one you created above). See "Remote Profiles — Concepts" on page 271. You must also specify either a port or a group so that the RAS 2000 will know on which serial port to dial out.

9. Enter a phone number in this remote profile if the dialer script is configured to use it, and should contain the name of your option profile, if different from **default**.
10. Save your configuration and reboot the RAS 2000 for these new profiles to be recognized.
11. Bring up an outbound connection; **ping** the IP address you entered as the **Remote Address** above.

When the RAS 2000 detects network activity for this address, it attempts to bring up the link. It first runs the dialer script, then the login script, and then (for PPP links) attempts to initiate PPP option negotiation, finally bringing up the connection. This first *ping* will most likely time out (that is, fail) before the link has time to be established. This is normal. A second *ping* should succeed, however.

12. Look at your syslog output for clues, if the link doesn't come up.

13. Check to make sure the RAS 2000's **route** command shows a route to the **Remote Address** through the appropriate PPP/SLIP interface. Read chapter 12, *Network Administration*, for other troubleshooting ideas.

■ End of Procedure

Setting up an Inbound PPP link

Use the following procedure to set up an inbound PPP link:

1. Read *chapter 10, Local Network Configuration* and chapter 11, *Remote Network Configuration*. If you are new to network concepts, read chapter 9, *Network Basics* also.
2. Make sure the RAS 2000's Internet address, broadcast address, and netmask have been set.
3. Set the RAS 2000's **Syslog Priority** to **`LOG_INFO`** (see page 207). Most problems bringing up a PPP link can be solved by using the syslog file.
4. Set the **Syslog Host** to the IP address of a network host capable of syslogging or set it to **`console`** and put a terminal on your console port. Syslogging to a host is generally a better choice because your host's syslog daemon can be configured to automatically store the output in a file.
5. Set up a user.

You can set up a user in the RAS 2000's NVRAM or on a network host running a RADIUS server. See chapter 7, *Configuring Users* and chapter 17, *User Authentication using RADIUS*. Remember that this **user** does not necessarily correspond to a particular person. The computer that wants to dial up and start a PPP connection will be running a login script that will log in as this user name. When the RAS 2000 sees that user name, it looks up that user's information, which contains the necessary details for bringing up the link.

6. Specify whether the user will use PPP, SLIP, or CSLIP if you are configuring a user through NVRAM.
7. Configure a serial port as **Login by Port**, **wait** (see page 68). Configure it with: modem enabled (see page 78), RTS input flow control (see page 77), and CTS output flow control (see page 76).

Note that the port configuration is really the same as for any dial-in user. If a user configured for telnet access were to dial in, he'd get a telnet connection. But when a user configured for PPP/SLIP/CSLIP logs in, the RAS 2000 tries to bring up a connection.

8. Create an options profile (see page 263) if different settings from the **default** are required.
9. Create a remote profile. You must supply the **interface address** which is often the same as your RAS 2000's IP address. You must also specify the **interface type** as **inbound**.
10. Set a remote address for the user if the user is configured in NVRAM.

If you want to make sure a particular user is assigned a particular remote address, you have to specify that user's name in the remote profile (see page 285). If the user was configured using RADIUS, there is more flexibility. See *Assigning Remote Profiles* on page 290 to see how user information learned from a RADIUS server is reconciled with information in the remote profile.

11. **Save your configuration and reboot the RAS 2000 for these new profiles to be recognized.**
12. Connect a terminal to the port, as a test, instead of your modem (changing cabling as appropriate).

Make sure the terminal's line speed and other settings match the settings for your port. See that you get a login prompt. Manually log in as the PPP/SLIP/CSLIP user you created. The RAS 2000 should present you with a banner (see page 278). This proves that the RAS 2000 recognizes the user as wanting a PPP/SLIP/CSLIP connection and has obtained the necessary information from RADIUS.

13. Disconnect the terminal and connect your modem, using the appropriate cable.
14. Do a **hangup port** port number command to kill off the session you had started with your terminal, in case it is still alive. Let your remote site try to dial up the RAS 2000.

■ End of Procedure

If there are problems bringing up the link, the **`whodo all`** command indicates whether the user was recognized as having logged in. You can also use the **ppp-stat interface-name** command to determine whether the RAS 2000 thinks the link is up. Look at your syslog output or syslog file to see if there may have been PPP negotiation problems, and in general to see how far along you got. Use the **`route`** command to ensure that a route was established corresponding to this link. Refer to chapter 12, *Network Administration* for other commands and techniques you might use.

When Things Get Really Dicey

If you are working on a configuration problem of great complexity and have read all the documentation, think you are doing everything right but still things are not working, what should you do? One possibility is to call Computone's tech support department and try to explain everything over the phone. This method is quite good for problems of mild to intermediate complexity. For harder problems it becomes increasingly difficult to convey all the nuances over the phone, so we recommend the you:

1. Save your RAS 2000's configuration to a host file.
2. Assemble any files containing relevant information.

These may include extracts from your syslog file, RADIUS user and client files, debugging output from various sources, lists of symptoms, and other information, as well as the RAS 2000 configuration file.

3. Combine the files into a single file (generally using *tar* or a similar utility).
4. FTP the resulting file to our site: **ftp.computone.com:/incoming/** is the host name and directory to use. Give your file a name that another customer would be unlikely to use.
5. Send email to **support@computone.com** explaining the nature of the problem and telling us what name you gave your FTP'd file. We will reply via return email.

CHAPTER 3

Using The Commands

In this chapter, you learn some general rules for using the command-line interface to the RAS 2000. If you are not already familiar with the command-line interface, you should read this chapter carefully. In later chapters you will learn how to use commands to configure your RAS 2000, and they assume that you are already familiar with these rules.

Why a "shell"

The RAS 2000's command-line interpreter is also called the "shell", because it is similar in function to the "shells" in UNIX systems. These were called "shells" because they were the part of the operating system the user would see, as opposed to the "kernel", which was the part he would eat. Anyway, the name has stuck; when you are configuring a user and want him to be able to type commands at an RAS 2000 prompt, you configure one of his *connections* to be **`shell`**.

Type Your Command at the Prompt

The command-line prompt consists of the name of the RAS 2000 Commnuications Server followed by a **#** sign or a **$** sign. The **#** indicates that you have *administrative privileges*, and the **$** that you do not. Users who are not configured to have administrative privileges are not allowed to make certain configuration changes.

When you see the prompt you can type your next command. When you are using the command shell, you are typing in **line mode**, which means that you enter a line of text, backspacing and re-typing as necessary, finally pressing the **Enter** or **Return** key to send the command. When you are in **line mode** there are also three special control keys you may use:

Table 3-1: Line-Mode Control Keys

Default	Name	changing the default	Action
`^C`	**Intr**	see chapter 5, *"Show Port" on page 64*	Partially-typed command ignored, new command-prompt. Command in progress may be stopped.
`^H (backspace)`	**Erase**	see chapter 5, *"Show Port" on page 64*	Backspaces the cursor and erases the last character typed.
`^U`	**Kill**	see chapter 5, *"Show Port" on page 64*	Partially-typed command ignored, spaces to next line for fresh command. No effect on commands in progress,

You may want to change these codes to be more suitable for your terminal or lifestyle. Some terminals, for example, send the **DEL** code from their *backspace* key instead of the more usually backspace code, **^H**. Some operators are used to hitting **DEL** as an interrupt character instead of typing **^C.** To change the defaults, see the page references in the table.

Conventions for Describing Commands

In this chapter and other chapters you see words appearing in an odd array of typefaces and styles. Here is a summary.

- Words in **Courier Bold** are to be typed as they appear.
- Words in *Italics* are descriptions which you replace with something more specific, depending on context.
- Brackets—[]—indicate that what is inside is optional.
- Curly brackets—{}—indicate that what is inside may be repeated one or more times.
- Words separated by a vertical bar—|—indicate choices; you may choose one.

Three Types of Words

RAS 2000 commands contain three types of words that are treated specially, the **command word**, the **verb**, and **keywords.**

Consider the following command:

```
set server domain computone.com
```

- "**Set**" is the verb and tells what action to perform. You are "setting" the server's configuration, not showing it, for example.
- "**Server**" is the command word which indicates the topic of the command. You are setting a "server" parameter, not a port or user.
- "**Domain**" is a keyword and describes which server parameter you are setting.
- "**Computone.com**" is not a command or a verb or a keyword; it is the specific value you are giving to the keyword, **domain**.

There are certain verbs you use again and again:

Table 3-2: Commonly Used Command Verbs

Verb	Definitions and Usage
show	Display the current values. The command **show port 3** displays the current configuration of port 3
set	Change one or more values for an existing record. The command **set user leary admin enabled** turns on administrative privileges for the existing user, leary.
add	Add a new record. This is used for large tables where the entries are identified by some name you have given them. For example, **add user tim** would add a new user to the user table.
delete	Remove an existing record. Like **add** this is used for tables where the entries are identified by a given name. For example, **`delete user tim`** gets rid of the entry created above.
kill, **hangup, output**, **echo**	These are additional verbs used only in certain commands.

Syntax Conventions

There are some conventions that are common to all the commands. Some of these are similar to conventions found in UNIX and DOS command shells, others have been added for flexibility and convenience.

Table 3-3: Syntax Examples

Example	Syntax Rules
`set port 0 modem enabled` `port set 0 modem enabled`	If one of the standard verbs is used, either the command word or the verb may come first. The following two commands are identical:

`SET PORT 0 MODEM ENABLED`
`set port 0 modem enabled`
`SeT PorT 0 MoDEm EnabLED`
	Command words, verbs, and keywords may appear in either upper or lower case. Specific names you have given to things are usually case-sensitive, and so must be entered in the correct case. The following three commands are identical.

`set port 0 username Leary` `set port 0 username leary`	These two are not equal because *leary* is a value, not a keyword.

Table 3-3: Syntax Examples

Example	Syntax Rules
`show route` **`route`**	When the command contains only the command word and the verb "show", the verb may be omitted. The following two commands are the same.
`set port 1 comment "Computer Room Terminal` **`set port 1 comment Computer\ Room\ Terminal"`**	Spaces are used to separate the individual elements of a command. If you need to supply a value that *contains* a space, you need to use one of two methods: Enclose the entire value between two double-quotes, or place a back-slash before each space
`set radius host 1 ""` **`set radius host 2 ""`**	To set something to *nothing*, let the value be a pair of double-quotes. This can be done whenever it is legal for a parameter to be blank. In this example, we are disabling RADIUS authentication by setting both of the hosts to **nothing**..

Help

Type **`help`** at the command prompt, and the RAS 2000 displays a list of the commands. Type **`help`** *`command-name`* at the command prompt for more detailed help on a particular command. The help screens use a similar array of brackets, braces, and so on, as what is used in this guide. There are two differences:

- Angle brackets—<>—are used where *italics* would be used in this book: to indicate *parameters* that you will replace with something more specific.
- The plus sign—**+**—appears to the left of certain commands. This indicates that the command (or this form of the command) can only be done by users with "administrative privileges" (see chapter 7, "Configuring Users: General Issues" on page 116). **The plus sign doesn't get typed.**

Sample Help Screen

Let's look at a typical help message (somewhat abridged):

Example 3-1: Help for the "remote" Command

```
# help remote
     remote      - Modify/Display a PPP/SLIP remote interface profile

+remote set <remote name>|<iface name> {parameter <value>}

        {parameter <value>} options are:
        [ifaddr <local ip address>]   [netmask <ip addr mask>]
        [type inbound|outbound|disabled]
        [address <remote ip address>]
        [port <number>|none]   [group <number>|none]
        [mtu <size>]   [async <mask>]

+remote show <remote name>|<iface name>|all|summary
```

In this example, the **+** signs before the word **remote** indicates that only users with administrative privileges can do these commands. In the **remote set** command, you can specify either the name of the remote or its interface name. The expression **{parameter <value>}** indicates that you should list one or more pairs consisting of a keyword and new value for that keyword. Next, the keywords themselves are listed with descriptions of what values may be applied. The brackets around **[type|inbound|outbound|disabled]** indicate that this keyword-value pair is optional (if you don't specify it the original values are unchanged). The keyword is **type** and the value must be either **inbound**, **outbound**, or **disabled**. In another example, the value supplied for the keyword **port** may be either the port number or the word **none**.

The last line is help for the **remote show** command:

```
remote show <remote name>|<iface name>|all|summary
```

There are no brackets to indicate that any of these parameters are optional, so you *must* supply either the remote name, its interface name (either of which uniquely identifies a remote profile), or one of the keywords **all** or **summary**.

Pagination

Many of the commands are capable of generating lots of output—certainly more than a single screen can hold. There is a simple method you can use to break this output into pages. At the end of the command-line, add a single vertical line—**|**—before you hit *enter* or *return*. This will cause the output to be broken into 23-line pages, pausing for keyboard input after each screenful. If you like a different number of lines at once, enter that number after the vertical bar. For example, **show user all | 10** displays information on all the users, 10 lines at a time. Anything other than a number after the vertical bar is ignored. This is done so that when you accidently type **| more** at the end of the command, no harm done.

When the pager is waiting for you to press a key, the key you press influences what the pager does next.

Table 3-4: Special Paging keys

Enter or Return	Display the next line of output
Spacebar	Display the next page of output
Intr (see page 28)	Terminate the command
Any other key	Exit the pager: remaining output continues unpaged.

About Specific Commands

The chapters that follow are grouped according to what you are learning to configure. When a setting can be changed using the menu or the command line, both methods are described together, along with explaining what the setting means. At the end of this chapter, there is an alphabetical table of all the commands, with page references to more complete discussions.

Table of Commands

The following table summarizes the commands and shows where each is explained in more detail. You can get a summary of commands by typing **`help`** at the command prompt, and a detailed summary of the syntax for a specific command by typing **`help`** *`command-name`*. The syntax summaries shown in this table are abbreviated versions of the ones in each command's help screen. The most detailed information is found in this manual near the indicated pages.

Table 3-5: Summary of Commands

Command	Usage	Description
`arp`	**`show arp`** *`ip-address | hostname`* **`add arp`** *`ip-addr | hostname`* *`ethernet-addr options…`* **`delete arp`** *`ip-address | hostname`*	Display or modify the RAS 2000's ARP table. See chapter 12, "ARP Table" on page 306 for details.
`boot`	**`show boot`** **`+set boot type disabled|bootp|tftp`** **`+set boot primary`** *`host boot conf`* **`+set boot secondary`** *`host boot conf`* **`+set boot retry`** *`count`*	Display or modify the RAS 2000's boot configuration. See chapter 10, *"Configuring Bootstrap Options" on page 217*.
`broadcast`	**`broadcast all`** | *`port-list message`*	Sends the message to all specified ports. See chapter 15, "Broadcast" on page 377 for more information.
`clear`	**`clear`**	Clears the current screen
`connection`	**`show connection`** *`[connection no.]`* **`+add connection`** *`type [arguments]`* **`+set connection`** *`number type [args]`* **`+delete connection`** *`number`*	Display or modify the global connection table. See chapter 7, *"Connection Option" on page 124* for more information.

`Bold-face keywords` ... Enter as shown.
`Italics`..... Supply a value.
`{keyword value}`..... Enter a keyword and its value, one or more times.
`Scylla | Charybdis` ... Vertical bar means you must enter one of the choices
`+` ..... plus sign indicates you must be administrator.

Table 3-5: Summary of Commands (Continued)

Command	Usage	Description
dial	**show dial** *scriptname* **+set dial** *scriptname* **line** *number text* **+add dial** *scriptname* **line** *number text* **+delete dial** *scriptname*	Display or modify the table of modem dialer scripts (Used for dial-out PPP and SLIP connections). See chapter 11, sections *"Dial Scripts" on page 255* and "Shell Commands for Dial Scripts" on page 257.
env	**show env** **env**	Display current environment variables. See chapter 15, *"Env" on page 391* for more details.
exit	**exit**	Logs you out. If you are logged in over a modem, the phone hangs up. See chapter 8, *"Logging Out" on page 163* for more information.
filter	**show filter** **show filter** *filtername* | *interface* **+set filter** *filtername* **rule** *number parameters* **+add filter** *filtername [interface] [parameters]* **+delete filter** *filtername* **+attach filter** *interface filtername* **+detach filter** *interface*	Display or modify the table of IP filters and the rules these filters contain. Dynamically attach or detach a filter from an interface. See chapter 10, "IP Filters" on page 239 for more information.
gateway	**show gateway** **+add gateway** *destination gateway* **+delete gateway** *destination gateway*	Display or modify table of *gateways*, or static routes which are loaded at start-up. See chapter 10, "Gateway Command" on page 228 for more information.
help	**help** **help** ***command***	Display a summary of all commands or the syntax of a single command. See *Sample Help Screens* section in this chapter.
host	**show host** **+add host** *hostname ip-address* **+delete host** *hostname* | *ip-address* **+set host** *hostname ip-address*	Display or modify the table of locally-defined host names.

Bold-face keywords ....Enter as shown.
Italics.....Supply a value.
{keyword value} .....Enter a keyword and its value, one or more times.
Scylla | Charybdis ....Vertical bar means you must enter one of the choices
+.....plus sign indicates you must be administrator.

Table 3-5: Summary of Commands (Continued)

Command	Usage	Description
login	**show login** *scriptname* **+set login** *scriptname* **line** *number text* **+add login** *scriptname* **line** *number text* **+delete login** *scriptname*	**Not the opposite of "logout".** Display or modify the table of login scripts (used by dial-out PPP and SLIP connections). See chapter 11, *"Shell Commands for Login Scripts" on page 261*.
logout	**logout**	**Not the opposite of "login"**. Another name for Exit. See chapter 8, "Logging Out" on page 163 for more information.
menu	**menu**	Enter the Menu system.
modeminit	**show modeminit** **+set modeminit** *<1-8> name commands*	Display or modify the table of modem initialization commands. See chapter 6, "Initialization Table" on page 107 for more information.
motd	**show motd** **+set motd line** *number message*	Display or modify the *message-of-the-day*, the message that is displayed to a user after he logs in. See chapter 8, "User Configuration Summary" on page 148.
nameserver	**show nameserver** **+add nameserver** *ip-address* [**port** *n*] **+delete nameserver** *ip-address*	Display or modify the table of domain name servers. (Where to send name resolution requests.) See chapter 9, "Nameservers and Other Domains" on page 179.
netstat	**netstat all** **netstat tcp | ip | icmp | udp** **netstat route | sonic | ppp | slip** **netstat connections**	Display tables of network statistics. See chapter 12, "Network Statistics" on page 313 for more information.
network	**show network** **+add network** *name ip_address* **+delete network** *name* **+set network** *name ip_address*	Display or modify table of locally-defined network names. See chapter 10, "Network Addresses" on page 213 for more information.

Bold-face keywords ... Enter as shown.
Italics ..... Supply a value.
{keyword value} ..... Enter a keyword and its value, one or more times.
Scylla | Charybdis ... Vertical bar means you must enter one of the choices
+ ..... plus sign indicates you must be administrator.

Table 3-5: Summary of Commands (Continued)

Command	Usage	Description
password	**password** **+password** *username*	Change your or someone else's password.See chapter 7, "Password" on page 123 for more information.
ping	**ping** *hostname* | *ip-address*	See if a host is reachable by sending an ICMP echo request. See chapter 12, "Checking Routes with Ping" on page 294 for more information.
port	**show port** *port-list options* **set port** *port-list {keyword value}* **set port** *port-list* **from** *port*	Display or modify the configuration of a serial port. See chapter 5, *"Configuring Serial Port Parameters" on page 67* for more information.
port	**kill port** *port-list* **hangup port** *port-list*	Terminate processes running on selected serial ports. See chapter 15, "Kill, Hangup Port" on page 376 for more information.
port	**output port** *port options* **echo port** *port*	Send test data to a port, or place the port in remote echo mode. See chapter 15, *"Echo Port" on page 376* for more information.
pppoption	**show pppoption** *profile* | **all** **+set pppoption** *profile {keyword value}* **+add pppoption** *profile {keyword value}* **+delete pppoption** *profile*	Display or modify table of pppoption profiles.
pppstat	**pppstat** *remote-name* | *interface-name* **pppstat all**	Display statistics for PPP and SLIP connections. See chapter 12, *"PPP (And Slip) Statistics" on page 321* for more information.
preamble	**show preamble** **+set preamble line** *number message*	Display or modify the *preamble*, the message displayed before a user logs in. See chapter 8, "Menus" on page 154 for more information.

Bold-face keywords ....Enter as shown.
Italics.....Supply a value.
{keyword value} .....Enter a keyword and its value, one or more times.
Scylla | Charybdis ....Vertical bar means you must enter one of the choices
+.....plus sign indicates you must be administrator.

Table 3-5: Summary of Commands (Continued)

Command	Usage	Description
production	**+set production enabled|disabled**	Enable or disable *production mode*. When production mode is enabled (the default) the **queues**, **ps**, and **streams** commands are not present. See chapter 15, "Production Command" on page 381 for more information.
profile	**show profile iview** *name* **+add profile iview** *name {keywd value}* **+set profile iview** *name {keywd value}*	Display or modify an IntelliView profile.See chapter 13, "Commands" on page 329 for more information.
profile	**show profile iprint** *name* **+add profile iprint** *name {keywd value}* **+set profile iprint** *name {keywd value}*	Display or modify an IntelliPrint profile.
profile	**show profile iset** *name* **+add profile iset** *name {keywd value}* **+set profile iset** *name {keywd value}*	Display or modify an IntelliSet profile
profile	**+add|set profile iview|iset|iprint** *name* **from** *name*	Copy configuration from one IntelliFeatures profile to another (existing or new).
ps	**ps**	Display a list of all processes (used for troubleshooting—not present in production mode). See chapter 15, "ps" on page 382 for more information.
queues	**queues**	Display a list of all data queues (used for troubleshooting - not present in production mode). See chapter 15, "Queues" on page 385 for more information.
radius	**show radius** **+set radius** *{keyword value}*	Display or modify settings for RADIUS authentication and accounting. See chapter 7, "RADIUS Menu and Commands" on page 140 for more information.

Bold-face keywords ... Enter as shown.
Italics ..... Supply a value.
{keyword value} ..... Enter a keyword and its value, one or more times.
Scylla | Charybdis ... Vertical bar means you must enter one of the choices
+ ..... plus sign indicates you must be administrator.

Table 3-5: Summary of Commands (Continued)

Command	Usage	Description
remote	**show remote all** **show remote summary** **show remote** *name*	Display settings of one or more remote profiles and associated interfaces. See chapter 11, "PPP/SLIP Menu" on page 251 for more information.
remote	**+set|add remote** *name {keyword value}* **+delete remote** *name*	Modify configuration of a remote profile. See chapter 11,"PPP/SLIP Menu" on page 251 for more information.
restore	**+restore** **+restore** *host-name file-name* **+restore factory**	Restore configuration from NVRAM, from a host (via TFTP) or to factory defaults. See chapter 14, "Saving & Restoring: Menu and Commands" on page 357 for more information.
rhosts	**show rhosts**	Display current rhosts settings. See chapter 8, "Commands Other Than cat" on page 160 for more information.
rhosts	**+set rhosts enabled | disabled**	Allow or disallow rsh commands (other than rsh, cat) from remote hosts.
rhosts	**+add rhosts** *ip-address* **+delete rhosts** *ip-address*	Specify IP address of host who is allowed to issue rsh commands.
rip	**show rip** **+add rip host** *ip-address* **+delete rip host** *ip-address* **+set rip list accept | reject** **+set rip enabled | disabled**	Display or modify RIP configuration: global settings. See also *server* and *remote*. See chapter 10, "RIP Configuration" on page 234 for more information.
rlogin	**rlogin** *host options*	Connect to a host via RLOGIN protocol. See chapter 16, "Rlogin" on page 408 for more information.
route	**show route** **+add route** *destination gateway* **+delete route** *destination gateway*	Display or modify current routing table. See chapte 12, "Routing Table" on page 309 for more information.

Bold-face keywords ....Enter as shown.
Italics.....Supply a value.
{keyword value} .....Enter a keyword and its value, one or more times.
Scylla | Charybdis ....Vertical bar means you must enter one of the choices
+.....plus sign indicates you must be administrator.

Table 3-5: Summary of Commands (Continued)

Command	Usage	Description
save	**+save** **+save** *host-name file-name*	Save configuration to NVRAM or to a host via TFTP. See chapter 14, "Saving & Restoring: Menu and Commands" on page 357 for more information.
server	**show server** **+set server** *{keyword value}*	Display or modify settings which apply to the RAS 2000 overall, or to its Ethernet interface. See chapter 10, "Displaying the IntelliServer Configuration" on page 202 for more information.
services	**show services** **+set services** *name* **port** *number* **+add services** *name* **port** *number*	Display or modify table of names for TCP/UDP service ports (*unrelated to serial ports*). See chapter10, "Show Services" on page 232 for more information.
shutdown	**shutdown now** *message* **showdown** *minutes message*	Send a message to all logged-in ports, then re-start the RAS 2000 immediately or after a delay. See chapter 15, "Shutdown Command" on page 378 for more information.
snmp	**show snmp** **+add snmp traphost** *ip-address* **+delete snmp traphost** *ip-address*	Display or modify current SNMP configuration. See chapter 10, "SNMP Configuration" on page 222 for more information.
streams	**streams**	Display a list of stream-buffer usage (used for troubleshooting—not present in production mode). See chapter 15, "Streams" on page 383 for more information.
systat	**systat** **systat** *seconds*	Display CPU and memory-usage statistics. See chapter 15,"Systat" on page 380 for more information.

Bold-face keywords ... Enter as shown.
Italics ..... Supply a value.
{keyword value} ..... Enter a keyword and its value, one or more times.
Scylla | Charybdis ... Vertical bar means you must enter one of the choices
+ ..... plus sign indicates you must be administrator.

Table 3-5: Summary of Commands (Continued)

Command	Usage	Description
telnet	**telnet** *host options*	Log into a host using TELNET protocol. See chapter 16, "Telnet" on page 396 for more information.
term	**show term** *number* **+set term** *number {keyword value}*	Display or configure a user-defined terminal type. See chapter 5, "User-defined Terminal Types" on page 90 for more information.
tip	**+tip** *port*	Connect directly to a serial port. See chapter 6, "Initializing Using TIP" on page 110 for more information.
udp	**show udp** **set udp checksum enabled|disabled**	Enable or disable UDP checksum checking. See chapter 15, "Udp" on page 393 for more information.
user	**show user** *name* | **all** **set user** *name {keyword value}* **set user** *name* **from** *name* **+add user** *name {keyword value}* **+add user** *name* **from** *name* **+delete user**	Display or modify a locally-defined user, or copy one user's configuration to another's. See chapter 7, "Configuring Users: General Issues" on page 116 for more information.
version	**version**	Display the version of the RAS 2000's software that is currently running. See chapter 15, "Version" on page 393, for more information.
whodo	**whodo** **whodo all**	Display a list of ports and how they are being used. See chapter 15, "Whodo" on page 379 for more information.

Bold-face keywords ....Enter as shown.
Italics.....Supply a value.
{keyword value} .....Enter a keyword and its value, one or more times.
Scylla | Charybdis ....Vertical bar means you must enter one of the choices
+.....plus sign indicates you must be administrator.

CHAPTER 4

Using the Menu Interface

In this chapter, you will learn general rules for using the menu interface to the IntelliServer. If you are not already familiar with our menus and forms, you should read this section because later chapters assume you are familiar with this material.

Menus and Forms: General Description

The menu interface consists of two types of screens: *Menus* and *Forms*. Menu screens allow you to select what you will do next, such as connect to a host, work with a configuration form, or see yet another menu. Form screens allow you to view and modify configuration settings for a particular item.

Unlike the command line interface, the menu interface is screen-based. This means that your terminal must be capable of cursor addressing, highlighting, and other functions. Equally important, the menu interface needs to understand what type of terminal you are using so it can send the proper commands to your terminal to perform these functions.

Tip	If your terminal displays "garbage" when you are trying to use the menu interface, but not at other times, make sure the port is configured for a terminal type that matches your actual terminal and that your terminal is operating in the emulation mode you think it is. See "Terminal Descriptions" on page 84 and"User-defined Terminal Types" on page 90.

The menu interface also makes use of control keys, arrow keys, and function keys. *Control keys* are entered by holding down the **`ctrl`** key on the keyboard and pressing a letter key. Control keys are indicated by using **`ctrl-`** or **`^`**, as in **`ctrl-E`** or **`^E.`**

The *arrow keys* have actual **`up-arrow, down-arrow, left-arrow`** and **`right-arrow`** symbols on them. *Function keys* are the keys usually labeled **`F1, F2, F3`**…on the keyboard and are labeled the same in this chapter. If your terminal does not support all the necessary arrow and function keys, there are pre-defined *control* keys you can use instead.

Starting the Menus

You can enter the menu interface in two ways: by typing the **`menu`** command at the command prompt or by configuring a user to automatically enter the menu interface when logging in.

Terminal Type

As soon as the menu interface starts, it tries to determine from the serial port's configuration what type of terminal you are using (page 84). If the local terminal type has been configured as *unknown*, the menu interface asks you to supply a terminal type now. (Serial ports connected to modems are frequently configured with an *unknown* terminal type because different types of terminals may connect at different times).

Screen 4-1: Prompting for Terminal Type

```
Select a terminal type from the list below and press <ENTER>:

  0 - ansi
  1 - wyse30
  2 - wyse50
  3 - wyse60
  4 - vt100
  5 - xterm
  6 - uterm0
  7 - uterm1
  8 - uterm2
  9 - uterm3
 10 - unknown
 11 - exit
```

Path: **First entering the menu interface if terminal type is unknown**

Conventions

This illustration of a menu screen uses some conventions which continue through the manual:

- The title (*"Prompting for Terminal Type")* is not actually part of the screen but is a label provided for clarity.
- The box at the bottom, marked "**Path:**" also does not appear on the screen. It is provided to show how to reach this screen.
- The horizontal bar between the prompt and the menu selections does not appear on the screen, it is shown here for clarity.

On This Screen

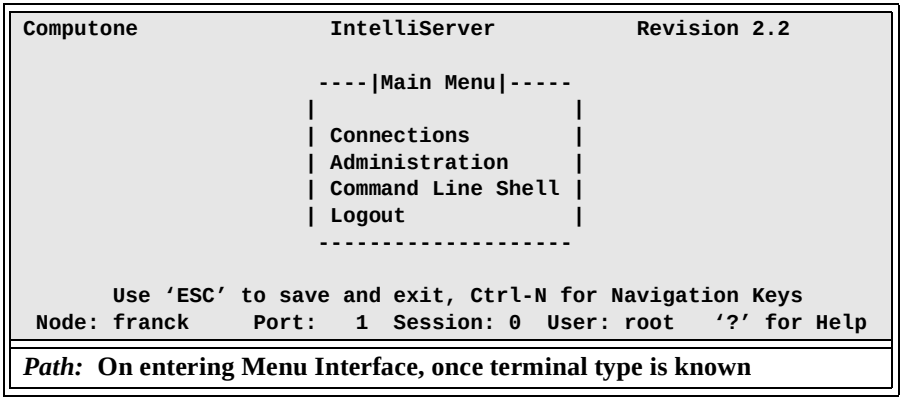
In this list, the **uterm** entries refer to user-defined terminal types which you learn to configure in chapter 5, *Configuring Serial Ports*. Your terminal does not have to be literally something from this list. It only needs to emulate one of these terminals closely enough to paint the screen correctly and recognize special keys. For example, PC-based terminal emulators frequently support ANSI-compatible and WY60-compatible emulation modes. If your terminal does not emulate one of the supported types and you select **unknown,** you are informed that you cannot use the menu interface and are logged out (the same as if you had selected **exit**).

There is one situation where you do select *unknown*. If you have configured a user to have a **Selected Connection Menu** or **Global Connection Menu**, you are given list of connections to choose from. You can either telnet to this host and do this, or rlogin to that host and do that. When the terminal type is known, these connection menus are presented in screen-based format, but when unknown the menus are presented in a line-by-line format. (See chapter 7, "Configuring Users: General Issues" on page 116).

Main Menu

When the menu interface learns what type of terminal you are using, it can display the main menu screen shown here:

Screen 4-2: Main Menu



This first screen illustrates elements that are common to all menu screens and forms. In future illustrations, only the center portion is shown. The white box marked "Path" shown in this illustration and later ones does not appear on your screen, it shows you how you reached this menu. Table 4-1 explains Screen 4-2.

Table 4-1 Screen Element Definitions

Screen Element	Definintion
Manufacturer and Product	*Computone IntelliServer* in this example.
Software version	In this example, 2.2. If you are having difficulty troubleshooting a problem, find out which version you are running before you call the tech support line.
Screen Body	Centered in the middle is the **body** of the screen, the only part shown in the future. It will be surrounded by a nice border if your terminal supports the appropriate line-drawing characters, or a border something like the one shown here if it doesn't.
Help Line	**Instructions & Help:** *"***Use ESC to save…***"* in this example. When you press the **?** or the **F1** key to display the help line for a selected field, this is where it appears.
Bottom Line	**Node name:** The name you have given this IntelliServer (its **Host Name**), here called *franck.* **Port:** The number of the serial port you are using: port 1 in this example. **Session:** The session number (will be zero unless you are using IntelliView). **User:** The name you gave when you logged in.

Before exploring the Main Menu screen in more detail, the different types of menus and forms are explained, as well as how to use special keys to move around in them.

Navigating in Menus and Forms

There are two types of screens in the menu interface: **menus** and **forms**. **Menus** give you a list of choices and when you select one of those choices you are given another screen, either a menu or a configuration form. **Forms** allow you to view and modify the current configuration of some item. You can move about from item to item making changes and when you are done, you choose whether to accept the changes you have made.

Menus

When you are in a menu screen, one of the choices is highlighted. Depending on your terminal it may be displayed in reverse-video, or brighter, or a different color. There are three things you can then do: choose what's highlighted, highlight something else, or go back to the parent menu screen. Here is a summary of the keys you can use.

Table 4-2 Menu Navigation Keys

Keys	Description
Enter **Return**	Choose the highlighted selection.
Tab **Right-Arrow** **Down-Arrow**	Highlight the next selection.
Left-Arrow **Up-Arrow**	Highlight the previous selection.
A...Z **a...z**	Highlight the next selection that begins with this letter.
Esc	Return to the parent menu screen. The *parent* menu is the one that originally led to the present menu, not necessarily the last screen you viewed. (For example, if you were on *menu 1* and selected *menu 2*, and then from *menu 2* selected *menu 3*, pressing **Esc** would return you to *menu 2*, and the next **Esc** would return you to *menu 1*.
F1 **?**	Displays a help message for this menu, if available.
ctrl-N	Display a list of navigation keys.

Forms

A form screen (or "configuration form") contains several data items whose values are displayed and which you may be able to modify. Each data item consists of a description and an **input area** where the item's current value is displayed. When you select an item to modify, its *input area* will be highlighted and you can then modify that value. When you have selected and modified everything you want to change, you then decide whether or not to accept the changes you have made.

Before telling you how to do all this, there are six different types of forms and three different types of data items. The rules vary slightly depending on which type of form and data items you are encountering.

The six types of forms are:

1. **Tables** are forms in which data is arranged in columns. For example, the **Nameservers** form is a table that contains the host IP address in the first column, and the TCP service port in the second column.
2. **Multi-page Tables** are tables which would have too many rows for a single screen and so the entries are presented one screen at a time. The **Host Addresses** form is one of these and contains several pairs of host names and IP addresses. If you modify something on one page of a multi-page table, you are asked whether to accept the changes before you can move to a different page.
3. **General Forms** are forms where the data is not arranged in columns. An example would be **IntelliServer Configuration** form, which contains a variety of things to configure about the IntelliServer.
4. **Multi-record Forms** are used when you need to provide a lot of information about many things. It is the same as a general form, but first you are prompted for the **name** of the particular item you are maintaining. For example, in the **Port Configuration** form, you are first prompted for a port number and then a general form is displayed for maintaining the port you have selected.
5. **Prompt and Confirm** forms prompt you to supply some information, possibly including confirmation, and then performs some special action. For example, the **Save Configuration** form prompts for information on how you want to save the configuration, and then saves it. The **rlogin to Host** form prompts for a host name and then attempts to rlogin to it.
6. **View Only** forms are used to display lists, of which you cannot modify the contents directly. **List Ports** is an example. These are often available in connection with multi-record forms.

There are three type of data items:

1. A **pick-list** is a data item whose value must be chosen from a pre-determined set. When a pick-list item is selected, you change the value by pressing the **space** bar. Press it repeatedly until the desired value appears and then move to the next field.
2. A **general** field is one which is not limited to a set of pre-defined values. You modify these fields by typing the desired value in the **input area**, thereby replacing the current value and then moving to the next field.
3. A **protected** field is one you cannot change. This includes information that is on the screen for reference purposes (the port number in **port configuration**, for example) and fields that you do not have administrative privileges to change. You can't select these fields.

Navigating in Forms

The table below explains how to use special keys to select data items in a form, modify the values, and exit the form. In many cases, either a function key or a control key can be used for the same purpose.

Table 4-3 Form Navigation Keys

Function Key	Key	Description
F1	**?**	Displays a one-line help message for the selected field and appears on the help line.
	Enter	Accepts the value of this data item and selects the next one. In **prompt & confirm** forms this may result in immediate action. In **multi-record** forms, you press enter after typing the name of the particular item you want to maintain.
F4	**Esc**	Exits this form, after asking whether to accept the changes.
	Space	If the selected data item is a **pick-list**, selects the next pre-defined value for the selected data item. Otherwise, a space is just a space.
	Tab	Selects the next data item in sequence. (Across the current row, then across the next row, and so on.)
	Left Arrow	In **tables**, selects the data item to the left, wrapping from the leftmost to the rightmost. In other forms, selects the previous data item in sequence.
	Up Arrow	In **tables**, selects the data item above, wrapping from topmost to bottom-most. In other forms, selects the previous item in sequence.

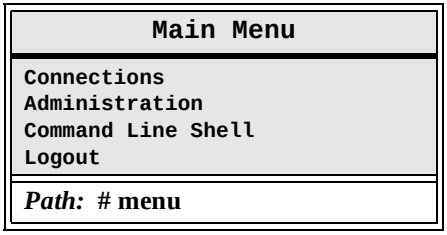
Table 4-3 Form Navigation Keys (Continued)

Function Key	Key	Description
	Right Arrow	In **tables**, selects the data item to the right, wrapping from rightmost to leftmost. In other forms, selects the next data item in sequence.
	Down Arrow	In **tables**, selects the data item below, wrapping from bottommost to top-most. In other forms, selects the next item in sequence.
	ctrl-B	In **multi-page tables**, selects the previous page (“back”).
F3	**ctrl-E**	Exits this form, accepting any modifications.
	ctrl-F	In **multi-page tables**, selects the next page ('forward”).
	ctrl-H	In **pick-lists,** selects the previous pre-defined value for the selected data item. During entry of **general** data, it acts as a backspace.
	ctrl-N	Displays a list of navigation keys.
	ctrl-R	Re-draws the screen.
	ctrl-U	Displays associated values. Often used when you must input a value that corresponds to something you configured on a different form. This will be documented under each specific use.
F4	**ctrl-X**	Exits this form after asking whether to accept the changes. (Same as Escape key, see above).
F6	**ctrl-Z**	Clears (blanks) the current input area.

The Main Menu in Detail

Since you have reviewed the navigation keys, its time to look more closely at the Main Menu screen. Screen 4-3 shows the Main Menu screen. On your screen, one of the selections will be highlighted.

Screen 4-3: Main Menu



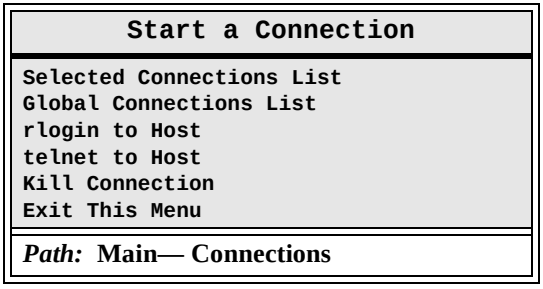
If you choose the last selection, **`Logout`**, the RAS 2000 first asks you to confirm that you want to log off. If you do, then you are logged off. If you have dialed in over a modem or have telnetted into the RAS 2000, you are disconnected.

If you choose **`Command Line Shell`**, you also exit the menu interface but you are not logged off. Instead, you enter the command interface described in chapter 3, "Using The Commands" on page 27. While most things can be done from either the menu or the command interface, there are some things that are more convenient to do from the menus and other things that are more convenient from the command line. There are also a few commands (e.g. **`term`**, **`ping`**, and **`filter`**) which do not have counterparts in the menu interface. From the command interface you can re-enter the menu interface by typing the command, **`menu.`** In addition to the four choices shown, there is a fifth choice, the **`Esc`** key (which always returns you to the "parent menu"). The Main Menu has no parent, so here **`Esc`** has the same effect as choosing **`Logout`**.

The remaining two choices, **Connections** and **Administration**, take you in completely different directions.

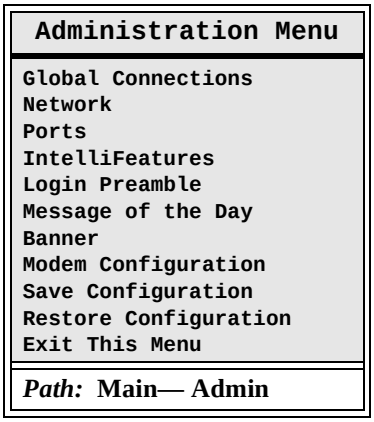
Connections takes you to a menu that allows you to launch telnet and rlogin sessions which you have previously defined in the *global connection* and *selected connection* tables. You can also launch telnet and rlogin sessions that are not included in the connection tables. If you have administrative privileges, you can also terminate sessions running on specific ports. This is discussed in more detail in chapter 8, "Logging Into the IntelliServer" on page 147.

Screen 4-4: Connection Menu



Administration allows you to view and modify the RAS 2000's configuration. It firsts sends you to the **Administration Menu**, where you choose the general area you want to configure, and from there to other, more specific menus

Screen 4-5: Administration Menu



and finally to the appropriate configuration form. The configuration forms are discussed later in great detail, but not about the menus that brought you to these forms. Look at the white box at the bottom. This box does not appear on your screen. It is used to explain how to reach the screen you are looking at. This example is telling you that from the **Main** menu you chose the selection **Admin**istration.

Menu Interface Summary

Table 4-4 is an overview of the RAS 2000's menus and submenus. The column marked **Form** indicates what type of configuration form is used. The column marked **related commands** shows commands with the same or similar function.

Table 4-4 Menu Interface Summary

See Page	Main Menu—Submenus	Form	Related commands
54	Connections		
	Selected Connection List	P	
	Global Connection List	P	
	rlogin to Host	P	**rlogin**
	telnet to Host	P	**telnet**
	Kill Connection	P	**kill, hangup**
55	Administration		
133	Global Connections	M	**connection**
200	Network		**network**
204	IntelliServer Configuration	G	**server**
251	PPP/SLIP Menu		
251	Login Scripts		
251	List Scripts	V	**show login**
259	Create Script	R	**add login**
259	Modify Script	R	**set login**
253	Delete Script	P	**delete login**
251	Options Profiles		
251	List Profiles	V	**show pppoption**
263	Create Profile	R	**add pppoption**
263	Modify Profile	R	**set pppoption**
253	Delete Profile	P	**delete pppoption**

Key to Form Types:

T .....Table
G.....General
R .....Multi-Record
M .....Multi-Page Table
P.....Prompt & Confirm
V.....View Only

Table 4-4 Menu Interface Summary (Continued)

See Page	Main Menu—Submenus	Form	Related commands
252	Remote Profile		
252	List Profiles	V	**show remote**
275	Create Profiles	R	**add remote**
275	Modify Profiles	R	**set remote**
253	Delete Profiles	P	**delete remote**
251	Dialer Scripts		
251	List Scripts	V	**show dial**
255	Create Script	R	**add dial**
255	Modify Script	R	**set dial**
253	Delete Script	P	**delete dial**
211	Host Addresses	M	**host**
215	Bootstrap	G	**boot**
140, 222	RADIUS/SNMP	G	**radius, snmp**
225	Name Servers	T	**nameserver**
227	Gateways	M	**gateway**
213	Network Addresses	M	**network**
230	Service Ports	M	**services**
60	Ports		
62	List Ports	V	**show port**
67	Configure a Port	R	**set port**
89	Duplicate a Port	P	**set port from…**
327	IntelliFeatures		
324	IntelliView		
328	List Profiles	V	**show profile iview**
338	Create Profile	R	**add profile iview**
338	Modify Profile	R	**set profile iview**
328	Delete Profile	P	**delete profile iview**

Key to Form Types:

T ..... Table
G ..... General
R ..... Multi-Record
M ..... Multi-Page Table
P ..... Prompt & Confirm
V ..... View Only

Table 4-4 Menu Interface Summary (Continued)

See Page	Main Menu—Submenus	Form	Related commands
327	IntelliSet		
328	List Profiles	V	`show profile iset`
342	Create Profile	R	`add profile iset`
342	Modify Profile	R	`set profile iset`
328	Delete Profile	P	`delete profile iset`
327	IntelliPrint		
328	List Profiles	V	`show profile iprint`
333	Create Profile	R	`add profile iprint`
333	Modify Profile	R	`set profile iprint`
327	Delete Profile	P	`delete profile iprint`
154	Login Preamble	G	`preamble`
154	Message of the Day	G	`motd`
107	Modem Configuration	T	`set modeminit`
357	Save Configuration	P	`save`
357	Restore Configuration	P	`restore`
53	Command Line Shell	P	
53	Logout	P	`logout, exit`

Key to Form Types:

T .....Table
G .....General
R .....Multi-Record
M .....Multi-Page Table
P.....Prompt & Confirm
V.....View Only

CHAPTER 5

Configuring Serial Ports

In this chapter, you learn how to use commands or menu screens to configure the IntelliServer's serial ports.

Since the main function of the IntelliServer is to provide various types of connections to your local area network through serial ports, this will be a big chapter. There are many questions you have to answer as you proceed. Some you can answer now; some you will answer as you understand better what the IntelliServer can do.

- What kind of device will be connected to the serial port—terminal, modem, printer, or something else?
- What are the device's pinouts? Which data-set signals are required?
- What are the line characteristics (e.g., speed, character size, and flow control)?
- What is the purpose of this connection? To allow a terminal to telnet to a host on the network? To allow remote access to your network through PPP/SLIP links? To support a printer?
- How does the configuration of this port affect the configuration of other parameters in the IntelliServer?

If you are new to some of this, it can be a bit intimidating. In practice, you will deal with some of these issues all the time, and all the issues some of the time, but you won't deal with all the issues all the time. You must be *aware* of them so that when things don't work as expected, you will have some intuition as to what is wrong.

What Does Port Configuration Include?

Any parameters that are likely to be different from one serial port to another, which are based on the choice of the port itself (as opposed, say, to which *user* has logged in or some other variable), are stored by the IntelliServer on a per-port basis, and so are part of the Port Configuration. There are many of these parameters, but they tend to fall into a number of distinct groups:

Functional Characteristics

What is this port going to be used for? A terminal running multiple telnet sessions to a host on the local LAN? A modem that supports dial-in clients that will establish PPP/SLIP connections to your network? A printer which will receive data from the print spooler on one of your UNIX hosts? A modem to support clients on your local network who wish to dial out and access BBS systems?

Physical Characteristics

These include the lines speed (baud rate), character size, parity, and number of stop bits. The settings will need to match those of the device that is connected to the port. When they are incorrect, nothing much will work: each side will seem to receive “garbage data” from the other.

Flow Control

If the IntelliServer cannot handle incoming data quickly enough, how shall it signal the attached device that he should stop sending data? If the attached device cannot handle the data as quickly as we are sending it, how will it signal us to stop? The key point here is that each side needs to agree. If you configure a serial port for XON/XOFF flow control, the device that is attached to that port needs to be configured the same, otherwise he won’t know how to tell us he’s full, and he won’t honor our request to stop when we’re full.

Modem Characteristics

These are all things that affect how a connection is established. Do we need to send an initialization string to an attached modem? Should we wait for carrier (the RS232 signal *Data Carrier Detect*) before we send a login prompt? For dial-out PPP/SLIP connections, what is the (modem-specific) dial script to use?

Application-specific Characteristics

Some settings are significant only in specific contexts. For example, the *User Name* is only meaningful when the port is configured as *Auto Login*. The scope of these settings is described in more detail later.

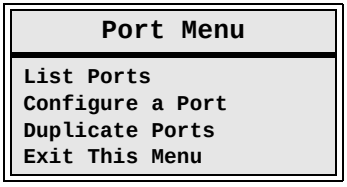
IntelliFeatures

IntelliFeatures include IntelliView, IntelliPrint, and IntelliSet. IntelliView allows multi-screened serial terminals to support separate sessions, using hot-keys to switch between them. IntelliPrint allows network access to printers attached to serial terminals, and IntelliSet is used to specify special line characteristics. In chapter 13, *IntelliFeatures*, you learn how to make collections of these features and give them names called *profiles*. You apply these profiles to a specific port by specifying its name as part of the port configuration.

Displaying Port Configuration: Menu

If you are using the menus, the current port configuration is displayed on the menu screen you will use to configure it. From the Main Menu, select "Administration", then "Ports". You will see the menu shown below:

Screen 5-1: Port Menu

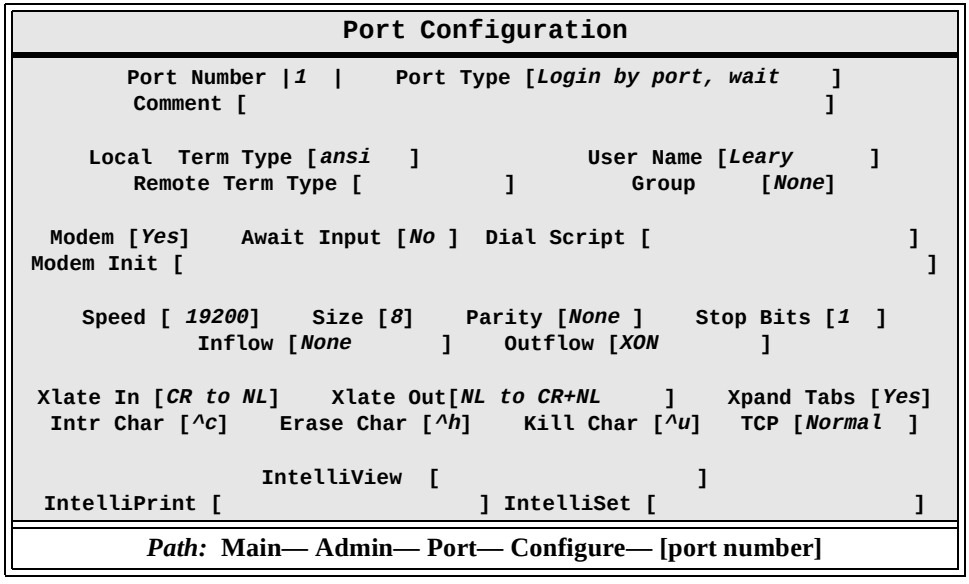


The first selection, *list ports*, displays a summary list of the ports, giving the port-number, port type, associated user name (if any), and any comments which have been previously stored for that port.

The third selection, *duplicate a port*, allows you to copy the configuration of one port to another.

The second selection, *configure a port*, next prompts you for the port number you wish to configure. After you enter the port number (0-63), you are taken to the port configuration form; a sample is shown below:

Screen 5-2: Port Configuration Form — Sample



There is nothing special about the particular values shown here in *italics*; this is just a sample. Actual values are discussed when configuration is explained.

Show Port

There are several commands to display port parameters. The first one, shown below, displays all the settings which were displayed on the menu's port configuration screen. It also reports the current settings for the port, which may differ from the stored settings. This is because newly changed settings may have not yet taken effect, and some applications will force certain settings temporarily (this is explained later). The current state of the data-set signals is also shown.

Example 5-1: Show Port

show port *port-list*
show port *port-list* **full**

```
# show port 2 full
Port Number:   2    Port Type: Login by port, wait
Comment:

Local  Term Type: wyse50         User Name: Leary
Remote Term Type:                Group    : None

Modem: Yes    Await Input: No    Dial Script:
Modem Init:

Speed:   9600    Size: 8    Parity: None    Stop Bits: 1
Inflow: None    Outflow: XON

Xlate Input: CR to NL    Xlate Output: NL to CR+NL    Xpand Tabs: Yes
Intr Char: ^c    Erase Char: ^h    Kill Char: ^u   TCP: Normal

IntelliView :
IntelliPrint:                        IntelliSet:

Current screen settings:
  modem; ospeed 9600; ispeed 9600; no parity; size 8; stop bits 1
  inflow ; outflow xon ; rows 24; cols 80; MSR = DTR RTS CD CTS dsr ri
  tabs ; ixlat CR to NL; oxlat NL to CR/NL; Intr: ^c; Erase: ^h; Kill: ^u;
```

The *port-list* is a series of numbers from **0** to **63**, separated by commas or dashes. Numbers separated by a dash (like **0-7**) select ports 0 through 7 inclusive. To select several ports that aren't consecutive, separate them by commas (like **0,3,5**). If the port number is omitted, the current port (the one you are typing on) is assumed. If you want information on all ports, type **ALL** in place of the port list.

If you are doing a full listing of several ports, the output will extend beyond a single screen, so you may want to page the output using the **|** key (see "Pagination" on page 33).

If the **show port** command is run from a *virtual port* (as when you have telnetted into the IntelliServer from some other host on your network), only the current settings are shown, because virtual ports are not configured like serial ports are.

Sometimes, the **show port** command will pause before displaying the current port settings, and then display a message similar to the following:

```
*** cannot get current settings, port is blocked
```

This indicates that the port is configured as *modem enabled* and there is a process waiting for carrier. You can confirm whether this is the case by using the command **whodo all**.

There are three commands which print summary information about the selected ports:

Example 5-2: Show Port — Access

show port *port-list* **access**

```
# show port 0-2 access
Port  User        Term Type  Login                      Group #
  0   root        wyse50     Login by port, wait         None
  1   leary       ansi       Login by port, wait         None
  2   tim         wyse50     Login by port, wait         None
```

Example 5-3: Show Port — Hardware

show port *port-list* **hardware**

```
# show port 0-2 hardware
Port  Speed   Parity  Bits  Stop  Modem? In Flow   Out Flow
  0     9600  None       8   1     Yes    None      XON
  1    19200  None       8   1     Yes    None      XON
  2     9600  None       8   1     Yes    RTS       CTS
```

Example 5-4: Show Port — Options

show port *port-list* **options**

```
# show port 0-2 options
Port  Erase    Intr     Kill
  0  ^h       ^c       ^u
  1  ^h       ^c       ^u
  2  ^h       ^c       ^u
```

Configuring Serial Port Parameters

You can configure serial port parameters by using the port configuration screen described on Screen 5-2 on page 63, or you can use commands to change individual parameters and groups of parameters.

When you change the configuration of a port, the changes will take effect the *next* time the port is opened. For example, if a user is currently logged into a port, and you change the line speed, the change will not take effect until that user logs off: the new login prompt will be issued at the new line speed. An exception is when you change the configuration of the port you are running on: you will then be given an option to allow the changes to take place immediately (but return to original values at next login), to take effect at next login, or both.

Port Type—How Will the Port be Used

Here you will tell the IntelliServer how the port will be used. Do you want to have someone log into this port? Do you want someone on your network to be able to dial out on a modem attached to this port? Do you want to attach a printer? Do you want to start up a PPP/SLIP link?

There are four port types that support terminals and dial-in connections: *Login by Port*, *Login by Screen*, *Auto-Login*, and *Auto-Login/Wait*. For these types, the connection is started by whatever is attached to the serial port. This may be a human sitting at a local terminal, a client who dials into an attached modem, or a computer that dials in and sets up a PPP connection by running a login script. When modems are used, the ports are usually configured so they detect incoming calls by waiting for the modem to assert *carrier* (**DCD**), and when *carrier* is dropped, to recognize that the connection has been dropped. See chapter 6, *Configuring Modems*.

There are two port types, *Reverse-TCP* and *Printer*, which can support connections that are started by processes running elsewhere on your network. These include printing and dial-out capabilities. There is one type, *Login-by-Port/TCP*, which supports dial-in and dial-out connections. There is a last port type, *Outbound Connection*, which supports dial-out PPP/SLIP/CSLIP connections to other networks.

Menu:	**Port Type [Disabled]**
Command:	**Set port** *port-list* **login disabled**

Nothing will happen on this port, *except*, that you can use commands such as **tip** and **output** to send data to it in order to test the port or configure a modem or other device.

Menu:	**Port Type [Login by Port, wait]**
Command:	**Set port** *port-list* **login byport**

With this selection, the port sends a login prompt to the attached terminal or modem. When the user logs in, the IntelliServer starts up whatever connections have been configured for that user.

If the port has been configured with an IntelliView profile, and the user has been configured for multiple sessions, the additional virtual screens are enabled, and when they become active they display the message: **Press <enter> to continue**. When a process on one of these screens completes, the "press enter" prompt returns. When all sessions on this port are logged off, the port drops **DTR** to terminate the connection and then issue a new login prompt.

ISP NOTE	To support dial-in users, an Internet Service Provider will usually attach modems to ports configured as *Login by Port*. He will configure the port as *Modem enabled* so that the IntelliServer will wait for an incoming call before it sends the login prompt. He may also insert modem-initialization strings for the port. He will usually not be using IntelliFeatures. The actual service being provided (telnet, rlogin, PPP…) is configured for each user, and may be stored either in NVRAM or on a separate computer, when RADIUS is used. See chapter 7, *Configuring Users*.

Menu:	**Port Type [Login by virtual screen]**
Command:	**Set port** *port-list* **login byscreen**

You generally use this setting only if you have configured the port to support

multiple sessions through IntelliView. Each virtual screen is sent its own login prompt, and the user must log into each virtual screen separately. When a session is ended, a new login prompt for that virtual screen is sent, but DTR is not dropped if there is any other session on this port still active.

Since there is generally only one actual *person* at the terminal at one time, multiple logins with different user id's on the same port are usually used when these different users are configured to launch specific functions. On one screen I might log in as *pr* and use *telnet* to log into the computer that does payroll; on the second screen I might log in as *ap* and use *telnet* to access the computer that does accounts payable.

Menu:	**Port Type [Auto-Login]**
Command:	**Set port** *port-list* **login auto**

(*and also in the port configuration form and command)*

Menu:	**User Name [*myname*]**
Command:	**Set port** *port-list* **username** *myname*

Sometimes, you will want to configure a port so that the IntelliServer will automatically start a connection without prompting for a login. If the port is configured for *Auto-Login*, you must also specify a user name for this port (*myname*, in the above example). The port would behave exactly as a *Login-by-Port* but instead of sending a login prompt, he assumes that the specified user has successfully logged in, and starts up his connections accordingly. When the session is over, the IntelliServer will (after waiting for *carrier* when appropriate) restart the sessions again.

ISP NOTE:	Some service providers who also supply BBS access and other special services implement them by configuring *Auto-Login* ports which automatically telnet or rlogin to a host on the network configured to provide these special services.

Menu:	**Port Type [Auto-Login, wait]**
Command:	**Set port** *port-list* **login autowait**

This is almost identical to *Auto-Login*, except that instead of launching the connection immediately, the port first sends a prompt: **Press <enter> to continue**, and when the operator does this, *then* the connection is launched.

This is designed to solve the quandary that occurs when a port configured as *Auto-Login* is attached to a local terminal that is always on but frequently unattended. The user logs off and walks away, and the IntelliServer immediately launches the connection. Suppose that connection is an attempt to rlogin to some host machine. So that machine prompts for a password. Since there is no one present to enter a password, the connection soon times out and is restarted, and times out and is restarted… If the port is configured as *Auto-Login, wait*, then the IntelliServer will remain at the "Press enter" prompt until someone does this, and you avoid the retries and time-outs.

Menu:	**Port Type [Reverse-TCP]**
Command:	**Set port** *port-list* **login revtcp**

When a port is configured as *Reverse-TCP*, the IntelliServer accepts a TCP connection from some other host on the network. Data received from that host is sent out the serial port, and data received from the serial port is sent to the host. This is a common method of supporting printers and other "non-login" serial devices. See chapter 18, *Reverse TCP and Printing*, where this is discussed in much more detail, and in this chapter on page 86, where a port's *group* and *tcp* parameters are discussed.

Menu:	**Port Type [Login by Port,TCP]**
Command:	**Set port** *port-list* **login byporttcp**

This is a combination of *Login-by-Port* and *Reverse-TCP*, and is designed to support bidirectional operation of a modem. You must configure the port as *modem enabled*, because the IntelliServer will use *carrier* (DCD) to sense incoming calls and determine whether there has been a disconnection.

When the port is idle and there is no incoming call, the IntelliServer accepts TCP connections for this port from hosts on the net, just like *Reverse-TCP*. If a connection is established, the client can access the modem, send dialing commands, and connect to other systems. Anyone trying to dial in will get a busy signal because the modem is off-hook.

If an incoming call comes in first, the port sends out a login prompt, like *Login-by-Port*, and as long as the incoming call is connected, the IntelliServer refuses or defers TCP connections from the network for that port. This is explained in more detail in chapter 18, *Reverse TCP and Printing*.

Menu:	**`Port Type [Printer]`**
Command:	**`Set port` *`port-list`* `login printer`**

This configuration is similar to *Reverse-TCP*, except that a port configured as a *printer* can also accept connections from **`rcp`** and **`rsh cat`** clients on your network. There are other differences as well: these are discussed in chapter 18, *Reverse TCP and Printing*.

Menu:	**`Port Type [Outbound connection]`**
Command:	**`Set port` *`port-list`* `login outbound`**

This configuration supports outbound PPP/SLIP/CSLIP links. The IntelliServer brings these links up automatically when it tries to route a packet to a network that it knows to be on the other side of one of these links. This process is discussed in more detail in chapter 11, *Remote Network Configuration*. Note that this port type supports only *dial-out* connections. To support clients who are dialing into the IntelliServer, you configure the port as *Login-by-Port*.

Physical Characteristics

These include the line speed, character size, number of stop bits, and parity. The most important thing is to ensure that these parameters match those of the device that is connected to the serial port.

Menu:	**Speed [*9600*]**
Command:	**Set port** *port-list* **speed** *speed*

This sets the line speed at which data is transmitted and received: *speed* must be one of the following:

50	150	1200	3600	19200	64000
75	200	1800	4800	38400	76800
110	300	2000	7200	56000	115200
134.5	600	2400	9600	57600	

In addition, you can define custom rates by setting up an IntelliSet profile and assigning that profile to a port. By using IntelliSet, you can also specify a *split baud-rate* where the port transmits at one speed, and receives at another. When line speeds and other parameters are defined using IntelliSet, those values override the ones chosen here. For more details, see chapter 13, *IntelliFeatures*.

Menu:	**Size [*8*]**
Command:	**Set port** *port-list* **charsize** *size*

This sets the number of data bits per character: *size* must be one of the following:

5	6	7	8

Menu:	**Parity [*none*]**
Command:	**Set port** *port-list* **parity** *parity*

This controls the parity bit sent with each characters: *parity* must be one of the following:

odd	even	space	mark	none

Menu:	**Stop Bits [*1*]**
Command:	**Set port** *port-list* **stopbits** *bits*

This controls the number of stop bits that are transmitted after each character. Choices are **1**, **1.5**, or **2** bits. One stop bit is generally sufficient except when you are connecting to devices that are very old, very slow, or very unusual. This has no effect on the receiver: one stop bit is always sufficient.

Flow Control Characteristics

These are the parameters which are designed to prevent the device that is transmitting serial data from overflowing the buffers of the device that is receiving the data. In order to do this, the receiving device is configured to notify the sender when the receiver's buffers are starting to get full. The sending device must be configured to understand this notification, and to stop sending data accordingly. Later, when the receiver's buffers again have room for more data, he notifies the sender to start transmitting again.

Sometimes the serial port on the IntelliServer is the sender, and it must avoid overrunning the terminal, modem, or printer to which it is attached. This is called *output flow control*. At other times, the IntelliServer is receiving data from a serial device, and it must be configured to signal when its buffers start to fill up so that the serial device stops sending data for a while. This is called *input flow control*.

On the IntelliServer, you configure flow control for each direction separately. The *output flow control* you choose needs to match the *input flow control* of the device you are sending data to, and vice-versa.

Tip	If the device connected to the IntelliServer's serial port seems to be losing data, try configuring the port and the device for a lower line speed. If the device loses data at high speeds but not at lower speeds, you should check the port's *output flow control*, and the device's *input flow control.* **Wrongly-configured flow control accounts for 90% of all "missing data" problems.** It accounts for 70% of the problems that remain after this cause has been completely ruled out.

Output Flow Control Options

Menu:	**Outflow [None]**
Command:	**Set port** *port-list* **outflow disabled**

Output flow control is disabled: the IntelliServer does not recognize any condition that means “stop sending!”

Menu:	**Outflow [XON]**
Command:	**Set port** *port-list* **outflow xon**

If an *XOFF* character is received (normally a byte with the binary value 10011), the IntelliServer stops sending data until an *XON* character is received (binary 10001). These flow-control characters are also stripped from the data stream and are never seen as actual “incoming data”. This is sometimes called XON/XOFF flow control, sometimes “software flow control” because it was traditionally implemented in software. It is also sometimes called “in-band” flow control, because the flow-control information is sent along the same wires as the data itself. This form of flow control can be used with terminals and printers and some types of file transfer, when the normal data passing between the devices does not contain bytes equal to the XON and XOFF values. This type of flow control is usually not suitable for PPP/SLIP connections, or binary file transfers that would be carrying data which might contain XON/XOFF bytes.

Menu:	**Outflow [XANY]**
Command:	**Set port** *port-list* **outflow xany**

This is a variation on XON/XOFF flow control intended for terminals. As with the previous, receiving an XOFF character makes the IntelliServer stop transmitting. But once the IntelliServer has stopped, the receipt of any data (including a second XOFF) causes the IntelliServer to resume. Since the default XON and XOFF values correspond the **ctrl-Q** and **ctrl-S** on traditional ASCII keyboards, this corresponds to the MSDOS convention of entering **ctrl-S** once to suspend output and again to resume.

With standard XON/XOFF flow control, if an operator accidently enters **ctrl-S**, this suspends output until (and unless!) the operator enters **ctrl-Q**. With this *XANY* variant, entering any key restarts the output, so an inexperienced operator is less likely to panic.

On the other hand, this selection is a poor one *if the terminal cannot keep up and has to send an XOFF to avoid losing data*. Why is this? Because most operators type ahead. If an operator is typing ahead while his terminal is starting to overrun (and has sent an XOFF character), many terminals still send those keystrokes and when the keystrokes are received, the IntelliServer resumes sending data and likely overruns the terminal. This is also a poor selection for many printers, because many send *multiple* XOFF bytes when their buffers start to become nearly full.

Menu:	**Outflow [CTS]**
Command:	**Set port** *port-list* **outflow cts**

Data is sent as long as the *clear to send* **(CTS)** data-set signal is asserted (or "raised"). When **CTS** is negated ("dropped") the IntelliServer stops sending data until the signal is asserted again. This is sometimes called "hardware flow control" because it was traditionally implemented in hardware; also called "out-of-band" flow control, because the flow control information is sent on a separate wire from the data.

This type of output flow control is recommended for PPP/SLIP/CSLIP links or any connections where you are transferring arbitrary binary data that could include XON or XOFF characters. When using CTS flow control with a modem, you connect the IntelliServer CTS pin to the modem's CTS pin, and ensure that the modem is configured for CTS flow control. When using CTS flow control with terminals, you need to check your terminal configuration and see what data-set signals it can use for flow control. Sometimes it is *request-to-send* **(RTS)** but more often it is *data-terminal-ready* **(DTR)**. Whatever signal is used, you must arrange your cable to connect that output signal to our CTS input, and configure the terminal appropriately. Printers are similar to terminals in this way, but there is an even greater variety of pins they may use for "hardware flow control", and because they are generally slower than terminals, they are less forgiving of incorrect configuration.

Menu:	**Outflow [XON & CTS]**
Command:	**Set port** *port-list* **outflow xoncts**

Menu:	**Outflow [XANY & CTS]**
Command:	**Set port** *port-list* **outflow xanycts**

Combinations of XON and CTS, or of XANY and CTS. The IntelliServer does not transmit data unless *both* conditions permit. A good example where this might be used is for a slower terminal (which needs robust flow control) whose operator wants to use **ctrl-S** to suspend and resume output. Many terminals have scroll-lock keys, however, and in that case CTS flow control alone would suffice. Perhaps you can think of another use?

Input flow Control Options

Menu:	**Inflow [None]**
Command:	**Set port** *port-list* **inflow disabled**

There is no input flow control. No attempt is made to notify the sender when the receive-buffers are becoming full. This is often used with terminals because key-strokes do not usually arrive fast enough to overrun computers.

Menu:	**Inflow [XOFF]**
Command:	**Set port** *port-list* **inflow xoff**

When the port's receive buffer becomes mostly full, an XOFF character is sent to the sending device to tell it to stop sending data. When the buffers start to get empty again an XON character is sent to tell it to restart transmission. This is the mate to *XON outflow* described above.

Menu:	**Inflow [RTS]**
Command:	**Set port** *port-list* **inflow rts**

When there is room in the IntelliServer receiver buffer, the *request-to-send* **(RTS)** data-set signal asserted. When the buffers become mostly full, this signal is negated ("dropped"). This is the "mate" to *CTS outflow* described above. When connecting to a modem, you wire the IntelliServer RTS output to the modem's RTS (an input), and configure the modem for RTS flow control. (You are probably using CTS flow control as well.) When connecting to a device (like another IntelliServer port) configured for *CTS outflow* you connect the IntelliServer RTS to the devices CTS (and probably vice-versa).

Modem Characteristics:

When your serial port is connected to a modem you have several concerns, and these differ depending on whether the modem is supporting dial-in connections, dial-out connections, or both. For dial-in connections, you must be able to send any needed configuration commands to the modem so it will be ready to receive a call. You must be able to detect when the call has come in. For dial-out connections, you must be able to send commands to the modem to make it dial a phone number and connect with a modem at the other end. For both kinds, you must be able to tell when the phone connection has been dropped, or to drop it yourself.

In addition to wires for transmit data, receive data, and signal ground, the serial port communicates with the modem using *data set signals*, so you must be sure your cable contains all the appropriate connections.

Tip	When connecting an IntelliServer's serial port to a modem, always connect the pins on the modem *to the pins on the IntelliServer which have the same name*. Consult your hardware reference for exact pinouts and sample cables, but this is the rule to remember.

There are four *data-set signals* of interest to the IntelliServer:

- **DTR**, or *Data Terminal Ready*, is asserted by the IntelliServer to tell the modem that it is ready (for example, to receive incoming calls). The IntelliServer drops **DTR** to tell the modem to hang up the line (for example, when a dial-in user logs off).

- **DCD**, or *Data Carrier Detect*, is asserted by the modem to indicate that it has made a connection with another modem and is ready for data to be transferred. The modem drops **DCD** to tell the IntelliServer that the line has been disconnected. When you want the port and the modem to be using *hardware flow control* (see page 75).
- **RTS** (*Request to Send)* is asserted by the IntelliServer to tell the modem that it can send the port more data, if RTS flow control has been configured for that port.
- **CTS** *(Clear to Send)* is asserted by the modem when it has room for data from the IntelliServer.

Your modem might handle its data-set signals in this way, right out of the box, or you may need to configure it by setting switches or, more commonly, by sending commands. Consult your modem manual and chapter 6, *Configuring Modems*, for more details.

Menu:	**Modem [*No*]**
Command:	**Set port** *port-list* **modem disabled**

This is the default, and means that the IntelliServer is to pretend that the **DCD** signal is always asserted. This is intended for connections to local terminals, printers, etc. This is sometimes called a “non-modem” port.

Menu:	**Modem [*Yes*]**
Command:	**Set port** *port-list* **modem enabled**

For dial-in connections, the IntelliServer waits for the modem to assert **DCD** before it sends the login prompt. For dial-in and dial-out connections, if **DCD** is dropped the IntelliServer recognizes that there has been a disconnection and terminate whatever processes have been using that port. For login ports, the user is marked as having logged off, for reverse-TCP ports the TCP connection is closed. When a port is configured in this way, it is called a “modem port”, even though it may be connected to a terminal or other equipment.

There is another difference between “modem” and “non-modem” ports. When a user logs into a modem port, the user has one minute from when the login prompt is sent to when he enters a correct password. If he does not log in successfully within a minute he will be disconnected. For non-modem ports there is also a one minute limit, but it begins when the user first enters his user name. (See also chapter 8, *Logging Into the IntelliServer*).

Menu:	**Await Input [*No*]**
Command:	**Set port** *port-list* **wait *disabled***

Normally, a port configured to send a login prompt will do so shortly after detecting that **DCD** is asserted. If you set this to *yes* (*enabled)* then after detecting carrier the IntelliServer waits until it receives some incoming data on the port before it sends the login prompt. There are some modems which raise **DCD** while they are still in command mode, which prevents the modem from mistaking the login message for a modem command. There is in case the built-in delay between sensing carrier and sending the prompt is not long enough.

Menu:	**Modem Init [***modem-commands…***]**
Command:	**Set port** *port-list* **init** *modem-commands*

This setting is used by ports that are configured for terminals or dial-in connections (see page 67). It defines a string of commands which the IntelliServer transmits to the modem before it waits for the next incoming call. This is not always required. Some modems can be configured ahead of time and never seem to lose their settings and never get “sick”. In chapter 6, *Configuring Modems*, examples are given of command strings you might want to send, and other ways to configure a modem.

When does the string get sent? Some user logs off. IntelliServer drops **DTR** to hang up the line. Waits a second. Raises **DTR**. *Sends the initialization string*. Waits for modem to assert **DCD**…Call comes in, next fellow logs in, works, logs off—and it’s *deja vu* all over again.

Important: The contents of this field *can* be the command string itself that is to be sent to the modem, but it can instead be the name of a command string. Name? In chapter 6, *Configuring Modems*, you are shown how to define a table of commonly-used modem commands. You give each set of commands, a name and enter the name here instead of the command string itself.

How does the IntelliServer know whether you have entered the name of an initialization string or the string itself? When it gets ready to send the string, it looks for an entry in the initialization table whose name matches what you have entered here. If it finds one, then it assumes you meant this to be a name; otherwise, it treats it as the string itself.

The initialization string is sent at the appropriate time, even if the port is configured as *modem disabled*. That way you can use the same mechanism to send initialization to local terminals or other devices.

The IntelliServer does not automatically send a carriage return or linefeed after this string. To send these or other control characters, you must specify them using the format given in Table 5-4, “Printing the Unprintable,” on page 94. Since the backslash and caret are used in the definitions of unprintable (control) characters, those characters themselves have to be entered in a special way, according to the table.

Menu:	**Dial Script [***scriptname***]**
Command:	**Set port** *port-list* **dialscript** *scriptname*

This is the name of a dialer script. This is used by ports configured for outbound PPP/SLIP/CSLIP links (see page 71). It specifies the commands that have to be sent to the modem so it will dial and establish a connection and allows the IntelliServer to wait for particular responses. Different modems may require different dialer scripts; that is why the dialer script is stored on a per-port basis, while the *login script* (which depends on the particular target of the call) is identified in the *remote profile*. Dialer scripts are discussed in chapter 11, *Remote Network Configuration*.

Application-specific Settings

Application-specific settings are ones which affect the operation of the port only when it is used for particular applications. For example, terminal descriptions have no effect on a port configured as Reverse-TCP, and input processing options have no effect on a printer. Application-specific settings include options for doing input/output processing, information about attached terminals, and other miscellany, and are explained in the following sections.

Input Processing

Any input processing specified here affects the port's operation when it is accepting line-based input, such as at the command prompt, or when in telnet command mode. At other times, the individual applications (telnet, rlogin, menu, etc.) force the input processing to an appropriate setting.

Menu:	**Xlate In [None]**
Command:	**Set port** *port-list* **ixlat disabled**

Using this option, no input processing is performed.

Menu:	**Xlate In [CR to NL]**
Command:	**Set port** *port-list* **ixlat cr_nl**

The carriage return key (ascii CR) is mapped to a linefeed character (ascii LF). This is the default and the only reason I can think of to change it would be to support a terminal that sends both carriage-return and linefeed when the Return key is pressed.

Menu:	**Xlate In [NL to CR]**
Command:	**Set port** *port-list* **ixlat nl_cr**

We are still trying to decide what this one is for.

Special Keys

When a terminal is connected to your port, certain keys can be given special significance.

Menu:	**Intr Char [^*c*]**
Command:	**Set port** *port-list* **intr** *key*

This defines the *interrupt key.* Use this key to quickly terminate commands before they have finished. In this example **^c** represents *control-c*. In either the menu or the command line, you type **^** and **c** to enter the value as shown. You could also enter *control-c* itself, unless it were already defined as one of the special keys. The **del** key has a special representation. You can either press the key itself (if not otherwise used) or enter **^** and then **?**.

Menu:	**Erase Char [^*h*]**
Command:	**Set port** *port-list* **erase** *key*

This defines the character used to backspace a single character and erase it.

Menu:	**Kill Char [^*u*]**
Command:	**Set port** *port-list* **kill** *key*

This defines the character used to cancel the entire line you are currently typing. This is used when you are doing line-oriented input such as at the command prompt or in telnet command mode.

Output Processing

These settings affect the operation of the port *only* when it is configured as a printer (see page 71).

One of the differences between *Printer* and *Reverse-TCP* ports is that these output processing options will take effect on *Printer* ports, while output processing is always disabled for *Reverse-TCP.* These processing options are provided to assist in sending output from your system to a text printer. There are two separate options: whether to provide tab expansion, and how to handle carriage-returns.

Menu:	**Xpand Tabs [*Yes*]**
Command:	**Set port** *port-list* **tabs enabled**

With this setting, the port will translate ascii **tab** characters to a sequence of spaces sufficient to achieve tab stops at 8-character intervals. This tab setting corresponds to the traditional tab processing performed on UNIX systems and is useful when printing output from a UNIX system using tools that expect this processing to be performed "downstream". If this parameter is set to *No* (or *disabled* using the command), then **tab** characters are sent unchanged.

Menu:	**Xlate Out [None]**
Command:	**Set port** *port-list* **oxlat disabled**

With this setting, the port will send carriage return (ascii **CR**) and linefeed (ascii **LF**) characters as-is with no translation.

Menu:	**Xlate Out [NL to CR+NL]**
Command:	**Set port** *port-list* **oxlat nl_crnl**

With this setting, a carriage-return (**CR**) is added after any linefeed (also called a "newline", hence the abbreviation) in the output. This is useful when printing output from systems (like UNIX, again) which use a single linefeed character to delineate the ends of lines. If you send such output directly to most printers, each new line will begin directly below where the previous line left off, creating a "barber-pole" effect.

Menu:	**Xlate Out [CR to NL]**
Command:	**Set port** *port-list* **oxlat cr_nl**

Carriage return characters are changed to linefeeds (newlines).

Menu:	**Xlate Out [Strip CR]**
Command:	**Set port** *port-list* **oxlat strip_cr**

Carriage returns which occur at the beginning of a line are thrown away.

Menu:	**Xlate Out [CR | NL to CR+NL]**
Command:	**Set port** *port-list* **oxlat crnl_crnl**

Carriage returns are added before any newline, and newlines are added after any carriage return. In other words, *either* a carriage return *or* a newline shall become *both* a carriage return and a newline. This may or may not be useful, but it is very poetic.

Terminal Descriptions

Many screen-based applications will not operate properly unless they are informed what type of terminal is being used. For example, the IntelliServer's menu will not be able to correctly position the cursor and draw boxes if it thinks you are using a Wyse 60 terminal when in fact you are using a VT100. When you telnet from the IntelliServer to a host on your network to run applications, the IntelliServer can inform the application what terminal type has been configured so that it will work properly.

Menu:	**Local Term Type [*ansi*]**
Command:	**Set port** *port-list* **term** *terminal-type*

This setting defines the terminal characteristics that will be used when the IntelliServer's menu interface is running on this port. This also defines the default terminal name that will be sent when you telnet or rlogin from this port to a host on your network. This default value may be overridden by other settings, however.

Because this information is used by the menus, the IntelliServer needs to understand the terminal characteristics that each terminal name represents. For that reason, there are a limited number of these supported. Your choices are:

unknown	wyse30	xterm

ansi	wyse50	uterm0	uterm1
vt100	wyse60	uterm2	uterm3

The last four terminal types are user-definable. If your terminal does not emulate one of the defined terminals, a section later in this chapter starting on page 90 explains how you can store your terminal's definitions under one of these four terminal types.

Menu: `Remote Term Type [wyse70 ]`

Command: `Set port port-list rterm terminal-type`

Above, you defined a *Local terminal name* for your port. But you may not want this name to be sent when you log into a host. What would he make of a terminal type *uterm0*, anyway? Even if you were using something common like a wyse30, your host computer may contain several variants on that name, corresponding to several uses of the port.

If you enter a name here, then by default it will be sent when you rlogin or telnet to a host, instead of using the one given for the *Local terminal name*. Since the IntelliServer does not need to know what this name actually means, it can be any name that will be understood by the login host.

The telnet and rlogin commands also support command-line arguments which, if used, can override these default terminal-types. If there is no command-line argument, the *remote term type* is used, and if no *remote term type* is defined, then the *local term type* is sent. The telnet and rlogin commands are described in chapter 16, *Connections*.

Tip	When you telnet to a UNIX system, the assigned tty device name will not indicate the physical location of the terminal running the session (as would be the case with directly-attached serial ports). However, the terminal name the IntelliServer sends will be stored in the environment variable **`TERM`**. If you configure a unique *remote terminal name* for each port, you can add code to your **`.profile`** script (see Example 16-2 on page 404) to sort out the port number and true terminal type.

Menu:	**Comment [*Muffy's terminal*]**
Command:	**Set port** *port-list* **comment** *comment*

Whatever you put here in the privacy of the home is your own business. Perhaps it will help you keep track of which terminal is which.

Reverse-TCP options

There are two options which are only meaningful if the port is configured for *Reverse-TCP* (see page 70) or as *Login-by-port/TCP* when the TCP connection is active. A reverse-TCP connection is so called because the historical purpose of terminal servers was to connect telnet and rlogin sessions begun on the serial ports, to hosts on the network. A reverse-TCP port accepts connections from hosts on the network, and gives them access to serial ports.

Menu:	**Group [*None*]**
Command:	**Set port** *port-list* **group** *group-number*

There are 16 groups of ports, numbered 0 to 15. Any port can belong to any group, or to no group at all. When something tries to start a reverse-TCP connection to the IntelliServer, it can specify a particular port, or a particular port group. When a port group is specified, the first available port in the group is used. (see chapter 18, *Reverse TCP and Printing*). A port group number can also be specified in a *Remote Profile* for an outbound PPP/SLIP/CSLIP interface (see chapter 11, *Remote Network Configuration*).

A port is configured as *Reverse-TCP* or *Login-by-Port/TCP* cannot be a member of the same group as a port configured as *Printer* or which uses *IntelliPrint*. This is because the first types suppress output processing, while the others perform it. If both types were members of the same group the results might depend on which printer happened to be available.

Menu:	**TCP [*Normal*]**
Command:	**Set port** *port-list* **tcp** *option*

There are three choices— first as they appear on the menu-screens:

Normal	CRNL->CR	Raw

and now as you would type them when using the commands:

normal	crnl_cr	raw

Normally, a reverse-TCP connection uses telnet protocol. Telnet server implementations differ in their treatment of carriage-return (CR) and linefeed (or newline, NL) characters. With some, if a CR-NL pair is received from the network, the two characters will be output. That is what our *normal* option does. With other telnet servers, if a CR-NL pair is received, the CR is sent but the NL is ignored. We call this the *CRNL->CR* option. These two options are provided for maximum compatibility.

The third option, *Raw*, causes the Reverse-TCP connection on that port to not use telnet protocol at all. Instead, the data received over the TCP connection is sent to the port exactly as received, and vice-versa. This is provided for compatibility with other vendors' products, as well as providing an easy-to-use interface for special applications.

IntelliFeatures

There are three types of IntelliFeatures, and they are discussed in detail in chapter 13, *IntelliFeatures*. There are three types of IntelliFeatures. *IntelliView* allows you to run multiple login sessions from a single multi-page terminal. *IntelliPrint* allows network access to a printer connected to your terminal's auxiliary port. *IntelliSet* allows you to specify custom lines speeds and override other aspects of a port's physical characteristics.

Groups of features are collected together in *profiles*, and you will give each profile a name. To apply a profile to a port, you will enter its name in the port configuration.

Menu:	**IntelliView [*wy60.2t*]**
Command:	**Set port** *port-list* **iview** *profile-name*

This specifies the name of the IntelliView profile you want to apply to this port (*wy60.2t* in this example, but there is nothing special about this one).

Menu:	**IntelliPrint [*wy60.p*]**
Command:	**Set port** *port-list* **iprint** *profile-name*

This specifies the name of the IntelliPrint profile you want to apply to this port (*wy60.p* in this example, but it could have been any profile you have already created).

Menu:	**IntelliSet [*pr.9600*]**
Command:	**Set port** *port-list* **iset** *profile-name*

This specifies the name of the IntelliSet profile you want to apply to this port (*pr.9600* in this example).

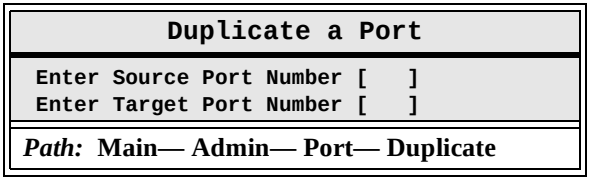
NOTE:	IntelliView and IntelliPrint are extremely terminal-specific, so they are only used when you know what type of terminal will be connected to the port, as for example when they are locally connected, or when a terminal of a known type will be dialing into a particular port via modem.

Duplicating Port Configurations

From the Menu

Often you will want to have several ports configured in identical or nearly identical ways. The fastest way to do this is to configure one port the way you want it and then copy the settings to other ports.

Screen 5-3: Copying the Port Configuration



If you are using the menu system, from the Main Menu, select “Administration”, then “Ports”, as you did on page 62. Then select “Duplicate a Port” and you will see the screen pictured above. Enter the port number that is already configured as the *source port*, and the port you want to configure as the *target* port.

From the Command Line

You can also use the command-line interface to copy settings from one port to another; in fact you can duplicate the settings to several ports. In this example, the configuration of port 1 will be copied to ports 8 through 15.

Example 5-5: Copying the Port Configuration

```
set port port-list from port
# set port 8-15 from 1
```

User-defined Terminal Types

In order to use the IntelliServer's menu system, your terminal must be capable of cursor addressing, highlighting text, and other functions. These features are implemented in different ways on different terminals. The IntelliServer contains descriptions of several popular terminals (see page 84), but it is possible that your terminal may not be sufficiently similar to any of these for the menus to work properly. This will not prevent the terminal from being used for commands, or to telnet and rlogin to other hosts, or even to use the selected connection menu. It only prevents you from starting up the main menus as you would to do administration.

If you want to use this terminal for the menus as well, all you have to do is configure one of the user-defined terminal types *uterm0-uterm-3* for your terminal, using the **`term`** command.

The most important command is *help term*, because you won't be able to find this book and you won't be able to remember the codes for the sequences. So here it is.

Example 5-6: Help Screen For the Term Command

```
# help term
     term        - Modify/Display user-definable terminal types

term set <number> {sequence <string>}

term show <number>

        {sequence <string>} options are:

        [cl <clear screen>]             [cm <move cursor>]
        [so <start highlight>]          [se <end highlight>]

        [CO <display cursor>]           [CF <hide cursor>]
        [GS <start graphics>]           [GE <end graphics>]
        [GH <horizontal bar>]           [GV <vertical bar>]
        [G1 <top left corner>]          [G2 <top right corner>]
        [G3 <bottom left corner>]       [G4 <bottom right corner>]
        [GL <vertical bar w/left hori>] [GR <vertical bar w/right hori>]
        [kl <left arrow key>]           [kr <right arrow key>]
        [ku <up arrow key>]             [kd <down arrow key>]
        [k1 <function key 1>]           [k3 <function key 3>]
        [k4 <function key 4>]           [k6 <function key 6>]
```

When you assign one of these terminal types to a port (as on page 84) they are named **uterm0**, **uterm1**, **uterm2**, and **uterm3**, but in this *term* command you specify them as **0**, **1**, **2**, or **3**.

The *sequence* refers to one of the codes **cl**, **cm**, **so**, **se**, **CO**, and so on. The codes indicates which terminal function you are describing, and is very similar to the codes used in UNIX *termcap* files. The *string* defines the sequence of characters which will perform that function. A special notation is used to specify this string, because the real bytes that are sent usually contain non-printing characters. Strings defining cursor addressing are especially complex since they must indicate how cursor addresses in general are to be sent to a terminal.

```
show term number
```

This displays all the strings you have defined for each sequence for this terminal type.

```
set term number sequence string
```

…and this command sets one of the sequences to the string you specify.

Sequence Codes Explained

The following tables explain the function of each sequence code. The first four are mandatory. You must define strings for them in order for the menu system to work. The next set of codes are not required but if your terminal supports them the menu screens will look better. The third set allows you to use your terminal's arrow and function keys, by specifying what sequence of characters these keys send.

These codes are considered *keywords* to the IntelliServer's command processor, so you can type them in upper or lower case. Some are shown here in upper case and some in lower case, because those are the correct cases used in the */etc/termcap* file.

Table 5-1: Sequence Codes — Required

cl	Clear the entire screen. *(this is "see-ell" not "see-one")*
cm	Set cursor position.
so	Start highlighting. This can be reverse video, high intensity, or any other method to make the selected text stand out. *("ess-oh")*
se	End highlighting.

Table 5-2: Sequence Codes—Optional Drawing

CO	*("see-oh")* Turn on the cursor. (Your termcap file may represent this as **ve**.)
CF	Turn off the cursor. (Your termcap file may use **vi** for this.)
GS	Start graphics mode, for terminals which change mode to draw line-drawing characters.
GE	End graphics mode, for terminals which use *start graphics mode.*
GH	Horizontal line-drawing character.
GV	Vertical line-drawing character.
G1	*("gee-one")* Top left corner line-drawing character.
G2	Top right corner line-drawing character.
G3	Bottom left corner line-drawing character.
G4	Bottom right corner line-drawing character.
GL	"Left tick" line-drawing character: a vertical bar with a horizontal bar extending to the left.
GR	"Right tick" line-drawing character: a vertical bar with a horizontal bar extending to the right.

Table 5-3: Sequence Codes—Keyboard

kl	*("kay-ell")* The characters sent by your terminal's left-arrow key.
kr	The characters sent by your terminal's right-arrow key.
ku	The characters sent by your terminal's up-arrow key.
kd	The characters sent by your terminal's down-arrow key.
k1	*("kay-one")* The characters sent by your terminal's **F1** function key.
k3	The characters sent by your terminal's **F3** function key.
k4	The characters sent by your terminal's **F4** function key.
k6	The characters sent by your terminal's **F6** function key.

Strings Explained

The *string* parameter defines the character sequence that the IntelliServer must send to perform the function selected by the sequence code. The strings are entered in the same format used within UNIX *termcap* files. Some UNIX systems use a *termcap* file for this; others use a set of *terminfo* files. If your terminal is already working with a UNIX system that uses the former, you should determine which *termcap* entry has been configured for this terminal, and simply transcribe the values you need from the *termcap* file. If you are getting your terminal's specification from another source, such as the terminal's programming manual, I will need to teach you the lingo. Remember, strings are **not** keywords, so they must be entered in correct upper or lower case.

Delays

If a string starts with a number, that number represents a *delay:* it is the number of milliseconds to wait after sending the string, before more data can be sent. This was done to support old slow terminals, so you probably will not need this, but it is part of the *termcap* specification, so we support it for completeness. The IntelliServer implements delays by sending a sufficient number of **NUL** characters to the terminal to take up the required amount of time. For example, the string **50\EC** would mean that the characters **ESC C** would be sent to the terminal, followed by enough **NUL** characters to occupy 50 milliseconds. The delay may be followed by an optional ***** character. This has no significance to the IntelliServer but is permitted for compatibility with *termcap* specifications.

Character Codes in Strings

There are two kinds of data which can be sent to a terminal. Data with a numerical values between 32 and 127 generally represent printable data; for example, the value 65 may represent the letter 'A'. Other numerical values represent commands or the beginnings of special sequences (generally called "escape sequences" because these usually begin with an ASCII "escape" character (numerical value 27). These are generally called "unprintable" characters.

For convenience, there are several methods of representing unprintable characters; we support the conventions used in UNIX *termcap* files, as well as other popular notations.

Table 5-4: Printing the Unprintable

Code	Decimal value	Description
\E	27	ASCII escape character
\n	10	ASCII linefeed (newline)
\r	13	ASCII carriage-return
\t	9	ASCII tab
\b	8	ASCII backspace
\f	12	ASCII form-feed
****		Represents a single backslash.
\^		Represents a caret.
\200	0	ASCII NUL
*****nnn*	octal *nnn*	The ASCII character with octal value *nnn*
^*X*	value of *X* minus 64	**CTRL**-*X*, where *X* can be A-Z or the following: **[ ] ? _ \ ^**

Note: the **** character has special meaning to the command line shell, so when you are using **set term** you need to type **** in place of each single one. For example, to represent a tab you would type **\\t**. When appearing in **show term** the codes appear as in the table. (See example on page 96).

Using this table and some knowledge of the ASCII character codes, you can see that there are at least three ways to represent the *escape* character, viz., **\E**

directly from the table, **\033** because 27 in octal is 33, and **^[** because the **[** character has a value of 91 (64 plus 27).

The cursor addressing string **cm** is more complicated to describe, because the description must give the IntelliServer a rule: how to send the cursor row and column address for different rows and columns. Terminals differ greatly in the way this is done, but fortunately not so greatly that they cannot be described by one or the other of these rules.

The row and column addresses are inserted as *parameters* into the cursor addressing string; the rules describe how these addresses are converted into values to be sent.

Table 5-5: Conversion Experience

Rules:	The rows and columns are assumed to be numbered from 0. Row 0, column 0 is the top left character on the screen. The first parameter is assumed to represent the row. Parameters all begin with **%**..
%%	Represents a single **%**, (when you want the symbol itself).
%d	Output the row or column as a decimal value: column 65 would be sent as **65**.
%2	Output the row or column as a 2-digit decimal value: column 65 would be sent as **65**.
%3	Output the row or column as a 3-digit decimal value: column 65 would be sent as **065**.
%.	Output the row or column as a character: column 65 would be sent as **A**.
%+*x*	Add *x* to the value, then output as a character: if the format string were **%+3**, then column 65 would be sent as **C**.
%r	Indicates that first parameter represents the column, not the row.
%i	Indicates the terminal expects the first row/column to be numbered 1, not 0. (used with %d, %2, or %3 parameters).

These conventions are explained in much more detail in the UNIX documentation for the *termcap* file.

Example: IBM 3151 Configuration

Don’t ask me why there isn’t already a built-in configuration for the IBM 3151. There would have to be at least *one* reasonably popular terminal that isn’t included, and this happens to be one of them. It’s a good thing, because now I have a useful example I can use.

As it happens, my UNIX box has an /etc/termcap file. Here is a fragment of that file:

Example 5-7: Fragment of /etc/termcap

```
I0|ibm3151|i3151|3151:\
   :so=\E4\101:se=\E4\100:\
   :us=\E4\102:ue=\E4\100:\
   :tc=3163
...
m2|ibm3163|i3163|3163:\
...
  :am:cl=\EK:bs:im=:ei=:cm=\eY%+\40%+\40:\
...
  :kd=\EB:kl=\ED:kr=\EC:ku=\EA:kh=\EH:\
  :GU=v:GU=n:
...
```

This is not the entire termcap entry, only enough to give you an idea where my information came from. Notice that colons (**:**) are used to separate one definition from another, and an equal sign (**=**) is used to separate the keyword from its value. Notice also that expression **tc=3163** means that definitions for the 3163 are to be used when they are not defined explicitly for the 3151. Other than that, the keywords and the corresponding values are exactly as needed for entry into the IntelliServer.

In Example 5-8 we are typing the commands which will configure terminal type *uterm0* as an IBM3151.

Example 5-8: Setting uterm0 for IBM 3151

```
(1)    # set term 0 CL \\EK
(2)    # set term 0 cm \\EY%+\\40%+\\40
(3)    # set term 0 so \\E4\\101 se \\E4\\100
(4)    # set term 0 gs \\E<A ge \\E<\\100
(5)    # set term 0 gh q gv x g1 k g2 l g3 m g4 j
(6)    # set term 0 GL u GR t
(7)    # set term 0 KL \\ED
       # set term 0 kr \\EC ku \\EA kd \\EB
       # set term 0 k1 \\Ea^M k2 \\Ec^M
       # set term 0 k4 \\Ed^M k6 \\Ef^M
(11)   # save
```

Some lines are marked with numbers to make it easier to describe what is going on. On lines 1-3 are the commands to set the parameters that are absolutely required for the menus to work: **cl** to define the clear-screen sequence, **cm** to define cursor motion, **so** and **se** to control highlighting. Since keywords can be typed in upper or lower case, I typed **cl** as **CL** so you wouldn't read it as "see-one", but you may use lower case. The values, e.g., **\\EK** must be typed in the correct case, **\\ek** would not do.

On lines 4 through 10 are sequences to define box-drawing characters which will make the menu prettier, and function key sequences that will make the menu easier to use.

Notice the doubling of the \ character as required by the rules in Table 5-4 on page 94.

On line 1 the sequence for clearing the screen is written **\\EK**, which represents a sequence of two bytes: ASCII escape character followed by the letter K.

The sequence on line 2 defines the rules for cursor addressing. The rules are explained in Table 5-5 on page 95, but in this case we did not have to understand them; the sequence comes right from the termcap file.

On line 3 you see I that I can define two or more sequences, **so** and **se** in this case, in the same command. You could have combined more on a line, or less. Whatever is easiest.

On line 4 I have typed the keyword **GH** as **gh**, **GV** as **gv**, etc., to try and make the typing easier. It was. Remember, you can only do this for keywords, not the values.

On line 5, the code for **g2** is an "ell" not a "one". On line 7 the keyword **kl** is typed in uppercase to avoid confusion. (There is in fact a "kay-one" defined elsewhere).

On line 7 the expression **^M** represents the ASCII carriage-return: so the terminal's **F1** key apparently sends the three-character sequence **escape**, **a**, **carriage-return**.

Finally I was so proud of having typed it all correctly that I decided on line 11 to save the configuration to NVRAM so it will be there after I reboot (but we don't need to reboot for these changes to take effect).

Making Sure it's Right

If you have been working along with me, let's check everything and see if yours looks like mine. Type the command, **`show term 0`**.

Example 5-9: Seized With Doubt, He Checks

```
# show term 0
cl: \EK                 cm:  \EY%+\40%+\40
so: \E4\101             se:  \E4\100
CO:                     CF:
GS: \E<A                GE:  \E<\100
GH: q                   GV:  x
G1: k                   G2:  l
G3: m                   G4:  j
GL: u                   GR:  t
kl: \ED                 kr:  \EC
ku: \EA                 kd:  \EB
k1: \Ea^M               k3:  \Ec^M
k4: \Ed^M               k6:  \Ef^M
```

Ok, mine is narrower but I cheated. Check your "ells" and "ones", and "ohs" and "zeros". Whoops! Did we forget something? Where are the codes for **`CO`** and **`CF`**? I can't find any! What do they mean anyway? Turn on the cursor, turn off the cursor. So maybe this terminal doesn't have commands for this. Do I care if the cursor is still turned on when I am in menus? Nah.

Notice that the backslashes are now single instead of double.

Finishing the Job

The user-defined terminal type you have just configured is the one called **`uterm0`**. You might like to start calling it something else, but **`uterm0`** is the

name by which the IntelliServer knows it. The point is, it's a good idea to make a little note somewhere (here for example) so you don't forget what's what:

uterm ___ = terminal mfg/model:	
cl:	cm:
so:	se:
CO:	CF:
GS:	GE:
GH:	GV:
G1:	G2:
G3:	G4:
GL:	GR:
kl:	kr:
ku:	kd:
k1:	k3:
k4:	k6:

All you need to do now is find out which port this terminal will be connected to, and set that port's *Local Term Type* to **uterm0**, as described on page 84.

Now you are ready to log in and try cranking up a menu, eh?

CHAPTER 6

Configuring Modems

In this chapter, you learn how to configure the IntelliServer RAS 2000™ Communications Server (RAS 2000) to work with modems. You also learn something about how to configure modems to work with the RAS 2000.

A checklist of some of the things you are to be concerned with:

- Use of modem for dial-in, dial-out, or both
- Type of data to be sent and received
- Type of flow-control between the modem and the serial port
- Process to make the modem dial a number
- Process to configure the modem for a particular line speed
- Connection of the modem to the serial port

Since there are many types of modems available from many different manufacturers, it is important that you have your modem's user manual nearby. This should include any necessary information about pinouts, jumper and switch settings, and command-mode programming.

Since there are many types of modems and all cannot be discussed here, what will be discussed is the characteristics of *many* modems. You must decide for yourself what things apply to your modem.

General Issues

There are some things you always must be aware of, regardless of what type of modem you are using or what you are using it for. These things are discussed first, before trying to configure anything.

Physical Characteristics

In chapter 5 *Configuring Serial Ports*, you learned how to configure a serial port's line speed, character size, and so on. You need to determine from your modem's instructions how to configure the modem so that its physical characteristics match the characteristics of the port.

Sometimes modems are configured using hardware switches, but more often it is done by sending "Hayes-compatible" commands to the modem from the serial port when the modem is in command mode. Sometimes a combination of commands and hardware switches must be used. Sometimes nothing needs to be done and the modem happens to be configured properly "out of the box". Usually, however, you have to do something.

Most modems distinguish between two types of line speeds (baud rates). The modem talks to the serial port at a certain speed, sometimes called the *serial* or *DTE* rate. But then the modem also talks over the phone line to ther modems at possibly some different speed, sometimes called the *line* or *DCE* rate. In earlier times, the *serial* rate and the *line* rate were always the same so computer equipment would need to adjust its port configuration to accommodate connections at different speeds. Today, most modems can be configured to use a constant *serial* rate, while adjusting their line rates to accommodate the caller's modems.

Always configure your modem to keep the serial line speed constant. This way the RAS 2000's port configuration does not need to change depending on the modem at the other end of the phone line. The examples at the end of this chapter all include configuration commands that do this.

Flow Control

You may also have to configure your modem to match the flow control that your port is using. For greatest flexibility, use RTS/CTS flow control:

- CTS asserted from the modem means the RAS 2000 can send data to the modem.
- RTS asserted by the RAS 2000 means it can receive data from the modem.

In-band flow control such as XON/XOFF can sometimes be used when CTS/RTS is not practical, but not when the data stream itself might contain XON/XOFF characters. It is generally safe for terminals, and when running file-transfer protocols which have been designed to avoid these characters. Even PPP can be used, but it must be configured properly. If you *don't* know what you're sending, RTS/CTS is safer.

Cabling

You must be certain to check your RAS 2000 Communications Server's Hardware Installation Guide for the correct pinout. RAS 2000 with DB25 connectors are designed to connect to *terminals* with straight-through cables; so the signals will have to be swapped around for modems. One rule holds true, regardless of the specific pinout: **A signal on the RAS 2000 should be connected to the signal on the modem having the same name.**

Signals

You must always connect the *signal ground* (***SG***), and the *transmit-data* (**TxD**) and *receive-data* (**RxD**) signals. If you want to know when an incoming call is established and when the remote side disconnects, you must connect *carrier-detect* (**DCD**). You must also ensure the RAS 2000's serial port is configured as a *modem port* (see "Modem Characteristics:"). Sometimes modems can be configured to assert **DCD** all the time, not just when the remote carrier signal is present. You usually do not want this, because how would you then know when the remote side has disconnected? On some modems the options are called *force DCD* and *DCD follows carrier*: you would want to choose the latter in this case.

The *data-terminal-ready* (**DTR**) signal from the RAS 2000 is asserted when the port is opened, dropped when the port is closed, and stays dropped if the port is disabled or the RAS 2000 is off. If you don't want your modems answering the phones when the RAS 2000 Communications Server is turned off, be sure **DTR**

is connected and make sure that the modem has not been configured to ignore **DTR**. If you are using CTS/RTS flow control (see chapter 5, "Output Flow Control Options") be sure to connect both of these signals as well. The RAS 2000 software does not require the use of the *data-set-ready* (**DSR**) and *ring-indicator* (**RI**) signals, so these do not need to be connected.

Dial In, Dial Out

If you are using the modem to support dial-in connections, you need to make sure it is configured for *auto answer*, and for **DCD** to follow the carrier signal (i.e., not be *forced*). Usually, you need to turn off status messages or result codes from the modem. Otherwise, when a user dials in, the modem may send a message such as "CONNECT 14400" to the RAS 2000 Communications Server after it has raised carrier, which the RAS 2000 Communications Server may interpret as a login attempt. When a port is configured to support *only* dial-in connections, command-echoing is often turned off as well.

If you are using the modem to support dial-out connections, you need to consider how the modem is dialed. This is usually done by sending it Hayes-compatible *AT* commands, but the real question is, what actually causes the number to be sent? If this is an outbound PPP/SLIP/CSLIP connection, then the *Dial Script* (see chapter 11, *Dial Scripts*) associated with that port has the commands. If this is a Reverse-TCP port connected to an interactive telnet session on a UNIX host, the user must type the commands.

RAS 2000 Communications Server Configuration

The RAS 2000 has two areas of configuration which are specific to modems; *Dial Scripts* and *Modem Initialization Strings*. The *Dial Scripts* are used by outbound PPP/SLIP/CSLIP ports and are discussed in more detail in chapter 11, *Remote Network Configuration*. The *Modem Initialization Strings* are discussed in this chapter.

Initialization Strings

Why Have Initialization Strings?

A modem initialization string is a series of commands which the RAS 2000 sends to the modem at the beginning of each session. This enables you to ensure that the modem is properly configured to receive the next incoming call. Why initialize the modem between each call? Usually it is possible to configure the modem as you want it and save its configuration (in the modem), and not worry about it again. But what if the modem supports dial-out as well as dial-in connections, and the last one to use it for dial-out messed up the configuration? What if the modem's own configuration storage is not reliable? Also, there are some modems which automatically adjust their DTE speed to match that of data it receives from the port. By sending these initialization strings, the RAS 2000 allows these modems to sense the correct rate.

Finally, by configuring the RAS 2000 to send a fixed initialization string for a certain type of modem, it becomes possible to add additional modems of the same type without having to initialize them by hand.

When Are the Initialization Strings Sent

When is the initialization string sent? The following is the sequence of everything that happens when a login port starts up.

1. Previous session ends, the port is closed and the **DTR** signal is dropped.
2. Port is still enabled for login, port is re-opened and **DTR** asserted after a one-second delay. This delay is to ensure the modem sees **DTR** low for long enough that it hangs up the phone.
3. Modem initialization string is sent approximately $^1/_4$ second after **DTR** is raised.
4. Port waits for the modem to assert **DCD**.
5. RAS 2000 waits one second before starting to send regular data, after seeing the **DCD** signal asserted.

   Now the RAS 2000 expects that data it sends to the modem will be sent to the modem that dialed it. There are some modems which assert **DCD** before completely coming out of command mode. If the delay was not added, some

modems could interpret the data sent (the login prompt, for example) as a command. This can cause a variety of bad things to happen, the best of which is that the caller is disconnected.

6. *Preamble* is sent if one is configured (see chapter 8 *Logging Into the IntelliServer*).
7. Login prompt is sent and the user enters his user name and password, if a login is required.
8. *Message of the Day* is sent after the login, if there is one.
9. Appropriate application is run based on the configuration of the port and that of the user who logged in.

Each port can have its own modem initialization string associated with it. You can enter the string itself as part of port configuration, as described in chapter 5 “Modem Characteristics:” . If you are using the same type of modem on each of a couple dozen ports and the strings are long, this gets old fast. To make your configuration easier and more intuitive, the RAS 2000 can store a table of up to eight different modem initialization strings, each with a name associated. You can choose this name to be something meaningful, like 'Hayes” or “Garfield”. In the port configuration, you can enter the *name* of one of these strings instead of the string itself. When the RAS 2000 is ready to send the initialization, if the string stored in the port configuration matches the name of one of the initialization strings, then the string from the table is sent. Otherwise, it assumes you had stored the string itself and sends that.

In the next section you learn how to maintain these tables.

Using the Menu

You can view and modify entries in the Modem Initialization Table by using the following table, reached by selecting *Modem Configuration* from the *Administration* menu.

Screen 5-4: Modem Initialization Table

```
                    Modem Initialization Strings

     Name                          Initialization string
1 [Hayes   ] [AT &C1 \\Q3\r                                          ]
2 [Garfield] [AT &C1 \\Q1\r                                          ]
3 [        ] [                                                       ]
4 [        ] [                                                       ]
5 [        ] [                                                       ]
6 [        ] [                                                       ]
7 [        ] [                                                       ]
8 [        ] [                                                       ]

Path: Main— Admin— Modem Configuration
```

In this example, entries 3-8 are empty. Entry 1 is named “Hayes” and contains the following commands:

- **AT** - begins the command line.
- **&C1-** only turn on DCD when remote carrier is present.
- **\Q3 -** enable RTS/CTS flow control. Although the command contains a single backslash, you must enter it twice in the table because backslashes are used to introduce special characters (see chapter 5, *Character Codes in Strings)*.
- **\r** - carriage return or end of command. Note the use of the backslash to start a sequence that represents a control character.

If a port is configured to have a modem initialization string “Hayes”, then the port would send **AT&C1\Q3** (***return)*** as the command. There is nothing special about these particular commands, they were chosen for a sample only.

To remove an entry from the table, select the input area for the name and press **ctrl-Z** to erase it or replace it with a different entry. Be careful when you

remove an entry because there is no check to make sure it wasn't being used. If the RAS 2000 can't find an entry in the table whose name matches the *modem init* specified for that port, it assumes what you configured was the initialization string itself, not the name of one.

Using Commands

You can also display and modify these table entries using the *modeminit* command. When you *show modeminit*, each table entry you have defined is displayed along with its table number. To enter a new name and initialization string for a particular entry number, use the command *set modeminit*, as shown in the next example.

Example 6-10: Show Modeminit

```
show modeminit
```

```
# show modeminit
   Name     Initialization String
1  Hayes    AT &C3 \\Q3\r
2  Garfield AT &C1 \\Q3\r
```

In Example 6-11, table entry #3 has been added, table entry #2 has been modified, and table entry #1 has been removed by blanking both the name and the initialization string. Always be careful when you remove an entry or change its name. If any ports were configured to use the old names you must change them as well.

Example 6-11: Set Modeminit

set modeminit *entry name string*
 # set modeminit 3 dennis ATL1N1\r **# set modeminit 2 Garfield "AT &C1 \\Q2\r"** **# set modeminit 1 "" ""**

ISP NOTE:	Usually you want the name stored in the modem initialization table to be some abbreviation of the modem's make or model number. Certain types of modems are going to require the same initialization regardless of which port they are connected to, and when you do port configuration it is easier to keep track of modem names than the initialization strings themselves. Using the strings in the port configuration should be limited to the exception, not the rule.

Initializing Using TIP

Sometimes sending initialization strings automatically is not enough. Sometimes you are going to want to send commands interactively to some modem to determine what its current settings are, to perform internal diagnostics, to make extensive configuration changes, or to try things out and see what they do.

The RAS 2000 provides a command, **`tip`**, that makes this easy to do.

1. Connect the modem to some port which is configured for the correct line speed, character size, flow control, and so on.
2. Verify the port is configured as a modem port. Otherwise **`tip`** complains to you. **`Tip`** does not require that DCD (carrier) be asserted for it to talk to the modem, since you want to configure the modem when it is idle.
3. Verify this port is not currently in use because **`tip`** complains if it is. If the port is in use, you can set its *login type* to *disabled* (see chapter 5, "Port Type—How Will the Port be Used") and then stop any existing processes by using the **`hangup port`** or **`kill port`** commands (see *Example 15-3 on page 376*). It may not be necessary to change the login type, but if your modem is presently forcing carrier, then when you kill the port, another process could start right up. Disabling the port prevents this possibility.
4. Log into the RAS 2000 and go to a command-line prompt.

This example shows what happens when you run the **`tip`** command.

Example 6-12: TIP Session

`tip` *`port-number`*

```
# tip 3
Escape sequence is '~.'
AT
OK
ATDT17705551212
CONNECT
Welcome to the Generic BBS
Type your login name below:
~.
#
```

If the port is already in use you get an error message. Otherwise, the **tip** command first displays a little reminder how to exit from the **tip** session: "*Escape sequence is ~.*" After that, **tip** sends everything you type on the keyboard out to the selected port, and everything that comes in the port is displayed on your terminal.

In Example 6-12 the commands **AT** and **ATDT17705551212** are typed from the keyboard and sent to the modem on port 3, which echoes the keystrokes.

The modem itself generated the messages **OK** and **CONNECT**, shown in italics for clarity. The message *"Welcome to the Generic BBS, Type your login name below:"* was sent by the system that was just dialed into.

When the user finished with the port, he typed **~.** (tilde dot), which is the *tip* command to exit. The pound sign (**#**) on the next line is the RAS 2000 command line interface.

When you are running **tip**, if keys you type are not echoed back it may be that the modem is not configured to echo commands. If keys are echoed back but you do not see responses from the modem, it may be that the modem has been configured not to send responses.You can also use **tip** to dial out from your modem in order to log into some other system with dial-up access. Presumably one not on your local network, otherwise you would have used *telnet* to log into it.

Initialization Strings Examples

Perhaps some of these initialization strings for a few sample modems give you an idea what is involved. Do not treat these examples as an absolute guide, because your modem or your circumstances may be different. Ultimately, you have to study your own modem's documentation to know what works for you.

In these samples, several spaces are added for clarity, but the modem doesn't require them. Every example has a **^M** (carriage-return) at the end.

There is one example of a modem configured for dial-out only. The other examples are for dial-in ports. Any necessary re-initialization for dialing out can be included in the dial script (see chapter 11, *Dial Scripts*).

TABLE 6-6. US Robotics Sportster 14400 (dial-out)

AT&F&C1&D2 V1 S0=0 S2=128 S7=55 &H1&R2 &K1 &N0 &B1^M	
AT	Informs the modem a command follows.
&F	Restore factory defaults.
&C1	Normal DCD operation - DCD signal follows carrier.
&D2	Normal DTR operation - not ignored.
V1	Result codes are verbal, not numeric.
S0=0	Disable auto-answer. You are configuring this modem for dial-out only. (It's probably connected to a outbound PPP port).
S2=128	No command escape code used.
S7=55	Wait 55 seconds for carrier before disconnecting.
&H1	Enable CTS flow control (disable XON/XOFF).
&R2	Enable RTS flow control.
&K1	Auto enable data compression.
&N0	Phone line interface - variable rate.
&B1	Serial port interface - fixed rate.

TABLE 6-7. **US Robotics Sportster 14400 (dial-in)**

AT&F&C1&D2 E0 Q1 S0=1 S2=128 S7=55 &H1&R2 &K1 &N0 &B1^M	
AT	Informs the modem a command follows.
&F	Restore factory defaults.
&C1	Normal DCD operation - DCD signal follows carrier.
&D2	Normal DTR operation - not ignored.
E0	Disable echoing of commands.
Q1	Result codes disabled.
S0=1	Pick up on the first ring because you are using this with a dial-in port.
S2=128	No command escape code used.
S7=55	Wait 55 seconds for carrier before disconnecting.
&H1	Enable CTS flow control (disable XON/XOFF).
&R2	Enable RTS flow control.
&K1	Auto enable data compression.
&N0	Phone line interface - variable rate.
&B1	Serial port interface - fixed rate.

TABLE 6-8. **Telebit T3000 (dial-in)**

AT&F&C1&D2E0Q1S0=1S2=128S7=57S50=0 S51=6S58=2 S68=255^M	
AT	Informs the modem a command follows.
&F	Restore factory defaults.
&C1	Normal DCD operation - DCD signal follows carrier.
&D2	Normal DTR operation - not ignored.
E0	Disable echoing of commands.
Q1	Result codes disabled.

TABLE 6-8. Telebit T3000 (dial-in) (Continued)

AT&F&C1&D2E0Q1S0=1S2=128S7=57S50=0 S51=6S58=2 S68=255^M	
S0=1	Pick up on the first ring because you are using this with a dial-in port.
S2=128	No command escape code used.
S7=57	Wait 57 seconds for carrier before disconnecting.
S50=0	Automatic speed determination (for speed between modems).
S51=6	DTE speed fixed at 38,400 (you are assuming that the modem is attached to a port configured for 38,400).
S58=2	DTE flow control is full-duplex RTS/CTS.
S68=255	DCE flow control follows S58 (RTS/CTS).

TABLE 6-9. Practical Peripherals PC288MT (dial-in)

AT&F1 E0Q1 S0=1 S2=128 S7=57 ^M	
AT	Informs the modem a command follows.
&F1	Restore “IBM compatible” factory defaults. This includes normal DCD and DTR operation (**&C1** and **&D2**) and RTS/CTS flow control (**&K3**) so those commands aren’t needed.
E0	Disable echoing of commands.
Q1	Result codes disabled.
S0=1	Pick up on the first ring, because you are using this with a dial-in port.
S2=128	No command escape code used.
S7=57	Wait 57 seconds for carrier before disconnecting.

CHAPTER 7

Configuring Users

In this chapter you learn about setting up user accounts on the IntelliServer. This involves several topics including authentication, task selection, and time accounting. Things you learn include:

- How to configure and store user accounts locally in the IntelliServer's NVRAM.
- How to configure the IntelliServer so it can receive user authentication from another host on your network, using the RADIUS protocol.
- How to configure users to automatically be dispatched to certain functions when they log in.
- How to make the IntelliServer send time and accounting information to another host on your network using RADIUS protocol and syslog.

Configuring Users: General Issues

Before learning the details of configuring users, it is necessary to define some of the terms and some of the issues involved.

A User is Not Just a Person

From the standpoint of the IntelliServer, a *user* is not the person who is sitting there working with it. **A *user* is the set of things that happen when someone (or something) logs into the IntelliServer in a certain way.** The following examples illustrate various users.

Examples:

1. At the login prompt, log in as “root” and give a password. The IntelliServer gives a command-line prompt because the user “root” in the IntelliServer has been configured to do this.
2. When a client runs software to dial up the IntelliServer and establish a PPP link, the IntelliServer knows *how* to set up this link because the client’s software logged into the IntelliServer and gave a certain user name.
3. When an *autologin* port (see page 69) automatically starts a telnet session with some host on your network as soon as the terminal is turned on, it is because that port was associated with a *user* and this user was configured to do this.

Where is User Information Stored

Users can be configured and stored in the IntelliServer itself or user information can be stored on other hosts on your network. You can store some users one way and some the other.

A user that is configured on the IntelliServer and stored locally in its *non-volatile-RAM* is called an ***NVRAM user***. This is the easiest way to configure a small number of users on a single IntelliServer. The local NVRAM is limited to storing about a hundred users, so if you must support a larger number you must store them on another host.

A user whose information is stored on another host on your network is known as a ***RADIUS user***, because the RADIUS protocol allows the user information to be sent from this host to the IntelliServer. RADIUS stands for *Remote Authentication Dial-In User Service* and is a standard designed to assure interoperability of "Network Access Servers" (of which the *IntelliServer* is an example) from different manufacturers. RADIUS is discussed in more detail in chapter 17, User Authentication using RADIUS).

There Are Three Kinds of Users

Users can be classified based on what needs to happen after they log in. The three kinds of users are:

Table 7-1 Three Kinds of Users

Users	Name	Description
1	Administrative	Wants to get a command line prompt or administrative menu so that he can perform maintenance and configuration on the IntelliServer, as well as telnet and rlogin to other hosts on the network.
2	Login User	Needs to be able to start a telnet session with this host, or an rlogin session with that one, or possibly to choose from among several telnet or rlogin sessions. the object is to get them logged into some other machine on the network, where they can run whatever application you have configured for them. When you configure a *login user* you need to supply information about each of these *connections* the user could establish.
3	PPP User (framed user)	When the user logs in, the intent may be to bring up a PPP or SLIP connection between the IntelliServer's local network and a client computer that has just dialed in. These users are called ***PPP users*** (although they may be using SLIP or CSLIP protocol instead) Sometimes these are called ***framed users*** because PPP, SLIP, and CSLIP are all protocols in which data is *framed* (i.e., separated into well-defined blocks marked by headers). When you configure a ***PPP user***, you need to provide networking information particular to this user so that routes between its network and yours will be set up correctly. A *framed user* exists only for the purpose of bringing up the PPP or SLIP link. Once the network has been extended by this connection, hosts on one side of this connection can connect to hosts on the other side. These connections may include rlogin and telnet sessions, and those users have no relationship to the *framed user* that caused the PPP/SLIP connection to be made.

Keeping Track of User Activity

For the IntelliServer, a *user* represents a job to do. For you, it may represent a source of income. For this to happen you have to be able to keep track of when specific users log in and out. The IntelliServer supports this through the RADIUS accounting protocol *and* through syslog messages.

Connection Tables

When you configure a login user you must specify information about the different rlogin or telnet sessions that can be run. These are called the users' *connections*, and each user can have up to eight. All connections for all users are stored in a *Global Connection Table*, which can store up to 128 connections.

In This Chapter

This chapter is mainly concerned with configuring *NVRAM users*.

Each *selected connection* has four settings associated with it:

- **`Lock:`** Normally any user can change his own selected connection table, even if he does not have administrative privileges. If the administrator *locks* the entries, then the user cannot change them unless *he* has administrative privileges. Generally the administrator locks *all* his selected connections, including any unused ones. If he doesn't, the user will be able to change the unlocked, unused ones to new connections of his own, thereby defeating the purpose of the feature.
- **`GC#`** represents the *global connection number.* All the selected connections you configure for all NVRAM users are automatically pooled into a table called the *Global Connection Table*. Normally this process is invisible to you. When you enter a connection that is absolutely identical to one already defined for another user, the same table entry (and so the same *global connection number*) will be used for the new user. When you change the selected connection for one of these users to something brand new, a new entry in the global connection table is created to hold the new connection, and so the GC# changes.
- **`Command`** is the command that will be automatically run for this user when he logs in, and **`Arguments`** are any command-line arguments that need to be associated with that command. For example, if the command were telnet, the argument might be the host name or IP address.

Commands to Display NVRAM Users

There is only one command to show a user's configuration. It displays all the information for the selected user except for the password, which is always invisible. The command **`show user all`** displays all information for all users (not a summary) so you want to page the output using the **`|`** option.

Example 7-1: Show User

```
show user user-name
show user all
```

```
# show user cass10
User Name: cass10
Comment: This hypothetical user is logged into one of two hosts

Connection option:          Selected connection menu
Initial number of Sessions: 1
Administration Privileges:  No

Sess Lock  GC#  Command   Arguments/Description
  0   No    13  rlogin    160.77.99.100 -l paymaster #Tomahawk Payroll
  1   No    14  rlogin    160.77.99.101 -l goaltender #Birds Payroll
  2   No     0  Disabled
  3   No     0  Disabled
  4   No     0  Disabled
  5   No     0  Disabled
  6   No     0  Disabled
  7   No     0  Disabled
```

This example is a bit more interesting. Since this connection option is *Selected connection menu*, there will be given some sort of menu after login. What will be on the menu? Two selected connections are configured, either to rlogin to one host as that host's user "paymaster" or to rlogin to a different host as that host's user "goaltender", so those are the choices. But, since "rlogin 160.77.99.100" is somewhat cryptic for a menu, the administrator has placed a comment in the arguments section of each connection: the comment is whatever appears after the **#** (pound) sign. When you are using a selected connection menu and there is a comment defined, the comment appears on the menu instead of the command and arguments. So when user *cass10* logs into the IntelliServer, a menu with the following options is displayed:

- Tomahawk Payroll
- Birds Payroll

Chooses one, and the IntelliServer rlogin's the user to the appropriate host.

NOTE	Remember, these commands only apply to users stored in NVRAM. User configurations stored on RADIUS server hosts are viewed using host software appropriate to the particular RADIUS server you are using.

New Users, Old Users

Modifying an existing user is much the same as creating a new one. If you are using commands, you create a new user using the *add user* command, while you will modify settings for an existing user with the *set user* command; aside from that difference the commands are the same. **For the purposes of this section, you are configuring an existing user.**

Using the Command to Configure Multiple Settings

When using the *set user* and *add user* commands, you can specify more than one keyword-value pair in one command. For example, you could type **`set user root connect full admin enabled`**, and the result is the same as if you had set the values for **`connect`** and **`admin`** separately.

Password

You can set the password using two different commands.

Command:

```
set user user-name password password
password user-name
```

You can use the *set user* command to set a user's password along with other elements of his configuration. However, since you would be typing the password with the rest of the command, it will be visible as you type it. For that reason, the alternative *password* command is provided. When you use the *password* command, you will be prompted for the password (and it won't echo back), and then you are prompted again for confirmation.

To Omit the Password Prompt

Normally, the IntelliServer will prompt for a password during login, even if no password was configured (in which case the proper response is to hit enter without typing a password). Sometimes you want to configure a user so when that user name is given, there will be no password prompt. To configure an NVRAM user to skip the password prompt entirely, set his password to **NOPROMPT** (exactly as shown - all upper-case - all one word.)

Comment

The **comment** is provided to allow you to store more extensive information about the user. The IntelliServer does not care what you put here.

Command:

```
set user user-name comment comment
```

NOTE: The comment must be enclosed in " ".

Connection Option

Think of this as the master control for what will happen when the user logs in. There are six choices:

- *Direct Connect per Screen*
- *Selected Connection Menu*
- *Full Connection menu*
- *Inbound SLIP*
- *Inbound CSLIP*
- *Inbound PPP*

The first three connection options are used to support administrative users and login users, those who need to do maintenance on the IntelliServer or to establish telnet and rlogin connections with other hosts on the network. The last three are used to support dial-in users who want to establish PPP, SLIP, and CSLIP connections to computers or networks at their sites.

Command: `set user user-name connect direct`

There are some terminals which support multiple pages of screen display. If you are using one of those terminals, our IntelliView feature allows you to configure separate sessions to run on each of these *virtual screens*. (To configure a port to use IntelliView, refer to page 61 and to chapter 13, IntelliFeatures). *Direct connect per screen* refers to these virtual screens.

When a *direct connect per screen* user logs in, each of the *selected connections* configured for that user can be started on one of his terminal's virtual screens. The number of connections actually started is the least of these three numbers:

- The *Initial number of sessions* (defined below) configured for this user.
- The number of selected connections defined for this user.
- The number of virtual screens supported by that port's IntelliView profile.

If the port is not configured to use IntelliView, then only the first selected connection is used.

If the *Initial number of sessions* is less than the number of selected connections and virtual screens, the user can later activate these screens by hitting **[break] - - -** (that is, the break key followed by three minus signs.)

When the user logs in, his first selected connection is immediately launched on his main screen. If there are other connections for other screens they are not started immediately. Instead they wait until the user selects that screen. Then there is a message, **Hit enter to continue**. After the user does this, *then* the connection is started. If the IntelliServer had tried to start the connection before the operator was ready, it might have timed out again and again.

Exception: If a *direct connect per screen* user logs into a port that has been configured as *Login-by-virtual-screen* *(page 68)* that port operates as a *selected connection menu* user instead. This is done to avoid the potential confusion of keeping track of which connection to apply to which screen when different users log into different screens. It is easier to remove the ambiguity by giving a menu of all the selected connections.

Command: `set user user-name connect selected`

When a user configured for the *selected connection menu* logs in, a menu is presented of the selected connections that were configured for this user. If the terminal type for this port has been defined, the menu will be like those described for the menu system, with borders, highlighted areas, and so on. If the terminal type is unknown, the user will be prompted to enter one of the supported terminal types. If the user is not using one of the supported types or doesn't know what their type is, "unknown" should be selected. While ports with "unknown" terminals are not allowed to run the full administrative menus, they *can* support connection menus. In this case they are displayed line-by-line, scrolling down the screen. For a discussion on terminal types and their effects, see chapter 7, Configuring Users.

If this user has logged into a port that supports multiple screens through IntelliView, each screen will have a menu of all the connections.

Connection Comments

Comments are anything in a connection's command-line arguments following a **#** (pound) sign. This allows an administrator to hide the details from a selected connection menu user.

Command:

```
set user user-name connect full
```

A *full connection menu* user is just like a *selected connection menu* user, except that this one's menu includes every connection in the global connection table. This includes every connection configured for every user, plus any that have been entered directly in the global connection table.

This menu is the same one a user would reach from the *Main Menu* by selecting **Connections** and then **Global Connections List** (see chapter 4, Using the Menu Interface).

The full connection menu differs from the selected connection menu in that it does not hide the commands and arguments when comments are used. The command, arguments, and comments are all displayed. It is presumed that anyone with access to the global connection menu wants to know everything that is going on.

Command:

```
set user user-name connect ppp
set user user-name connect slip
set user user-name connect cslip
```

The last three options I have grouped together because they are nearly the same, the only difference is whether a PPP, SLIP, or CSLIP connection is to be established. When this type of user logs in, the IntelliServer searches through *Remote Profiles* you have configured until it finds one with a suitable configuration; this profile contains additional network information which is used for establishing the link. This process is described in more detail in chapter 17, User Authentication using RADIUS.

Initial Number of Sessions

This applies to users who have been configured as *Direct Connect per Screen* (see page 124) and limits the number of sessions that will be initially started after the user logs in. For example, suppose a user with three selected connections defined logs into a port configured to support a terminal with three virtual screens. If the initial number of sessions were two, then the first two sessions would be started, but the third would not be started until the user presses **break - - -**. (That is, the break key followed by the hyphen three times).

Command:

```
set user user-name initsessions number
```

Since the IntelliServer supports a maximum total number of sessions which is lower than the maximum number of configurable connections per port times the maximum number of ports, the initial number of sessions can be configured low for those who would want to run multiple sessions only occasionally. This way, seldom-used connections won't start until necessary.

Setting Administrative Privileges

A user with administrative privileges is allowed to perform unlimited maintenance within the IntelliServer. While running the command shell any valid command can be performed. If in the menu system, any valid input field can be modified. A user without administrative privileges is only allowed to perform limited maintenance. The IntelliServer will issue error messages as attempts are made to do certain commands (see page 34) and many user fields will be *protected* (see page 51) in the menu system.

Command:

```
set user user-name admin disabled
set user user-name admin enabled
```

A user must have administrative privileges to *telnet* into the IntelliServer (see page 158).

Configuring Selected Connections

The selected connections configured for each user, together with the *connection option*, determine what a user can do when logged in. Selected connections are ignored for users configured as *Incoming PPP*, *SLIP*, or *CSLIP*, but for the other types of users these connections define what the user will be able to do.

Command:

```
set user user-name session number disabled
set user user-name session number gc number
set user user-name session number keyword value
```

You can define up to eight selected connections for a user; in my illustration I show just two lines from the menu screen. When using the third form of the **`set user`** command there you can specify one or more pairs of keywords and values. Table 2 shows the keywords that can be used.

Table 7-2 Keywords in Set User Command

Keyword	Values	Comments
lock	**enabled** *or* **disabled** *shown in menu as* **Yes** *or* **No**	When an entry is locked, this user cannot change it using the menu unless he has administrative privileges. To make the user completely secure, remember to lock *all* his selected connections, including the disabled ones. (You can never change someone else's selected connections or the global connection table unless you have administrative privileges).
gc	*entry number from global connection table*	If the connection you want is already in the global connection table, and you know its number, it saves re-typing it all out. If you enter this number in the menu, the command and arguments field will be filled in when you tab into those areas. If you skip past this area and enter the command and arguments separately, an appropriate number will be filled in here.
command	**shell menu rlogin telnet** *In the menu this is a pick-list.*	The command line shell and main menu can be used to administer and maintain the IntelliServer, as well as to start connections. Telnet or rlogin starts a login session to the specified host.

Table 7-2 Keywords in Set User Command (Continued)

Keyword	Values	Comments
host	*host name or ip address (applies to telnet and rlogin commands)*	In the menu, the *host* and the *arguments* are entered together in the same input area. Using the command, there are separate keywords, but this for clarity. When both are used the values for **host** and **args** are combined to create a single combined string (*host* first) to be stored.
args	*additional command-line options*	
disabled	(none)	In the commands, this disables the specified selected connection for that user. In the menu, this is done by setting the **command** field to **disabled.**

Connection Examples

The following are examples of setting up selected connections.

Suppose you want to set up an administrative user who will get a command-line prompt on up to two virtual IntelliView screens. The commands would be:

Example 7-2: Adding Sessions To An Administrative User "Bill"

```
# set user bill session 0 command shell
# set user bill session 1 command shell
# set user bill initsessions 2 admin enabled
# set user bill connect direct
```

Suppose you want to set up a user to rlogin automatically as user "veep" to some host on your network as soon as he logs into the IntelliServer.

Example 7-3: Adding Sessions To User "Al" Who Doesn't Get To Do Much

```
set user al session 0 command telnet host 160.77.99.102 args "-l veep"
set user al initsessions 2 admin disabled connect direct
```

Special Menu Considerations

If you enter a command and arguments which are exactly the same as an entry already in the global connection table, then the existing entry is used. But the arguments must be absolutely identical.

Orphan Connections

When new connections are added to a user, new connections could be added to the global connection table, or existing connections could be re-used. When a user's connection is changed, the original entry in the global connection table remains. It would have to remain if any other user were using it, but in fact it also is retained even when no user is currently using it. This is intentional and it means that the global connection table retains something of a "catalog" of all previously used connections.

At some point you may want to purge some of these stale entries from the global connection table. Use the Global Connection Configuration Form (Screen 7-1 on page 134) or related commands for this purpose.

Duplicating User Configurations

Often you will want to set up a user who is nearly identical to an existing user by copying the settings from user to another, and then make the changes as appropriate. This can only be done from the command line.

In a classic example, the settings of existing user cronos are used to create a new user, zeus. Then zeus's, settings are copied to a third (existing) user, jupiter.

Example 7-4: Duplicating User Configuration

```
set | add user user-name from user-name
```

```
# add user zeus from cronos
# set user jupiter from zeus
```

Passwords are not copied along with the other information. The new user zeus does not inherit cronos' password, but starts out without a password until one is assigned. In the second example, what will jupiter's password be afterward? He will have no password. Both the *add user…from* and the *set user…from* commands leave the user without a password until a new one is given.

Deleting a User

You can delete a user by using the command line. The command to delete a user is shown below. There is no confirmation required and the user is deleted immediately.

Example 7-5: Command For Deleting A User

```
delete user user-name
```

```
# delete user loki
# show user loki
  loki not found
```

Global Connection Table

Whenever new selected connection is added to a user's configuration, the entry is automatically added to the *global connection table*. This is a master table that contains all the connections configured for all users. When you are using the global connection menu, you can choose to run any of the connections in this table.

You may also add, modify, and delete entries in the global connection table directly, without working through user configuration. Why? Because in most installations with lots of login users, there tends to be more users than there are places to go. If the global connection table number is known for a particular connection, then you can configure the user more quickly. More importantly, if some system-wide parameter changes, you are more likely to be able to make a single change and affect all appropriate users. For example, perhaps lots of users are configured to rlogin to a certain host in order to perform a specific function. But later, this function is moved to a different host on your network. You *could* change each user separately or do the following:

- Look at the user configuration form for one of these users. In his selected connection table will be the global connection number of that connection. Remember it.
- In the global connection menu, find the entry and change it. All other users using that entry will be updated as well.

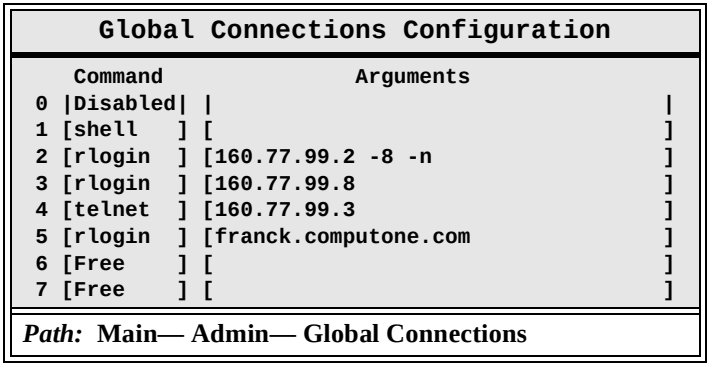
This is possible because the IntelliServer automatically forces users with identical connections to share a single global connection entry. Remember, entries must be completely identical. Even the spacing must be identical or separate entries are created.

If two users were configured with identical connections, and you wanted to make a change for one user only, you would have made the change using user configuration. This would automatically create a new entry in the global table for the user's new connection.

Global Connection Table Form

You can use the menu to view or modify the global connection table. Use the arrow keys to select the input area you wish to modify, or the **`ctrl-F`** and **`ctrl-B`** keys to page forward or back. You get 16 entries per page: we cheated in the illustration.

Screen 7-1: Global Connections — Configuration Form



The commands are pick-lists, you may choose between *menu*, *shell*, *rlogin*, *telnet*, or *disabled*. A sixth choice, *Free*, means that the entry is available to store a new global connection.

To delete an existing global connection, you can try to set its command to *Free*. If there is a user configured with this as one of his selected connections, the IntelliServer won't allow you to make the change. If you want to see a list of all the users who are using a particular global connection, hit **`ctrl-U`** when you have selected that connection. When you modify one of these entries, it affects all users whose selected connection table contains the entry.

Global Connection Table Commands

Viewing the Connection Table

If you want to see all the global connection table from the command-line, use the following commands:

Example 7-6: Displaying the Global Connection table

```
show connection
show connection connection-number
```

```
# show connection
GC#  Command   Arguments
  0  Disabled
  1  shell
  2  rlogin    160.77.99.100 -8 -n
  3  telnet    160.77.99.101 -8 -n
  4  telnet    wrc.whitehouse.gov #Visit the Pres'
  5  rlogin    jp2.vatican.it #Audience with Pope
#
#
#  show connection 3
GC#  Command   Arguments
  3  telnet    160.77.99.101 -8 -n
#
```

If you don't specify a particular connection number, current entries are listed (so you may want to paginate by typing the | key at the end of the command). If you specify a connection number, only that one is shown.

Notice that connection numbers four and five are using comments, probably for someone to use on a selected connections menu. These two also specify a (hypothetical) host in name.domain format rather than by Internet address. You can use either, so long as the IntelliServer has been properly configured to access external nameservers (see page 129).

Modifying The Table

The commands to modify the global connection table are similar to the commands used to modify the selected connection tables for each user (described on page 129).

Example 7-7: Add Connection Command

add connection *command [***host** *hostname] [***args** *args]*
(valid commands are **disabled**, **shell**, **menu**, **telnet**, *and* **rlogin***.)*

```
# add connection telnet host 160.77.99.110 args “-8 -n”
#
#
# show connection 3
GC#  Command   Arguments
 3  telnet 160.77.99.101 -8 -n
#
```

Example 7-8: Set Connection Command

set connection *number* **command** *command [***host** *hostname][***args** *args]*
(valid commands are **disabled**, **shell**, **menu**, **telnet**, *and* **rlogin***.)*

```
# set connection 3 command rlogin host 160.77.99.101 args -8
#
#
# show connection 3
GC#  Command   Arguments
 3  rlogin 160.77.99.101 -8
#
```

Example 7-9: Delete Connection Command

```
delete connection number
# delete connection 2
```

In these examples, a new global connection for a binary-mode telnet to host 160.77.99.110 was added. Then, to modify table entry 3 it was displayed to make sure it was the right one. The command was then changed from telnet to rlogin, changing some command-line arguments as well, because telnet and rlogin *use* different arguments. Then it was displayed again to make sure. Finally, connection number 2 was removed.

Had connection number 2 been included in any user's selected connection list, the IntelliServer would have complained and not removed the entry.

RADIUS Users

In addition to storing user configurations in the IntelliServer's NVRAM, it is also possible to store them on another host on your network. When a user attempts to log in, the IntelliServer will send the login name and password to this host, and the host will send back a reply indicating whether the user is permitted to log in, and what type of services the IntelliServer should provide him. This reply contains much of the same information you have learned to configure for NVRAM users, but sometimes more. In the case of users configured for PPP, SLIP, and CSLIP connections, the reply contains networking information such as the user's remote IP address and routes to any networks which can be reached through his address.

We call these "RADIUS" users because we use RADIUS protocol to send these *authentication requests* from the IntelliServer, and receive the *authentication replies* from the host. As of this writing, RADIUS is an IETF draft standard, adopted and supported by a number of manufacturers. By conforming to this standard, interoperability is encouraged between IntelliServers and devices of similar function from other manufacturers.

There are actually several advantages to storing the users on a host machine in this way:

1. The IntelliServer can store only about a hundred users. The number that can be stored on a host with a large hard disk is virtually unlimited.
2. Many Internet providers use multiple IntelliServers connected to a single phone switch. A particular customer might dial into any one of a number of IntelliServers. If users had to be configured locally, each IntelliServer would need to contain each user just in case he dialed in *there*.
3. When you configure users on a host machine you can store more information than can be kept for an NVRAM user.

RADIUS protocol can also send notification to a host when users log in and out. This is known as "RADIUS Accounting" and if your users are charged based on connect time, these notices can be used for billing purposes.

Discussed next is how to configure the IntelliServer so that it will send authentication requests and accounting information to remote hosts.

RADIUS Configuration

It is easy to configure the IntelliServer to support RADIUS authentication and accounting.

On some host, somewhere on your network, you have installed a software package known as a "**RADIUS server**". This package includes configuration files that have a list of users and their associated configuration, and a means for you to create and maintain this list. There will be a 'daemon" program which runs in the background and listens on the network for authentication requests from "**RADIUS clients**" (including IntelliServers). There will also be configuration files that control which clients this RADIUS server is authorized to respond to, and additional security keys to ensure that the requests are actually coming from the authorized source.

There are different implementations of RADIUS servers, and each differs in the details of how they are installed and how their supporting files are maintained. There, however, will always be three elements:

- The RADIUS Server software (including the RADIUS daemon).
- A user authentication file and some means to maintain it.
- A list of authorized clients, with associated security keys.

Properly configured, the IntelliServer is a "RADIUS client". When a user tries to log in, it sends authentication requests to the RADIUS server. When the RADIUS server gets the request, it looks up the user's information, and sends a reply back to the IntelliServer. What information does the IntelliServer need for this?

- The IP address or host name of the RADIUS server or servers.
- The security key to be used. The IntelliServer uses this to encrypt the password in the authentication request (if the password were not encrypted it could be learned by examining network traffic). The RADIUS server uses it to guarantee that the requests are coming from an authorized source, and the IntelliServer uses it a third time to guarantee that the reply it receives is from the authorized RADIUS server.

Two requirements; two things to configure. You must tell the IntelliServer the IP address of the host it needs to contact. Since the IntelliServer supports a backup, or secondary RADIUS host, you can configure two addresses. The requirements of security dictate that the authentication requests and replies are encrypted. To support this, you configure the IntelliServer with an encryption key called the *RADIUS secret*. The RADIUS host is also configured with this same key.

The RADIUS server won't reply unless the requests come from a RADIUS client (IntelliServer) with the right key, and the IntelliServer won't provide service to the user unless the reply comes from a RADIUS server (host) with the proper key as well.

There is a similar situation with RADIUS accounting. Two hosts to identify via IP addresses. The hosts that perform accounting functions are not assumed to be the same ones performing authentication, but they may be. You could even be doing accounting when all users are configured in NVRAM because they are separate functions as in authentication, there is need to store a security key so that the IntelliServer can tell that its accounting records were received by a host authorized to process them.

RADIUS Menu and Commands

Menu

To find the RADIUS menu, from the Main Menu select **Admin**, **Network** then **RADIUS/SNMP**. (SNMP and RADIUS are unrelated. They are only on the same menu screen because each has only a few fields to configure).

Screen 7-2: SNMP/RADIUS Configuration Form

```
                    Configure RADIUS/SNMP

  Primary RADIUS Host            [160.77.99.110                ]
Secondary RADIUS Host            [160.77.99.111                ]
  Primary RADIUS Accounting Host [160.77.99.112                ]
Secondary RADIUS Accounting Host [160.77.99.113                ]
  RADIUS Retry Count             [0 ]
RADIUS Retry Time                [0 ]
RADIUS CHAP Secret
         [F6tomClancY9                                        ]
Accounting CHAP Secret
         [                                                    ]

 SNMP Trap Host1             [0.0.0.0        ]
 SNMP Trap Host2             [0.0.0.0        ]
 SNMP Get Request Community  [           ]
 SNMP Set Request Community  [           ]
 Enable SNMP                 [No ]
```

Path: **Main— Admin— Network— RADIUS/SNMP**

Commands

From the command-line, you can type **`show radius`** to display the RADIUS configuration, and use the **`set radius`** command family to modify individual items.

Example 7-10: Show RADIUS Command

```
# show radius
RADIUS/Accounting Configuration:

RADIUS Server Host #1        : 160.77.99.110
RADIUS Server Host #2        : 160.77.99.111
RADIUS Accounting Host #1    : 160.77.99.112
RADIUS Accounting Host #2    : 160.77.99.113
RADIUS CHAP Secret           :
    F6tomClancY9
RADIUS Accounting CHAP Secret:
```

In these examples there are primary and a secondary RADIUS Hosts defined and have given the hosts' IP addresses. Host names could have been used instead, but it is customary to supply IP addresses to identify hosts who provide automatic continuous service. To use host names on the IntelliServer, the names must be defined in the IntelliServer's host table, *or* the IntelliServer must know the IP address of one or more nameserver hosts — see page 225. Resolving host names through a nameserver creates additional network traffic which is avoided when the IP address is used. Notice also that in this example separate hosts are used for RADIUS authentication and accounting functions. Often the RADIUS (authentication) host will be the accounting host as well for convenience of account administration, but this is not required.

Commands and Form Items in Detail

Appearing below are each of the fields in the RADIUS configuration form together with their associated shell commands.

Menu:	**Primary Radius Host [*160.77.99.110*]**
Command:	**set radius host 1** *name-or-IP-address*

Menu:	**Secondary Radius Host [*160.77.99.111*]**
Command:	**set radius host 2** *name-or-IP-address*

When a user attempts to log in, if his name and password are not found in the IntelliServer's NVRAM file, the IntelliServer will send a RADIUS authentication request to the *Primary Radius Host* and wait for a reply. If there is a reply, then this user is allowed or denied access according to whether the reply is an acceptance or a rejection.

If there is no reply from the *Primary Radius Host*, the IntelliServer will attempt to re-send the request a few times. If there is still no reply, it assumes that the host is off-line and it attempts to send the authentication request to the *Secondary Radius Host*. If there is no reply from this host after a few retries, access is denied to the user.

If either *Radius Host* field is blank, the IntelliServer does not send the corresponding request. A network with only one RADIUS server running would have the *Primary Radius Host* defined, but the *Secondary Radius Host* left blank. If you are not running RADIUS at all, you would leave both hosts blank.

TIP:	In configuration forms, you erase the input area by typing **`ctrl-Z`**. In the command shell you clear a field by setting the value to a *empty string*, i.e., two double-quotes with nothing in between.

When you change the *Primary* or *Secondary Radius Host*, or the *Radius CHAP Secret*, the change takes effect at the next user login attempt, not counting login retries from a user already trying to log in.

Menu:	**`Radius CHAP Secret`** **`[F6tomClancY9 ]`**
Command:	**`set radius secret`** *`shared-secret`*

This authentication key must match one stored in your *Primary Radius Host* (and *Secondary*, if used). This key is used in three ways:

1. By the IntelliServer, to encrypt the user password before it is sent in the authentication request.
2. By the RADIUS server, to decrypt the password and to ensure the request has come from an authorized RADIUS client (i.e., your IntelliServer and not some snooping software).
3. By the IntelliServer, to confirm that the authentication reply has in fact come from the authorized RADIUS server, not some Trojan-horse software.

Menu:	**`Primary Radius Accounting Host [`*`160.77.99.112`*`]`**
Command:	**`set radius acct 1`** *`name-or-IP-address`*

Menu:	**`Secondary Radius Accounting Host [`*`160.77.99.113`*`]`**
Command:	**`set radius acct 2`** *`name-or-IP-address`*

When you are using RADIUS Accounting, the IntelliServer sends a "start" notice when a user logs in and a "stop" notice when the user logs out or is disconnected. When the RADIUS Accounting server receives this notice, it sends an acknowledgment back to the IntelliServer so the latter knows the notice was received. If the IntelliServer does not receive an acknowledgment after a short time, he will continue to send out duplicate notices for several minutes or until an acknowledgment is received. If an acknowledgment is still not received, the IntelliServer will send a *syslog* error message (see page 193) to your *syslog host*.

Menu:	**Accounting CHAP Secret** **[ ]**
Command:	**set radius acctsecret** *shared-secret*

The IntelliServer uses this key to validate the responses from the accounting server. **Note: some RADIUS accounting servers do not support reply authentication. If this is the case, you must leave this blank.** For example, the "radiusd" reference implementation does not authenticate accounting replies as of this writing.

Here is the exact procedure used for sending Accounting notices:

1. A user logs in, logs out, or is disconnected, the IntelliServer sends an accounting notice. A syslog message of priority LOG_NOTICE is also sent to the *syslog host*, if one is defined (see page 205).
2. The IntelliServer sends a notice to each, if there are both a *Primary* and a *Secondary Accounting Host* defined.
3. The IntelliServer waits for an acknowledgement for each *Accounting host* defined. If an *Accounting CHAP Secret* has been defined, the acknowledgment must be authentic or else it isn't counted. If an acknowledgement is not received, a notice is sent again and again, waiting longer and longer each time for a response. If an acknowledgment is received (from *both* hosts, if they are both defined), then the process is done.
    - If both *Primary* and *Secondary Accounting Host's* are defined, but we only get an acknowledgment from one of them, trying stops after 510 seconds but a *syslog* message of priority LOG_WARNING is sent.
    - If an acknowledgment is not received from **any** *Accounting Host* after 2000 seconds, trying is stopped to send a *syslog* message with this information at the LOG_WARNING level, and a message that the "…accounting host is unresponsive…".

There are three key differences between RADIUS authentication and accounting:

- With RADIUS Authentication, a user cannot log in until authenticated. With Accounting, the user does not have to wait for the accounting start record to be acknowledged before he is granted service; *a fortiori* he does not have to wait for the accounting stop record to be acknowledged before he logs off!
- With RADIUS Authentication, the Secondary host is used only when there is no reply from the Primary. With Accounting, the IntelliServer presumes the Secondary host is there for redundancy, and so sends duplicate information to each host.
- With RADIUS Authentication, the *CHAP Secret* is mandatory. With Accounting, it may be used or not depending on the Accounting server software.

CHAPTER 8

Logging Into the IntelliServer

This chapter discusses what happens when a user logs into the IntelliServer. This includes the following:

- Differences between modem and non-modem ports.
- Modem Initialization strings.
- Preamble and Message-of-the-Day (MOTD).
- Login Prompt.
- Logging in over the network via telnet.
- Using the remote shell (rsh) to run IntelliServer commands.
- RADIUS and Syslogging.

Some of this information has been touched on in other chapters; for more details you should refer to the following:

- To configure a port so that a user can log into it: chapter 5, *Configuring Serial Ports*.
- To configure modems, and your IntelliServer to support them: chapter 6, *Configuring Modems*.
- To configure users and to control what happens when the user logs in: chapter 7, *Configuring Users*.

Logging Into A Serial Port: Checklist

Port Configuration Summary

If a user is to log into a serial port, the port must be configured in one of three ways:

- *Login by Port, wait* — see page 68.
- *Login by Virtual Screen* — see page 68.
- *Login by Port/TCP* — see page 70.

These configurations prompt the user to enter a user name and password. There are two other port configurations which cause a user to be logged in, but in this case the user name is pre-configured for that port and is automatically logged in without prompting anyone for a user name or password.

- *Auto-Login* — see page 69.
- *Auto-Login, wait* — see page 70.

Ports can be configured as *modem ports* or *non-modem* ports (see page 78). *Modem* ports wait for carrier (**DCD**) before prompting for a login, and log the user out when the carrier is lost. *Non-modem* ports ignore carrier.

User Configuration Summary

What happens after a user logs in is controlled by the user configuration. The two most important settings are the following:

- *Connection Option* — a master control which determines what type of service this user receives, and how the *selected connections* are interpreted. See page 124.
- *Selected Connections* — a list of up to eight commands (see page 128) that may be run when the user logs in. These commands may be presented as a menu, or started automatically on different virtual screens, depending on how the *Connection Option* is configured.

Users can be stored in NVRAM or on a separate host using RADIUS protocol (see page 138 and chapter 17, *User Authentication using RADIUS*).

Other Configurations

You can also configure messages to appear during login:

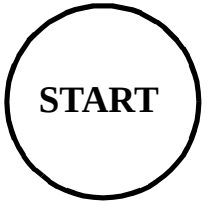
- The *Preamble* appears before the login prompt.
- The *Message of the Day* appears after the user logs in successfully.

Logging Into A Serial Port: Sequence Of Events

These are the events surrounding a users' logging into a serial port. This table is a bit more detailed than the one given in chapter 6, *Configuring Modems*.

The events described are cyclical, one user disconnects, another connects and so on. Starting the description, the port is disabled.

1. When a port is disabled and not in use, the data-set signals **DTR** and **RTS** are not asserted.
2. A user enables the port using a command like **`set port 3 login byport`**.
3. The IntelliServer checks to make sure the port is not already in use.

   Ports configured for *Login-by-Port/TCP* might already have a TCP connection running on them, making them ineligible for login, or someone might be using the **`tip`** command to talk to the port. If the port is in use, The IntelliServer waits two seconds and checks again.
4. The data-set signals **DTR** and **RTS** are asserted, if the port is not already in use.
5. The IntelliServer provides a 200 millisecond delay to give the modem plenty of time to detect **DTR** and **RTS** before any commands are sent.
6. The commands are sent now, if a modem initialization string was configured for this port. Any modem connected to the port would now be ready to answer incoming calls.
7. The modem port waits for the modem to assert carrier (**DCD**), if the port was configured as a *modem port*.

   When a *modem port* is connected to a local terminal, the port's **DCD** is usually wired to the terminal's **DTR** or **RTS** (whichever is *not* being used for flow-control). In that case, the IntelliServer would be waiting not for an incoming call, but for someone to turn the terminal on. If this is a non-modem port, it is assumed that carrier is present. Skip to step 12.
8. The IntelliServer waits for 1 second before continuing, after carrier (**DCD**) is detected by a *modem port*.

This allows any attached modem or device to stabilize, before an attempt to send data to it. For example, there are some modems which assert carrier before coming out of command mode. Data intended for transmission to the remote modem may be interpreted as a command. After this 1 second delay, any data that might have been received so far is flushed before the preamble or login prompt is sent.

9. The IntelliServer again checks to make sure some other process did not begin to use the modem while it was waiting for carrier, after carrier has been detected (or is presumed present for non-modem ports).

   (For example, if the port was configured as *Login-by-Port/TCP*, there might be a carrier because a TCP connection was originated from a network host which has dialed *out* the modem). If the port is in use now, the IntelliServer waits for two seconds. Return to step 3. If the port was not in use, continue.

10. The IntelliServer waits here until some data is received from the port, then it continues, if the port was configured as *await input* (page 79).

    If the port is configured as *Auto-Login* or *Auto-Login, wait*, the user specified for that port is automatically logged in: skip ahead to step 20. Otherwise continue.

11. The *Preamble* is sent, next. This is a message that you have configured for all users to see before they would log in. (See Figure 8-1).

Figure 8-1: Sample Preamble

```
Welcome!
You are now reading the
preamble configured by our
system administrator.

IntelliServer Release 1.3.0
Server Name login:
```

12. The login banner and prompt are displayed after the *preamble*.

    If this is a *modem* port, the user has one minute to log in, starting now. If he does not do so within this time, skip to step 19.

13. The user enters his login name. If this is a *non-modem* port, the user has one minute to log in successfully, starting when the login name is first entered. If the user fails to log in within the time limit, skip to step 19.

    If this user was defined in the IntelliServer's local NVRAM, and the password was set to NOPROMPT, then there is no password prompt: skip to step 17.

14. The password prompt is displayed and keyboard echoing is turned off in preparation for the user entering the password.
15. The user enters the password.
16. The IntelliServer searches NVRAM to see if there is a user configured with this login name and password. If one is found, skip to step 17.

    If there are no RADIUS hosts configured and the user wasn't in NVRAM, this login attempt failed; skip to step 18. If there is a primary RADIUS host defined, the IntelliServer sends an authentication request to that host and waits for a reply. If there is no reply this request is re-sent a few times, and if a secondary RADIUS host is defined it is tried as well. If there is still no reply, or if one of the replies is an *access rejection*, skip to step 18.

17. Login was successful: the user was authenticated either through NVRAM or through RADIUS. The *message-of-the-day* (MOTD) is displayed. Skip to step 20.

Figure 8-2: Sample Message of The Day

```
IntelliServer Release 1.3.0
Server Name login: guest
password:

Thank you for logging in.
You are now reading the
message of the day.
```

18. Login was unsuccessful. After a very short delay, the user is again prompted for login name (without the preamble and login banner, this time) and for password, as if starting back in step 13. The user has five attempts, or one minute to log in, whichever comes first. If login is successful, go to step 17.
19. The IntelliServer disconnects this user by closing the port because all login attempts were unsuccessful or the time limit expired, Skip to step 24.

20. The service configured for this user now begins. If a syslog host is defined (see page 205) a syslog message is sent at LOG_NOTICE priority reporting the user name, port number, and the name of the service the user is being provided (e.g., telnet, rlogin, menu, shell, PPP).
21. If the IntelliServer is configured for RADIUS accounting,(page 143) a “start record” containing the user name, port number, and service is sent to the RADIUS accounting host (or hosts).
22. The user continues the communications session.
23. The user logs off or is disconnected and two kinds of notices are generated. If a syslog host is defined (see page 205) a syslog message is sent at LOG_NOTICE priority, reporting the user name, port, and reason for disconnection. If the IntelliServer is configured for RADIUS accounting (page 143) a “stop record” is sent to the RADIUS accounting host.

24. The port is closed, the **DTR** and **RTS** signals are both dropped. Dropping **DTR** to a connected modem should cause it to hang up the phone line if the modem is properly configured. If the port is configured as a *modem port* the IntelliServer waits for a full second before returning to step 3, otherwise it returns there immediately.

Configuring the Preamble and Message of the Day

The only difference between the *preamble* and *message-of-the-day,* is that one (the preamble) is sent before the user logs in, and the other is sent afterward. They are two different messages, appearing at two different times, but they are configured in the same way.

Menus

The forms for configuring the preamble and message of the day are shown below. The actual input fields are much longer, but I shortened them in the illustration to save space. This makes my examples more cramped than they would be.

Screen 8-1: Login Preamble Configuration Form

```
Login Preamble

1 [\t\tWelcome to Space World\n\n ]
2 [New Users: Log in as guest\n\n ]
3 [To report trouble, dial M \n\n ]
4 [x\nxx\nxxx\nxxxx\nxxxxx\n      ]
5 [                               ]

Path: Main— Admin—Login Preamble
```

Screen 8-2: Message of the Day Configuration Form

```
Message of the Day

1 [ \n\n Remember we will be \n ]
2 [ !!!-OFFLINE-!!!  \n         ]
3 [ on 14 July... \n\n          ]
4 [                             ]
5 [                             ]

Path: Main— Admin—Message of the Day
```

There is *one* preamble and *one* message of the day; the same for all users.

The configuration forms divide the messages into five lines of sixty-five columns each. This is **not** how they will be displayed to the users, however. You are to control the line spacing by inserting control characters as appropriate.The table, "Printing the Unprintable" on page 94, shows how you would enter these control characters. In my examples, **\n** represents a new-line and **\t** represents a tab character.

There is no implicit newline at the end of each of the five lines of configuration, so you could start a line of output text near the end of one line (on the form) and continue it on the next.

Commands

The message of the day and preamble are easier to configure using the menus, but they can also be displayed and configured using commands.

Example 8-1 and Example 8-2 show the commands to display the preamble and message of the day. As with the menus, the tab and newline characters are represented as **\t** and **\n** respectively. To the user logging in, the preamble (and the login banner that follows) would look something like this:

Example 8-1: Command To Display Preamble

```
show preamble
```

```
# show preamble
1 [\n\nWelcome to Triangle BBS\n\n]
2 [     1\n    222\n   33333\n]
3 [  4444444\n 555555555\n\n]
4 [New users log in as scalene\n\n]
5 []
```

Example 8-2: Command to display Message of the Day

```
show motd
```

```
# show motd
1 [We will be closed sometime for ]
2 [the unbestimmt festival\n]
3 []
4 []
5 []
```

This is a good time to point out that the login banner contains two lines: the first line always says “IntelliServer Release” followed by the release number of its software. The next line has the IntelliServer’s *node name* (a.k.a. its host name) followed by a login prompt.

Figure 8-3: IntelliServer Banner

```
Welcome to Triangle BBS

     1
    222
   33333
  4444444
 555555555

New users log in as scalene

IntelliServer Release 1.3.0
triangle login:
```

The commands to modify the preamble and motd operate on a line at a time:

Example 8-3: Set MOTD Command

set motd line *line* **message**
 # set motd line 1 “Did you think\n” **# set motd line 2 “the preamble\n”** **# set motd line 3 “was too short?\n”** **# set motd line 4 “If so remember\n”** **# set motd line 5 “Burma-shave\n”**

Example 8-4: Set Preamble Command

set preamble line *line* **msg**
 # set preamble line 1 “Welcome!\n” **# set preamble line 2 “”** **# set preamble line 3 “”** **# set preamble line 4 “”** **# set preamble line 5 “”**

Notice that a line is erased by setting it to empty double quotes.

Telnet access to the IntelliServer

There are times when you like to be able to configure the IntelliServer without having to come in through a serial port. All the serial ports might be in use or might all be connected to printers. The IntelliServer might be some distance away, accessible to you over the network but not through any serial ports. You might be needing to configure it *because* you are having trouble with the ports. No problem.

The IntelliServer supports up to two simultaneous telnet sessions from remote hosts. That is, if the IntelliServer's IP address were 160.77.99.109, you could walk up to a UNIX box on the same network and type **`telnet 160.77.99.109`** and the IntelliServer is prompting you for a login name.

There are two differences between logging into the IntelliServer over a serial port, and using telnet:

1. Using telnet, you can only log into the IntelliServer as an NVRAM user who was configured with administrative privileges. You cannot log in as a RADIUS user (i.e., a user defined on a remote host and authenticated using RADIUS protocol), and you cannot log in as a user who does not have administrative privileges.
2. A user logging in using telnet will always get the command-line interface ("shell") after login, regardless of the *connection option* in that user's configuration.

This telnet facility was intended to allow administrators to perform remote maintenance on the IntelliServer, so this should not need to support more users than can be stored in NVRAM.

Remote Shell Access

Using telnet to log into the IntelliServer allows you to administer the IntelliServer interactively, but sometimes you may want to run a series of IntelliServer commands automatically from a host system. For example, you might want to kill all the ports on the IntelliServer to put them in a known state when your host machine starts up. You might want to automatically shut down the IntelliServer at certain times, or temporarily disable certain ports when the host machine is unavailable. To meet these needs, the IntelliServer can be configured to accept **`rsh`** (remote shell) connections from hosts on your network.

The **`rsh`** (remote shell) command is available on most UNIX hosts, but on some systems it is called **`remsh`** and on others it is not part of the usual execution path. You will need to consult your host's UNIX documentation for details on implementation on your system. This confusion arises because some varieties of UNIX have historically used **`rsh`** to represent the *restricted shell*, an entirely different command.

Example 8-5: Unix rsh Command (Usual Syntax)

`rsh` *`host command arguments`*
`rsh 160.77.99.201 cat 2` **`rsh 160.77.99.201 kill port 3`** **`rsh myserver set port 4 login disabled`**

The *host* will either be the IntelliServer's IP address, or a host name that your UNIX box can convert to an IP address. The command and arguments are what you would have typed at the IntelliServer's command prompt.

Special Case — rsh cat

The IntelliServer treats the *rsh…cat* command differently from other commands. This is one of the ways to send output from a host to a serial printer and is described in chapter 18, *Reverse TCP and Printing*. The IntelliServer allows you to use this command even when the other commands are disabled.

Commands Other Than cat

Because of security considerations, the IntelliServer's factory default is to ignore remote commands other than *cat*. If you want to allow other commands to be sent, you have to first use the IntelliServer's **`rhosts`** command to enable remote commands and optionally specify an IP address from whom these commands may come.

Example 8-6: Enabling And Disabling Remote Commands

```
set rhosts enabled | disabled
```

```
# set rhosts enabled
# set rhosts disabled
```

By default, remote commands (other than *cat*) are ignored. To allow them to be processed, set rhosts enabled.

Example 8-7: Specifying A Particular Host For Remote Commands

```
add rhosts ip-address
delete rhosts ip-address
```

```
# add rhosts 160.77.99.102
# delete rhosts 160.77.99.102
# add rhosts 160.77.0.0/16
```

This form of the command allows you to specify a particular IP address from whom remote commands may be sent. The use of *add* and *delete* suggest that there can be a list of allowed addresses, but at present the list can only contain one entry. To change the address, delete the old one and add the new one.

If remote commands are enabled and there is no specific IP Address allowed, then remote commands are accepted from any host. If the IP Address is a host address, then remote commands will be honored only from that host. You can also specify a network address, (including an optional bit-count to indicate subnetting) if you want to allow remote commands from a group of hosts. In the example above, **`160.77.0.0/16`** allows remote commands from any host on that Class B network. The addresses don't have to correspond to real networks; you could allow remote commands from IP addresses 160.77.99.200 and

160.77.99.201 by specifying **`160.77.99.200/30`**. This requires the thirty most-significant bits of the address would need to match, and the last bit could be a zero or one.

Example 8-8: Displaying The Current rhosts Settings

```
show rhosts
```

```
# show rhosts
Remote commands: enabled
Allowed hosts: 160.77.0.0/16
```

Restrictions and Limitations

The following are the restriction and limitations of the rsh command:

- The IntelliServer will accept remote commands (other than *cat*) only from the *root* user. If you are running these commands interactively from your UNIX host, you need to have logged in as *root* or have obtained root privileges there. If the commands are run automatically from a script, the process running it needs to have an effective user ID of *root*. Consult your system's UNIX documentation if you are unfamiliar with *effective user ID*s and similar concepts.
- The rsh command is not intended as a substitute telnet for interactive configuration. Consequently, certain commands (such as *menu*) are not supported.
- Command output pagination using the **|** key is not supported.
- Remote commands may respond more slowly than if they were run directly from the IntelliServer.

Finger

In its default configuration, the IntelliServer supports *finger* commands issued from network hosts.

Example 8-9: Using The Finger Command From A Network Host

`finger` *@host*

```
# finger @jeeves
Welcome to the Computone IntelliServer "jeeves"
Running cnx kernel release 1.3.0, version 951031
port  session   owner   command
0     0         root    telnet 160.77.99.203
1     0         root    init awaiting DCD
Systat: 0% user, 3% system, 96% idle, 666K free
Up for 7 days, 7 hours, 7 minutes, 7.7 seconds
connection closed by remote host
```

In this example, the *finger* command was issued on some network host. *Jeeves* is presumed to be the host name of some IntelliServer, which the UNIX host was able to resolve to an IP address. The IntelliServer sent back a reply showing his host name, current software release number, a summary of activity on each port (similar to the IntelliServer's **`whodo all`** command) and a system status report (similar to the IntelliServer's **`systat`** command).

If you do not want the IntelliServer to respond to *finger* requests, set the service port for fingerd to 0, then save your configuration and reboot. To re-enable, change the port back to its default (79), save configuration, and reboot. See "Service Ports" on page 230 for details.

Logging Out

Once you have logged into the IntelliServer, there are five ways you can be logged out:

1. From the IntelliServer's command prompt, enter the **`exit`** or **`logout`** command, or enter **`ctrl-D`** at the beginning of the line.
2. From the IntelliServer's Main Menu, select *Logout* and confirm.
3. If you have dialed into the IntelliServer through a modem port, hang up the phone from your end.
4. If you are logged in as a user who automatically telnets or rlogins to a certain host, when you disconnect from that host you will be logged off the IntelliServer as well.
5. The IntelliServer's system administrator can do a *kill port* or *hangup port* command directed to the port you are on (see page 376).

CHAPTER 9

Network Basics

In this chapter you will learn the basic elements of network configuration. If this is too …elementary for you, feel free to skip ahead. Otherwise this chapter will answer the following questions:

- What are IP addresses?
- What are network addresses and subnets?
- What are host names and domains?
- How is network traffic routed?
- What are Ethernet addresses?
- What is ARP?
- What are PPP, SLIP, and CSLIP?
- What are dialer and login scripts?
- What is syslogging?
- What are IP filters?
- What is RIP?

Basic Definitions

Table 9-1 defines terms used in this chapter.

Table 9-1. Term Definitions

Term	Definition
Host	A specific computer, router, or similar piece of equipment which can receive and send data on your network. It usually refers to a computer which is running one or more *server* and *client* processes.
Client Process	Software that initiates network contact with another piece of software, usually running on a different host.
Server Process	Software that waits for a client process to contact it, then provides the client with the necessary services. For example, when you use *telnet* to log into a remote host, the *telnet* program on your host is the client, and a process called *telnetd* running on the remote host would be the server.
Ethernet	A physical interface that is used to support local area networks.
Ethernet Addresses	Addresses used to identify the source and intended destination of packets that are sent over an Ethernet LAN.
ARP (Address Resolution Protocol)	A protocol for determining the correct Ethernet address of a host, when its IP Address is known.
PPP (Point to Point Protocol)	Protocols for sending network traffic over serial lines.
SLIP (Serial Link Interface Protocol)	Protocols for sending network traffic over serial lines.
CSLIP (Compressed SLIP)	Protocols for sending network traffic over serial lines.
Internet Protocol (IP)	The protocol that allows packets from one host to be routed to the proper destination host. These data packets can travel over an Ethernet LAN, and over PPP, SLIP, or CSLIP links (as well as over other network protocols.) Higher-level protocols like TCP and UDP rely on IP protocol to get the packet to its destination.
IP Addresses	Addresses used in the headers of IP packets to identify the source and intended destination. An IP address can identify either a specific host or a network of hosts.
Router	A host residing on, or with links to, multiple networks. It can receive packets from one network and send them to another.

Term	Definition
Interface	The part of a host that links it to a network. For a PPP link, the interface is associated with a serial port, for Ethernet networks it is associated with the host's Ethernet chip-set.
TCP	A protocol which establishes a reliable connection between two processes, generally on separate computers. Higher-level protocols like telnet and rlogin rely on TCP to ensure that data is not lost in transmission and that data is not sent faster than it can be processed.
UDP	A protocol which establishes *unreliable* packet-based connection between two processes. Higher level protocols like TFTP and RADIUS, which use UDP packets, must themselves include provisions to ensure data is transferred reliably.

IP Addresses

IP Addresses are used to identify a data packet's source and destination. The source identifies which host sent the packet, and the destination identifies which host is supposed to receive it. This includes packets sent to an Ethernet LAN, as well as packets sent over PPP and SLIP links.

Address Classes

There are a few very large networks which have lots of hosts and there are very many small networks which have only a few hosts. An IP address has two parts: the first, or *network portion*, identifies which network the host is on. The second, or *host portion*, identifies the particular host on the network. To accommodate networks of different sizes there are three *address classes* and each divides the network and host portions differently.

An IP address is a number of the form ***nnn.nnn.nnn.nnn*** where *nnn* represents some number between 0 and 255. (Why these numbers? Because these are the smallest and largest numbers that can be stored in an 8-bit *byte* of data. Expressed in binary notation, 255 is 11111111 — every bit "set"— and 0 is 00000000 — every bit "clear").

The byte values 0 and 255 themselves have special meaning, so IP addresses for hosts use the values 1 through 254.

The first number in an IP address determines which class it belongs to.

Table 9-2. Three Classes of IP Address

1.1.1.1 – 126.254.254.254 ***nnn.hhh.hhh.hhh***	Class A addresses - For very large networks. The first number designates the network, the last three designate the host in that network.
128.1.1.1 – 191.254.254.254 ***nnn.nnn.hhh.hhh***	Class B addresses - For medium-sized networks. The first two numbers designate the network, the last two designate the host.
192.1.1.1 – 223.254.254.254 ***nnn.nnn.nnn.hhh***	Class C addresses - For smaller networks. The first three numbers designate the network, the last one designates the host.

If you have a local network and want it to be directly on the Internet, you need to be assigned an official network address. This guarantees that everyone's networks and host IP addresses are unique. Getting a properly registered network address is outside the scope of this manual.

Special Addresses

An IP network has three special numbers associated with it.

- An IP address whose host portion is all zeros refers to that network as a whole and is called the "**network address**".
- The **broadcast address** for a network is used for sending a packet to every host on the network. It is usually configured to be the network address with the host portion set to 255 (in binary, all 1's).
- The **netmask** is not really an address; it is number that indicates which portion of the IP address represents the network portion. Expressed in binary, the netmask has a '1' bit corresponding to each bit in the network portion.

If two hosts have IP addresses with identical network portions, they are said to be on the same network.

Table 9-3. IP Address Examples

`5.0.0.0`	Sample Class A Network Address
`5.255.255.255`	Broadcast address for this Class A network
`255.0.0.0`	Netmask for a Class A Network
`5.1.1.1  5.2.2.2  5.42.12.3  5.100.230.223`	Some host addresses on this Class A network…
`3.1.1.1  16.2.54.23`	…and a *different* Class A network
`160.77.0.0`	Sample Class B Network Address
`160.77.255.255`	Broadcast address for this Class B network
`255.255.0.0`	Netmask for a Class B Network
`160.77.99.101  160.77.111.20  160.77.42.223`	Some host addresses on this Class B network…
`160.78.100.100  150.77.99.101`	…and a *different* class B network.
`192.9.99.0`	Sample Class C Network Address

Table 9-3. IP Address Examples

192.9.99.255	Broadcast address for this Class C network
255.255.255.0	Netmask for a Class C Network
192.9.99.2 192.9.99.100 192.9.99.202	Some host addresses on this Class C network…
192.9.98.1 193.9.99.2	…and a *different* Class C network.

Subnets

Consider the number of possible networks of each class that could be assigned, and the number of hosts there can be on any one of those networks.

The first thing this table tells you is, when you apply for a registered IP network, don't bother asking for a class A. There aren't many to go around. On the other hand, unless you have more than 64 thousand hosts on your network, you probably don't need one.

Table 9-4. Network Limits

Class	Maximum Networks	Maximum Hosts per Network
A	127	16,000,000
B	16,000	64,000
C	2,000,000	254

Even class C addresses allow for up to 254 hosts on a single network and this is often more hosts than will be at a single site. Wouldn't it be more convenient if there were classes in between these classes; or, classes for networks even smaller than Class C?

Suppose you are managing a Class C network with ten hosts, six on a LAN in Toronto, and four on a different LAN in Frobisher Bay and the two sites are connected through a PPP link. Does it make sense to treat all ten hosts as though they were all members of a single network? You would like to say that Toronto has one network and Frobisher Bay a different one, and that these networks are linked somehow. With only the single Class C network at your disposal, how do you do this; by using *subnets*. Subnets allow you to generalize the Class A, B, C structure to allow more flexibility in what parts of the number are considered network and what parts are host addresses.

Subnets and Binary Notation

In order to understand how subnets work, you have to look at the IP addresses as they would be represented in binary notation. As we mentioned before, each of the four numbers in an IP address represent a single 8-bit *byte* of data and the four *bytes* together comprise the IP address.

The different classes of network differ in how many bits are assigned to the network and how many to the host. Class A assigns 8 bits to the network, Class B assigns 16, and Class C assigns 24 bits. Subnets work by assigning additional bits to the network portion that would otherwise be assigned to the host portion of the address. In practice, this is done either by specifying a different netmask for your network or by explicitly giving the number of bits used for the network address.

NOTE: You cannot use subnets to "supernet". You also cannot use a subnet mask or bit count to specify fewer network bits than would be allowed by the IP address class.

The following table shows the standard netmasks for each of the standard classes, together with some possible subnet masks. See how the masks are used to divide a standard class network into 2, 4, 8, 16… separate subnets.

Table 9-5. Subnet Masks For Each Class

Class / Subdivisions	Netmask	Netmask (binary)	Network Bits
A	255.0.0.0	**11111111 00000000 00000000 00000000**	8
A / 2	255.128.0.0	**11111111 10000000 00000000 00000000**	9
A / 4	255.192.0.0	**11111111 11000000 00000000 00000000**	10
A / 8	255.224.0.0	**11111111 11100000 00000000 00000000**	11
A / 16	255.240.0.0	**11111111 11110000 00000000 00000000**	12
A / 32	255.248.0.0	**11111111 11111000 00000000 00000000**	13
A / 64	255.252.0.0	**11111111 11111100 00000000 00000000**	14
A / 128	255.254.0.0	**11111111 11111110 00000000 00000000**	15
B	255.255.0.0	**11111111 11111111 00000000 00000000**	16
B / 2	255.255.128.0	**11111111 11111111 10000000 00000000**	17
B / 4	255.255.192.0	**11111111 11111111 11000000 00000000**	18
B / 8	255.255.224.0	**11111111 11111111 11100000 00000000**	19

Table 9-5. Subnet Masks For Each Class

Class / Subdivisions	Netmask	Netmask (binary)	Network Bits
B / 16	255.255.240.0	**11111111 11111111 11110000 00000000**	20
B / 32	255.255.248.0	**11111111 11111111 11111000 00000000**	21
B / 64	255.255.252.0	**11111111 11111111 11111100 00000000**	22
B / 128	255.255.254.0	**11111111 11111111 11111110 00000000**	23
C	255.255.255.0	**11111111 11111111 11111111 00000000**	24
C / 2	255.255.255.128	**11111111 11111111 11111111 10000000**	25
C / 4	255.255.255.192	**11111111 11111111 11111111 11000000**	26
C / 8	255.255.255.224	**11111111 11111111 11111111 11100000**	27
C / 16	255.255.255.240	**11111111 11111111 11111111 11110000**	28

Smaller subnets for Class C addresses are not shown because any smaller would not allow more than six hosts per subnet. Also, no network was shown being broken into more than 128 subnets, but to proceed further just pick netmasks from the next smaller network class. For example, to break a Class B network into 1024 subnets, use the netmask 255.255.255.192.

As you can see from the table below, the process of subnetting a class A, B, or C network is essentially the same.

Table 9-6. Subnet Mask Rules

First…	Then…	To divide your network into ___ subnets
255, one or more times…	**128**	2
	192	4
	224	8
	240	16
	248	32
	252	64
	254	128

By convention, subnet masks must have consecutive bits set. You would not break up a Class C network by using a netmask of 255.255.255.130, for example, because in binary this would be 1…10000010. There are the same *number* of bits set as in 255.255.255.192, but now they aren't consecutive.

The rule that netmask bits must be consecutive allows you to express a netmask as a single *bit count* equal to the number of bits in the netmask that are set (i.e. the number of bits of network address). These bit counts are frequently used together with network addresses to indicate the degree of subnetting. By convention, the bit count is written after a slash following the network address. It is easier to write about network 160.77.128.0/18 than to write "160.77.128.0 with subnet mask 255.255.192.0". The IntelliServer uses this notation for subnetted network addresses in its routing tables and in IP filters.

You will sometimes see the bit counts used when the network is not subnetted. For example, network address 160.77.0.0/16 refers to an ordinary Class B network with 16 bits of network address, not subnetted.

Subnetting — A Detailed Example

The concept of network addresses, host addresses, and subnetting will be clearer after you study a detailed example. Compare Table 9-7 with Table 9-8.

Table 9-7. Class B Without Subnets

160.77.0.0 or **160.77.0.0/16**	Class B network address without subnets.
160.77.255.255	Broadcast address for this network.
160.77.0.1 – 160.77.255.253	Range of possible host addresses on this network.
160.77.192.0	Example of one host address on the network.

Table 9-8. Class B With Subnet Mask 255.255.192.0

160.77.0.0/18 **160.77.64.0/18** **160.77.128.0/18** **160.77.192.0/18**	The network addresses of each of the four networks created when the subnet mask 255.255.192.0 (bit count 18) is used.
160.77.0.1 – 160.77.63.254	Range of possible host addresses on subnet 160.77.0.0/18.
160.77.63.255	Broadcast address for subnet 160.77.0.0/18.
160.77.64.1 – 160.77.127.254	Range of possible host addresses on subnet 160.77.64.0/18.
160.77.127.255	Broadcast address for subnet 160.77.64.0/18.
160.77.128.1 – 160.77.191.254	Range of possible host addresses on subnet 160.77.128.0/18.
160.77.191.255	Broadcast address for subnet 160.77.128.0/18.
160.77.192.1 – 160.77.255.254	Range of possible host addresses on subnet 160.77.192.0/18.
160.77.255.255	Broadcast address for subnet 160.77.192.0/18.

In this example we have broken up a Class B network into four subnets. Where there was a single network, there are now four. Each network has its own network and broadcast addresses and some of these addresses could have been host addresses before subnetting. For example, IP address 160.77.192.0 would have been a host address on the original network, but now is the network address of one of the new networks.

Whenever you subnet, potential host addresses are lost, corresponding to the new network and broadcast addresses that are created. This is one reason you will never see a Class C network broken into 128 subnets: of the two addresses available for each subnet, one is the network, one is the broadcast, and one is…oops!

When you subnet, the network address of one of the subnets looks like the original network address, except for the subnet (or bit count). Without knowing whether the network was subnetted, and the size of the subnet, you wouldn't know whether "IP Network Address 160.77.0.0" contains hosts 160.77.0.1 - 160.77.255.254, or only the hosts 160.77.0.1 - 160.77.63.254, or some other range corresponding to some different subnet mask. This is why the IntelliServer includes the subnet bit counts in its routing tables.

Host Names and Domains

Although Internet Protocol identifies hosts and networks by their IP addresses, these addresses are not very practical for a human being to use. If you had to remember that Computone's FTP site was, say, 160.77.1.10 and Generic General's WEB page was on 160.77.99.101, and so on, it would get tiresome very quickly. That is why it is possible to identify a host by a name, rather than by its IP address.

Converting Name To IP address — Host Table

In order to send packets to their proper destination they must contain actual IP addresses. When a host is specified by name, that name has to be converted into an IP address. The most direct way to convert a host name to an IP address is to have a file or table stored on your local host computer which lists a variety of host names and their IP addresses. On UNIX machines there is an /etc/hosts file for this purpose. In the IntelliServer there is a host table.

If you were to type the command:

```
telnet fred
```

and there was an entry in your file or table something like this:

```
fred 160.77.99.150
```

then the telnet program would find "fred" in the local table and substitute the given IP address.

Converting Name To IP Address — Name Server

If you have a small network and aren't connected to other networks, keeping a host file on every computer is not a real problem, but imagine if you had a network with 50 hosts and you decided to add a new one. Just log into each of your 50 computers and modify the host tables. Not very practical. Therefore, on larger networks, a single host (or small group of hosts, for redundancy) is given the responsibility of storing host names and addresses. Such a host is called a *nameserver*. Its job is to listen for requests from other hosts and supply IP addresses for particular names. In the above example, if *telnet* had not found *fred* in his local table, he might have sent a request to a nameserver asking whether *it* might know what IP address corresponds to *fred*. That nameserver might have

fred in his table, or he might be configured to check other nameservers. If an IP address is finally discovered, the nameserver sends it back in a reply.

The process of converting a host name to an IP address is known as *name resolution*. When names are *resolved* through an external nameserver, the protocol used is called *Domain Name Service*, or DNS for short.

Domains

Earlier in this chapter we discussed the fact that you must have a registered IP address if you want to have a direct connection to other networks on the Internet. This is necessary to prevent different sites from using identical IP addresses. The use of host names presents a similar dilemma, but one that has been solved by the convention of "domain names".

In much the same way you are given a registered Internet addresses, you can request a registered "domain name" as well. This domain name needs to be added to the end of any host name you assign.

Domain names are organized in a hierarchic structure. The universe of names (the "root domain") has been divided into basic groups, each with its own domain name and each potentially with its own administrator. For example, the domain **.com** assigns domains to commercial networks, **.gov** to government agencies, **.edu** to educational institutions, and **.org** to non-profit organizations. There is also a whole collection of domains for international use: **.de** for Germany, **.nl** for Holland, and **.fr** for France, for example. Notice that the individual elements of a domain name are separated by periods.

Computone is a commercial establishment, so we would get our domain name from whoever administers the **.com** domain. They assign us a domain, **.computone.com**. Every domain *they* assign is going to have **.com** at the end of it because if they assigned one ending in **.nl**, it might conflict with some name assigned by someone in Holland.

At this point, Computone can assign host names without fear of conflict, as long as the names are unique within the domain, they will be unique in the universe. This hierarchy can continue. If Computone had two big departments, each with its own network administrator, we could assign each administrator a separate domain, say, **.eng.computone.com** and **.chan.computone.com.** If both administrators were named Fred, their favorite hosts would probably get named **fred.eng.computone.com** and **fred.chan.computone.com**, respectively. Potential problem averted.

Nameservers and Other Domains

Now that you understand host names and domain names better, let's return to the example given earlier.

1. The user types something like **`telnet fred`**, and there is no fred in any local host table.
2. The IntelliServer knows (because we configured it so) that our local domain is **`.computone.com`** so it tacks that on and sends a request to the nameserver, "what is the IP address for **`fred.computone.com`**?" The nameserver sees from its own tables that it has names for this domain, and fred is there, so it returns the IP address.
3. Later, your type something like telnet **`rs.internic.net`**. In this case, the host name already has a domain associated (it's has periods in it), so **`.computone.com`**. is not tacked on. A request is sent to the local nameserver which checks its own tables and doesn't have a listing for **`rs.internic.net`**. However, its configuration files indicate that there is some other nameserver that *will* have names for that domain. So, our local nameserver sends a request to *that* nameserver, which may send a request to yet another nameserver. Finally, the replies come back and the local nameserver can send its reply.

If you are planning to configure one of your hosts to be a nameserver, give yourself some time to read that system's documentation. Then give yourself some more time to play with things.

Generally, name resolution needs only performed when a program starts up. In the telnet examples above, once an IP address was found for fred, *telnet* remembered it and did not need to resolve the name again as long as the session was up. These name-address assignments are *not* cached on a host-wide basis, however. If two different telnet sessions were started to the same host, each would perform its own name-resolution as it starts up.

Tip	If you can reach a host when you supply its IP address, but not when you supply the name, there is something wrong with the name resolution. There may be something wrong with the nameserver's configuration or the name may be missing from a host table.

Tip	Inordinate delays when first contacting a host that disappear when the host's IP address is used could be because of name resolution latencies, usually a result of a problem elsewhere on the network.

IP Addresses and Routing

Routing is the process of directing an IP packet to its proper destination. When there is only one network, routing is trivial and so it is easy to ignore the issue. When there are several networks and you need to route packets from one network to another, you can no longer ignore the issue.

When a host has a packet to send (either one it has generated itself or one it received from the network), it could do one of five things with it:

1. Send the packet to an appropriate process running on this host, because the packet is addressed to the host itself.
2. Send the packet to a local network. This would include packets addressed to other hosts on the same Ethernet LAN, for example.
3. Send the packet to a host connected to a PPP or SLIP interface.
4. Send the packet to a *different* host on the local network or PPP/SLIP interface; that host being expected to forward it to the correct host.
5. Discard the packet because we don't know what to do with it.

How does the host decide what to do?

Routing Table

To determine how a packet should be disposed of, the host first considers whether the packet is for itself. This is easy because the host knows its own IP address (or IP addresses, when the host is on more than one network). Packets for *this* host are sent to the appropriate protocol or process to be dealt with locally.

For packets addressed elsewhere, our host uses a routing table. Each entry (or *route)* in the routing table has a destination address, a gateway address, and an interface.

- If the destination address is a host address, this is a route to a specific host. A route to a specific host takes precedence over other, more general routes.
- If the destination address is a network address, this route applies to any destinations with host addresses on this network.
- If the destination address is zero, this is a default route. Packets sent to destinations not otherwise accounted for are sent via this route.

Table 9-9 shows the routing table in a hypothetical host. This host is connected to a local Ethernet LAN, and its IP address on *that* network is 160.77.11.101. But, this host is also connected to another network via a PPP connection to a host residing on that network. *That* host's IP address is 160.77.99.200, and to *its* network, our host's IP address is 160.77.99.101. Our local Ethernet LAN is assumed to contain a router which connects us to our Internet Provider, and a second router which connects us to a small network in Lithonia. This network is illustrated in Example 7-1 on page 184.

Table 9-9. Typical Routing Table, With Comments

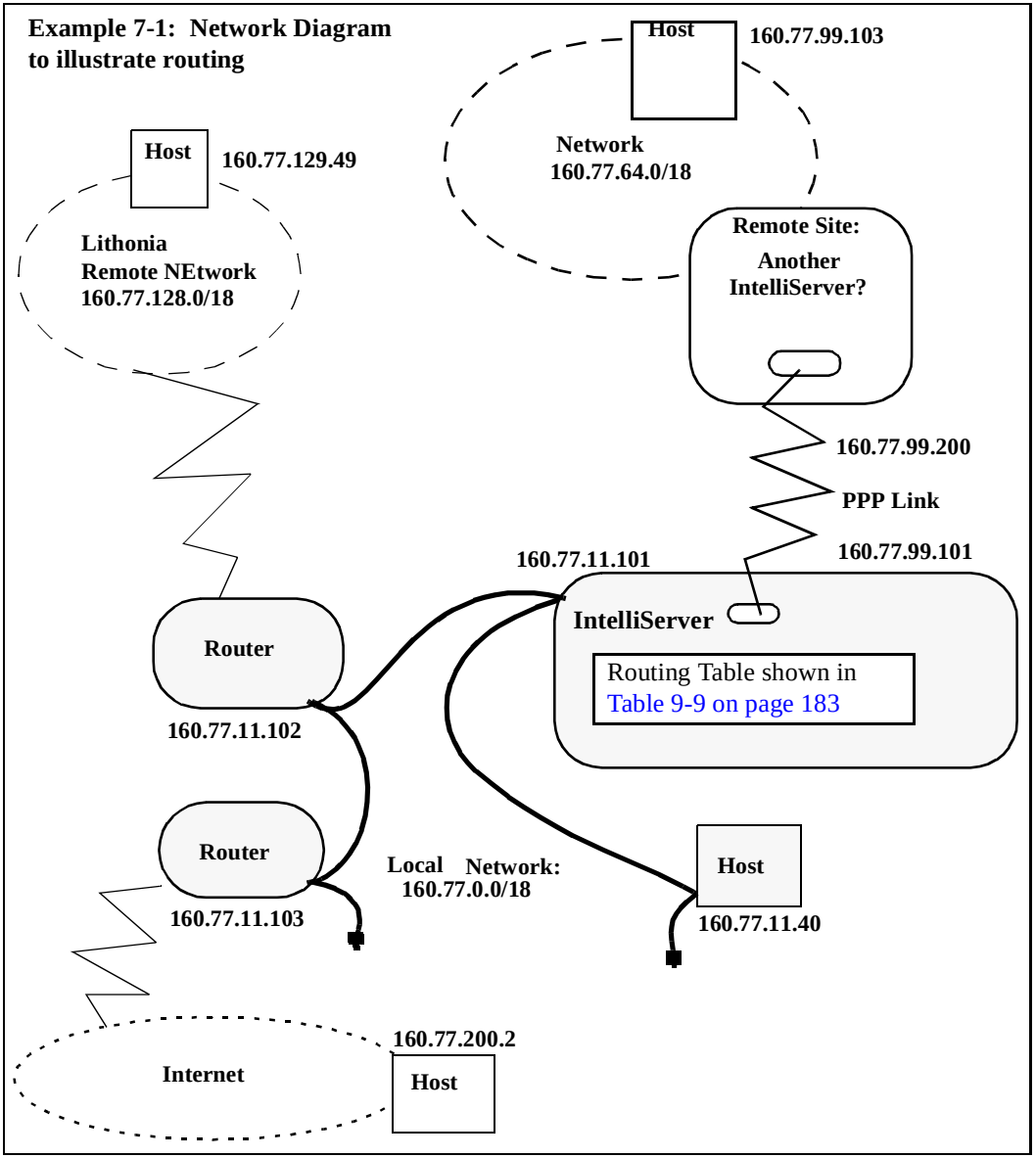
No.	Destination	Gateway	I'face	Comments
1	160.77.0.0/18	160.77.11.101	ether	**Interface route:** Hosts belonging to this subnet are presumed to on our local Ethernet LAN, so they are sent to the Ethernet interface.
2	160.77.64.0/18	160.77.99.200	ppp01	Packets for any host on this other subnet need to be sent to the specific host 160.77.99.200. How do I reach him?
3	160.77.99.200	160.77.99.101	ppp01	**Interface route:** Host address 160.77.99.200 is reached through this PPP interface.
4	160.77.128.0/18	160.77.11.102	ether	Packets for this subnet (presumably the one in Lithonia) are sent to a router on our local LAN: that router's address is 160.77.11.102
5	0.0.0.0	160.77.11.103	ether	Packets that cannot be routed in any other way are sent to a second router on our local network, whose address is 160.77.11.103.

In this example, 160.77.11.101 is the IP address associated with *this* host's Ethernet interface. 160.77.99.101 is the IP address of *this* host's PPP link to a host on a remote network; *his* IP address is 160.77.99.200.

If the gateway used in a route is an IP address associated with one of the host's *own* interfaces, then we call this an *interface route*.

Network Diagram

Confused? Here is a diagram of the network represented by the routes shown in Table 9-9:



Exercise — Routing Sample Packets

The only way to understand how the routes in Table 9-9 actually work is to try and use it to route some packets by hand.

Routes are always checked in this order:

1. Is there a route to this specific host?
2. Is there a route to a network that would include this host?
3. Is there a default route?

This way, more specific routes take precedence over less specific ones. In the examples below, routes searched for but not found are not discussed, only the found and used.

Here are some examples:

- **`160.77.11.101`** — That's *this* host. The packet has arrived at its final destination and will be sent upstream to whatever is running on this host that wants it.
- **`160.77.11.40`** — The network portion of this host address matches the subnet **`160.77.0.0/18`**, and there is a route to that subnet on entry 1, Table 9-9. Furthermore, this is a route to our Ethernet interface so it's decided; send the packet to the Ethernet interface. If the host is on the local LAN like it is supposed to be, the Ethernet interface will figure out how to deliver it to this host.
- **`160.77.129.49`** — This host address is *not* on the subnet described in entry 1, some of the bits in the network portion are different. (Note that if **`160.77.0.0`** had not been subnetted, this host *would* have been on that net). This host is on the subnet described in entry 4. The gateway is **`160.77.11.102`**, and this is not one of our interfaces. *That* host is on the subnet described in entry 4. The Ethernet interface is to deliver this packet to **`160.77.11.102`** on the local LAN.
- **`160.77.99.200`** — There is a route specifically for this host in entry 3 of Table 9-9, because it is the host on the other side of our PPP link to our other network. This is an interface route and the packet should be sent over the PPP link.

- **160.77.99.103** — This host would be on the subnet described in entry 2. The gateway is **160.77.99.200** and is not one of our own IP addresses so this is not an interface route. How do we get to **160.77.99.200** then? Entry 3 gives the answer; send this packet over the PPP link.
- **160.77.200.2** — There is no route specifically to this host, nor is there a route to any network or subnet that would contain it (it would have been on **160.77.192.0/18**). Because there is a default route on line five, use it. The gateway is **160.77.11.103**. How do you reach it? It is on the **160.77.0.0/18** subnet, so the Ethernet interface sends this packet to **160.77.11.103**, as per line entry 1.

Ethernet Addresses and ARP

In the last section you learned about sending packets to the local LAN through the Ethernet interface as though that were the end of it. So far as Internet Protocol and routing is concerned, that is true and their work is done. However, the Ethernet interface (that is, the Ethernet hardware and the software that supports it) still has the work of sending the packet to the correct host on the LAN.

To send a packet to the correct host on Ethernet network, that host's Ethernet address must be known.

Example	Suppose you want to send someone a letter. On the outside of the envelope is her mailing address and inside is the letter. You want her to get the letter quickly, so you give the letter to a guy down the street with a FAX machine. He rips it open, FAXes a copy of the letter *and* the envelope to a guy down the street from *her*, then *that* guy re-creates the envelope and the letter from the FAX and carries it to her house. To FAX the information, our guy needed to know the other guy's FAX number. He couldn't get that number from the address on the envelope, so he probably called directory assistance. Our guys with FAX machines symbolize the Ethernet network. The letters in their envelopes were like IP packets with their destination addresses. The FAX number represents the Ethernet address and calling directory assistance to get the FAX number symbolizes the way ARP (Address Resolution Protocol) is used to determine the Ethernet address when the IP address is known.

What is an Ethernet Address

An Ethernet address is a six-byte number that identifies the actual Ethernet hardware. For example, when you install an Ethernet controller card in your personal computer, that controller comes with its own Ethernet address. When you bought your IntelliServer, it came configured with its own Ethernet address. These addresses are assigned by the hardware manufacturer and you cannot change them. Ethernet addresses are guaranteed to be unique because every manufacturer of Ethernet equipment is assigned their first three bytes uniquely by the IEEE organization. If the manufacturer guarantees that no two boards *he* produces have the same Ethernet address, then nothing anyone else manufactures has the same address either (because the first three bytes would be different).

Ethernet addresses are usually expressed as a series of six bytes, in hexadecimal notation, separated by colons, (e.g., **`80:4e:5f:33:13:01)`**. In this example, **`80:4e:5f`** would have been assigned by IEEE to this particular manufacturer, and the manufacturer would have assigned **`33:13:01`** to this particular Ethernet product. For each set of three numbers assigned by IEEE, the manufacturer is limited to a mere 16 million pieces. Companies with larger productions than this need to acquire additional 3-byte sequences from IEEE.

Now you know what *an* Ethernet address is, but how could hosts possibly keep track of what *the* Ethernet addresses are on the local LAN? After all, you don't even get to pick the numbers so they aren't organized like IP addresses, and they can change unexpectedly, like when the guy in the next cube shuts down and changes the Ethernet controller because the old one was faulty.

Using ARP to Determine Ethernet Addresses

Since it would be impractical to manually maintain tables of IP addresses and corresponding Ethernet Addresses, there is a protocol, called *Address Resolution Protocol* (ARP) which does this automatically. When a host wants to send a packet to some other host on the local network but does not know its Ethernet address, it broadcasts a request to everyone on the local network, saying in essence, "Does anyone know what the Ethernet Address is for IP address 160.77.99.103?" Since the question is broadcast to all the hosts on the local LAN, it should be seen by the host we are looking for. It knows its own Ethernet address and IP address and so it sends back a reply: "160.77.99.103 can be reached via Ethernet address 80:4e:5f:ca:ff:ee".

Now that the sending host knows the Ethernet address, it can send the packet. Suppose it gets another packet for the same host. Does it start all over and send an ARP request again? That would not be wise, because each ARP request is broadcast to every host on the network. If you were going to do this for every packet, you might as well have broadcast each packet to everyone. The other hosts on your network have better things to do than read broadcast messages and throw them away. So, once your host has learned the Ethernet address for a particular IP host address, it retains the information in an *ARP Table.* It always checks its local ARP table first before sending an ARP request.

Do ARP entries stay in the ARP table forever? Generally, no. It is possible to store a permanent ARP entry in the table, but normal entries are dropped from the table if they have not been used for a long time and there is usually a way to purge entries from the table manually. This handles the case where some Ethernet card has been changed, but the IP address has stayed the same. Anyone on the network with ARP table entries made before the swapped out the card have stale information. By removing the ARP reference manually, you don't need to wait for the entry to expire. With the old entry gone, the host will need to perform another ARP request, and in doing so gets the new information.

Proxy ARP

Usually when an ARP request is sent to the network, the target of the request can answer for itself. "Yes, I have that IP address and here is the Ethernet address". But sometimes, it is necessary for a *different* host to answer on its behalf: "Yes, I know who that IP address belongs to, and here is its Ethernet address". A very useful instance of this is when a host is configured to report its *own* Ethernet address as that of the target. This is known as *Proxy ARP.*

If a host is configured for Proxy ARP and it reports its own Ethernet address as being the target host's, it had better expect to receive packets destined for that host. When it receives the packets, it will use its own routing table to send the packets to some other interface. Proxy ARP, then, would not be used if the target host were actually connected to the local LAN. If it were so connected, it could answer ARP requests on its own behalf and receive its own packets. Nor can proxy ARP be used when the target's host IP address is not a member of the local LAN's network. Only host addresses that are members of the local network (as determined by the network portion of their addresses) would have been sent to the Ethernet interface in the first place.

Referring again to Table 9-9, could the router on line 4 (host address **160.77.11.102**) perform Proxy ARP for a host which has an IP address of **160.77.65.3**? Perhaps it could be *configured* to do this, but it wouldn't matter. The routing table says that traffic for this host gets sent to a PPP interface, not to the LAN at all. Could this router perform Proxy ARP for a host with an IP address of **5.10.100.212**? This starts to look promising but the routing table has nothing specific for this IP address or its network, so the default route will be used and the default route *does* go to the Ethernet interface. *But,* this route tells the Ethernet interface that *it* must deliver the packet to host **160.77.11.103**, the gateway. So the Ethernet interface won't try to find **5.10.100.212**'s Ethernet address, it will try to find the gateway's and the gateway will answer for itself.

Could the router (at **160.77.11.102**) perform proxy ARP for a host whose address was **160.77.11.98**? Yes. According to the routing table a host with that address should be found on the local Ethernet network (line 1 of Table 9-9). So the Ethernet interface uses ARP to try to find …**98**'s address, but …**102** answers "Its with me!", putting itself in the position of being able to route packets to this other host *without actually having to tell anyone else to add an explicit route*.

Rule	Proxy ARP can be used to route packets to a target host which is not physically connected to the local Ethernet network, but whose IP address indicates that it would be so connected. The host which performs proxy ARP on its behalf is responsible for forwarding any packets it receives, and so it must have an accurate route to the target host. Other hosts on the local network believe this target host is on the local network, and do not require a special route for it.

As you can see, under certain circumstances Proxy ARP can be useful because it simplifies routing. The danger is that its operation is so *transparent* you often don't realize it is working. Later on you change something in your network and things don't work any more. You check the routing tables and nothing seems to have changed. You check some more and now aren't sure why things were working before.

PPP, SLIP, and CSLIP

As useful as a local Ethernet LAN can be, without remote access it has not achieved its full potential. SLIP, CSLIP, and PPP are protocols for sending IP traffic via asynchronous serial ports instead of over Ethernet or other media. These serial ports can be attached to modems, ISDN terminal adapters, and leased-line equipment in order to access similar equipment at a remote site.

Since SLIP, CSLIP, and PPP links are just another way of carrying IP traffic, hosts on two different networks connected by one of these links can communicate with each other the same as if they were on the same local network (aside from speed considerations). When this is so, the link is *functionally transparent*. When you log into a host using telnet, your telnet program and the host's telnetd program have no idea and do not care whether the hosts are on a single Ethernet LAN, linked by a PPP connection, or linked by a combination of several PPP links and routes through local area networks.

Because of this functional transparency and the widespread availability of SLIP and PPP software, these protocols are increasingly popular. There is not room here for a complete discussion of PPP and SLIP protocols, but a summarize of some important facts is provided

Differences Between SLIP and PPP

The following are some of the differences between SLIP and PPP:

- SLIP supports fewer options and configuration parameters than PPP. In some situations this makes a SLIP link easier to bring up than a PPP link. On the other hand, fewer configuration options also means fewer ways of dealing with potential compatibility issues.
- PPP uses a *frame check sequence* (f*cs)* to check data integrity, SLIP does not.
- SLIP has very minimal packet overhead, although PPP overhead can also be low in certain configurations.
- To reduce packet overhead, PPP can optionally use a technique called *Van Jacobsen Compression*. SLIP can also use this technique, but then it is called CSLIP.

- Since PPP has so many configuration options, the hosts on each end of a PPP link need a means of agreeing on which options they use. This is done through a part of the PPP protocol called *negotiation.* SLIP links do not have such a mechanism and any options necessary for interoperability have to be configured ahead of time to agree.
- PPP has a mechanism to *escape* selected control characters so that these characters never occur in normal data flow. This allows XON/XOFF flow control to be used. SLIP has no such mechanism and so requires you to use hardware flow control (RTS/CTS, for example).

Outbound Connections

In the IntelliServer, outbound PPP and SLIP connections are normally started on demand. When the IntelliServer needs to route a packet and the routing table says to send it to a PPP/SLIP interface, the IntelliServer checks to see if the link is up. If the link is up, the packet is sent over the link.

If the PPP/SLIP link is *not* yet up, the IntelliServer tries to bring up the connection. It does this in two steps.

1. It dials the modem on the remote site. This is done using a *dial script* which was assigned to that port. If the link is running over a leased line, the dial script can often be omitted.
2. It logs into the remote site using a *login script* associated with that site. If you are dialing a commercial provider for this link, his machine will be prompting for a login name and password so it will know who to bill for the service and know how to configure the link.

When this phase is complete, PPP links are ready to start negotiating options. SLIP links are ready to start transferring packets.

Inbound Connections

In the IntelliServer, inbound PPP and SLIP connections are supported by configuring the port as an ordinary login port, but configuring certain users so that when *they* log into the server the information about that particular user is used to help bring up the link. When other users log in, they might get a command prompt, or telnet session or something different.

In most cases, it is not a human who would be logging into the IntelliServer to bring up a PPP or SLIP link. It would be a login script run automatically by the PPP or SLIP software on the other end.

Syslog

What is Syslogging

Do you remember the teletype that used to sit in the middle of the mainframe room that was printing the "system log". When important things happened something would print on the log; the operator would peek over there from time to time to make sure things were alright.

Network syslogging is an extension of this idea, only now things are much more flexible. There are two parts to syslogging:

1. The *syslog host* - a computer on the network running software called a system log daemon (on UNIX hosts usually called *syslogd*).
2. One or more *syslog clients* - computers and devices configured to send syslog messages to the syslog host.

Syslog messages are sent as UDP datagrams and the syslog host does not confirm receipt by returning any acknowledgment. Therefore, syslog messages are intrinsically unreliable. While in most networks most syslog messages will be delivered most of the time, it is still possible that a message can be lost. Messages are especially apt to be lost if an extremely large number are sent to the same host very quickly, and especially if that host is otherwise busy.

Each syslog message contains three parts:

1. The message text (that is, the message itself) - By convention this message begins with something to indicate which of the sender's processes generated the message. That is, messages generated from the *init* process might be expected to begin with *"init:"*, for example.
2. The priority - This identifies the urgency of the message. Syslog clients can be configured to send only messages of a certain priority or higher. Syslog hosts can be configured to store messages based on priority: messages of a certain urgency and higher being sent to a certain file, messages of a certain priority or lower are discarded, and still others are recorded elsewhere.
3. The facility - The facility serves to classify the source of the message. In that way, messages from user processes on the computer can be distinguished from system messages and other types. The syslog daemon can be configured to record messages from different facilities in different files. This differs from

the separation that results for messages with different priorities. When messages of a certain priority are sent to a file, any messages with greater priority would also be sent to that file.

There are two more pieces of information that the syslog host will usually record in the log files:

1. The source (IP address or host name) of the syslog message. The IP address is obtained from the message's packet header.
2. The time the message was received. This is obtained from the syslog host's resident clock.

Syslogging provides an important debugging tool in many situations, so you should be comfortable with it. On the IntelliServer it is especially useful when there is trouble bringing up a PPP connection, and when there is a problem involving Reverse-TCP ports.

IntelliServer Syslog Tips

The following information about syslog may be helpful.

- By default, your syslog host will often store syslog messages from the IntelliServer in the same files as messages from other sources. Sometimes this is good, but at other times you want to separate the messages from a particular IntelliServer. To do this, change the syslog facility for that server, save and reboot. Configure your syslog host to send messages from that facility to a separate file.
- If you are trying to get a PPP link to come up, set your IntelliServer's syslog priority to LOG_INFO. This level generates syslog messages to track the PPP negotiation process.
- To debug Reverse-TCP ports on the IntelliServer, configure the IntelliServer for a priority of LOG_VERBOSE (probably called "LOG_DEBUG" on most syslog hosts). This will generate syslog messages that record all the data sent to and received by any Reverse-TCP host.
- If you cannot get your hands on a syslog host, the IntelliServer can be configured to send the messages to its console instead. If you need to store the messages, attaching a plain terminal to the console would not be a good idea, because the information would just scroll off. If you are debugging something and are needing to look at the syslog information, a better choice would be a computer running a terminal emulation package, one that will store whatever is displayed. In some cases even a printer has been attached.

IP Filtering

Whenever you allow for remote access to your network, there is a question of security. How can you allow remote users to access certain services on your network while denying them access to others? For example, suppose your company's network is connected to the Internet and you have an FTP site that needs to be accessible to everyone so that people can download upgrades. But, there are many other host computers on this same local network. How do you allow the world to reach your FTP site but not allow it to FTP to any different host? How would you prevent outside users from logging into any host on your network including the FTP site?

You *might* rely on login passwords to provide security by reasoning that outside users won't be able to log into unauthorized hosts because they won't know the correct user names and passwords. This reliance on passwords as the *only* protection against unauthorized access is not a good idea.

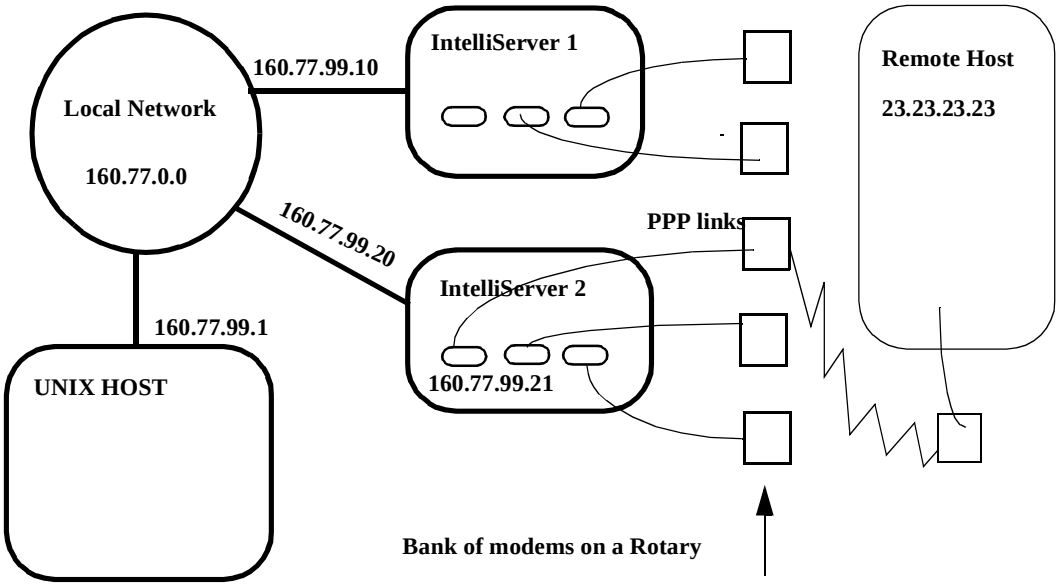
Outside access to your local network will be through some sort of *router.* For example, an IntelliServer connected to your local network might be configured with a PPP link to an Internet provider. The IntelliServer would be acting as a router in this situation by routing network traffic between the Ethernet interface and the PPP link, based on the IP addresses found in the headers. Since all the network traffic has been encapsulated into IP packets, you can use the information in the packet headers themselves to enforce a certain level of additional security.

Each IP packet header contains the IP addresses of the source and intended destination. It also contains the source and destination TCP or UDP ports. This allows the packet to be routed to the appropriate process when it reaches the correct host.

For incoming network traffic, the destination IP address represents a specific host on your network and the port number represents a specific type of service this host may provide. If a router were able to accept or reject these IP packets based on information in the header, this could be used to provide some security for your local network. This technique is known as *IP Filtering.* A set of rules is defined for each interface ("Discard any packets addressed to 160.77.99.23 that come in over the PPP link". "Discard any packets addressed to TCP port 23"). Packets that don't qualify according to the rules are discarded. When properly constructed, an IP filter makes it very difficult for an outside user to access forbidden hosts or services.

Routing Information Protocol (RIP)

Sometimes it is not enough to know a route, you must be able to share it with others. Consider the following example:



In this example, shown are two IntelliServers connected to a local network (160.77.0.0). There is also a UNIX host on the local network with IP address 160.77.99.1. Each of the IntelliServers is configured to support multiple dial-in PPP links through modems attached to various serial ports.

The modems are all connected to phone lines on a single rotary. That means there is a single phone number and the call is passed to the first available line. Here is the important point: *sometimes an incoming call will find a modem on one IntelliServer and sometimes another.*

In this illustration, a remote host (IP address 23.23.23.23) has dialed in and established a PPP link to one of the IntelliServers. When the link comes up, the IntelliServer has a route to destination address 23.23.23.23 through gateway 160.77.99.21, the interface address of one of our inbound PPP links. Likewise, the remote host will have a route to the IntelliServer, and it may have been configured with a static route to our network through the PPP link. So at this point, the IntelliServer could be expected to reach the remote host, and vice-versa.

Will the UNIX host be able to reach the remote host? Yes, the IntelliServer can reach the remote host and the UNIX host can reach the IntelliServer since they are on the same local network. Does it follow that the UNIX host can therefore reach the remote host? No, the UNIX host can only reach the remote host if *it* has a route to that host.

How would the UNIX host get such a route? Could you just configure a static route into this UNIX host? It would route traffic for 23.23.23.23 to the IntelliServer at IP address 160.77.99.21 which is running the PPP link. Why that particular IntelliServer? Because it is running the PPP link. *What about the next time this client dials in?* The rotary might send him to a port on a *different* IntelliServer, then what good is that static route?

Notice that the IP address of the remote host was chosen so that Proxy ARP was not used. Had the remote host's IP address been a member of the local network, Proxy ARP would have sufficed and RIP would not be required. Since the remote address in this example is not a network member, the more general solution through RIP is required.

In this example there would be no static routes required. The IntelliServers would be configured to broadcast their routing tables to the local network using RIP. The UNIX host would be configured to listen for these packets and update its own routing table according to information received. In this configuration it is also important that each IntelliServer listen for RIP packets as well as send them. Otherwise, how would one IntelliServer know about the new routes created by the other (assuming that is important to you)?

It is also possible to send RIP packets over the PPP lines. This becomes necessary when there is another network *behind* the remote host's IP address. How would a host on such a network become aware of routes created by the other IntelliServer? The first IntelliServer would bring up the PPP link and then broadcast the routes to the local network. The second IntelliServer would get a copy of the RIP message, using it to adjust its own routing table. In the fullness of time the IntelliServer would send its own RIP packet to the remote host, so that it could update his routing tables, and so on.

RIP is an important tool when you have to support dynamically assigned IP addresses or network structure; IntelliServer-specific issues are addressed on page 234.

CHAPTER 10

Local Network Configuration

In this chapter you will learn to configure settings that affect the IntelliServer's operation on its local network. Such as:

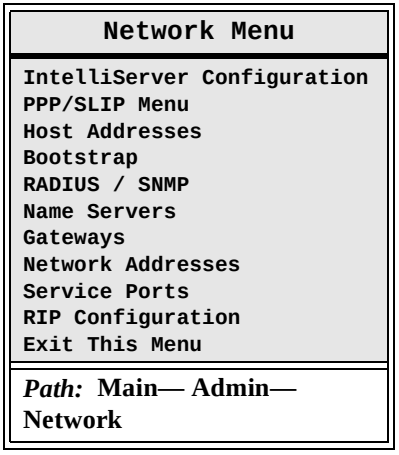
- How to set up the IntelliServer's basic network information.
- How to configure host and network tables.
- How to specify nameservers.
- How to specify syslogging.
- How to configure your IntelliServer to read its configuration and boot from the local network.
- How to write and install IP filters.

If you are already familiar with the basics of network operation, press on. If you are a beginner, make sure you have already read chapter 9, *Network Basics*.

Network Configuration Menu

The configuration forms for local and remote networks are reached from the Network Menu shown here:

Screen 10-1: Network Menu



The following is a brief description of each menu selection:

Table 10-1: Network Menu Selections

Menu Selection	Description
IntelliServer Configuration	Form which has assorted network configurations settings and a few other parameters that apply to the IntelliServer as a whole, or to the IntelliServer's Ethernet interface.
PPP/SLIP Menu	Leads to forms which are used in configuring PPP and SLIP links. These are discussed in the following chapter.
Host Addresses	Table for assigning IP addresses to names.
Boot Strap	Stores information used when booting over the network.

Table 10-1: Network Menu Selections

Menu Selection	Description
RADIUS/SNMP	Form stores IP addresses and other information used with these two protocols.
Name Servers	Lists the IP addresses of hosts which can provide DNS name resolution.
Network Addresses	Table for assigning IP addresses to names.
Gateways	Form that stores static routes to be loaded when the IntelliServer starts up.
Service Ports	Tables to modify if your network does not provide services in the standard way.
RIP Configuration	Form that stores information used by the Routing Information Protocol (RIP).
Exit	Closes NetworK Menu and takes you back to the ADMIN menu.

Displaying the IntelliServer Configuration

When you are using the menus, you can display the current settings by viewing the IntelliServer Configuration Form shown here. This is the same form you will use if you want to modify these settings.

Screen 10-2: IntelliServer Configuration Form

```
          IntelliServer Configuration

Host Name              [jeeves         ]
IP Address             [160.77.99.110  ]
Subnet Mask            [255.255.0.0    ]
Broadcast Address      [160.77.255.255 ]
Domain Name            [computone.com      ]
Syslog Host            [160.77.99.120      ]
Syslog Facility        [LOG_LOCAL5]
Syslog Priority        [LOG_VERBOSE]
Console Port Number    [0  ]
Ethernet Address       |00:80:69:80:09:9d|
Force AUI Port         [No ]
IP Filter              [melita             ]
RIP                    [both  ]

Path: Main— Admin— Network— IntelliServer
```

From the command line, you would show the current settings using the command "**`show server`**" as shown below. You will notice a certain passing similarity between the two displays. There is nothing special about the values shown in *italics*, this is just sample data.

Example 10-1: show server Command

```
show server
```

```
jeeves# show server
Server configuration:
  Name                : jeeves
  IP Address          : 160.77.99.110
  Subnet Mask         : 255.255.0.0
  Broadcast Address   : 160.77.255.255
  Domain Name         : computone.com
  Syslog Host         : 160.77.99.120
  Syslog Facility     : LOG_LOCAL5
  Syslog Priority     : LOG_VERBOSE
  Console Port Number: 0
  Ethernet Address    : 00:80:69:80:09:9d
  Force AUI Port      : No
  IP Filter           :
  RIP                 : both
```

Notice that the IP address, subnet mask, and broadcast address are appropriate for a Class B network. The host name, (*jeeves* in this example) is what appears before the command prompt. The Ethernet Address is shown but it cannot be maintained from the menu. It should never be changed except to change it *back* to its correct value should the NVRAM become corrupt.

Modifying IntelliServer Configuration

You can configure *IntelliServer* parameters using the configuration form (Screen 10-2 on page 202*)* or you can use the **`set server`** command from the shell.

Host Name

This is the name you are giving the IntelliServer itself. When the IntelliServer boots, this name will be displayed as the "Node Name" in the bootup banner message (see Screen 2-1 on page 14). it will also appear in the command prompt. **The name you provide here is *not* automatically added to this IntelliServer's local hosts table**, so the name you supply here may or may not be the name by which this IntelliServer is known on the network.

Menu:	**`Host Name  [`*`jeeves`*`  ]`**
Command:	**`set server name`** *`host-name`*

If you change the host name, the old name will continue to be used until after you save the configuration and reboot.

IP Address, Subnet Mask, Broadcast Address

These are the subnet mask and broadcast address for the Ethernet network the IntelliServer is on, and the IntelliServer's own IP address on that network. This does not affect IP addresses used for PPP and SLIP connections you may configure later.

When you enter an IP address for the first time, if you do not enter the subnet mask and broadcast address they will default to appropriate values, *assuming your network is not subnetted*. If your network is subnetted, you will have to enter the correct subnet mask and broadcast address appropriate to your network. Read chapter 9, *Network Basics*, immediately if you are not following this.

Menu:	**`IP Address  [`*`160.77.99.110`*`  ]`** **`Subnet Mask  [255.255.0.0  ]`** **`Broadcast Address [160.77.255.255 ]`**
Command:	**`set server address`** *`ip-address`* **`set server subnet`** *`subnet-mask`* **`set server broadcast`** *`broadcast-address`*

NOTE: If you change the IP address, subnet mask, or broadcast address after the network is already up, the changes will not take effect until after you save the configuration and reboot.

Domain Name

When the IntelliServer resolves host names into IP addresses, it appends the domain name onto any host name that does not already have a domain associated with it.

Menu:	**Domain Name [*computone.com*]**
Command:	**set server domain** *domain-name*

For example, if your domain were **computone.com** and you typed **ping pong** then the IntelliServer would try to find an IP address for **pong.computone.com.** If you had typed instead **ping pong.internic.net** the default domain would not be added on, because you have already provided one explicitly.

Syslog Host

This is the IP address of some host on your network which is configured to receive *syslog* messages. On UNIX hosts, this is frequently done by a daemon called *syslogd* based on configuration information in a file, */etc/syslog.conf*. If you want to use syslogging and are not already familiar how to configure your host, you will need to refer to that host's manuals or on-line manpages.

Menu:	**Syslog Host [*160.77.99.120*]**
Command:	**set server sysloghost** *host*

Instead of an IP address, you can specify a host name, but *only* if that host name is defined in the IntelliServer's local *Host* table (see page 211). The IntelliServer will not attempt to resolve the syslog host's name through an external name-server. If the syslog host is set to **console** then syslog messages will be displayed on the IntelliServer's console port (usually port 0). If the syslog host is left blank (cleared in the menu using **ctrl-Z** or set in the command to a pair of empty double-quotes), then syslogging is disabled.

Changes to the syslog host, facility, or priority will take effect as soon as the IntelliServer process *doing* the syslogging restarts. This means that syslog messages generated by commands like *telnet* and *rlogin* would be affected by the new settings the next time these commands were run. Syslog messages generated by internal processes like the *logger* (which reports log-in and log-out events) would not be affected by the changes until you save the configuration and reboot.

Syslog Facility

When the IntelliServer sends syslog messages to a syslog host, it identifies them by *facility*. Ultimately, this controls how the syslog host will log these messages: the syslog host will have been configured to record messages from some facilities into one file, and from other facilities into another.

Menu:	**Syslog Facility [*LOG_USER*]**
Command:	**set server facility** *facility-name*

The facility is **LOG_USER** by default, but you can set it to any of the following:

LOG_USER	**LOG_LOCAL1**	**LOG_LOCAL2**	**LOG_LOCAL3**
LOG_LOCAL4	**LOG_LOCAL5**	**LOG_LOCAL6**	**LOG_LOCAL7**

There will probably be processes running on your syslog host which also syslog at the **LOG_USER** facility, so leave the IntelliServer's facility at the default if you want all the messages to be logged together. If your IntelliServer has been configured to generate lots of syslog messages, then you may want them to appear in a file all their own. In this case, you would set it to one of the other facilities that no one else is using and see that your syslog host is configured to put those messages in a separate file.

Syslog Priority

The *priority* indicates which conditions on the IntelliServer will generate messages, and is also used by the syslog host to determine in which file to log the messages.

Menu:	**Syslog Priority [*LOG_NOTICE*]**
Command:	**set server priority** *priority-name*

Table 10-1 shows the eight possible priority settings:

- Higher in the table - only the most urgent conditions will generate syslog messages.
- Lower in the table - less urgent errors and warnings will be sent (as well as all the conditions above).
- Still lower in the table - more mundane information is added until the last entry turns on every last bit of syslogging information that can be had (far too much to leave on all the time).

Table 10-2: Syslog Priorities on the IntelliServer

LOG_EMERG **LOG_ALERT**	• Nothing at all is presently sent from the IntelliServer at this high of a priority: choosing either of these essentially turns off syslogging.
LOG_CRIT	• "Impossible" conditions suggesting software bugs or a hardware fault. • Resource problems or other conditions likely to affect multiple ports until the problem is resolved.
LOG_ERR	All the above, plus: • Normal errors, such as could be caused by configuration mistakes or other user errors.
LOG_WARNING	All the above, plus: • Unusual conditions which do not necessarily mean there is a problem, but which may be informative if there *is* a problem.

Table 10-2: Syslog Priorities on the IntelliServer (Continued)

LOG_NOTICE	All the above, plus: • Users logins, logouts, and disconnects. • PPP/SLIP links coming up and dropping. • Starting and stopping connections to Reverse-TCP ports.
LOG_INFO	All the above, plus: • PPP negotiation and additional information related to bringing up PPP and SLIP connections. • Additional RADIUS and Accounting information.
LOG_VERBOSE	All the above, plus: • **EXTREME** debugging output including dumps of all data sent to and from Reverse-TCP ports. This level corresponds to what is probably called **LOG_DEBUG** on your syslog host.

Console Port Number

This is the port number that is used for displaying system messages, such as the banners which are displayed when the IntelliServer starts up. This is also the port which is used to display syslog messages when the syslog host is **console.**

Menu:	**Console Port Number [*0*]**
Command:	**set server console** *port-number*

The default console is port 0 but you can set it to any port 0-15. To disable console messages entirely, set the console to port **255.**

Ethernet Address

This is the Ethernet address assigned to this IntelliServer and is displayed for informational purposes. You cannot change it from the menu and you should not change it at all, although a command is provided for use at the factory. If this number is changed accidently, the correct Ethernet address for your IntelliServer is provided on a sticker on the back.

Menu:	**Ethernet Address |00:80:69:80:09:9d|**
Command:	**set server ethernet** *Ethernet-address*

When the Ethernet address is changed, the new address does not take effect until the configuration is saved and the IntelliServer rebooted. Furthermore, the new Ethernet address will not be not be saved with the rest of the configuration unless it is saved using the command "**`save ether`**". Otherwise the original Ethernet address will be restored.

Force AUI Port

The IntelliServer's Ethernet interface has both an RJ-69 connector for 10Base2 ("thinnet") cables and an AUI connector, used for attaching transceivers for 10Base5, 10Base-T or other compatible media.

There are some transceivers which do not draw enough current from the AUI to be detected automatically. Setting "Force AUI Port" to "Yes" (enabled) overrides the automatic detection circuit and forces the AUI to be used regardless of whether a transceiver is sensed.

Menu:	**`Force AUI Port [`*`No`*`]`**
Command:	**`set server aui`** *`option`*

In the **`set server aui`** command, *option* is either **disabled** or **enabled.** Any change to this option takes effect only after the configuration is saved and the IntelliServer rebooted.

NOTE: Use this option CAUTIOUSLY: If you accidently set it when you are *not* using the AUI connector and then later save your configuration, when you reboot the IntelliServer you won't have network access and you won't remember why.

IP Filter

This is the name of an IP filter you have defined (see page 239). The IP filter you specify here is applied only to traffic received on the IntelliServer's Ethernet interface. Separate IP filters can be applied to individual PPP and SLIP interfaces.

Menu:	**`IP Filter [`*`melita ]`**
Command:	**`set server filter`** *`filter-name`*

RIP

This specifies whether RIP (Routing Information Protocol) will be used *over the IntelliServer's Ethernet interface*. RIP can be separately enabled or disabled on each PPP or SLIP interface you configure.

Menu:	**`RIP [Both]`**
Command:	**`set server rip`** *`option`*

There are four choices for *option:*

Table 10-3: RIP Interface Options

`send`	Routing information packets will be periodically broadcast over this interface and the IntelliServer responds to specific routing information requests it receives over this interface.
`listen`	The IntelliServer listens for any routing information packets broadcast to this interface by other hosts, updating its routing table as appropriate.
`both`	The IntelliServer both broadcasts and listens for these information packets over this interface.
`none`	The IntelliServer neither broadcasts nor listens for RIP packets over this interface.

Host Addresses

The IntelliServer uses its Host Address Table to resolve host names into IP addresses. Hosts not found in the local table are resolved through external nameservers.

If you are using the menus, you can display and modify the host address table using the form shown in Screen 10-3. Each host name on the left is assigned the IP address on the right. New entries to the table are available as soon as they are added.

Screen 10-3: Host Address Configuration Form

Host Table

	Host Name	IP Address
1	*vanderbilt*	*160.77.99.113*
2	*jeeves*	*160.77.99.110*
3		0.0.0.0
4		0.0.0.0
5		0.0.0.0
6		0.0.0.0
7		0.0.0.0
8		0.0.0.0

Path: **Main— Admin— Network— Host Addresses**

You may also display and modify the host table using the commands shown in Example 10-2 throgh 10-5. The **`add host`** command is used to add a new host into the first open slot in the table. The **`set host`** command is used to change the IP address of a host whose name is already present in the table. The **`delete host`** command removes the entry which has the specified host name or IP address.

Example 10-2: Show Host Command

```
# show host
 Host                Address
vanderbilt           160.77.99.113
jeeves               160.77.88.110
```

Example 10-3: Add Host Command

```
add host hostname ip-address
# add host bertie 160.77.99.114
```

Example 10-4: Set Host Command

```
set host hostname ip-address
# set host bertie 160.77.99.115
```

Example 10-5: Delete Host Command

```
delete host hostname|ip-addr
# delete host 160.77.99.115
# delete host bertie
```

Network Addresses

The IntelliServer uses its Network Address Table to resolve network names into IP addresses. If you are using the menus, you can display and modify the network address table using the form shown in Screen 10-4. Each network name on the left is assigned the IP address on the right. New entries to the table are available as soon as they are added.

Screen 10-4: Network Address Configuration Form

Network Table

	Network Name	IP Address
1	*local*	*160.77.0.0*
2	*frobisher*	*160.78.0.0*
3		0.0.0.0
4		0.0.0.0
5		0.0.0.0
6		0.0.0.0
7		0.0.0.0
8		0.0.0.0

Path: **Main— Admin— Network— Network Addresses**

You may also display and modify the network table using the commands shown below.

Example 10-6: Show Network Command

```
# show network
 Network              Address
local              160.77.0.0
frobisher          160.78.0.0
```

Example 10-7: Add Network Command

```
add network netname ip-addr
# add network fox 160.79.0.0
```

Example 10-8: Set Network Command

```
set network netname ip-addr
# set network fox 160.76.0.0
```

Example 10-9: Delete Network Command

```
delete network name|ip-addr
# delete network 160.76.0.0
# delete network fox
```

The **add network** command is used to add a new network name into the first open slot in the table. The **set network** command is used to change the IP address of a network whose name is already present in the table. The **delete network** command removes the entry which has the specified network name or network IP address.

At this point it will not be apparent to you why you would *need* to configure anything into this table, because the network names defined here have no relationship to domain names. In fact, this table is just a convenience allowing you to use names for networks when specifying routes.

Displaying Bootstrap Configuration

The IntelliServer has the option of running the version of software stored in its internal PROM, or of running a later version stored on one of the hosts on your local network.

Screen 10-5: Network Bootstrap Configuration Form

```
                Network Bootstrap

 Boot Type:                    [TFTP    ]
 Maximum Network Boot Tries:   [4  ]

 Primary TFTP Host:            [160.77.99.222     ]
 Primary TFTP Boot File:
     [/usr/lib/cnx/cnx131                ]
 Primary TFTP Config File:
     [/usr/lib/cnx/jeeves.cfg                    ]

 Secondary TFTP Host:          [                  ]
 Secondary TFTP Boot File:
     [                                   ]
 Secondary TFTP Config File:
     [                                   ]
```

Path: **Main— Admin— Network— Bootstrap**

Booting a newer software version over the network is the most common method of upgrading when new releases of IntelliServer software become available. It also has the option of using the configurations stored in internal NVRAM, or of using configurations stored in a file on one of the network's hosts.

If you are using menus, the network bootstrap options can be displayed and modified using the form shown in Screen 10-5. If you are using commands, you can display the current settings using the **`show boot`** command, shown below:

Example 10-10: Show Boot Command

```
show boot
```

```
Boot configuration:
  Boot type                   : TFTP
  Maximum Network Boot Tries  : 4
  Primary TFTP Host           : 160.77.99.222
  Primary TFTP Boot file      : /usr/lib/cnx/cnx131
  Primary TFTP Config file    : /usr/lib/cnx/jeeves.cfg
  Secondary TFTP Host         :
  Secondary TFTP Boot file    :
  Secondary TFTP Config file  :
```

There is nothing special about the values in *italics*, these are just samples. In the next section we will show how to configure each of these options using either the configuration form or using commands.

Boot Type

This option determines whether the IntelliServer tries to get its software and configuration information from the network. With **`boot type disabled`**, the IntelliServer runs using the software stored in PROM and the configuration stored in its local NVRAM. It does not attempt to get any of this information over the network.

Menu:	**`Boot Type [Disabled]`**
Cmd:	**`set boot type disabled`**

With **`boot type TFTP,`** the IntelliServer uses TFTP protocol to download a *TFTP Boot File* and *TFTP Config File* from a *TFTP Host.* You can specify *Primary* and *Secondary* files and hosts and if the primary fails, the secondary is used.

Menu:	**`Boot Type [TFTP]`**
Cmd:	**`set boot type tftp`**

With **`boot type BOOTP`**, the IntelliServer uses BOOTP protocol to find a host on the network which has been configured to supply this IntelliServer with new software to run or a configuration file to load. When BOOTP is used, you do not need to specify primary and secondary TFTP hosts, boot files, and configuration files. All configuration is done on the host that provides BOOTP services for your network.

Menu:	**`Boot Type [BOOTP]`**
Cmd:	**`set boot type bootp`**

BOOTP protocol acts as a “front end” which provides configuration information to the IntelliServer (i.e., the names of the boot file and configuration file to use). To actually download these files, the IntelliServer uses TFTP just as it would use with **`boot type TFTP.`**

When the IntelliServer is configured to net-boot, it first must bring up its own software in order to start up the networking code so that it can *do* the net-booting. If you are watching the console, you will see this older version’s messages and banners.

Since it knows it must net-boot, the IntelliServer configures itself to allow space in DRAM to download the new software; most serial ports are de-activated and non-essential processes are removed. After it is loaded, the new software is started and if you are watching the console you see *its* power-up messages and banners, which look *almost* like the first set, and then you are running.

When Net-booting Fails

If the net-booting should fail after a pre-determined number of retries, it will finally bring itself up using the software and configuration in PROM and local NVRAM. To do this, it cannot simply stop trying to TFTP the files. It has to actually re-boot itself again, because it had previously reconfigured its DRAM for net-booting. This meant temporarily deleting things which it now must re-load from PROM to recover. The net result is that if you are watching the console you will see a double set of banners in this case as well.

An IntelliServer which has not yet been configured with an IP address will also use BOOTP protocol in order to learn this and other information. This happens regardless of the **`boot type`** settings.

Boot Tries

This controls the number of times the IntelliServer attempts to boot from the network before it gives up and uses the software and configuration stored locally. If the retry count is set to **`0,`** then it continues to retry forever.

Menu:	**`Maximum Network Boot Tries [4]`**
Cmd:	**`set boot retry`** *`retry-count`*

Note	IntelliServer software prior to 1.3.0 did not implement the retry count properly. Retry counts other than 1 would always try forever. If you are net-booting 1.3.0 but have an earlier version in PROM, you will continue to have this behavior, because only the version resident in PROM is capable of interpreting the retry count.

Primary TFTP Host, Boot File, Config File

These are used when the *Boot type* is set to TFTP. The *Primary TFTP Boot Host* is the IP address of the first host the IntelliServer tries to download its files from. The *Primary TFTP Boot File* contains a copy of IntelliServer software (usually a newer version than you are running from firmware) and the *Primary TFTP Config File* contains a configuration file that had earlier been saved from an IntelliServer to this host (see chapter 14, *Saving and Restoring Configurations*).

Menu:	**`Primary TFTP Host [160.77.99.222]`** **`Primary TFTP Boot File:`** **`[/usr/lib/cnx/cnx131 ]`** **`Primary TFTP Config File:`** **`[/usr/lib/cnx/jeeves.cfg ]`**
Command:	**`set boot primary`** *`hostname bootfile config`*

Be careful using the command because you have to specify the host, the bootfile name, *and* the configuration file name in the one command. It does not use keywords to specify the parameters separately. If you want to leave either file blank, put an empty pair of double quotes on the command line to represent the file name.

If **`Primary TFTP Boot File`** is left blank, the IntelliServer runs its firmware version. If the **`Primary TFTP Config File`** is left blank, it uses the version stored in NVRAM.

Secondary TFTP Host, Boot File, Config File

If the IntelliServer is unable to boot from the Primary host, it next attempts to get the information from the Secondary host, if it is configured.

Menu:

```
Secondary TFTP Host [160.77.99.223]
Secondary TFTP Boot File:
     [/usr1/cnx/cnx131      ]
Secondary TFTP Config File:
     [/usr1/cnx/jeeves.cfg   ]
```

Command:

```
set boot secondary hostname bootfile config
```

If both primary and secondary boot hosts are configured, the process works as follows:

1. If the primary and secondary boot files are both blank, the IntelliServer runs the software stored in its PROM. Skip to step 5.
2. If a primary boot file is defined, download it and reboot. If this succeeds, skip to step 5.
3. If there was no primary boot file defined, or if the file could not be loaded, *and* there was a secondary boot file defined, try to download it. If this succeeds, skip to step 5.
4. At least one of the boot files had been defined, but the IntelliServer was unable to load it. It re-boots itself and tries again. It continues to retry until the *Retry Count* specified above has been exhausted. After that it stops trying and uses the version in PROM.
5. Now the proper version of software is running. If a primary config file has been specified, try to TFTP that file and use it as the working configuration. If successful, skip to step 8.
6. If a secondary config file has been specified, try to TFTP it and use it as the working configuration. If successful, skip to step 8.
7. If neither TFTP config file has been specified, or neither could be loaded, use the configuration stored in NVRAM.
8. All ready.

From this outline, you can see that it is possible to load a boot file from one host and a configuration file from a different host by defining a boot file for the primary host and a configuration file for the secondary host.

SNMP is configured using the RADIUS/SNMP configuration form. The top portion of the form is used for configuring RADIUS, as was discussed on page 115 of chapter 7, *Configuring Users.*

Screen 10-6: SNMP/RADIUS Configuration Form

```
                    Configure RADIUS/SNMP

  Primary RADIUS Host            [                              ]
Secondary RADIUS Host            [                              ]

  Primary RADIUS Accounting Host [                              ]
Secondary RADIUS Accounting Host [                              ]

RADIUS CHAP Secret
         [                                                     ]
Accounting CHAP Secret
         [                                                     ]

 SNMP Trap Host1  [160.77.99.175  ]
 SNMP Trap Host2  [0.0.0.0        ]
 Enable SNMP      [Yes]
```

Path: **Main— Admin— RADIUS/SNMP**

The three settings in the lower portion of the form are used for SNMP configuration.

Overview

SNMP, or *Simple Network Management Protocol*, requires the following:

- One or more SNMP *managers*. A manager is a network computer that is running one or more SNMP management applications.
- One or more SNMP *agents*. Agents are network computers and devices (such as the IntelliServer) that can respond to queries from SNMP managers.

The SNMP managers use UDP datagrams to send commands and queries to the agents and the agents send back responses (also using UDP). Agents also can send unsolicited messages, called *traps*, to report important conditions such as shutdown and start-up. The hosts that receive these traps are called *trap hosts*.

Trap Hosts

The IntelliServer can send *trap* messages to as many as two *trap hosts*, which you can configure using the configuration form or through commands.

Menu:	**SNMP Trap Host1 [*160.77.99.175*]** **SNMP Trap Host2 [*0.0.0.0*]**
Command:	**add snmp traphost** *ip-address* **delete snmp traphost** *ip-address*

When using the configuration form, you enter IP addresses in either or both of the spaces provided. If there is only a single trap host, SNMP Trap Host2 is set to **0.0.0.0,** as in the current example.

When using the commands, you either **add** a new trap host or **delete** an existing one. You cannot add a new trap host if there are already two configured. You need to delete one first because this changes the IP address for that host to 0.0.0.0. To change an existing entry, you delete it first and then add a new one.

Enabling & Disabling

When SNMP is enabled, the IntelliServer responds to queries from SNMP managers and sends trap messages to any trap hosts that may be configured. Note that the IntelliServer responds to queries from SNMP managers even when no trap hosts are configured.

Menu:	**Enable SNMP [Yes]**
Command:	**set snmp enabled|disabled**

When SNMP is disabled, it does not listen for queries from SNMP managers and it does not send trap messages, even if trap hosts are configured.

If SNMP is disabled, the IntelliServer re-allocates memory that would have been needed for SNMP support to make the memory available for other processes. This means that when you first enable SNMP, the change cannot take effect immediately because there is no way for SNMP to reclaim the resources it had sacrificed. **When you enable SNMP, the change does not take effect until after you save the configuration and reboot.**

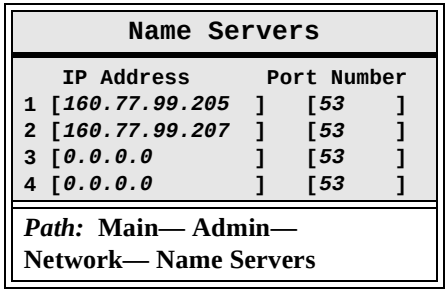
Displaying SNMP Configuration

The current settings are shown on the configuration form (Screen 10-6) or you can display them using the **`show snmp`** command.

Configuring Name Servers

A name server is a host that has been configured to resolve host names into IP addresses. The IntelliServer sends it some host's name, and it sends back a reply with its IP address.

Screen 10-7: Name Server Configuration Form



The IntelliServer can be configured with up to four name servers. If the IntelliServer cannot find the IP address of a host listed in its local host table (page 211), then it sendS a *name resolution request* to the first, nameserver (if defined), then to the second, (if the first had no answer), then to the third, then to the fourth.

You can configure up to four name servers using the menu form shown here on Screen 10-7. For each nameserver, its IP address and service port number are shown. The service port defaults to 53 as shown in the example, and does not need to be changed unless some host was providing name services on a non-standard service port.

NOTE: The IP addresses of the name servers, not their names, must be entered here.

You can use the **show nameserver** command to display the current nameservers. You can also use commands to add and delete entries from the nameserver table.

Example 10-11: Show Nameserver Command

```
# show nameserver
  IP Address       Port Number
160.77.99.206         53
160.77.99.207         53
```

To change an entry, you delete it and add it correctly. Since the IntelliServer tries to use the nameservers in table order, if you are using the commands you should add the nameservers in the order you want them used. This may require deleting and re-adding entries to change the order.

Example 10-12: Add and Delete nameserver commands

add nameserver *ip-address [***port** *number]*

```
# add nameserver 160.77.99.205 port 7035
# add nameserver 160.77.99.210
```

delete nameserver *ip-address*

```
# delete nameserver 160.77.99.205
```

Configuring the Gateway Table

The gateway table contains *static routes* which are automatically added when the IntelliServer starts up and when any new SLIP or PPP links are brought up. Internet Protocol (IP) uses these routes to ensure that data reaches its proper destination. For details on routing tables, see "IP Addresses and Routing" on page 181.

Why is the gateway table re-read when SLIP and PPP connections come up? There may be some routes in your gateway table whose destinations are unreachable when the IntelliServer is first started up, because those destinations are reached through SLIP or PPP links that are not yet up. Such a route cannot be added at that time, but *could* be added after the required SLIP or PPP link has come up.

Gateway Configuration Form

You can display and configure the gateway table using the gateway configuration form shown in Screen 10-8 . (The illustration is shortened for space reasons; the real table is larger).

Screen 10-8: Gateway Configuration Form

```
                Gateway Table

     Destination              Gateway
1 [160.88.31.2          ] [160.77.99.223 ]
2 [160.88.128.0/17      ] [160.77.99.221 ]
3 [160.88.0.0/17        ] [160.77.99.222 ]
4 [default              ] [160.77.99.220 ]
5 [                     ] [              ]

Path: Main— Admin— Network— Gateways
```

The fourth line in the example shows a *default* route. You can enter the destination as either **`default`** or **`0.0.0.0`**, there is no difference. Any IP packet with no other route defined is sent to host **`160.77.99.220,`** (presumably on our local network).

The second line shows a route to one subnet of a Class B network that has been split into two subnets, and the third line shows a route to the other subnet. Traffic for the first subnet is sent to **160.77.99.221**, but traffic for the second is sent to **160.77.99.222**. Subnets, and the use of */nn* notation to express them, are explained in "Subnets" on page 171.

The route on line one is a route to a specific host, **160.88.31.2**. Were it not for that route, traffic for this host would be sent to **160.77.99.222** because **160.88.31.2** is a member of network **160.88.0.0/17.** Because host routes take precedence over network routes, its traffic is sent instead to **160.77.99.223**.

When the IntelliServer reads this table to add its routes to the routing table, it adds them in the order they appear; therefore, the order of routes in this table is important. Do not have routes that depend on other routes further down in the table.

Gateway Command

You can display the present contents of the gateway table by using the **show gateway** command in Example 10-13. This example shows the same routes that were shown configured in Screen 10-8 above.

Example 10-13: Show Gateway Command

```
# show gateway
Destination       Gateway
default           160.77.99.220
160.88.128.0/17   160.77.99.221
160.88.0.0/17     160.77.99.222
160.88.31.2       160.77.99.223
```

There are also commands to add and delete entries from the gateway table, see Example 10-14. In this example, static routes are added to two specific hosts. Then, a host route that had already existed is deleted.

Example 10-14: Add and Delete Gateway Commands

add gateway *destination gateway*
 # add gateway 160.88.31.3 160.77.99.223 **# add gateway 160.88.31.4 160.77.99.224**

delete gateway *destination gateway*
 # delete gateway 160.88.31.2 160.77.99.223

Note that you must specify both the destination *and* gateway of any entry you want to delete.

Note	When you add entries to the gateway table, they are **not** added immediately to the IntelliServer's routing table. Instead, they are added when the IntelliServer has occasion to scan the gateway table (as when the IntelliServer starts up and when any new PPP or SLIP link becomes active). If you want to make immediate changes in the IntelliServer's routing table, use the **route** command.

Service Ports

When client and server processes communicate with each other using Internet Protocol, IP addresses in the *IP header* are used to ensure that the data is sent to the proper host computer. The *IP header* also contains source and destination *port numbers*, which serve to identify which *particular* client or server on a host is the source or destination of that data. It is unfortunate that these numbers are called *ports*. When you are talking about the IntelliServer, "ports" usually refers to *serial ports*, but *these* port numbers have nothing to do with serial ports; they are just numbers used in Internet Protocol.

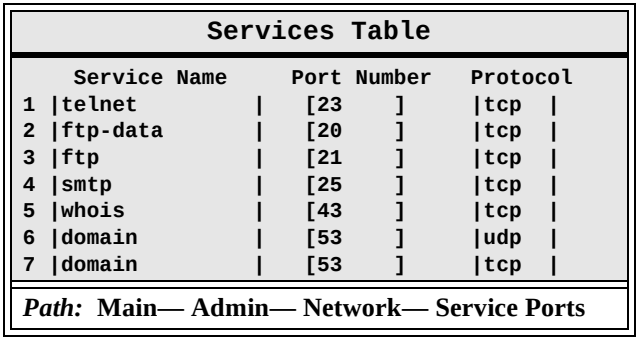
Processes which provide standard services listen on particular *well-known ports*. Client processes which want to get a particular type of service from a host try to make a connection to that *well-known port*. After it does, the server process can assign the client a different port number that applies to that particular session between those particular processes.

Standard well-known port numbers have been assigned to standard services and are listed in the IntelliServers *Service Ports* table. You will probably never need to change the entries unless your network is extremely unusual, but the table is provided nonetheless.

Configuration Form

You can display and modify the *service ports* table by using the configuration form shown in Screen 10-9. Several protocols are listed, each with its own *well-known port*. The column marked "Protocol" shows whether TCP or UDP protocol is used for that service.

Screen 10-9: Service Ports Configuration Form



```
                    Services Table

     Service Name      Port Number    Protocol
1 |telnet          |   [23    ]     |tcp  |
2 |ftp-data        |   [20    ]     |tcp  |
3 |ftp             |   [21    ]     |tcp  |
4 |smtp            |   [25    ]     |tcp  |
5 |whois           |   [43    ]     |tcp  |
6 |domain          |   [53    ]     |udp  |
7 |domain          |   [53    ]     |tcp  |

Path:  Main— Admin— Network— Service Ports
```

Look at the first entry: “telnet, port 23, tcp”. This means that if the IntelliServer wants to telnet into some host, it needs to contact TCP port 23 on that host. This is a multi-page table and only one page is illustrated here. All the services are shown in Example 10-15, “Show Services Command”.

The only input field you are allowed to change is the *port number.* As with other multi-page tables, you use the tab and arrow keys to move around a single page, and the **`ctrl-F`** and **`ctrl-B`** keys to move forward and back one page at a time. *The services table may contain entries for protocols that the IntelliServer does not support.*

Show Services

You can also display the services table using the **`show services`** command shown in Example 10-15.

Example 10-15: Show Services Command

```
# show services
Service            Port  Protocol
telnet             23       tcp
ftp-data           20       tcp
ftp                21       tcp
smtp               25       tcp
whois              43       tcp
domain             53       udp
domain             53       tcp
bootp              67       udp
tftp               69       udp
finger             79       tcp
www                80       udp
www                80       tcp
nntp               119      tcp
snmp               161      udp
snmptrap           162      udp
login              513      tcp
rcp_print          514      tcp
syslog             514      udp
tcp_direct_base    9000     tcp
tcp_group_base     10000    tcp
radius             1645     udp
radacct            1646     udp
logger             8        udp
```

Most of the entries shown correspond to standard well-known ports and would not be changing. There are four exceptions to this:

1. The service ports for tcp_direct_base and tcp_group_base define the well-known port used to listen for incoming telnet connections to serial ports configured as *Reverse-TCP* or *Login-by-Port/TCP.* A connection to serial port 0 would be made to TCP port 9000; to serial port 1, through 9001; to serial port *N*, through TCP port *N+9000.* These are *not* standardized well-known ports, so it *might* be necessary to change them if they conflict with another service on your network using these numbers.

2. The service ports for *radius* and *radacct* are based on the draft standard for RADIUS authentication and accounting. These would need to be changed should the standard change.
3. The UDP service designated *logger* is completely internal to the IntelliServer. It could be anything that doesn't conflict.
4. To "turn off" the *finger* daemon, set its service port to 0.

Changes to service ports take effect when the associated process starts up. For most practical purposes this means the changes don't take effect until after the changes are saved and the IntelliServer rebooted.

Set Services, Add Services

To change a service's port number, use the **`set services`** command. To add a new service not already in the list, use the **`add services`** command. The only time you should need to add new services to the table is when you are upgrading to newer IntelliServer software that uses services not already in the default table.

Example 10-16: Add and Set Services Commands

```
set services service port number
```

```
set services tcp_direct_base port 9500
set services finger port 0
```

```
add services service udp|tcp port no.
```

```
# add services radius udp port 1645
# add services radacct udp port 1646
```

In these examples, the base TCP port for Reverse-TCP port connections was changed from 9000 to 9500 because of an imaginary conflict. *Finger* service (responding to *finger* requests from clients)was also disabled by setting its service port to 0. Also demonstrated was the **`add services`** command by adding two services that are supported in one software version but not in another.

RIP Configuration

RIP (Routing Information Protocol) is used when the IntelliServer needs to share routing information with other hosts. By *listening*, it learns routes from other hosts and by broadcasting or *sending*, it tells other hosts about the routes *it* knows.

There are three elements to RIP configuration:

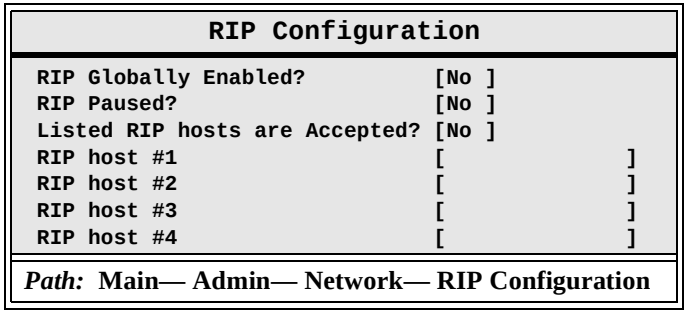
1. Each interface is separately configured to *listen* for RIP packets, to *send* (i.e. broadcast) RIP packets, to do both, or neither. This configuration is not done in a special menu but as part of the menus for configuring that interface. In other words, the RIP options for the Ethernet interface are configured as part of "IntelliServer Configuration", for example Screen 10-2 on page 202. The RIP options for particular PPP and SLIP interfaces are configured as part of their "remote profiles", which are studied in the next chapter.
2. There is a list of specific hosts from whom the IntelliServer is authorized to accept RIP information or queries. RIP packets originating from other hosts are ignored — ***or*** — there is a list of specific hosts from whom the IntelliServer is *not* authorized to accept RIP information or queries. RIP packets originating from these hosts are ignored.
3. There is a global "enable/disable" control. When disabled, the IntelliServer does not listen or send RIP packets on any interface.

The last two items (host list and global control) are described in this section.

Displaying RIP Configuration

The present RIP configuration is displayed and configured using the RIP Configuration form shown in Screen 10-10. You can also display this same information using the **`show rip`** command.

Screen 10-10: RIP Configuration Form



Modifying RIP Configuration

You can modify the RIP Configuration using the RIP Configuration form or by using commands.

Menu:	**RIP Globally Enabled? [Yes]**
Command:	**set rip enabled|disabled**

When RIP is globally enabled, the IntelliServer sends and receives RIP packets, subject to all other restrictions you have configured. For example, some interfaces may be configured to neither send nor receive RIP packets, others to do one or the other, or both. For another, the IntelliServer may be configured to only accept RIP packets that come from certain hosts.

When RIP is globally disabled, the IntelliServer does not send or receive RIP packets on any interface. The factory defaults have RIP globally disabled, but each interface can be separately enabled to listen and broadcast RIP packets. After you enable RIP globally, you then need to explicitly *disable* it for any specific interfaces you do not want handling RIP traffic.

When RIP is disabled, it frees up its memory for use by other processes. **Therefore, if RIP is disabled and you (globally) enable it, the change does not take effect until after the configuration is saved and the IntelliServer rebooted**.

Menu:	**RIP Paused? [No]**

This option is currently not implemented but is reserved for future use.

Menu:	**Listed RIP Hosts are Accepted? [Yes]**
Command:	**set rip list accept|reject**

This, together with the *RIP host* list defined below, controls which hosts are allowed to send RIP information to the IntelliServer.

When set to *yes (accept)*, the RIP host list is a list of the only hosts from whom the IntelliServer accepts RIP routing information; RIP packets from other hosts is ignored. When set to *no (reject)*, it is a list of hosts whose RIP packets should be ignored; packets from other hosts are accepted.

If there are no hosts defined in the RIP host list, then it is always taken to be a list of hosts to reject. Since there is no one on the list, packets from all hosts are accepted.

Menu:

```
RIP Host #1    [                ]
RIP Host #2    [                ]
RIP Host #3    [                ]
RIP Host #4    [                ]
```

Command:

```
add rip host ip-address
delete rip host ip-address
```

This is either a list of hosts to *accept* or to *reject*, as described earlier. If it is a list of hosts to *accept*, the IntelliServer attends to routing information received from any host on the list, but to no other. If it is a list of hosts to reject, the IntelliServer attends to routing information from other hosts, but not these.

There can be up to four hosts in the list. When you are using the configuration form, you enter host names into the available slots explicitly — a blank spot is considered available. You can change the IP address of an existing entry by modifying it in place.

When you are using the **`add rip host`** command, the IntelliServer adds this new host into an available slot. If there are no slots available, you need to **`delete`** one of the existing ones. To change an existing entry, you delete it and add its replacement.

RIP Implementation Details

The following details of our RIP implementation are useful to those already familiar with the details of RIP protocol:

The IntelliServer uses a “Split Horizon” algorithm. This means that the IntelliServer does not broadcast a route back to the interface with which the route is associated. For example, suppose a router on the IntelliServer’s local network broadcast a default route and the IntelliServer added it to its routing table. When it came time for the IntelliServer to broadcast its routes, it *could* send this route to various PPP and SLIP connections it may have, but it would not broadcast the route back to the local network. If it learned of a route to a remote network through one of his PPP connections, it might broadcast it to its local network, but it would not send it back over that PPP connection.

The IntelliServer does not replace *native* routes with routes learned from RIP. *Native* routes include all routes *not* learned through RIP, including static routes from the *Gateway* table, routes added by hand using the *add route* command, routes to interfaces added automatically at start-up and when PPP/SLIP connections come up, and routes learned from RADIUS attributes. The IntelliServer knows these routes more intimately than those it learns from RIP and so does not allow them to be replaced by a less authentic source.

The IntelliServer does not broadcast a route to any remote address for whom it is performing proxy-ARP. Such a route would be redundant and increase others’ routing tables needlessly.

In the IntelliServer’s **`show route`** command (see “Routing Table” on page 309) entries learned from RIP are marked **`RIP`** while *native* routes are not. In addition to broadcasting its routes periodically as required by RIP protocol, it also responds to RIP commands from network hosts. (The IntelliServer does not have to be *listening* on this interface, but it must be configured to *send* if it is to reply to these requests).

In response to a *Query* command for a specific route, the IntelliServer sends information about that route. In response to a *Query* for all routes, it sends information on the routes that it would have sent during one of its periodic broadcasts. In response to a *Poll* command for all routes, it sends every route in its table, regardless of “Split horizon” and other considerations.

IP Filters

An IP filter protects your network from unauthorized intrusion by restricting the types of IP packets which are allowed to travel through an interface. Each IP filter is a set of rules that designate which types of packets are allowed to pass and which are not. You assign a name to each set of rules you define and that name is used when you want to assign those rules to a particular interface.

Separate interfaces can use different rules. For example, the Ethernet interface to your local network might have no IP filter at all, but the PPP interface to your Internet provider might have an extremely restrictive filter. In that way hosts that reside on your local network can interact with each other in ways prohibited to outside hosts whose network packets would have to pass through the PPP interface.

Commands

IP filters are defined and maintained using commands. There is no menu interface available for this. The commands are outlined first, and then explained in more detail.

The **`show filter`** command lists all the filter names you have defined. You can define up to eight IP filters on a single IntelliServer. Since a single filter is commonly assigned to multiple interfaces, it is unusual to define more than one or two separate filters.

Example 10-17: List All Filter Names

```
show filter
```

```
# show filter
    name              rules
1,  firewall          12
2.  melita             3
3.  *unused*           0
4.  *unused*           0
5.  *unused*           0
6.  *unused*           0
7.  *unused*           0
8.  *unused*           0
```

Filters that have not been defined yet are marked as ***unused***. There is no indication on this display of whether a given filter has actually been assigned to some interface. If you supply a filter name, the rules for that filter are displayed as shown in Example 10-18. In this example, rule #1 prohibits any incoming packets destined for IP address 160.77.99.200. Rule #2 prohibits any incoming packets whose source address is from network 160.77.0.0/16. Rule #3 prohibits incoming packedts addressed to TCP port 23 on any host.

Example 10-18: List an IP Filter's Rules

show filter *filter-name*
 # show filter melita **filter: melita** **rule [actions | matching criteria]** **1. [deny | in dst 160.77.99.200]** **2. [deny | in src 160.77.0.0/16]** **3. [deny | in tcp port 23]**

To create a new IP filter, use the **add filter** command shown in Example 10-19. When this filter is first created, it has no rules associated with it. When creating a new filter, do not give it a name that is also an interface name: *ether, ppp00, ppp01*…If you do it will lead to confusion (see page 246).

Example 10-19: To Create A New Filter

add filter *filter-name*
 # add filter ram

Once a filter has been created, you can add rules with the **add filter** command shown in Example 10-20. This is almost the same as the command to create a new filter. *The **add filter** command creates a new filter when there are no rules listed. If there are rules, they are added to the existing filter.*

Example 10-20: To Add A Rule To An Existing Filter

add filter *filter-name rule*
 # add filter ram allow in tcp port 23 **# add filter ram allow in tcp ports 35-39** **# add filter ram deny out udp port 45**

When new rules are added to a filter, they are added to the end. If you want to insert a new rule at the beginning or between two existing rules, you can specify the position as shown in Example 10-21.

Example 10-21: To Insert A Rule Into An Existing Filter

```
add filter filter-name before rule no. rule
add filter filter-name after rule no. rule
```

```
# add filter ram before rule 1 deny in tcp port 37
```

Example 10-22: To Change An Existing Rule

```
set filter filter-name rule number rule
```

```
# set filter ram rule 1 deny in tcp port 38
```

You change an existing rule by using the **set filter** command. You can remove a filter either by setting the rule to nothing, or by deleting it: both are shown in Example 10-23. When you delete a rule from a filter, the other rules move up to close the gaps so there is no empty spot left.

Example 10-23: To Remove A Rule From A Filter

```
delete filter filter-name rule number
set filter filter-name rule number ""|blank
```

```
# delete filter ram rule 3
# set filter ram rule 3 blank
# set filter ram rule 3 ""
```

As you can see from Example 10-24, the command to delete an entire filter is almost the same as the command to remove a single rule from the filter. *If you are using the* **`delete filter`** *command to delete a rule from a filter, make sure you have typed* **`rule`** *and the rule number, otherwise you will delete the entire filter.*

Example 10-24: To Delete An Entire Filter

`delete filter` *`filter-name`*
`# delete filter ram`

Command	Rule listed?	Action
`add filter`	No	A new filter is created
`add filter`	Yes	New rule is added to the filter
`delete filter`	No	The filter is deleted
`delete filter`	Yes	Rule is removed from the filter

At this point you know how to build an IP filter by creating it and adding a set of rules. Next is explained how to create the rules.

Making the Rules

A rule consists of an *action* and a *test*. As each IP packet is filtered, the rules are applied in order. If the packet matches a rule's *test*, the *action* associated with that rule is performed. If the action calls for the matching packet to be allowed or denied, further testing stops. For this reason the order of rules is important. More specific tests should be specified before more general ones. If one rule defines an exception to a more general rule, the exception needs to be listed first. This is why you are allowed to insert rules into specific places in a filter.

Actions

There are four possible actions you can specify; we will describe them first, then explain how to construct the tests.

TIP	When you are first developing a filter, set all the actions to *log*. Such a filter will not discard anything, so it should have no effect on your network operation. You can use the command show filter interface-name to display the number of packets that have passed each test.

Table 10-4: IP Filtering Actions

Action	Description
allow	This IP packet is allowed to pass.
deny	This IP packet is discarded.
deny errors	This IP packet is discarded *and* the IntelliServer sends an ICMP error message.
log	Do not allow or deny this packet based on the results of this test, but keep a count of how many IP packets have matched. Statistics are kept for all rules. The log action allows you to keep statistics on a condition without making an allow/deny decision based on it.

Tests

Tests are constructed of many types of building blocks. A single test may contain several conditions which must all be true for the packet to match.

Table 10-5: IP Filtering Tests

Key-word	Parameters	Definition	Comments
in		The test is applied to inbound packets.	You have to specify either **in** or **out**. Specify both if you want the test to apply to all packets.
out		The test is applied to outbound packets.	
src	*ip address* *ip address/bits*	The IP packet's source address needs to match the address in this rule.	To match IP packet addresses from a particular network or subnet, specify the number of bits to be tested after the IP address. For example, to match any IP address from the class B network 160.77.0.0, specify it as 160.77.0.0/16. This is the same notation used for specifying sub-nets in routes and is explained in chapter 9, *Network Basics*.
dst	*ip address* *ip address/bits*	The IP packet's destination address needs to match the address in this rule.	
tcp		This must be a TCP packet.	If a service port or range of ports is specified in this rule, UDP or TCP ports in that range are matched. If no service ports are specified, all UDP or TCP ports are included.
udp		This must be a UDP packet.	
icmp		This must be an ICMP packet, for example, "ping".	

Table 10-5: IP Filtering Tests (Continued)

Key-word	Parameters	Definition	Comments
syn		This matches any TCP packet that has the SYN flag set. This flag is always set in the first packet sent over a TCP connection, so this test could be included in a rule to prevent certain new TCP connections from being started up.	
port	*port number*	The destination port in the IP header must match this one.	Since we are talking about network headers, *port* refers to the TCP or UDP service port associated with a connection; it has nothing to do with serial ports. When you use the **port** keyword, you must also specify either TCP or UDP, since the same port numbers could apply to either.
ports	*range* *(e.g. 1-35)*	The destination port in the IP header must match this range.	
ports reserved		The destination port in the IP header must be one of the well known reserved ports 1-1023.	

Sample Rules

Here are some sample rules. See if you can figure out what they are supposed to do.

```
allow in dst 160.77.99.30 tcp port 21
```

This rule allows all incoming packets destined for port 21 (used for FTP connections) of host 160.77.99.30. Specifically, this allows an outsider to establish an FTP connection to one particular host.

```
deny in tcp reserved
```

This rule denies all incoming packets addressed to any well-known port of any host. This rule prevents an outsider from starting a TCP connection to any of the standard network services, except for ones specifically permitted by earlier rules in the filter.

deny in src 160.77.128.0/17
This rule forbids any incoming packets from host addresses in the range 160.77.128.1 - 160.77.255.254.

Attaching a Filter to an Interface

For the IP filter to be used, it must be assigned to an interface. When you configure each interface you can specify an IP filter to be automatically attached when the interface is configured, or you can attach and detach them manually.

Configuring an interface to attach an IP filter automatically is covered in the documentation for each interface. To configure your Ethernet interface, see "IP Filter" on page 209. Configuring a PPP or SLIP interface is done through "Remote Profile" configuration, see "IP Filters" on page 239. These changes to the IP filter assignments do not take effect until the next time the interface is brought up.

To manually attach or detach a filter from an interface, use the following commands:

Example 10-25: Attaching A Filter To An Interface

attach filter *interface-name filter-name*
attach filter ether melita # attach filter ppp00 firewall

Example 10-26: Detaching A Filter From An Interface

detach filter *interface-name*
detach filter ether

Displaying Filter Statistics

IP filtering keeps track of the number of packets that match each rule. You can view this information by using the show filter command, but instead of the filter name, specify the interface name. This is why you should not give filters the same name as interfaces.

Comparing Example 10-27 with Example 10-18 on page 240, the only difference is that the matches are reported.

It is not meaningful to report the number of matches when only the filter is specified because the filter may be attached to multiple interfaces. The match counts are kept separately for each interface.

Example 10-27: List Interface's Filtering Statistics

show filter *interface-name*

```
# show filter ppp01
interface: ppp01, filter: melita
rule   matches [actions | matching criteria]
 1.       1022 [ deny | in dst 160.77.99.200 ]
 2.       3424 [ deny | in src 160.77.0.0/16 ]
 3.          3 [ deny | in tcp port 23 ]
```

Example 10-28: Creating A New IP Filter

```
# add filter gnu
# add filter gnu allow in dest 160.77.99.27 tcp port 21
# add filter gnu allow in dest 160.77.99.23 tcp port 20
# add filter gnu allow in dest 160.77.99.45 udp port 53
# add filter gnu deny in tcp ports reserved
# add filter gnu deny in udp ports reserved
# set filter gnu rule 1 allow in dest 160.77.99.23 tcp port 21
# add filter gnu before rule 1 allow dest 160.77.99.23 tcp port 119
# show filter gnu
filter: gmu
rule  [actions | matching criteria]
1.    [ allow  | in dest 160.77.99.23 tcp port 119 (nntp) ]
2.    [ allow  | in dest 160.77.99.23 tcp port 21 (ftp) ]
3.    [ allow  | in dest 160.77.99.23 tco port 20 (ftpdata) ]
4.    [ allow  | in dest 160.77.99.45 udp port 53 (domain) ]
5.    [ deny   | in tcp reserved ports (1-1023) ]
6.    [ deny   | in udp reserved ports (1-1023) ]
# attach filter ppp01 gnu
#
```

In this final example, a new filter was created and some rules added. Then, one of the rules was corrected and a new rule added at the beginning. Then the result displayed. Notice that the service names associated with TCP and UDP ports are also displayed even though not entered. The IntelliServer gets these names from the *services table* ("Service Ports" on page 230).

CHAPTER 11

Remote Network Configuration

In this chapter you learn how to configure the IntelliServer to support PPP, SLIP, and CSLIP links to remote hosts and networks.

You will learn:

- How Remote Network Configuration requires maintaining dialer and login scripts, remote profiles, and options profiles.
- The difference between inbound and outbound connections.
- How to use menus and commands to display and modify the remote network configuration.
- How user information from RADIUS affects how a PPP/SLIP interface is configured.

Remote Network Configuration — Overview

In chapter 9, *Network Basics*, you learned that an *interface* is the part of a host that links it to a network. Interfaces are assigned names for administrative purposes. On the IntelliServer, there is one Ethernet interface, named *sonic*. There are up to 32 remote interfaces, named *ppp00…ppp31*.

Each remote interface corresponds to a single PPP, SLIP, or CSLIP link. A particular remote interface may be linked to a certain serial port, or it may be assigned to different serial ports at different times.

When an interface is established, it draws on information from a number of places:

- *Remote Profiles* contain basic information about the interface: whether it is inbound (i.e., dial-up) or outbound, whether it is dedicated to a particular serial port or user, which protocols may be used, the IP address of the remote site, and so on.
- *PPP Option Profiles* contain additional protocol options used in bringing up PPP and SLIP links.
- *Dial Scripts* are used by outbound (dial-out) links to specify what commands need to be sent to an attached modem to make it dial up the remote site.
- *Login Scripts* are used by outbound links to specify how to log into the remote site, in order that the remote site will bring up its end of the link.
- *RADIUS information*: On inbound links, when the user dialing in as a RADIUS user configured for PPP, SLIP, or CSLIP service, the RADIUS user data base can supply additional information that is used in configuring the interface.

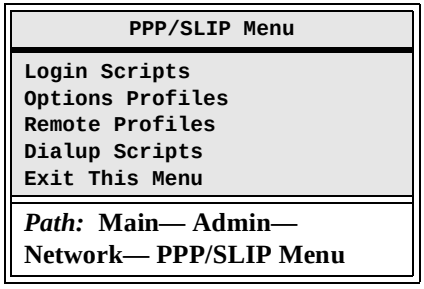
Discussed in this chapter is: how to configure remote profiles, PPP option profiles, dial scripts, and login scripts. Also discussed are the relationships between these configurations and the information that is sent from RADIUS, for the benefit of quick reference. A complete explanation awaits chapter 17, *User Authentication using RADIUS*.

PPP/SLIP Menu

The configuration forms for supporting PPP, SLIP, and CSLIP links are reached through the PPP/SLIP Menu shown here in Screen 11-1.

Each selection on this menu represents a different set of *Multi-Record Forms* (see page 50). For example, a single Remote Profile contains a screen full of information. There can be several Remote Profiles: you assign each a name that can later be used to refer to it. The same is true of the Login Scripts, and of Option Profiles, and of Dialup Scripts. Consequently, they are all maintained in the same manner.

Screen 11-1: PPP/SLIP Menu



Pick any of the selections on the PPP/SLIP Menu, and you get a menu similar to the one shown for Remote Profiles in Screen 11-3. Whether you have chosen Login Scripts, Options Profiles, Remote Profiles, or Dialup Scripts, the process is identical: the next menu gives you the option of creating new ones, or listing, modifying, and deleting existing ones.

Screen 11-2: List Remote Profiles

```
                              Remotes

Remote Name  Remote Address  Local Address   Iface Proto Port
yves         160.77.99.124   160.77.99.224   ppp00 slip  Any
forpppp      160.77.99.125   160.77.99.225   ppp01 any   Any

Path: Main— Admin— Network— PPP/SLIP— Remote— List
```

When you choose to **List** most things, only a list of names is displayed. When you list Remote Profiles, additional summary information is displayed along with each name.

Screen 11-3: Remote Profile Menu

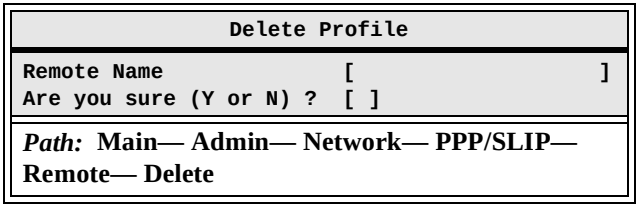
```
          Remote Profile Menu

  List Profiles
  Create Profile
  Modify Profile
  Delete Profile
  Exit This Menu

Path: Main— Admin— Network—
PPP/SLIP— Remote Profiles
```

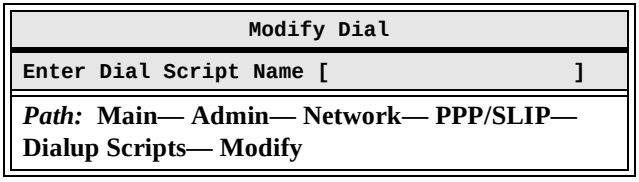
When you choose to **Delete** something, you will be prompted to enter its name, and then to confirm, as Screen 11-4 is an example.

Screen 11-4: Delete (Remote) Profiles



When you choose to **Modify** something, you are prompted to enter its name, as illustrated by Screen 11-5 (this time using the Dial Scripts as an example).

Screen 11-5: Modify Dial (prompts for name)



Perhaps you do not recall the names of any of the things that already exist? Then enter **ctrl-u**. This brings up a screen showing all the existing ones (option profiles, dial scripts, remote profiles…). You can use the arrow keys to highlight the one you want and then press **enter** to select it. After you enter the name of the thing you want to modify, the appropriate configuration form is displayed with current settings shown.

When you choose to **Create** something, the appropriate configuration form is displayed with the name left blank. You fill it in when you are filling in the other information. Several of the input areas may contain default settings.

Table 11-1 shows the maximum number of Remote Profiles, Dialup Scripts, etc., that can be configured. If you attempt to create more, the IntelliServer complains with an error message like “Create Limit Reached”.

Table 11-1: PPP/SLIP Configuration Limits

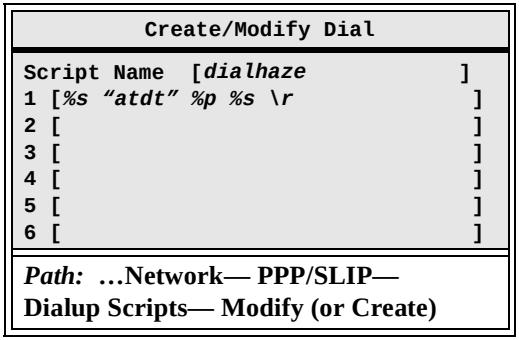
Type	Maximum Number
Remote Profiles	32
Dialup Scripts	8
Login Scripts	32
Option Profiles	**8**

There are enough login scripts to assign one per interface, if need be. There are fewer Dialup Scripts and Option Profiles, because they are more re-usable.

Dial Scripts

The Dial Script Configuration Form is shown here in Screen 11-6. Dial scripts define what needs to be sent to a modem so that it dials out and connects to another modem.

Screen 11-6: Dial Script Configuration Form



The contents of your dial script depends somewhat on the modem. Since a given serial port is attached to a given modem, dial scripts are associated with serial ports. The **`Script Name`** *(dialhaze* in this example) is the unique name of this Dial Script. During serial port configuration, you assign this to the port's *Dial Script* as described on page 81.

The remainder of the form consists of six lines of forty-two columns each, but there is nothing special about the arrangement into rows and columns. Before the script is run, trailing blanks are removed from each line and all the lines are run together. (See "Configuring the Preamble and Message of the Day" on page 154 for a more detailed description of this approach).

Compared to the Preamble or Message of the Day, a Dial Script is more complex. Not only must the script send out strings of data, it sometimes needs to wait until certain responses are *received* before continuing. Therefore a Dial Script is build not from simple data strings, but from *script commands*. There may be several commands on a line, but a command must not be split across lines.

The following table shows how script commands are constructed:

Table 11-2: Dial and Login Script Commands

Command Definition / Examples	Description
%s *string* **%s "ATDT5551212\r"** **%s "hello there"** **%s hellothere**	Transmit the string to the serial port. If the string contains any spaces, enclose the entire string in quotes. Control characters can be represented using the codes in Table 5-4 on page 94; in this example, **\r** represents a carriage-return.
%w *time string* **%w 10 connect\r** **%w 5 "carrier"** **%w** *time* **%w 10** **%w** *string* **%w carrier\r** **%w "10"** **%w "1derful"**	Wait until the specified string is received from the serial port, or the time (in seconds) elapse, whichever comes first. You may omit *either* the time or the string. • If the time is omitted the script will wait forever for the string. • If the string is omitted the script will wait the specified time unconditionally. • If a time and a string are *both* given, getting the string first is considered good. Timing out first is considered bad. **When an interface is using a dial or login script to bring up a connection, if a *wait* command times out before the string is received, the connection attempt will be stopped and the line disconnected.** If there is only one thing after the **%w** how does the script know whether it is supposed to be the time or the string? If it is a number, it is assumed to represent a time. Otherwise it is a string. If you want to wait forever for a certain string *and the string is a number*, then enclose it in quotes so it won't be mistaken for a time. Control characters are represented in these strings the same as for the **%s** command.
%p	Send the phone number stored in the associated Remote Profile. This command allows the same dial script to support several outbound connections with different phone numbers. Otherwise, separate dial scripts would have been needed.

Shell Commands for Dial Scripts

Dial scripts can also be configured from the command line using the **`add dial`** command to create new dial scripts, the **`set dial`** command to modify existing dial scripts, and the **`delete dial`** command to remove a dial script.

Example 11-1: Add Dial Command

```
add dial script-name line line# text
# add dial daisy line 1 “%s ATDT18005551213\\r”
```

Example 11-2: Set Dial Command

```
set dial script-name line line# text
# set dial daisy line 1 “%s ATDT18005551212\\r”
# set dial daisy line 2 “%w 10 CONNECT\\r %w 5”
# set dial daisy line 3 “%s \”hello world\\r\””
```

Example 11-3: Delete Dial Command

```
delete dial script-name
# delete dial daisy
```

To display the current contents of the a dial script, use the **show dial** command.

Example 11-4: Show Dial Command

show dial *script-name*|**all**

```
# show dial daisy

  Script  Name  daisy

  Line 1 [%s ATDT18005551212\r   ]
  Line 2 [%w 10 CONNECT\r %w 5   ]
  Line 3 [%s "hello world\r"     ]
(…)
  Line 6 [                       ]
#
```

To display the contents of all dial scripts, use **show dial all**. If you have several dial scripts defined, you may want to type **|** at the end of the command to paginate the output.

In Example 11-1 and 11-2 you see typed **"…123\\r"** instead of **"…123\r"**. In another example, you see typed **\" hello world\"** when what was wanted to appear was **"hello world"**. This was done because backslashes and double-quotes are special to the command shell. If you want one of these to *appear*, you must precede it with a backslash.

Login Scripts

The Login Script Configuration Form is shown here in Screen 11-7.

Screen 11-7: Login Script Configuration Form

```
                Create/Modify Login

Script Name  [lincoln       ]
1 [%w 30 "gin:"                        ]
2 [%s "abraham\r"                      ]
3 [%w 15 "word:"                       ]
4 [%s "0pnsesme\r"                     ]
5 [                                    ]
6 [                                    ]

Path:  …Network— PPP/SLIP—
Login Scripts— Modify (or Create)
```

When the IntelliServer starts to bring up an outbound PPP or SLIP connection, it first uses the dialer script to make the modem dial the remote site. At the remote site, a modem answers the call and the host computer may be configured to issue a login/password prompt. Then, it bring up its side of the link (or hangs up) based on what user name and password are provided.

Login scripts are run immediately after the dial scripts, allowing the IntelliServer to provide automatically the necessary responses to a remote site's login, password, or other prompts.

The nature of your login script is determined by the remote site you are contacting. Therefore, a login script is associated with a *remote profile*, not with a serial port (as are dial scripts).

The **`Script Name`** *(lincoln* in this example) is the unique name of this Login Script. During Remote Profile configuration, you can assign this to the profile's *Login Script* as described on page 286.

The rest of the form contains the body of the script. Like Dial Scripts, it is composed of commands. The rules for forming these commands are the same as for Dial scripts, found in Table 11-2 on page 256.

In the example shown in Screen 11-7, the IntelliServer waits up to thirty seconds for data matching **`gin:`** to come in. Presumably, this indicates the remote host has prompted us for our login name. Then the IntelliServer sends **`abraham`**, our login name. Then, it waits for the password prompt, as indicated by the fragment **`word:`**. Finally the IntelliServer sends our password, **`0pnsesme`**. The login script is finished, and the IntelliServer brings up our side of the link and so does the remote site.

Shell Commands for Login Scripts

Login scripts can also be configured from the command line using the **`add login`** command to create new login scripts, **`set login`** command to modify existing login scripts, and **`delete login`** command to remove a login script.

Example 11-5: Add Login Command

`add login` *`script-name`* **`line`** *`line# text`*
`# add login cabin line 1 “%s snow\\r:”`

Example 11-6: Set Login Command

`set login` *`script-name`* **`line`** *`line# text`*
`# set login cabin line 2 “%w 5 word:”` **`# set login cabin line 3 “%w 2 %s \\r”`** **`# set login cabin line 4 “%w 5 Reserved”`**

Example 11-7: Delete Login Command

`delete login` *`script-name`*
`# delete login cabin`

To display the current contents of a login script, use the **show login** command.

Example 11-8: Show Login Command

show login *script-name*|**all**

```
# show login cabin

Script   Name  cabin

Line 1  [%s snow\r                ]
Line 2  [%w 5 word:               ]
Line 3  [%w 2 %s \r               ]
Line 4  [%w 5 Reserved            ]
Line 5  [                         ]
Line 6  [                         ]
                #
```

To display the contents of all login scripts, use the **show login all** command. When there are several login commands defined, you may need to type **|** at the end of the command to paginate the output.

Note in Example 11-6 “\\r” instead of “\r” is typed. This is because the backslash is special to the command shell, and must be preceded by another one when you mean to *use* one. See Example 11-2 on page 257 for more examples.

Options Profiles

Options Profiles (sometimes called SLIP/PPP options) are used for storing configuration parameters that do not change very often. These parameters are also likely to be shared by a number of interfaces at a given site.

An Options Profile is created with a particular collection of settings and it is given a name. To assign these settings to a particular interface, you enter the name in that interface's Remote Profile. This reduces the number of separate parameters that an individual Remote Profile must contain.

Screen 11-8: PPP/SLIP Options Configuration Form

```
        Create/Modify PPP/SLIP Options

Profile Name                    [default      ]
Use Passive Mode                [No ]
Address/Control Compression     [Yes]
Protocol Field Compression      [Yes]
Address Negotiation Mode        [Enabled]
ASYNC Map Negotiation           [Yes]
Magic Number Negotiation        [Yes]
Maximum Receive Negotiation     [Yes]
Maximum Receive Size            [1024]
Van Jacobsen Compression Mode   [Disabled]
Enable Proxy ARP                [Yes]
Bring Up Slip Link Immediately  [No ]
Prompt Slip Login For Address   [No ]
```

Path: **…Network— PPP/SLIP— Options Profiles— Modify (or Create)**

At factory default there is a single Options profile defined, called **`default`**. When new Remote Profiles are created, this default Options profile is assigned to them where it remains until you change it.

Screen 11-8 shows the PPP/SLIP Options Configuration Form, as it would appear for the default profile.

You can also display the contents of one or all options profiles with the **show pppoption** command (Example 11-9). When you specify **all**, the complete information for all profiles is displayed, so you may want to paginate the output by typing a **|** at the end of your command. The information that is displayed is exactly the same as shown in Screen 11-8, so this need not be illustrated separately.

Example 11-9: Show Pppoption Command

```
show pppoption option-name
show pppoption all
```

```
# show pppoption all
```

Configuring Option Profiles

Option Profiles can be configured through the configuration form (Screen 11-8) or by using the **pppoption** command.

Creating New Option Profiles

How to create new Option Profiles was discussed on page 253. The process is the same as for Dial Scripts, Login Scripts, and Remote Profiles. From the command line, create a new Option Profile by using the add pppoption command shown in Example 11-10. As shown in the example, you can define one or more parameters in the same command. This is true when adding a new Option Profile as well as when setting the parameters of an existing one.

Example 11-10: Add, Set pppoption Commands

```
add pppoption option-name {parameter value}…
set pppoption option-name {parameter value}…
```

```
# add pppoption def1 accompress yes prompt no
# set pppoption def1 protocomp no
```

Option Profile Parameters

In this section you will examine each setting in detail to discover how it affects the operation of the interface. Shown for each item is its line in the configuration form as well as how to configure it using the **set pppoption** command.

Some parameters only apply to SLIP connections, others only to PPP connections; some apply to inbound connections only, and some to outbound. This is indicated in the appropriate section.

When a PPP connection is brought up, one site will have initiated the connection by dialing up and logging into the other. This is the *outbound* side of the connection. When a PPP link first comes up, *negotiation* messages are exchanged between the two sides of the link. This enables the two sides to exchange information about their respective configurations and reconcile any differences.

Normally, the *outbound* side of a connection is responsible for sending the first negotiation message. The *inbound* side waits passively to receive it. Inbound connections on the IntelliServer always wait for the other side to initiate the negotiation.

Passive Mode only affects an outbound PPP connection on the IntelliServer. Specify **No** (the default) if it should initiate the PPP negotiations, or **Yes** if it should passively wait for the other side to do so.This option has no effect on SLIP or CSLIP connections because these protocols do not involve negotiation.

Menu:	**Use Passive Mode [*No*]**
Command:	**set pppoption** *option-name* **passive yes|no**

The *Address/Control Compression* controls the local compression of address and control fields in the PPP header. Specify **Yes** (the default) to compress these fields, or **No** to leave them uncompressed.

Menu:	**Address/Control Compression [*Yes*]**
Command:	**set pppoption** *option-name* **accompress yes|no**

The *Protocol Field Compression* controls the local compression of the protocol field in the PPP header. Specify **Yes** (the default) to compress it, or **No** to leave it uncompressed.

Menu:	**Protocol Field Compression [*Yes*]**
Command:	**set pppoption** *option-name* **protocomp yes|no**

Address Negotiation Mode applies to PPP connections only and controls whether the IntelliServer performs address negotiation. Specify **Disabled** to turn off address negotiation, or **Enabled** (the default) to use the negotiation process defined in RFC1332.

Menu:	**Address Negotiation Mode [*Enabled*]**
Command:	**set pppoption** *option-name* **addrmode enabled|disabled**

Enabling Address Negotiation on inbound PPP connections allows the IntelliServer to learn the caller's IP address and inform the caller of our IP address through the PPP negotiation process. While address negotiation can be enabled for outbound connections as well, the IntelliServer needs to know the remote site's correct IP address ahead of time because it is the attempt to *access* that address which causes the IntelliServer to bring up the interface.

The IntelliServer will not alter its *own* interface's IP address as a result of PPP address negotiation, either for inbound or outbound connections.

The ASYNC Map is used by PPP to prevent certain control characters (such as XON and XOFF) from occurring in the data stream. The map indicates which characters are proscribed. Specify **Yes** (the default) to allow the IntelliServer to negotiate this map with the remote system. Specify **No** to force the IntelliServer to use the ASYNC Map specified in the Remote Profile.

Menu:	**ASYNC Map Negotiation [*Yes*]**
Command:	**set pppoption** *option-name* **async yes|no**

The *magic number* is a arbitrary 32-bit number which is randomly chosen by each side of a PPP link. During negotiation, each side sends the other its *magic number*. It would be unusual for two different hosts to randomly choose the same random number, so if the magic number we receive is the same as our own, it is assumed that something has gone wrong (perhaps the modem is running in loop-back mode) and the IntelliServer must be talking to itself. Since this is a bad thing, the IntelliServer drops the connection (hang up the modem, etc.). Choose **Yes** (the default) if you want the IntelliServer to check the magic numbers, or **No** if you want to ignore them.

Menu:	**Magic Number Negotiation [*Yes*]**
Command:	**set pppoption** *option-name* **magic yes|no**

The MRU (*Maximum Receive Unit*, *Maximum Receive Size*, or *size*) represents the maximum number of bytes the IntelliServer can receive in a single PPP packet. This is a partner to the MTU, or *Maximum Transmit Unit*, which is configured in the Remote Profile and defines the largest packet the IntelliServer can *send*.

Menu:	**Maximum Receive Negotiation [*Yes*]**
Command:	**set pppoption** *option-name* **mru yes|no**

Menu:	**Maximum Receive Size [*1024*]**
Command:	**set pppoption** *option-name* **size** *size*

Each side of the link has an MRU, usually constrained by internal buffer sizes and an MTU. The first step to harmony is making sure that one side's MTU is not greater than the other side's MRU. With PPP, this is done through *Maximum Receive Negotiation.* If *Maximum Receive Negotiation* (or *mru*) is **Yes**, (and assuming the remote side of the link is so configured) each side informs the other of its own MRU. If the recipient's MTU is larger, it temporarily reduces it accordingly.

For SLIP and CSLIP connections, the effective MRU is always 1536 bytes. A large value is chosen because there is no mechanism, other than mutual agreement at configuration time, to agree on a smaller value.

Van Jacobsen (VJ) Compression is a method of compressing TCP/IP headers in PPP or SLIP packets. With SLIP protocol, both sides must agree beforehand whether to use it. (SLIP with VJ Compression is called CSLIP). With PPP, the two sides can negotiate whether to use VJ Compression.

Menu:	**Van Jacobsen Compression Mode [*Disabled*]**
Command:	**set pppoption** *option-name* **vjmode enabled|disabled**

On the IntelliServer, connections are designated PPP, SLIP, or CSLIP. Since CSLIP always uses VJ Compression, and SLIP never does, this option affects only PPP links. Set it to **Disabled** (the default) if you do not want to use VJ Compression or to **Enabled** if you do. VJ Compression is defined in RFC1144.

If *Enable Proxy ARP* option is set to **Yes** (the default), the IntelliServer responds to ARP requests for the remote IP address on this interface, as long as the link is up. For example, suppose the IntelliServer's IP address (on the local Ethernet network) was 160.77.99.30. Suppose the remote IP address (the host at the other end of this PPP link) was 160.77.99.17. If a host on the IntelliServer's local network wanted to access this remote host it would think from the IP address that it is on the local network. So, it would perform an ARP request to learn the Ethernet Address. The IntelliServer replies giving its *own* Ethernet address and enabling it to receive packets destined for that host. This is explained more fully under "Proxy ARP" on page 189. If the option is set to **No**, Proxy ARP is not performed.

Menu:	**Enable Proxy ARP [*Yes*]**
Command:	**set pppoption** *option-name* **proxy yes|no**

The *Bring Up Slip Link Immediately* option applies to outbound SLIP and CSLIP connections. By default this option is set to **No** and the IntelliServer attempts to bring up the outbound link when it is first required to route network traffic to the IP address at the other end. If you choose **Yes**, then the IntelliServer attempts to bring up the line immediately on start-up. Furthermore, if the link goes down (because of a modem disconnect, for example) the IntelliServer attempts to bring it up immediately.

Menu:	**Bring Up Slip Link Immediately [*No*]**
Command:	**set pppoption** *option-name* **bringup yes|no**

Suppose you were a business with a branch office at a remote site. You *could* configure one IntelliServer with a outbound interface and dialer and login scripts, and configure another with an inbound interface. The outbound site would log into the inbound site and finally the link would come up.

Can we simplify? Let's start with a leased line. No dial script. What about logging in? Instead of configuring the remote site with an inbound interface, let's try configuring *both* sides as outbound, with the *Bring Up Slip Link Immediately* option set to **Yes**. Leased line? Neither side dials. Both sides outbound? No login prompt issued. No login script needed. No PPP? No negotiation. No waiting for network traffic to bring up the link!

What makes this work is the fact that SLIP and CSLIP do not have a negotiation phase. Each side is always prepared to be sending and receiving network traffic. With the *Bring Up Slip Link Immediately* option set, the ports are droning SLIP and CSLIP from the moment of birth.

Menu:	**Prompt SLIP Login for Address [*No*]**
Command:	**set pppoption** *option-name* **prompt yes|no**

This applies to inbound SLIP and CSLIP connections. When set to **Yes**, the IntelliServer prompts the user to enter his IP address. After the address is entered, the link is brought up using that IP address as the "Remote Address". This facilitates multiple sites being able to use a single interface at different times. This option does not apply to PPP connections which are able to use PPP address negotiation for this purpose. This option is also not required when remote dial-in users are configured on a RADIUS server because each user's IP address can be stored in the RADIUS server's user database (see chapter 17, *User Authentication using RADIUS*).

Remote Profiles — Concepts

Each PPP, SLIP, or CSLIP connection is associated with an *interface*. The interface is a body of software responsible for:

- Storing all options (configured and negotiated) about this connection. This includes information stored in the Options Profile as well as the results of PPP negotiations.
- Preparing data packets to be sent over the serial line and performing various types of compression, creating headers, and so on.
- Preparing data packets which have been received from the serial line and stripping headers, uncompressing data, and so on.
- Keeping track of the connection status (Is the link down? Up? Are we dialing? Negotiating?).
- Keeping statistics on behalf of this connection.

Each Remote Profile is associated with a unique interface. The IntelliServer supports a maximum of 32 interfaces (not counting the Ethernet interface) so you can define a maximum of 32 Remote Profiles. Regardless of the number of ports on your IntelliServer, you cannot run more than 32 PPP, SLIP, or CSLIP links at one time.

Each Remote Profile stores information necessary to bring up its interface. Since each Remote Profile is associated with an interface, the two are often used interchangeably. For some commands you can enter either a Remote Profile name or its interface name; there is a subtle distinction. The Remote Profile is a place to store configurations while the interface is the dynamic entity that is managing a connection. Remote Profiles have names that you can assign; interfaces have predefined names such as *ppp00, ppp01…ppp31*.

Remote Profiles (and therefore interfaces) can float. This means that an interface does not need to be tied to any particular user, or anchored at any particular remote IP address, or docked at a particular serial port. An interface *can* be tied to any or all of those things, but it does not have to be.

Inbound vs. Outbound Profiles

Each Remote Profile, with its accompanying interface, is configured to be either *Inbound*, *Outbound*, or *Disabled:*

- A *Disabled* interface is not available for bringing up a connection.
- An *Outbound* interface is designed to initiate a PPP, SLIP, or CSLIP connection in response to network demand (or to do so automatically at start-up, when so configured).
- An *Inbound* interface is designed to wait for a remote site to initiate a connection. When the connection is established, the IntelliServer extends its knowledge of the network to include this new piece. When this connection is dropped and a different connection is made, the IntelliServer's network knowledge changes again.
- Outbound interfaces are designed to start with network requirements and turn them into serial connections.
- Inbound interfaces are designed to start with a serial connection and turn it into a network entity.
- Remote Profiles for Inbound and Outbound interfaces have different requirements.

Outbound Interfaces in Detail

An outbound interface creates a serial connection to a remote network because the IntelliServer has network traffic that wants to reach this network. This implies that the IntelliServer must already know something about that remote network. If there is a single remote host, the IntelliServer must know its IP address (at least, the IP address it uses on the serial link itself). The IP address of this remote host is the *Remote Address* in the Remote Profile.

If there is a network behind the remote host, the IntelliServer should be configured with a suitable network route through the remote host *before* the connection is established. Otherwise, traffic to arbitrary hosts on that network would not be able to bring up the link. These are generally configured as static routes in the IntelliServer's *Gateway Table* (see Screen 10-8 on page 227).

At start-up, the IntelliServer loads static routes from the *Gateway Table* as well as routes to the *Remote Addresses* in the Remote Profiles of outbound interfaces. When the appropriate network traffic appears, it can now be routed to the interface.

Now the interface has some packets to send, but there is no physical connection yet. What must it do?

1. It has to know which serial port (or from which group of serial ports) to choose. This is the *Serial Port* or *Group* in the Remote Profile.
2. It must know what, if anything, must be done to dial up and log into the remote site. These are the *Dial Script* (assigned to the port, which you just chose), a phone number (from the Remote Profile), and *Login Script* (assigned to our Remote Profile).

Once the physical connection has been established and the outbound interface has initiated any PPP negotiation, the operation of an outbound interface is the same as for an inbound one.

Inbound Interfaces In Detail

This discussion begins not with an interface, but with a serial port. The serial port is configured as *Login-by-Port*, and someone else's outbound interface dials in and logs in. When it logs in it provides a user name. This is the key. The user name is the thing that distinguishes one caller from another.

The IntelliServer must now somehow translate the user name into information it can use to bring up the connection and add the remote host and its network to your own. This may be done in two ways:

1. If the user is configured in the IntelliServer's NVRAM, network information is obtained from a Remote Profile. Remote Profiles can be assigned to specific users or ports for this purpose, or can be assigned as available to any user.
2. If the user is configured on a network host using RADIUS, network information specific to this user can be stored. Nothing specific to the remote network needs to be determined from the Remote Profiles (however it is also possible for RADIUS users to be assigned temporary IP addresses drawn from the pool of *Remote Addresses* in the Remote Profiles.)

It is also possible to configure the interface so that the remote network information is determined by the port, rather than the user (simply by assigning the Remote Profile with this information to a specific port). This is uncommon where there are multiple lines, because these are often arranged in rotaries where a single incoming phone number accesses any of a number of modems.

Once the connection is up, the IntelliServer's routing table (and possibly ARP table) is expanded with information about the new network and host. This information is retained until the connection is dropped, then it is purged. This is the basic idea. There are more details to explain, but not until after "Remote Profile Configuration" is discussed.

Remote Profile — Configuration Form

Remote Profiles can be configured using the Remote Profile Configuration Form shown in Screen 11-9.

Screen 11-9: Remote Profile Configuration Form

```
                          Configure Remote

Remote Name    [                              ]
Remote Address [0.0.0.0        ]

Interface Address [               ]   Interface Name |ppp02|
Interface Netmask [0.0.0.0        ]   Interface Type [Disabled  ]

Serial Port [Any ]   Group [None]   MTU [0   ]   Async Map [000a0000]
Delay Between Redials [0  ]   Inactivity Timeout [0   ]  RIP [both  ]

Dial-In User [          ]      Phone Number    [                    ]
Login Script [            ]    Options Profile [default             ]
Protocol     [ANY     ]        IP Filter       [                    ]

Authentication Protocol  [None]
CHAP Name/PAP ID         [                    ]
CHAP Secret/PAP Password [                                         ]
```

Path: **…Administration --Network— PPP/SLIP— Remote Profiles— Modify (or Create)**

You can also display the configuration of one remote or all remotes using one of the forms of the **show remote** command shown in Example 11-11. In the first two cases, all the configuration information shown in the configuration screen is displayed, so there is no point in showing that again. If you specify **all** remotes, the output could be lengthy so be sure to paginate using **|**.

Example 11-11: Show Remote Command

```
show remote remote-name
show remote all
show remote summmary
```

```
# show remote frobisher
```

The **show remote summary** command is a little more interesting because just enough information is displayed to help you keep track of what is what.

Example 11-12: Show Remote Summary Command

```
# show remote summary
```

local iface	local address	remote name	remote address	proto user	port script	group options	in/out filter
ppp00	160.77.99.111	sevy	160.77.99.211	SLIP	--	--	in
				ford	--	default	--
ppp01	160.77.99.112	vyse	160.77.99.212	ANY	--	--	in
				grip	--	default	--

There are also commands to add, modify, and delete remote profiles:

Example 11-13: Add Remote Command

```
add remote remote-name {parameter value}
# add remote burgess ifaddr 160.77.99.88
```

Example 11-14: Set Remote Command

```
set remote remote-name {parameter value}
set remote interface-name {parameter value}
# set remote burgess address 160.77.99.86
# set remote ppp04 type inbound option default
```

Example 11-15: Delete Remote Command

```
delete remote remote-name
# delete remote burgess
```

When a remote is added in Example 11-13, it defines an interface address at the same time. This is necessary because each Remote Profile is required to have a valid interface address, and there is no default value. In Example 11-14, more than one pair of parameter and value is entered, and this is alright as well.

Configuring Remote Profiles

In this section, each of the elements in the Remote Profilewe is examined.

Menu:	**Remote Name [*saskatoon*]**
Command:	**add remote** *profile-name* **ifaddr** *ip-address*

You assign the *Remote Name* when the profile is created and it stays with the profile as long as it exists and cannot be explicitly changed except by deleting one remote and adding another. In addition to identifying the profile for administrative purposes, the *Remote Name* has one other important significance. For inbound interfaces, the *Remote Name* is displayed as part of a banner message after the user logs in correctly.

This banner message will sometimes be used as part of a client's login script in order to extract IP addresses and other information. Here is an example: of such a banner message.

```
Computone IntelliServer jeeves - SLIP login

jeeves 160.77.99.33 00_Your_Address: 160.77.99.44
```

- The message **`Computone IntelliServer`** always appears.
- Next is this IntelliServer's host name, **`jeeves`** (see page 204).
- Next is a hyphen, followed by the message **`PPP login`**, **`SLIP login`** or **`CSLIP login`**.
- The next line starts with the IntelliServer's host name again.
- Next is the IntelliServer's IP address (for the interface being used).
- Next is the *Remote Name*, **`00_Your_Address:`** (The colon is part of the name). More about this later…
- Finally, there is *its* IP address (The *Remote Address* from the Remote Profile, or as provided from RADIUS).

If the dial-in user is bringing up the link manually, it might need to read our banner, find its IP address, and enter it somewhere. So the words **Your_Address** are included in the remote name for its benefit. Contrariwise, the dial-in user might be using the login script from some commercial Internet access package to automatically bring up the link. Often, these scripts can be configured to automatically detect the remote address it is being assigned, and configure its system as appropriate. How do these scripts work? You tell them something like, "Look for a colon. The first one you see, the next thing after it will be the IP address you have to use". Something like that. So the remote name (*all the remote names configured at this site), end* with a colon, or any other special character that won't appear elsewhere in the banner. Finally, why the **00_** at the beginning? Because each remote name must be unique. Assuming you have configured several, they would be all the same except for **00_**, **01_**, **02_**…

Do not choose a *Remote Name* that is one of the interface names, i.e., *ppp00, ppp01,…ppp31*. There are some commands that take either a remote name or an interface name as an argument. If the names are not distinct, the command cannot distinguish which you mean.

Menu:	**Remote Address [*160.77.99.211*]**
Command:	**set remote** *profile-name* **address** *ip-address*

The *Remote Address* is the IP address of the PPP, SLIP, or CSLIP interface at the other end of the link. When the link is brought up, this address is used unless a different one has been assigned through PPP address negotiation or information from the RADIUS user file. It is possible to leave this field set to **0.0.0.0**, in which case the correct IP address *must* be supplied by other means. Here are the rules:

- For *Outbound* interfaces, the *Remote Address* must be set to the correct value, because it is the attempt to route to this address that brings up the link. This address cannot be subsequently changed by PPP address negotiation.
- For *Inbound* interfaces, if the IP address of the remote interface is supplied from the RADIUS user database, or if it will be available from PPP address negotiation, the *Remote Address* in the Remote Profile can be left "open", i.e., set to **0.0.0.0**.
- For *Inbound* interfaces, if the IP address of the remote interface will not be available by other means, the *Remote Address* must be set to some valid address. This technique is widely used by ISP providers to supply temporary IP addresses to dial-in users. The *Remote Addresses* you have assigned to var-

ious inbound interfaces comprise a pool of available IP addresses that are assigned dynamically as users dial in. See "Assigning Remote Profiles" on page 290 for an explanation of how this is done.

Menu:	**Interface Address [*160.77.99.111*]**
Command:	**set remote** *profile-name* **ifaddr** *ip-address*

The *Interface Address* is the IP address of this end of the PPP, SLIP, or CSLIP link. If two IntelliServers were connected via a PPP connection, each one's *Interface Address* would be the other's *Remote Address*.

The *Interface Address* must be set to some valid address, but ***Interface Addresses* in different Remote Profiles are not required to be different.** In fact, it is common for Internet Providers to use the Ethernet's IP address (page 208) for all interfaces. In some situations, you may need to use a different IP address. For instance, this could be an outbound interface to a site which expects you to have a particular IP address. This could happen if the remote site had another IntelliServer and it were configured to assign a specific IP address for specific users.

If you are configuring your IntelliServer with an outbound interface to an Internet provider, there is something you need to remember. Since the *Interface Address* cannot be altered through PPP negotiation, the provider must be configured to assign you your *own* IP address. This might be an address from your network you have chosen, or any other address that you can know ahead of time. The provider could be configured to accept your current interface address through PPP negotiation. It *cannot* assign you a temporary IP address randomly chosen from a pool. These temporary addresses are intended to support Internet access to single hosts, while the IntelliServer is designed to link your local *network* to remote hosts and networks (including the Internet).

If you are using the IntelliServer to connect to the Internet, the real goal is to provide access to other hosts on your local network (or other remote networks). There is not much point in the IntelliServer's being there alone. If your local network is to have full access to the Internet, you must have your own *registered* IP network address. If you do not already have this, your prospective Internet provider can tell you how to proceed.

Menu:	**Interface Netmask [*255.255.0.0*]**
Command:	**set remote** *profile-name* **netmask** *ip-netmask*

This is provided for compatibility with earlier versions of the IntelliServer. Its original function has been largely replaced by the new capability of entering netmasks directly into route destinations (see page 204, for example).

If you do not enter or change anything, this automatically defaults to an appropriate value for the *Interface Address* you supplied. For example, if the Interface Address is from a Class B network, the Interface Netmask will be set to 255.255.0.0. Don't change it.

The *Interface Name* is automatically assigned to a Remote Profile when it is created. It is permanently attached to that profile and cannot be changed. The *Interface Name* can be used in place of the *Remote Name* in the **set remote** and **show remote** commands.

Menu:	**Interface Name |*ppp02*|**
Command:	**set remote** *interface-name {parameter value}*

The *Interface Type* specifies whether this Remote Profile's interface will support inbound connections, or initiate outbound connections. The default value is **disabled**, so you must remember to set this value one way or the other before the interface can be used.

Menu:	**Interface Type [Disabled]** **Interface Type [Inbound]** **Interface Type [Outbound]**
Command:	**set remote** *remote-name* **type disabled|inbound|outbound**

The distinction between inbound and outbound interfaces is explained in detail at "Inbound vs. Outbound Profiles" on page 272.

The *Serial Port* and *Group* settings are used differently for inbound and outbound interfaces.

Menu:	**Serial Port [*Any*]**
Command:	**set remote** *profile-name* **port** *number*|**any**

Menu:	***Group [None]***
Command:	**set remote** *profile-name* **group** *number*|**none**

An ***Outbound*** interface needs to know which serial port to use when it initiates the connection. Usually there will be a particular phone line and modem dedicated to this, so you provide the serial port number the modem is connected to. To allow more than one alternative, set the *Serial Port* to **Any** and enter a *group number* for the *group* instead.

If a serial port is to be used for an outbound connection, it must be configured as an *Outbound Connection*, as shown on page 71. To configure serial ports with *group numbers*, assign this group number to all of the ports you want to be members. This is explained during serial port configuration, on page 86.

An ***Inbound*** interface does not need to be told which serial port to use. The connection has already been initiated over one of them, the one you're using! For inbound connections, the *Serial Port* is used to restrict the use of this interface to a particular serial port. If you want this interface to be used regardless of the serial port, set its *Serial Port* to **Any** (the default).

There are two cases to restrict an interface to a serial port. One is to make that port "special". Someone who dials into *that* phone number gets into *that* serial port which has its own dedicated Remote Profile with its own special settings. Ironically, the other case occurs when everything else is the same. In other words, a collection of Remote Profiles may be so configured that any connection may share them. In that case tying a particular interface to a particular port is not restrictive and it simplifies administration. If you are not covered by one of these cases, it is safer for you to keep the *Serial Port* set to **Any**. The *Group* is ignored by inbound interfaces.

The *Maximum Transmit Unit*, or *MTU*, is the maximum number of bytes that this interface sends in a single packet. For SLIP and CSLIP connections, it is important that this number not be larger than the other end's MRU, the largest packet it is prepared to *receive*. This must be done by prior configuration, because SLIP and CSLIP do not negotiate options.

Menu:	**MTU [*1536*]**
Command:	**set remote** *profile-name* **mtu** *size*

For PPP connections, this value can be negotiated. If Maximum Receive Negotiation has been enabled in the associated Options Profile, (see page 268) each side informs the other of its MRU, so that the other side can reduce *its* MTU (for this connection) accordingly.

The *Async Map* is used with PPP connections to prevent selected control characters from appearing in the data stream. When any proscribed character is sent, the interface substitutes a special character sequence. When any special character sequence is received, it is replaced with the corresponding original character.

Menu:	**Async Map [000a0000]**
Command:	**set remote** *profile-name* **async** *value*

How does the map work? There are thirty-two ASCII control characters with decimal values 0 through 31. The *Async Map* is a 32-bit number written in hexadecimal notation. Each bit, starting with the **rightmost**, corresponds to different control character: A map of **00000001** would represent the character value 0 (ASCII **NULL**) and a map of **80000000** would represent the character value 31 (ASCII **US**).

Remembering that each hexadecimal "digit" corresponds to four bits, consider the example **000a0000**. The first sixteen bits (from the right!) are clearly zero. Next is a hexadecimal **a**, which in binary is **1010**. The ones correspond to bits 17 and 19, and the rest of the bits are clearly zero. So, this mask traps the control characters with decimal values 17 and 19, which are the XON and XOFF characters.

Menu:	**Delay Between Redials [*0*]**
Command:	**set remote** *profile-name* **delay** *seconds*

If an ***Outbound*** interface fails to bring up a connection, it waits this minimum amount of time before attempting again. This setting is ignored for *Inbound* connections.

The *Inactivity Timeout* is the maximum number of seconds the interface allows the connection to remain established when there is no data passing through it. If a connection remains inactive for longer than this, the IntelliServer drops the connection — closes the serial port, drops DTR, and hangs up the modem. This timeout applies to both inbound and outbound interfaces.

Menu:	**Inactivity Timeout [*0*]**
Command:	**set remote** *profile-name* **timeout** *seconds*

If the *Inactivity Timeout* is set to **0**, there is no timeout. A link could remain inactive forever unless shut down by other means (e.g., if the other side had an inactivity timer).

Menu:	**RIP [*both*]**
Command:	**set remote** *profile-name* **rip none|send|listen|both**

This specifies whether IntelliServer will:

- **listen** for Routing Information Protocol (RIP) packets from this interface
- **send** RIP packets to this interface
- **both** send and listen for RIP packets, or
- **none** of the above

For more about RIP, see "RIP" on page 210.

The *Dial-In User* is only used for *inbound* interfaces. If a user name is supplied, then this Remote Profile can only be assigned to connections that were initiated when this user logged in. If the user name is left blank, this Remote can be assigned to connections initiated by any user.

Menu:	**Dial-In User [*erik*]**
Command:	**set remote** *profile-name* **user** *user-name*

If a dial-in user is configured in the IntelliServer's NVRAM, there is no way to assign it network information there. If you need this user to be associated with specific network information (a remote IP address, for example) you create a Remote Profile with the required settings and supply the user name there. When this user dials in, the Remote Profile with his name is assigned and the settings are used to bring up the interface.

It is sometimes possible to leave the *Dial-In User* blank, even when supporting NVRAM users. This occurs when you want a large number of users to dynamically share the same pool of network settings. The user dials in, and since there is no Remote Profile with its name on it, an available innominate one is assigned. This has the network information that applies to this user while the connection is up.

If the dial-in user is configured on a network host using RADIUS, the same rules apply. A Remote Profile with a user name can only be assigned to that user. The purpose of the restriction is different, however. For NVRAM users, it was to assign specific network information to a certain user because it could not be included in its own NVRAM configuration. But RADIUS users *do* have network information available in *their* database, so that is not the issue. **In this case, you dedicate a remote to a specific user in order to guarantee that a line is always available for that user, while still allowing it to access your site through a common phone number.**

The *Login Script* is used only by ***Outbound*** interfaces. It allows the interface to log into the remote host. You must supply the name of one of the Login Scripts you defined earlier. See "Login Scripts" on page 259 for details. If this is left blank, then no login script is used. It is unusual for a dial-in connection to *not* require a login, so you will usually be specifying one.

Menu:	**Login Script [*lincoln*]**
Command:	**set remote** *profile-name* **login** *script-name*

One instance where you do not need a login script is the SLIP connection described under the "Bring up Slip Immediately" option on page 269.

Menu:	**Protocol [*Any*]**
Command:	**set remote** *profile-name* **protocol** *type*

Your choices are as follows:

Any	SLIP	CSLIP	PPP	Disabled

For an ***Outbound*** interface, you must specify either SLIP, CSLIP, or PPP, because the interface needs to know which protocol to use before it can bring up the link.

For an ***Inbound*** interface, the desired protocol is learned directly from the user. If it is an NVRAM user it is configured specifically as either a SLIP, PPP, or CSLIP user. If it is a RADIUS user, similar information is stored in that database. For inbound interfaces, then, this is used to optionally restrict this Remote Profile's use, much like the *Serial Port* and *Dial-in User* are. If you want this Remote Profile to be available for any protocol, set the *Protocol* to **Any**. If you want to restrict it, specify SLIP, CSLIP, or PPP.

The setting **Disabled** exists for compatibility with earlier versions of the IntelliServer. It is no longer needed because a Remote Profile's interface can be disabled by setting the *Interface Type* to **disabled** (see page 281.)

The *Phone Number* is used only by ***Outbound*** interfaces with dial or login scripts. If the script contains the command **%p**, this phone number is inserted at that point. See "Dial and Login Script Commands" on page 256. Being able to configure the phone number here conserves dial scripts.

Menu:	**Phone Number [ ]**
Command:	**set remote** *profile-name* **phone** *phone-number*

This is the name of a PPP/SLIP Option profile which contains additional configurations used in bringing up the interface. They are made a separate entity because they are not so likely to vary as the parameters stored in the Remote Profile, and the few variations are apt to be common to several profiles. See "Configuring Option Profiles" on page 265 to learn more.

Menu:	**Options Profile [*default*]**
Command:	**set remote** *profile-name* **option** *option-profile-name*

This is the name of a PPP/SLIP Option profile which contains additional configurations used in bringing up the interface. They are made a separate entity because they are not so likely to vary as the parameters stored in the Remote Profile, and the few variations are apt to be common to several profiles. See "Configuring Option Profiles" on page 265 to learn more.

This is the name of an IP filter you have defined. If this is blank, no IP filter is attached and the traffic through this interface is unrestricted. It is common for different interfaces to have different filters. Certain things might be allowed in the privacy of the home (i.e., the local Ethernet Network) which might not be allowed through any of the remote interfaces. Other traffic might be allowed over an outbound interface to a remote branch of the same business, that would not be allowed over the interface that goes to an Internet provider (ISP).

Menu:	**IP Filter [*melita*]**
Command:	**set remote** *profile-name* **filter** *filter-name*

The *Authentication Protocol* determines which authentication protocol can be used as part of the PPP negotiation. When set to **None** (default), no PPP authentication is performed. When set to **PAP**, PAP protocol is used for authentication, and when set to **CHAP**, CHAP protocol is used. CHAP is considered the more secure method because the authentication fields are sent encrypted.

Menu:	**Authentication Protocol [*None*]**
Command:	**set remote** *profile-name* **authority none|pap|chap**

When PAP protocol is used, the *PAP ID and Password* are sent to the remote system when it requests them. When CHAP protocol is used, the *CHAP Name* and *Secret* are sent. Since you aren't using both protocols at once, the PAP or CHAP parameters are stored in the same place.

Menu:	**CHAP Name/PAP ID [ ]**
Command:	**set remote** *profile-name* **id** *name-or-id*

This needs to be a matter of some pre-arrangement. If the IntelliServer has the outbound interface and the remote site expects CHAP authentication, it had better enable it and know the correct CHAP name and secret to configure. So, this is not always a question of how you *want* to configure things.

Menu:	**CHAP Secret/PAP Password [ ]**
Command:	**set remote** *profile-name* **secret** *secret-or-password*

This setting applies to ***Outbound*** as well as ***Inbound*** interfaces. For *Outbound* interfaces, the IntelliServer supplies the information when requested by the site you are logging into. For *Inbound* interfaces, you request the information from the site that logged into yours.

Assigning Remote Profiles

When a user dials into the IntelliServer to bring up an inbound interface, there is certain information that is known right away:

- Port number used
- User name
- Pprotocol (SLIP, CSLIP, or PPP) needed (either from the NVRAM configuration or from the RADIUS authentication reply)
- Its IP address (for RADIUS users. RADIUS supplies an IP address using the *Framed-Address* attribute. There are two addresses which have special meaning: 255.255.255.255 means the IP address is unspecified and must be determined from PPP address negotiation. 255.255.255.254 also means that the IP address is unspecified, but the IntelliServer should assign one from a pool.)

Assignment Rules

Given the port, user, protocol, and address, the IntelliServer must assign a Remote Profile and its interface to finish bringing up the connection. There are four rules for doing this:

1. The Remote Profile's interface cannot already be in use. One Remote Profile per interface per connection. Its interface type must be *Inbound*.
2. The Remote Profile must be compatible with the port number, user name, protocol, and IP address specified for this user.
3. More restrictive Remote Profiles (that are still compatible) are assigned in preference to less restrictive ones.
4. If other network options are specified for a RADIUS user, these options supersede the ones configured in the Remote Profile.

Rules for Compatibility

How does the IntelliServer decide whether a Remote Profile is *compatible* with the connection's requirements?

- If the Remote Profile specifies a *Serial Port*, it must match the one receiving the connection. If set to **Any**, the serial port does not matter.
- If the Remote Profile specifies a particular *Dial-in User*, that name must match the user that logged in. If left blank, it can be used with any user name.

- If the Remote Profile specifies a particular *Protocol*, it must match the protocol this user desires. If set to **Any**, then it can be used with any protocol.
- If there is no IP address associated with the user, the Remote Profile must contain a valid *Remote Address*. An IP address is associated with the user if an actual IP address is supplied through RADIUS (using the "Framed-Address" attribute) or if the value 255.255.255.255 is supplied. The latter option is not an address, but at least it is a promise that a real address is forthcoming during PPP address negotiation.

Assignment Priority

Considering now only the Remote Profiles that are compatible, how will the *most* suitable one be found? This is what the IntelliServer looks for in order, from highest to lowest priority:

- A Remote Profile which specifies both the *Serial Port* and the *Dial-in User*. If this Remote weren't assigned here, it never *could* be assigned, because no one else will be using this port while this connection is using it.
- A Remote Profile which specifies the *Serial Port*, for the same reason as above.
- A Remote Profile which specifies the *Dial-in User*. This means the dedication of a Remote Profile to a particular user takes precedence over other considerations.
- A Remote Profile which species a *Remote Address*, if this address matches a specific one provided from RADIUS. This is a bit like having a matching user name.
- A Remote Profile where the *Remote Address* is zero, and the IP address provided from RADIUS is either an actual one or 255.255.255.255 (use address negotiation); **OR**, a Remote Profile where the *Remote Address* is valid (non-zero) and the user is an NVRAM user or a RADIUS user with 255.255.255.254 specified for an address (assigned from the pool). In either of these conditions, there would be a total of one remote IP address offered from all the parties.
- A Remote Profile with a non-zero *Remote Address*, and also a valid IP address from RADIUS. The one from RADIUS supersedes the one stored in the Remote Profile.

CHAPTER 12

Network Administration

In this chapter you learn techniques and commands that are useful for network administration, including the following:

- Using the **`ping`** command to check routes.
- Displaying and modifying the ARP table.
- Displaying and modifying the routing table.
- Displaying network status and statistics.
- Troubleshooting.

What Does Ping Do

The **`ping`** command sends an *ICMP Echo Request* packet to the designated host and waits for a reply. When it receives the reply it reports that the host is *alive*. This is a basic command to use when you want to verify that two hosts can communicate with each other.

In Example 12-1, a host is pinged using its host name (**`ping yacht`**) so the IntelliServer had to resolve the name into an IP address. This it did either through its Host Table (page 213) or by sending a name resolution request to one of the Name Servers configured on page 225. Once the IP address was determined, the echo request was sent and a reply received. The IntelliServer shows both the host name you supplied (**`yacht`**) and the IP address it found (**`160.77.99.78`**).

Example 12-1: Ping Command

`ping` *`hostname | ip-address`*

```
# ping yacht
yacht (160.77.99.78) is alive
# ping yot
yot unknown host
# ping 160.77.99.2
160.77.99.2 (160.77.99.2) is alive
# ping 160.77.99.214
no reply
#
```

Next, a different host called yot is pinged, but this time the IntelliServer could neither find the host in its local table, nor obtain the IP address from a name server. So, it replies **`yot unknown host`**. In the third sample, it is particularly easy for the IntelliServer to determine the IP address, **`160.77.99.2`**. It doesn't have to determine whether the host name is in its table or whether any name-servers are working. This host has responded, so **`ping`** reported *both* the name you supplied *and* the IP address in the result.

In the last example, there was no reply when you tried to ping **`160.77.99.214`**. Why? Most of the time it will be one of these reasons:

- If an IP address is supplied to the ping command, no name resolution is required, so the problem is not the inability to resolve the host name.
- The ping request may have not been sent. The IntelliServer may not have a route to the target host.
- The ping request may not have been sent successfully because the IntelliServer is not connected to the local network, or a required PPP/SLIP/CSLIP link is not up.
- The ping request may have been sent, but some intermediate host may not have a route that allows it to deliver the packet.
- The ping request may not have been delivered because the target host is not connected to its local network, or a required PPP/SLIP/CSLIP link is not up (including the case where the target host does not exist or is turned off).
- The ping request may have been delivered but target host may not have had a route back to the sender so it could send a reply.
- The target may have sent a reply, but an intermediate host may not have been able to route the reply to its destination (the original sender).

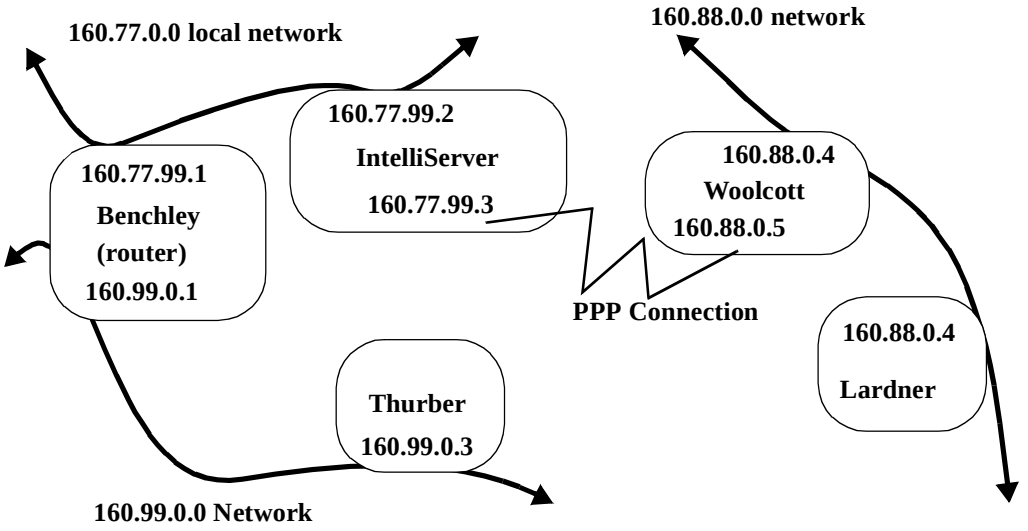
ping:	Ping verifies that: 1. The target host exists. 2. Both the sending host and the target are connected to a network interface. 3. If the sending host and the target are connected via a PPP/SLIP connection, the connection is up (active). 4. *Each* host has a route to the other or to its network (including default routes when applicable). 5. All intermediate hosts have the necessary routes to forward the packets.

When Ping Fails

Suppose you are unable to ping some host from the IntelliServer. How do you find the problem? That depends on where the host is on the network relative to the IntelliServer. Consider Example 12-2. The IntelliServer is on a local network which also includes host *benchley,* which in turn is a router to a second network containing host *thurber.* The IntelliServer is connected via a PPP link to host *woolcott,* which is on a network with host *lardner.*

Each of these four hosts illustrates something different. Let us try to ping each one in turn.

Example 12-2: Who can you Ping?



Pinging a Local Host

Suppose you tried to ping *benchley* from the IntelliServer and received no reply. It is right there on our local network, so there are only a few things to check:

1. If you were pinging *benchley* by name, try using its IP address, 160.77.99.1 instead. If that works where using the name did not, there is a likely problem with name resolution. If you are using a nameserver, make sure the IntelliServer has its correct IP addresses and that *its* name server daemons are running. Assuming that the IP address didn't work either…
2. See whether the IntelliServer can ping other hosts on the same network (160.77.0.0 in this example). If not, then check its physical connection to the network and recheck its configuration. Remember that when you change the IntelliServer's basic network configuration (IP Address, Netmask, etc.) you need to save the configuration and reboot before the changes take effect.
3. Check the IntelliServer's routing table (see "Routing Table" on page 309) to make sure its route to the local network is correct. In this example the IntelliServer would have automatically added a route to network 160.77.0.0 through its interface at 160.77.99.2. Make sure there is not also some specific route to *benchley* that shouldn't be there.
4. Check the IntelliServer's ARP table (see "ARP Table" on page 306) to see whether it was able to resolve the IP address into an Ethernet address. If not, this suggests a problem with ARP services on either the IntelliServer or *benchley.*
5. Remember, for a ping to work the echo request needs to be delivered *and* the reply needs to find its way back. If the IntelliServer *can* ping other network hosts, see whether *benchley* can. If not, you should probably insert a bookmark here and read the *"Benchley Configuration Guide"* for a while.
6. Make sure you don't have some IP filter installed on either the IntelliServer or on *benchley*, either of which might inadvertently be keeping the other one's packets out. (See pages 195 and page 209).

Pinging a PPP Target

Now you have sorted things out and can ping *benchley.* Now you try to ping *woolcot*, and it fails. Well, *woolcot* is not on our local network; it is on the other end of a PPP connection from the IntelliServer. Let's suppose it is an outbound connection, i.e., the connection is initiated by the IntelliServer in response to network traffic wanting to go there.

1. Check the IntelliServer's routing table (page 309) to make sure that there is a route to the remote host (160.88.0.5) through the interface address of one of the IntelliServer's PPP links (160.77.99.3, in this example). If this is not there, check your Remote Profile and see that it has been configured properly: all the correct addresses, configured as an *outbound* interface, specifying *PPP* protocol. Don't forget, since outbound interfaces are configured at start-up, you should save your configuration and reboot whenever you add new Remote Profiles or change the configuration of any outbound ones. Did you remember to do this?
2. Try the ping again. The first ping, the one that created the demand that brought the link up, will probably time out by the time the link *is* up, but once the link is up further pings should proceed as expected.
3. If the link is up and you still don't get replies to your ping, check *woolcott's* routing tables and make sure it is configured properly. The ping requests from the IntelliServer will have a source address of 160.77.99.3. Does it know where that address is?
4. If the link would not come up, stop what you are doing and read all about syslogging, starting with pages 193 and 205 of this guide. You will need to configure a syslog daemon on one of your network's hosts, if there isn't one running already. While the IntelliServer *can* syslog to its console port (0 by default), you will be getting a lot of output and it is better to have it in a file. The key here is to set the IntelliServer's *syslog priority* to LOG_INFO, then save configuration and reboot (*S.C.A.R*). As the IntelliServer attempts to bring up the PPP link, it will *syslog* a trail that should help you see how far along it got, and perhaps to see where things have gone wrong.
5. If the link still won't come up, there is usually some form of logging you could enable on *woolcott's* end. There may be more clues from its perspective.
6. If you're really stuck and need to call our tech support number, have the results of these steps handy. Especially the syslog.

Pinging Through a Router

Suppose, finally, *woolcott* is sorted out. Now for *thurber* (in no particular order). Again looking at Example 12-2 on page 296, *thurber* is on a different network. It cannot be reached directly, but can be reached him through *benchley*, our router. Suppose the IntelliServer can't ping *thurber.*

First thing, make sure each local network is ok. Make sure that the IntelliServer can ping *benchley* (you just did), and that *benchley* can ping *thurber.* (If *benchley* can't ping *thurber*, you get out the *benchley* book again…) In order for a ping request to reach *thurber* the IntelliServer needs to know to send it to *benchley.* Check the IntelliServer's routing table to make sure there is a route to network 160.99.0.0 through host 160.77.99.1 (for this example, anyway). How would this route have gotten there? Since both networks on *benchley* seem pretty static, configuring a static route in the IntelliServer's gateway table would be one way (see page 227). If you need to add one now, add it to the gateway table but also add it to the current routing table using the *route* command, so it will take effect immediately.

Note in this example that a route to host 160.99.0.3 *could* have been used instead of a route to the 160.99.0.0 network. But surely there are other hosts on that network. Would you add a separate host route for each? A single route to the network is easier.

Now that (presumably) *benchley* has the packet for *thurber*, it should be able to deliver it because it was able to ping *thurber* before. If *thurber* has it, it will want to send a reply. Send it where? Just as a route was needed through *benchley* to reach it, it needs one to reach us. Something like a route to network 160.77.0.0 *through* host 160.99.0.1. (Wasn't *benchley* 160.66.99.1 in *our* routing table, and now it is different? Yes, it has two different IP addresses, one for each local network it is attached to.

Sometimes it is useful to confirm whether a ping request was sent. The **`netstat icmp`** command (page 314) lists the number of echo requests and replies generated and received. The trick is, now have *thurber* try and ping us. Presumably it will fail as before. But now look at the ICMP statistics and see whether there were any echo requests received. If there were, the problem is in the IntelliServer-to-*thurber* direction. Otherwise, it is in the *thurber*-to-IntelliServer direction.

Pinging a Host on a Remote Network

Finally, having pinged *benchley* and dialed up *woolcott,* you'll now see if you can ping *lardner.* You see, this is not really any different from accessing *thurber*:

Lardner needs a route to the IntelliServer's network through *woolcott,* and the IntelliServer needs a route to *Lardner's* network through *woolcott* as well. The situation differs slightly from the previous, because this part of the network might be dynamic. *Woolcott* might be an Internet provider that sometimes will be attaching a different network to that PPP interface. The IntelliServer side is straightforward; if the link be up, *woolcott* is our host. But suppose *woolcott* was an IntelliServer as well. It can't have a route to our network (through the first IntelliServer) in its gateway table, because our IntelliServer might not be there. Besides, if it had lots of dial-in customers there wouldn't be room for all those static routes anyway. This is why routes through PPP interfaces can be stored for each dial-in access customer ("user") in the RADIUS user database. Alternatively, this route could have been acquired using RIP protocol, assuming each end were properly configured.

A Leap of Faith

Now someone tells you that *lardner* can't ping *thurber.* How do you sort that one out?

First, make sure everyone can ping its nearest neighbors (in terms of network hops), then increase the jump: *lardner* to *woolcott?* yes; *lardner* to *IntelliServer?* yes; *lardner* to *benchley?* yes. The goal is to determine the exact line between what works and what does not.

Look at the routing tables at each host along the way and envision what should happen: *lardner* needs to know that *thurber* lies beyond *benchley,* which in turn is on a network that lies beyond the IntelliServer, which dwells on the other side of *woolcott.* Three extra routes on *lardner* would be enough:

- to destination network 160.99.0.0 through gateway host 160.77.99.1,
- to destination network 160.77.0.0 through gateway host 160.77.99.3, and
- to destination host 160.77.99.3 through host 160.88.0.4.

This plus *lardner's* standard route to network 160.88.0.0 through its own address (160.88.0.4) will get you there.

Naturally, *thurber* needs the same routing information, but seen from the other direction.

Suppose *benchley* were the *only* router attached to *thurber's* network. Also, suppose that *woolcott* were the only link to the outside world from *lardner's* perspective.. Instead of putting in all those routes, *lardner* could have a single default route (destination 0.0.0.0) through 160.88.0.4 (*woolcott*) and *thurber* could have a default route through 160.99.0.1 (*benchley).* Using default routes, *lardner* would send *thurber's* packet to *woolcott*, not because it understood the network, but because it now trusts *woolcott* to know what to do with it. *Woolcott* still needs to know how to reach *thurber* through the IntelliServer, but if it is the only window to the outside world, the IntelliServer might be *its* default route as well. Before you get *too* carried away by default routes, remember that there can be only one default route for any particular host. Pick the most popular gateway and make it the one. Anything that needs to be pointed in a different direction must be entered explicitly.

Sample Syslog Output

In the last section it was suggested that syslogging might be helpful if you want to understand why a PPP link might not be coming up. Here are two annotated examples of the syslog output you might expect from bringing up a PPP connection.

Example 12-3 shows an Inbound PPP connection coming up (successfully, of course), and Example 12-4 on page 304 shows an outbound PPP connection coming up. In each case, syslogging has been enabled at the SYS_INFO level. The shaded rows do not appear in the syslog file, but contain notes about the syslog messages in the row to follow. The messages in your syslog file will not be exactly like these. IP addresses and other details will be different, and the wording may change slightly with different software releases. But, these should give you the general idea.

Example 12-3: Annotated Syslog for Inbound PPP Link

User nppp successfully logged into the IntelliServer (user name and password checked out, and the IntelliServer recognized it as a PPP user):
Sep 29 09:35:41 16.77.22.120 ppp: user nppp calling on port 6
There was an interface (remote profile) available for this connection:
Sep 29 09:35:41 16.77.22.120 ppp: found remote interface ppp01 (gw_2) ifaddr 192.9.100.1
IntelliServer sets up routes and timers, looks up Options profile:
Sep 29 09:35:41 16.77.22.120 ppp: del old routes thru ppp01 **Sep 29 09:35:41 16.77.22.120 ppp: del old host routes to 192.9.100.2** **Sep 29 09:35:41 16.77.22.120 ppp: add host route to 192.9.100.2 thru 192.9.100.1** **Sep 29 09:35:41 16.77.22.120 ppp: ppp01 dialin timers, delay=0 inactivity=0** **Sep 29 09:35:41 16.77.22.120 ppp: using options profile 'svr4'**
Start-up phase complete, starting up PPP state machine. The command-line arguments are shown to confirm correctness of options.
Sep 29 09:35:41 16.77.22.120 ppp: exec pppfsm +usr nppp +unit 1 +nv 1 +as 0 +pass +mtu 1500 +in -ac -pc +mru 1500 +addr draft -vj ttyA06
Sent at LOG_NOTICE level for any user who logs in (this message does not appear synchronously with other events shown here):
Sep 29 09:35:42 16.77.22.120 logger: [15] Port 6:0 User nppp Start ppp
Starting the PPP state machine: which interface was assigned to which port?
Sep 29 09:35:42 16.77.22.120 ppp01: main: connected ppp01 to ttyA06 **Sep 29 09:35:42 16.77.22.120 ppp01: subs: set MTU 1500**

Example 12-3: Annotated Syslog for Inbound PPP Link (Continued)

Since this is PPP, the remote site (who dialed into us) starts the negotiation process:

```
Sep 29 09:35:43 16.77.22.120 ppp01: fsm: (lcp-listen) rcvd conf req, id 2
```

The IntelliServer sends some information:

```
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: send MRU, 1500
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: send ASYNC MAP, 0x0
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: send MAGIC NUMBER, 0x9b3c948
Sep 29 09:35:43 16.77.22.120 ppp01: fsm: (lcp-listen) send conf req, id 145
Sep 29 09:35:43 16.77.22.120 ppp01: main: timeout in 3 seconds
```

IntelliServer receives information from remote, likes it, sends ACKs:

```
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: rcvd MRU
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: 1500
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: ACK
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: rcvd ASYNCMAP
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: 0x0
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: ACK
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: rcvd MAGIC NUMBER
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: 0xcaabd05c
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: ACK
Sep 29 09:35:43 16.77.22.120 ppp01: lcp: send CONF ACK
Sep 29 09:35:43 16.77.22.120 ppp01: fsm: (lcp-acksent) rcvd conf ack, id 145
Sep 29 09:35:43 16.77.22.120 ppp01: main: untimeout
```

Address negotiation…

```
Sep 29 09:35:43 16.77.22.120 ppp01: subs: set MTU 1500
Sep 29 09:35:43 16.77.22.120 ppp01: ipcp: send IP ADDR, 192.9.100.1
Sep 29 09:35:43 16.77.22.120 ppp01: fsm: (ipcp-closed) send conf req, id 63
Sep 29 09:35:43 16.77.22.120 ppp01: main: timeout in 3 seconds
Sep 29 09:35:43 16.77.22.120 ppp01: fsm: (ipcp-reqsent) rcvd conf req, id 3
Sep 29 09:35:43 16.77.22.120 ppp01: ipcp: rcvd IP ADDR
Sep 29 09:35:43 16.77.22.120 ppp01: ipcp: his 192.9.100.2
Sep 29 09:35:43 16.77.22.120 ppp01: ipcp: ACK
Sep 29 09:35:43 16.77.22.120 ppp01: ipcp: send CONF ACK
```

We like each other. Fix up routes in case negotiation changed any of them:

```
Sep 29 09:35:43 16.77.22.120 ppp01: fsm: (ipcp-acksent) rcvd conf ack, id 63
Sep 29 09:35:43 16.77.22.120 ppp01: main: untimeout
Sep 29 09:35:43 16.77.22.120 ppp01: subs: del routes thru ppp01
Sep 29 09:35:43 16.77.22.120 ppp01: subs: del old routes to 192.9.100.2
Sep 29 09:35:43 16.77.22.120 ppp01: subs: set VJCOMP on=0 slotids=0
Sep 29 09:35:43 16.77.22.120 ppp01: ipcp: LINK UP 192.9.100.1:192.9.100.2
```

Example 12-4: Annotated Syslog for Outbound PPP link

Network traffic detected for the other end of this PPP link:

```
Sep 29 11:02:06 16.77.22.120 pppd: outbound request rcvd: ip dst 195.1.1.1: proto
icmp
```

IntelliServer runs the dialer and login scripts. User name and password are “line1”:

```
Sep 29 11:02:06 16.77.22.120 pppd: ppp02 dialout timers, delay=0 inactivity=0
Sep 29 11:02:06 16.77.22.120 pppd: dialing host h interface ppp02 tty ttyA01
Sep 29 11:02:06 16.77.22.120 pppd: send(^M)
Sep 29 11:02:06 16.77.22.120 pppd: timer(30 secs)
Sep 29 11:02:06 16.77.22.120 pppd: want(gin:)
Sep 29 11:02:06 16.77.22.120 pppd: rcvd(^M ^M IntelliServer Release 1.2.2^M topgun
login: )
Sep 29 11:02:06 16.77.22.120 pppd: send(line1^M)
Sep 29 11:02:06 16.77.22.120 pppd: timer(30 secs)
Sep 29 11:02:06 16.77.22.120 pppd: want(word:)
Sep 29 11:02:06 16.77.22.120 pppd: rcvd(^M topgun login: )
Sep 29 11:02:06 16.77.22.120 pppd: rcvd(line1^M Password:)
Sep 29 11:02:06 16.77.22.120 pppd: send(line1^M)
```

Dialup and login were successful: look up Option profile and start the PPP state machine:

```
Sep 29 11:02:06 16.77.22.120 pppd: connected to h
Sep 29 11:02:06 16.77.22.120 pppd: using options profile ‘iserver’
Sep 29 11:02:06 16.77.22.120 pppd: exec pppfsm +unit 2 +nv 2 +as a0000 +out +mru
1500 -addr -vj ttyA01 195.1.1.2:195.1.1.1
Sep 29 11:02:06 16.77.22.120 ppp02: main: connected ppp02 to ttyA01
```

Since we initiated the call, we start. (Also note we timed out the first time and had to retry):

```
Sep 29 11:02:06 16.77.22.120 ppp02: subs: set MTU 1500
Sep 29 11:02:06 16.77.22.120 ppp02: lcp: send MRU, 1500
Sep 29 11:02:06 16.77.22.120 ppp02: lcp: send ASYNC MAP, 0xa0000
Sep 29 11:02:06 16.77.22.120 ppp02: lcp: send MAGIC NUMBER, 0xbd499e
Sep 29 11:02:06 16.77.22.120 ppp02: lcp: send PROTO COMPRESSION
Sep 29 11:02:06 16.77.22.120 ppp02: lcp: send ADDR/CNTL COMPRESSION
Sep 29 11:02:06 16.77.22.120 ppp02: fsm: (lcp-closed) send conf req, id 59
Sep 29 11:02:06 16.77.22.120 ppp02: main: timeout in 3 seconds
Sep 29 11:02:09 16.77.22.120 ppp02: fsm: (lcp-reqsent) timeout
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: send MRU, 1500
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: send ASYNC MAP, 0xa0000
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: send MAGIC NUMBER, 0xbd499e
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: send PROTO COMPRESSION
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: send ADDR/CNTL COMPRESSION
Sep 29 11:02:09 16.77.22.120 ppp02: fsm: (lcp-reqsent) send conf req, id 60
Sep 29 11:02:09 16.77.22.120 ppp02: main: timeout in 3 seconds
Sep 29 11:02:09 16.77.22.120 ppp02: fsm: (lcp-reqsent) retransmit #1
```

Example 12-4: Annotated Syslog for Outbound PPP link (Continued)

And here is the first set of responses:

```
Sep 29 11:02:09 16.77.22.120 ppp02: fsm: (lcp-reqsent) rcvd conf req, id 139
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: rcvd MRU
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: 1500
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: ACK
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: rcvd ASYNCMAP
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: 0xa0000
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: ACK
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: rcvd MAGIC NUMBER
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: 0x10fea0c6
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: ACK
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: rcvd PROTO COMPRESSION
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: ACK
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: rcvd ADDR/CNTL COMPRESSION
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: ACK
Sep 29 11:02:09 16.77.22.120 ppp02: lcp: send CONF ACK
Sep 29 11:02:09 16.77.22.120 ppp02: fsm: (lcp-acksent) rcvd conf ack, id 60
Sep 29 11:02:09 16.77.22.120 ppp02: main: untimeout
```

Tell him we like him:

```
Sep 29 11:02:09 16.77.22.120 ppp02: subs: set MTU 1500
Sep 29 11:02:09 16.77.22.120 ppp02: ipcp: send no options
Sep 29 11:02:09 16.77.22.120 ppp02: fsm: (ipcp-closed) send conf req, id 197
Sep 29 11:02:09 16.77.22.120 ppp02: main: timeout in 3 seconds
Sep 29 11:02:09 16.77.22.120 ppp02: fsm: (ipcp-reqsent) rcvd conf req, id 80
Sep 29 11:02:09 16.77.22.120 ppp02: ipcp: send CONF ACK
Sep 29 11:02:11 16.77.22.120 ppp02: fsm: (ipcp-acksent) rcvd conf req, id 81
Sep 29 11:02:11 16.77.22.120 ppp02: ipcp: send CONF ACK
```

See if he likes us. He does!

```
Sep 29 11:02:12 16.77.22.120 ppp02: fsm: (ipcp-acksent) timeout
Sep 29 11:02:12 16.77.22.120 ppp02: ipcp: send no options
Sep 29 11:02:12 16.77.22.120 ppp02: fsm: (ipcp-acksent) send conf req, id 198
Sep 29 11:02:12 16.77.22.120 ppp02: main: timeout in 3 seconds
Sep 29 11:02:12 16.77.22.120 ppp02: fsm: (ipcp-acksent) retransmit #1
Sep 29 11:02:12 16.77.22.120 ppp02: fsm: (ipcp-acksent) rcvd conf ack, id 198
Sep 29 11:02:12 16.77.22.120 ppp02: main: untimeout
Sep 29 11:02:12 16.77.22.120 ppp02: subs: set VJCOMP on=0 slotids=0
Sep 29 11:02:12 16.77.22.120 ppp02: ipcp: LINK UP 195.1.1.2:195.1.1.1
```

ARP Table

The ARP table is used to keep track of what Ethernet addresses correspond to which IP addresses, and was introduced in "Using ARP to Determine Ethernet Addresses" on page 188. It is sometimes beneficial to look at this table, at other times you may want to add entries manually.

Example 12-5: Show ARP Command

```
# show arp
carl (160.77.99.3) at 00:00:11:22:33:44, 0 minutes, permanent, published
phillip (160.77.99.55) at (incomplete), 1 minutes
emmanuel (160.77.99.2) at 00:00:6b:82:50:6c, 5 minutes
? (160.77.99.77) at 00:00:c0:7e:55:4b, 0 minutes
```

Example 12-6: Show ARP Command (single entry)

```
# show arp 160.77.99.2
emmanuel (160.77.99.2) at 00:00:6b:82:50:6c, 5 minutes
```

The **`show arp`** command displays the current ARP table. This includes the following data:

- Host name - If the IntelliServer can determine it from the IP address stored in the ARP table.
- IP address.
- Ethernet address, if known. - If there was no response to the IntelliServer's ARP request for this address, the address is marked **`(incomplete)`**.
- Number of minutes this entry has been in the table since it was last referenced.
- Option flags - **`published`**, **`permanent`**, or **`trailers`**. If an entry is marked *published*, the IntelliServer will respond to any ARP requests it receives for that IP address's Ethernet Address. If marked *permanent*, the entry will not expire; other entries will be removed from the table if there is no activity for that host for a long period of time. The option flag, *trailers* is reserved for future use.

In Example 12-5, you see a permanent, published entry for host name *carl*. This is probably there to support proxy-ARP for that host. The ARP table entry for *phillip* is incomplete. The IntelliServer was able to determine that the host name for 160.77.99.55 was *phillip*, but there was no response when the IntelliServer tried to get its Ethernet address. This suggests perhaps *phillip* is not on the network now. In the final example, there is an ARP entry for 160.77.99.77 which contains an Ethernet address, but the host name is unknown. This indicates that the host itself is on the network and is responding, but this IP address is not in the IntelliServer's host table, nor known to any nameserver the IntelliServer might be configured to use.

Entries can be added into the ARP table in three ways:

1. Entries are added automatically when the IntelliServer tries to access a new IP address on his local network. Then the IntelliServer sends the ARP request, and when the reply is received the Ethernet address in the ARP table is filled in. These entries are temporary and will expire when they have not been used for a long while.
2. Entries can be added as a result of bringing up a PPP, SLIP, or CSLIP connection where *Proxy ARP* has been specified. This is discussed in "Proxy ARP" on page 189, and in the discussion of the option profile's *Enable Proxy ARP* option, on page 269. These entries will be *permanent* and *published*. The information is shared with others and the entry won't be removed until the PPP, SLIP, or CSLIP connection goes down.
3. Entries can be added manually using the **`add arp`** command. If you add *permanent* entries, they will not be removed unless you delete them manually with the **`delete arp`** command. This command can also be used to manually delete entries added by other means.

Permanent ARP entries are only permanent as long as the IntelliServer stays up. They are not stored in NVRAM the way static routes in the *Gateway Table* are, for example. This is generally not an issue, because usually the only permanent entries you deal with are the ones created automatically for proxy ARP on behalf of remote hosts.

ARP Table — Changing it Manually

Sometimes you may want to add or delete ARP table entries by hand. Use the **`add arp`** and the **`delete arp`** commands.

Where *options* are one or more of the following:

published	temporary	trailers

Examples of the **`add arp`** command are shown in Example 12-7. The meanings of *published* and *trailers* are as described above under the *show arp* command. *Temporary* means **not** permanent. That is, if you use the **`add arp`** command, it is assumed permanent unless you add the temporary option. This is because you will usually want entries you add by hand to persist until you delete them.

Example 12-7: Add Arp Command

```
add arp ip-address|hostname ethernet-address options
# add arp 160.77.99.4 00:00:c0:7e:ee:30 published
# add arp jeeves3 00:00:c0:7f:ee:20
```

The **`delete arp`** command removes the entry with the associated IP address from the ARP table.

Example 12-8: Delete Arp Command

```
delete arp ip-address|hostname
# delete arp 160.77.99.4
# delete arp jeeves
```

When will you want to change the ARP table manually?

- When you have replaced the Ethernet controller on one of your network hosts, the new controller will have a different Ethernet address, but the IP address is the same. You can use the **`delete arp`** command to remove any stale ARP table entry rather than waiting for it to expire.
- If you had been relying on proxy ARP to send traffic for a remote host to the IntelliServer (so it could forward it through a PPP link), and forgot to enable the option, you can add the necessary permanent published entry now to see whether that fixes the problem.

Routing Table

The Routing Table tells the IntelliServer where to send IP packets based on the IP address. The basics of routing are discussed in "IP Addresses and Routing" on page 181 so it is assumed you have read that section or are already familiar with the basic concepts.

You can use the **`show route`** command to view the current routing table, as in Example 12-9 below:

Example 12-9: Show Route Command

```
show route
Interface     Destination            Gateway    Count  Flag
sonic0         160.77.0.0/18    160.77.28.204     8571  up net
ppp01       160.77.99.200       160.77.99.101        0  up host
ppp01         160.77.64.0/18    160.77.99.200       46  up net gateway
sonic0       160.77.128.0/18    160.77.11.102   102343  up net gateway
sonic0            0.0.0.0       160.77.11.103      400  up net gateway
```

The routes in this example may seem familiar; they are the ones described in Table 9-9, "Typical Routing Table, With Comments" on page 183. You should refer there, and to "Network Diagram to illustrate routing" on page 184, if you want to find out what these entries would represent in the real world.

The interface indicates whether packets will be sent to the local Ethernet controller (sonic0) or to one of the PPP, SLIP, or CSLIP interfaces. The destination specifies the host or network you want to reach, and the gateway specifies where such packets would be sent. The count represents the number of times this routing table entry has been used to route packets. The flags give the status of each route, as in Table 12-1:

Table 12-1: Routing Flags

`up`	The route is usable. In the case of outbound PPP, SLIP, CSLIP routes, this does not imply that the link itself is up. If you are not sure, you have to use the **`pppstat`** command to find out.
`host`	This is a route to a host.
`net`	This is a route to a network.
`gateway`	This is a route to a network that is not our own local one.
`rip`	This route was added because of information received in a RIP packet.

Automatic Routes

Routes are added automatically at various times:

- Direct routes through the IntelliServer's Ethernet interface and any outbound PPP, SLIP, or CSLIP interfaces are added at start-up time.
- Routes through the IntelliServer's inbound PPP, SLIP, and CSLIP interfaces are added when these links come up, and deleted when the links go down.
- Static routes listed in the Gateway Table (page 227) are added at start-up and whenever new inbound PPP, SLIP, and CSLIP connections are brought up. If the table has routes through any of these inbound connections, the routes will be added correctly after the new interface route has been added.
- Routes through an inbound PPP, SLIP, CSLIP connection that are specified in the RADIUS user database are added when the connection is brought up, and deleted when the connection is dropped.
- Routes learned via RIP protocol (page 210) are added and deleted according to information received from the rest of the network.

Routing Table — Changing it Manually

Sometimes you will want to add and delete routing table entries by hand. Use the **`add route`** and the **`delete route`** commands. Example 12-10 shows how to add routes to the table.

Example 12-10: Add Route Command

```
add route destination gateway
add route default gateway
```

```
# add route 160.88.128.0/17 160.77.99.35
# add route 160.88.128.20 160.88.160.3
# add route sixtysix 160.77.99.3
# add route default 160.77.99.4
```

The *destination* can be the IP address of a specific host, a network, or the word **`default`**. When present, a default route is used when no other route would apply.

When you add a route to a host, normally you specify the host's IP address. If you specify a host name instead, the IntelliServer attempts to resolve the host name to an IP address, and then stores the IP address in the table. Similarly, when you are adding a route to a network, you can either enter the network number or give a network name you have defined in the IntelliServer's network table (see page 213).

The gateway can also be specified as either a host name or an IP address. The gateway is always one of the IntelliServer's interface addresses; the *Interface Address* in one of the Remote profiles (page 280) or the IP address of the IntelliServer's Ethernet interface (page 204).

When you add a route, the IntelliServer checks to see that the gateway is reachable based on the routes that are already in the table. If the gateway cannot be reached based on the routes already present, or if it can be reached only via a default route, the new route will not be added. Consider again Example 12-9 on page 309. If the routes had been added in the order shown, the table would appear as shown.

Suppose however you had tried to manually enter the route to the 160.77.64.0/18 network (through host 160.77.99.200) before you had added the route to host 160.77.99.200. None of the existing routes would have been able to reach the gateway you specified, except for the default route. While you *could* have added the new route, why bother? It would be like saying "Richard Hancock, *and all human beings*, go through the door on the left". Why single out poor Richard, then?

When routes are added and deleted automatically by the various mechanisms on page 310, the necessary interface routes are added before any routes through the interfaces. The entries in the Gateway table are entered in the order they appear.

Whenever a route is added, *any existing routes to the same destination are automatically removed.* This applies whether the routes are being added manually via the **`route`** command or automatically when PPP/SLIP connections are established. This is done because multiple routes to the same destination would be meaningless. (If you are using RIP protocol, the IntelliServer keeps a separate table of routes it learns from other hosts. This table *can* have multiple routes to the same destination, but only the *shortest* route is loaded into the routing table itself).

You delete routes using the **`delete route`** command shown here.

Example 12-11: Delete Route Command

```
delete route destination gateway
delete route default gateway
```

```
# delete route 160.88.128.0/17 160.77.99.35
```

Why would you need to manipulate routes manually? Sometimes this is done to correct a configuration error made in the gateway table or RADIUS routes. Rather than fix the gateway table or RADIUS user file and then have to bring up the link again, it may be just possible to patch the routing table on the quick, to make sure the change would have the desired effect. (Changes to the routing table are immediately effective).

You can also do this to adapt to changes in your network configuration. (They moved the wires: you don't reach this network through this host anymore, you reach it through *that* host…) Naturally, if everyone is using RIP, the IntelliServer might be informed of the routing change automatically and you wouldn't have to adjust it manually after all.

Network Statistics

Network protocols are designed with the knowledge that network traffic is not always perfect. Packets may be damaged or delayed in transmission, data may be incorrectly routed, and host machines can be mis-configured.

The different modules that comprise the IntelliServer's networking software keep records called *network statistics*. They count the number of packets which cross into their territory, noting especially those that are disfigured or maimed, or which present any difficulties for the local authorities. Then, when the system administrator *(this would be you)* notices a problem with the network, you can review these statistics using the **`netstat`** command. By seeing which modules report difficulties and which do not, you can sometimes get clues where to look for the problem.

The netstat command comes in 9 variations:

Table 12-2: Netstat Variations

Command	Module	Typical traffic
`netstat icmp`	ICMP	Ping
`netstat udp`	UDP	Syslog messages, domain name resolution, RADIUS
`netstat tcp`	TCP	Telnet, rlogin, reverse-TCP connections
`netstat ip`	IP	All network traffic through the IntelliServer
`netstat route`	Routing section of IP	
`netstat sonic`	Ethernet Driver	All Ethernet traffic
`netstat ppp`	PPP	Traffic over any PPP links
`netstat slip`	SLIP	Traffic over any SLIP or CSLIP links
`netstat connections`	Lists TCP and UDP connections and their status.	TCP, UDP
`netstat all`	Lists all of the above statistics, in the order listed in this table.	

Netstat ICMP

Example 12-12 shows sample output from the command, **`netstat icmp`**. It counts the various types of ICMP messages sent and received by the IntelliServer, as well as noting how many of these packets might have had various things wrong with them.

Example 12-12: Netstat Icmp

```
netstat icmp
```

```
-- icmp --
   0 total icmp errors
   0 bad packet was icmp
 117 sent echo reply
   0 sent dest unreachable
   0 sent source quench
   0 sent shorter route
  42 sent echo request
   0 sent time exceeded
   0 sent ip header bad
   0 sent timestamp request
   0 sent timestamp reply
   0 sent information request
   0 sent information reply
   0 sent address mask request
   0 sent address mask reply
   0 rcvd bad icmp code
   0 rcvd packet too short
   0 rcvd bad checksum
   0 rcvd min/max length error
   0 rcvd responses generated
  42 rcvd echo reply
   0 rcvd dest unreachable
   0 rcvd source quench
   0 rcvd shorter route
 117 rcvd echo request
   0 rcvd time exceeded
   0 rcvd ip header bad
   0 rcvd timestamp request
   0 rcvd timestamp reply
   0 rcvd information request
   0 rcvd information reply
   0 rcvd address mask request
   0 rcvd address mask reply
```

The ICMP statistics you are most often concerned with are the following:

- **Sent echo request** — each time the IntelliServer tries to *ping* a host.
- **Rcvd echo reply** — each time a reply is receive from the other host in response to the *ping* that was sent by the IntelliServer.
- **Rcvd echo request** — each time a host sends a ping *to* the IntelliServer.
- **Sent echo reply** — each time the IntelliServer replies to the *ping* request.

If some host has been trying to *ping* the IntelliServer but has been getting no reply, you can see whether echo requests are being received, and whether it has sent the replies.

Netstat UDP

If there is anything wrong with UDP packets, there is not much to check.

Example 12-13: Netstat UDP

```
netstat udp
```

```
-- udp --
   102610 input packets
     5616 output packets
        0 bad header pullup
        0 bad checksum
        0 bad length
```

Example 12-14: Netstat Tcp

```
netstat tcp
```

```
-- tcp --
    3 connections attempted
    6 connections accepted
    9 connections established
    0 connections dropped
    0 connection attempts dropped
   14 connections closed
 2849 pkts timed for round trip
 2845 round trip time updated
  160 delayed acks sent
    0 links dropped
    0 retransmit timeouts
    0 persist timeouts
   25 keep alive timeouts
   25 keep alive probes sent
    0 keep alive - links dropped
 3028 sent total packets
 2840 sent data packets
28775 sent data bytes
    0 data pkts retransmitted
    0 data bytes retransmitted
  177 sent ack only pkts
    0 sent window probes
    0 sent urgent only pkts
    0 sent window update only pkts
   11 sent syn
```

Example 12-14: Netstat Tcp

```
 4900 rcv total packets
 2730 rcv pkts in sequence
 3596 rcv bytes in sequence
    0 rcv bad checksums
    0 rcv bad offsets
    0 rcv too short
    0 rcv duplicate pkts
    0 rcv duplicate bytes
    0 rcv partially duplicate pkts
    0 rcv partially duplicate bytes
    8 rcv out of order pkts
    0 rcv out of order bytes
    0 rcv pkts after window
    0 rcv bytes after window
    0 rcv pkts after close
    0 rcv window probes
   33 rcv duplicate acks
    0 rcv acks for unsent data
 2847 rcv ack packets
28787 rcv bytes acked
    3 rcv window update pkts
    0 connections that lingered
    0 lingers aborted by signal
    0 linger timer expired
    0 linger timer canceled
    0 buffer wait calls
```

Netstat TCP

TCP protocol is a more complicated territory than others, so there is more to report:

- The number of connection attempts, successful and otherwise.
- The number of good packets of various types sent and received.
- The number of received packets that appear damaged in some way (too short, bad checksum, etc.).
- Other network anomalies: (duplicate packets, retransmissions, etc.).

Unless you are quite familiar with TCP protocol, many of these statistics will be meaningless to you. If you have to call our tech support with a network problem, this is good information to have.

Netstat IP

About the only thing the IP layer will detect wrong, is a problem with the IP header itself. Barring any of those problems, you get a count of the total number of IP packets sent and received (including all the interfaces).

Example 12-15: Netstat IP

```
netstat ip
```

```
-- ip --
107512 total pkts rcvd
  8686 total pkts sent
     0 bad checksum
     0 cant pullup hdr
     0 less data than hdr says
     0 bad hdr len
     0 data len less than a hdr
     0 frags rcvd
     0 frags dropped
     0 frags timed out
     0 pkts forwarded
     0 cant forward to dest
     0 redirects frwrd same net
     0 pkts lost
```

Netstat Route

This command gives statistics about how packets have been routed. Each separate process keeps track of the most recent routes it has found for the packets it is sending, so that it will not have to look through the routing table for each packet. This is what *lookups that were cached* signifies.

This report also summarizes the results of routing table lookups according to the type of route:

- Routes to hosts.
- Routes to networks.
- The default route (called “wildcard” here).
- Lookups where there was no route found.

Example 12-16: Netstat Route

```
netstat route
```

```
-- route --
    0 bad redirects
    0 routes created by redirects
    0 routes modified by redirects
 8739 lookups that were cached
    1 lookups that failed
    0 lookups matching host route
   37 lookups matching net route
    0 lookups matching wildcard route
```

Netstat Sonic

This command shows statistics about the Ethernet Interface, such as the total number of packets and bytes sent, and various anomalies.

Example 12-17: Netstat Sonic

```
netstat sonic
```

```
-- sonic --
   206544 input packets
 33514662 input bytes
     6211 output packets
   520414 output bytes
        0 rcv buffers exhausted
        0 transmit deferred
        0 excessive deferral
        0 collisions
        0 excessive collisions
    97793 unbound pkts dropped
        0 too many fragments
        0 upstream queue full
```

Collisions and deferrals generally indicate a problem with the network's physical layer. Unbound packets are not necessarily bad, they are just packets (possibly broadcast) sent to the IntelliServer using a protocol the IntelliServer doesn't support.

Netstat PPP, SLIP

The **`netstat ppp`** and **`netstat slip`** commands give nearly identical information, except for checks on some fields that are present in PPP headers but not in SLIP (or CSLIP), and FCS (Frame Check Sum) errors because the FCS is used only by PPP.

Example 12-18: Netstat PPP

```
netstat ppp
```

```
-- ppp --
  141 in pkts
20768 in bytes
    0 in pkts too short
    0 in bytes flushed
    0 in could not alloc
    0 in missed all stations
    0 in missed ifield
    0 in bad proto field
    0 in short header
    0 in bad fcs
  432 out pkts
42132 out bytes
    0 out pkts too short
    0 out could not alloc
```

Example 12-19: Netstat SLIP

```
netstat slip
```

```
-- slip --
   1021 in pkts
  90012 in bytes
      0 in pkts too short
      0 in bytes flushed
      0 in could not alloc
   1233 out pkts
  97102 out bytes
      0 out pkts too short
      0 out could not alloc
```

The total counts for *all* PPP or SLIP (including CSLIP) interfaces is shown.

Netstat Connections

The final member of the netstat nonalogy is different from the others — it is not history, but geography. It reports the state of all current TCP connections, and shows which service ports the IntelliServer is listening on for connections.

The service port appears at the end of each IP address. If the service port appears in the IntelliServer's services table (page 230), the name of the service is used instead of the number.

The first line in Example 12-20 shows a connection started when someone telnetted into the IntelliServer. The next three lines show that you are listening for connections for telnet, rcp, and finger services, and the fifth line shows a session created when an IntelliServer user rlogin'ed into another host.

The remaining lines show connections created when various processes in the IntelliServer send syslog messages to the syslog host, here 160.77.99.30.

Example 12-20: Netstat Connections

```
netstat connections
-- connections --
proto   local address          remote address         state
tcp     160.77.99.200.telnet   160.77.99.27.1202   established
tcp     *.telnet               *.*                   listen
tcp     *.rcp_print            *.*                   listen
tcp     *.finger               *.*                   listen
tcp     160.77.99.200.1286     160.77.99.28.rlogin established
udp     160.77.99.200.1027     160.77.99.30.syslog
udp     160.77.99.200.1026     160.77.99.30.syslog
```

PPP (And Slip) Statistics

One further tool is available to report the status of PPP, SLIP, and CSLIP links, the **`pppstat`** command.

If you type **`pppstat all`**, then the status of all interfaces, *ppp00* through *ppp31*, are listed. In Example 12-21 the interfaces after *ppp03* were left off because they were all disabled anyway.

Example 12-21: Pppstat All Command

`pppstat all`

```
gripen# pppstat all
ppp00, yves, inbound, SLIP, disconnected
        5 redial delay     120 inactivity timer
        0 in bytes           0 out bytes
        0 in pkts            0 out pkts
        0 in errors          0 out errors
ppp01, forpppp, inbound, PPP, disconnected
        5 redial delay     120 inactivity timer
        0 in bytes           0 out bytes
        0 in pkts            0 out pkts
        0 in errors          0 out errors
ppp02, disabled
ppp03, disabled
...
```

Example 12-22: Pppstat Command

`pppstat ppp01`

```
ppp01, forppp, inbound, PPP, disconnected
        5 redial delay     120 inactivity timer
        0 in bytes           0 out bytes
        0 in pkts            0 out pkts
        0 in errors          0 out errors
```

If you supply the name of an interface *or* the name of a remote profile, the status of that interface alone is given. In Example 12-22, the command **`pppstat forppp`** would have given the same output.

The first line is probably the most important. In these examples the interfaces are shown as being *disconnected*. Since these are inbound interfaces, it probably means that the user hasn't dialed in to bring them up. If it had attempted to do so, this would be telling us it didn't succeed.

CHAPTER 13

IntelliFeatures

In this chapter, you learn to configure and use IntelliFeatures. These include the following:

- IntelliPrint, which supports network access to printers attached to your terminal's *AUX* port.
- IntelliView, which supports independent multiple sessions on terminals with multi-page displays.
- IntelliSet, which allows you to configure and "lock in" physical characteristics of selected serial ports.

IntelliFeatures — Overview

There are three types of IntelliFeatures:

- IntelliView - allows you to have multiple login sessions on a single multi-page terminal.
- IntelliPrint - allows you to send printer output to your terminal's Aux port.
- IntelliSet - allows you to define special serial protocol options.

A collection of IntelliFeatures specifications is called a *profile*. Generally, a profile corresponds to a particular kind of terminal or serial device you are using. For example, the Aux port on a Wyse 60 terminal is accessed differently from the Aux port on a VT100, so these two terminals would require different IntelliPrint profiles.

There are separate types of profiles for IntelliView, IntelliPrint, and IntelliSet. When you have defined a profile and given it a name, you can assign that profile to one or more ports by supplying the profile's name during port configuration (see page 89). In this way, you only have to specify a particular *kind* of device once, regardless of how many you are using.

IntelliView

To support IntelliView, your terminal must be capable of maintaining two or more *pages* of display memory. Often, the number of pages supported depends on the display mode. For example, a Wyse 60 in 132x43 (Wy60 emulation) mode might support only a single *page*, where in 80x25 (Wy60 emulation) it might support two pages, and in "Econ-80" mode it might support three or more pages. If you are not sure about your particular terminal, you may need to read its manual and (failing to find the information there) experiment.

Multi-page terminals change which page they are displaying in response to a special command code sent from the IntelliServer. Different terminals use different codes, so the IntelliServer needs to know what this will be. You also must pick one or more functions keys that tell the IntelliServer you want to switch sessions. When the IntelliServer sees that one of these keys has been pressed, it knows it must direct *subsequent* keystrokes to a different virtual session, and also that it must send the appropriate screen-switch sequence. All this information about screen-switch commands and function keys (here called *hot-keys*) is stored as part of the IntelliView profile.

IntelliView allows you to run a separate session on each of the terminal's display pages. On one page (or "screen") you might be logged into one host, while on another page you are logged into a different one. You would use designated function keys to flip between the two sessions.

Because each separate session uses additional memory within the IntelliServer, the total number of sessions you can run at the same time is limited, and depends on whether you are using a 2-Meg or 4-Meg IntelliServer, whether or not some of the ports are running PPP or SLIP connections, and other configuration details. As a rule of thumb, the 2-Meg IntelliServer can support up to 96 sessions, and the 4-Meg IntelliServer can support up to 128. The RAS 2000 supports up to 128 sessions also.

In order to use the terminal's function keys to flip between pages, the IntelliServer needs to be told what codes these keys send. Generally this is not a problem, but there are some applications which send commands to the terminal to re-program its function keys to specific values. If this includes a function key you want to use for IntelliView, you may find yourself unable to switch screens once you have entered that application. In that case, you will need to investigate re-configuring that application or using different function keys for IntelliView.

IntelliPrint

To support IntelliPrint, your terminal must have an AUX port, and support commands that will:

- route subsequent data to that port (instead of to the display), and
- route subsequent data to the display (and no longer to the Aux port).

These commands are special data sequences that would be sent from the IntelliServer before and after any data directed to the printer. An IntelliPrint profile contains these sequences, as well as other information for setting the relative priorities between data for the printer and data for display.

When you have associated an IntelliPrint profile with a serial port, you can configure your network hosts to send data directly to that printer, independently of what is happening on the terminal. For example, your terminal could be logged into a host and running an application. While in the application, you decide you want to print a report. The report is sent to your system's print spooler, which you have configured to send output to the printer attached to your terminal. While the output is printing, you are still able to use your terminal.

It is also possible to connect a printer directly to one of the IntelliServer's serial ports, but in that case the port would be configured differently, and you would not be using *IntelliPrint*.

Chapter 18, *Reverse TCP and Printing*, explains how to configure ports for printing and also gives some recommendations on how to configure your network hosts.

IntelliSet

IntelliSet profiles include a mixed bag of specifications that can be thought of as an extension of port configuration. In fact, there are many parameters in common. The difference is that features you specify with IntelliSet *override* the settings in Port Configuration.

IntelliSet parameters also resist any attempts by applications to change them. For example, a telnet session normally puts a serial port into raw mode, disabling any output processing that might have been specified under port configuration. If, however, this had been specified in an IntelliSet profile, the output processing continues.

Configuration Forms

The forms for configuring IntelliFeatures Profiles are reached through the IntelliFeatures Menu shown here in Screen 13-1.

Screen 13-1: IntelliFeatures Menu

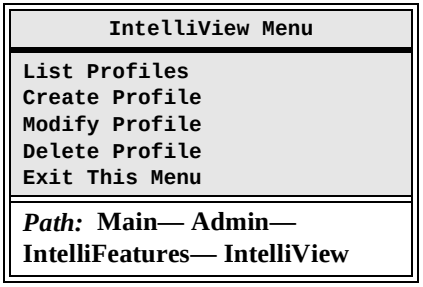
```
      IntelliFeatures Menu
IntelliView
IntelliSet
IntelliPrint
Exit This Menu
```

Path: **Main— Admin— IntelliFeatures**

Each selection on this menu represents a different set of *Multi-Record Forms* (see page 50). For example, a single IntelliView profile contains a screen full of information. There can be several different IntelliView profiles: you assign each a name that can later be used to refer to it. The same is true of IntelliPrint and IntelliSet profiles, so they are all maintained in the same manner.

Pick any of the selections on the IntelliFeatures Menu, and you will get a menu similar to the one shown for IntelliView in Screen 13-2.

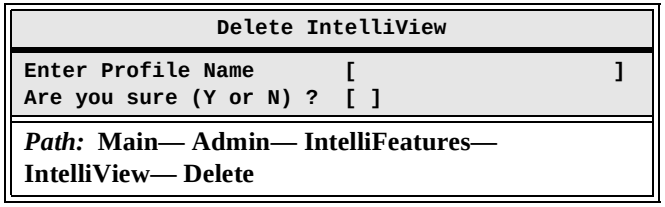
Screen 13-2: IntelliView Menu



Whether you have chosen IntelliView, IntelliSet, or IntelliPrint, the process is identical. The next menu gives you the option of creating a new profile, or of listing, modifying, or deleting existing profiles. When you choose to **List** profiles, the names of all existing profiles of this type are displayed.

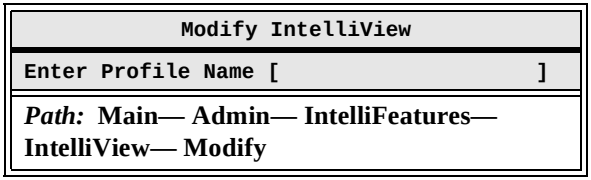
When you choose to **Delete** a profile, you are prompted to enter its name, and then to confirm, as shown in Screen 13-3.

Screen 13-3: Delete (IntelliView) Profile



When you choose to **Modify** something, you are prompted to enter its name, as shown in Screen 13-4.

Screen 13-4: Modify IntelliView (Prompts for Name)



Perhaps you do not recall any of the existing profiles' names. Then enter **ctrl-u**. This brings up a screen showing all the existing profiles of this type. Then, use your terminal's arrow keys to highlight the one you want and press **enter** to select it. After you enter the name of the profile you want to modify, the appropriate configuration form is displayed showing that profile's current settings.

When you choose to **Create** a profile, the appropriate configuration form is displayed with the profile name left blank. You fill it in when you are filling in the other fields. Several input areas may already contain default settings.

Commands

The **profile** command can also be used to create, modify, and display IntelliFeatures profiles, as shown in the examples below:

Example 13-1: Creating a New IntelliFeatures Profile

```
add profile iview|iprint|iset name settings...
add profile iview|iprint|iset name from oldname
```

```
# add profile iprint alfred printon ^A printoff ^B
# add profile iprint befred from alfred
# add profile iset sigfried addcr enabled
```

In Example 13-1 a new IntelliPrint profile called *alfred* was created, and then another called *befred* with its settings copied from alfred. Then, a new IntelliSet profile called *sigfried* was added.

Example 13-2: Modifying an IntelliFeatures Profile

```
set profile iview|iprint|iset name settings...
set profile iview|iprint|iset name from oldname'
```

```
# set profile iprint befred printoff ^C
```

In Example 13-2 one of *befred's* IntelliPrint parameters was changed. Note that it is possible for different *types* of IntelliFeatures profiles to have the same name. For that reason, it is always required that you specify **`iview`**, **`iprint`**, or **`iset`** to specify whether you are working on an IntelliView, IntelliPrint, or IntelliSet profile.

Example 13-3 shows commands to display IntelliFeatures profiles.

Example 13-3: Displaying IntelliFeatures Profiles

show profile iview|iprint|iset *name*
show profile iview|iprint|iset

```
# show profile iview
wy60.2t
wy60.3t
dumb.8t

# show profile iview wy60.2t
Profile Name:                 wy60.2t
Toggle Sequence:              ^AK^M
Hot Key Timeout:              0
Number of Screens Configured: 2

 Scr Hot Key Sequence  Output Sequence
  0  \200              \Ew0
  1  \201              \Ew1
  2
  3
  4
  5
  6
  7

# show profile iprint wy60
Profile Name:         wy60
Start Print Sequence: ^[d#
End Print Sequence:   ^t
Print Delay:          10
Print Interval:       5
NL to CR/NL:          Yes
Expand Tabs:          No

# show profile iset sigfried
Profile Name:  sigfried
Custom Baud 1: 0
Custom Baud 2: 0
Outgoing Baud: -------
Incoming Baud: -------
Size:          -
Parity:        -----
Stop:          ---
Inflow:        -----------
Outflow:       -----------
Modem:         ---
NL to CR/NL:   Yes
Expand Tabs:   ---
```

If you don’t supply the name of a specific profile, then the IntelliServer displays all the profile names of a given type. In our example, there are three IntelliView profiles, called *wy60.2t, wy60.3t,* and *dumb.8t.* If you do supply the name of a profile, then all the settings for that profile are displayed. The meaning of all these settings will be discussed in the next sections.

Configuring IntelliPrint

The IntelliPrint configuration form is shown in Screen 13-5. This example shows the configuration for a profile called **wy60**, which is included in the factory defaults.

Screen 13-5: IntelliPrint Configuration Form

```
          Create/Modify IntelliPrint

Profile Name          [wy60                 ]
Start Print Sequence  [^[d#             ]
End Print Sequence    [^t               ]
Print Delay           [10 ]
Print Interval        [5  ]
NL to CR/NL           [Yes]
Expand Tabs           [No ]
```

Path: **…IntelliFeatures— IntelliPrint— Modify (or Create)**

If you are creating a new IntelliPrint profile, you enter the name on the first line. If you are modifying an existing one, the name is already there, and you are not allowed to change it.

The following section explains each of the settings, as well as how to change them using commands:

Menu:	**Start Print Sequence [^[d#** **]**
Command:	**set profile iprint** *name* **printon** *sequence*

To make a terminal send data to its Auxiliary port, the IntelliServer must first send a command saying *"send all subsequent data to the aux port, until I say otherwise"*. Not in so many words, of course, but some sort of special data sequence must be used. In this example, for the Wyse 60 the sequence consists of the three characters: **escape d #**. The escape character is represented here by the symbols **^[**. It is often necessary to represent the *escape* code and other unprintable bytes in IntelliFeatures profiles (refer to Table 5-4 on page 94 to help you remember how to do this).

How do you find out what codes to use for your particular terminal? There are several listed at the end of this chapter. If the answer is not there, consult your terminal's manual or programmers' guide. What is called the *start print sequence* is often called *start transparent print*. The print sequences to use are determined by which *terminal* you are using, not by your choice of printer. The printer never "sees" these sequences, they are interpreted by the terminal.

Menu:	**End Print Sequence [^*t#*]**
Command:	**set profile iprint** *name* **printoff** *sequence*

When the IntelliServer has been sending data to the printer and now wants to send data to be displayed again, it must send a command to the terminal. For a Wyse 60, the command to do this is the single character, **ctrl-t**.

This brings to mind a concern:

Note: If you want to send data to a printer attached to your terminal, that data had better *not* contain the terminal's *end print* sequence. When the terminal sees such data, it will *not* send it blindly to the printer. It will rightly interpret it as a command and send the following data (that you had intended for the printer) to the display.

This is usually not a problem with printed text or even with PostScript output, because control codes usually do not appear. It is more likely to be a problem with a printer using various *native* graphics modes.

The *Print Delay* tells how long the IntelliServer must wait after any display output, before it sends any data for the printer. (The command uses different terminology: there it is called *start*, but the meaning is the same). The delay is measured in tenths of a second.

Menu:	**Print Delay [*10*]**
Command:	**set profile iprint** *name* **start** *time*

Why would you want such a delay? Data for display often contains control sequences for cursor-addressing, highlighting, and so on. These sequences consist of two or more bytes of data and most terminals get confused if such a sequence is interrupted by a command to start transparent printing. The delay ensures that all the bytes of any control sequence have a chance to be completely sent before a command to start printing is sent.

The *Print Interval* defines a delay to be inserted between successive blocks of print data. (The command calls this the *delay*, not to be confused with the *Print Delay* which the command calls *start*.) The *Print Interval* is measured in tenth-seconds.

Menu:	**Print Interval [*5*]**
Command:	**set profile iprint** *name* **delay** *time*

Why a delay between successive blocks of print data? To make the IntelliServer send its printer data more slowly. To see why this is important, you need to understand how *flow control* works. If the IntelliServer sends data to the terminal more quickly than the terminal can handle it, the terminal notifies the IntelliServer to stop sending data. You specify how this is to be done using your terminal's setup screen, and in the IntelliServer's port configuration. Similarly, if the terminal sends data to the printer more quickly than the printer can print it, the printer notifies the terminal that *it* must stop sending data. This is specified in a different place in your terminal's setup screen, and on your printer by using a configuration menu or switches.

Most terminals can display faster than most printers can print. If you were to send the print data as quickly as you would send display data, the printer's buffers would fill up and it would tell the terminal to stop sending data. Unable to send data to the printer, the terminal's buffers would fill up, and it would notify the IntelliServer to stop sending data. This is not bad in itself and if everything is configured appropriately, no data will be lost. There is just one problem. If the terminal's buffers get full, it means no data can be sent to the terminal, including data for display. Depending on your printer, this means that there could be long periods during which your terminal would be unusable for display purposes, because it would be clogged with print data.

To prevent this from happening, experiment with different *Print Interval*'s to find one that just starts to slow the rate of actual printing. When this happens, you know that you are now sending data just slower than the printer is capable of printing. (For faster rates, increasing the print interval seems to have no effect, because the printing speed is limited by the speed of the printer).

These define whether the IntelliServer adds Carriage-Returns before linefeeds and expand tabs in data to be sent to the printer.

Menu:	**NL to CR/NL [*Yes*]**
Command:	**set profile iprint** *name* **addcr enabled|disabled**

Menu:	**Expand Tabs [*No*]**
Command:	**set profile iprint** *name* **tabs yes|no**

In most cases it is better to keep these options disabled and configure your host software to perform whatever processing is appropriate. This may include configuring your application, setting print-spooler options, or selecting *iservd* or *iservcat* options. (See chapter 18, *Reverse TCP and Printing*, for more about the *iservd* and *iservcat* utilities). If you enable processing here, it applies to any output sent for printing. It is usually safe to use these options when you are printing ordinary text or even PostScript files. However, when using some printers in native graphics modes, a *tab* character may not always *mean* a tab, and a *linefeed* character might not signify a linefeed. They might be part of some graphics command. So it would be inappropriate to alter them. Since the IntelliServer does not know what your data is supposed to be, it cannot judge whether your data should be processed or not. Back on the host, you *do* know what data is what, because different types of output are generated at different times by different applications, and different types of processing can be applied to each.

If you are trying to print text files and they are coming out looking strange, you could try turning on these options. For example, if you were expecting this:

```
alpaca  llama  vicuna
cat     lion   tiger
a       b      c
```

and instead got this:

```
alpaca llama vicuna
                    cat lion tiger
                                    a b c
```

then you should try NL->CR/NL to eliminate the *barber-pole* effect, and tab expansion to line up the columns.

The IntelliView configuration form is shown here in Screen 13-6. This example shows the configuration for a profile called **wy60.2t**, which is included in the factory defaults.

Screen 13-6: IntelliView Configuration Form

```
          Create/Modify IntelliView

Profile Name                    [wy60.2t     ]
Toggle Sequence                 [^AK^M       ]
Hot Key Timeout                 [0  ]
Number of Screens Configured    |2|

   Hot Key Sequence      Output Sequence
0 [\200              ] [\Ew0                ]
1 [\201              ] [\Ew1                ]
2 [                  ] [                    ]
3 [                  ] [                    ]
4 [                  ] [                    ]
5 [                  ] [                    ]
6 [                  ] [                    ]
7 [                  ] [                    ]
```

Path: **…IntelliFeatures— IntelliView— Modify (or Create)**

This IntelliView profile supports a Wyse 60 terminal with two pages of screen memory. Three *hot keys* are defined, all of them function keys:

- Press **F12** to switch between screens. When you press it, it sends the three-byte sequence **^A K ^M**.
- Press **Ctrl-F1** to switch to screen 0 (the main screen). If your Wy60 is configured for 8 bit data, this key sends a single byte, octal value 200.
- Press **Ctrl-F2** to switch to screen 1. This key sends a single byte of octal value 201.

If you are creating a new IntelliView profile, you enter the name on the first line. If you are modifying an existing one, the name is already there and you are not allowed to change it.

The following section explains each of the settings, as well as how to change them using commands:

Menu:	**Toggle Sequence [^AK^M]**
Command:	**set profile iview** *name* **toggle** *sequence*

The *toggle key* is a function key that switches you to the *next* screen, whichever one you are on. You don't have to define a *toggle* key; you can instead define one function key to select the first screen directly, another function to select the second, and so on. You can also define a *toggle key* **and** a separate function key for each screen. But you must do one or the other, else there will be no way to select a particular screen and what good would it be then?

Many administrators find it best to define only the toggle key, without defining separate keys to select each screen. Why? Because each function key defined for this purpose is trapped by the IntelliServer and is **not** sent upstream to the application with the rest of the keyboard data. That means *this* function key had better not be one you need to run your applications. The fewer keys defined, the less the chance of a conflict.

Once you have decided which function key to use, how do you find out what control sequence it sends? If the information you need is not at the end of this chapter, the most straightforward way is to consult the terminal manual. If that is not available and you are still stuck, trial-and-error methods involving the UNIX **od** command (or the functional equivalent on other operating systems) can sometimes help. If it gets to that point and you don't know how to proceed, call our tech support department for help.

Once you know what codes the function key sends, enter them using the notation from Table 5-4, "Printing the Unprintable," on page 94.

Menu:	**Hot Key Sequence Output Sequence** **0 [\200] [\Ew0]** **1 [\201] [\Ew1]**
Command:	**set profile iview** *name* **scan0** *sequence* **set profile iview** *name* **out0** *sequence* **set profile iview** *name* **scan1** *sequence* **set profile iview** *name* **out1** *sequence*

The *hot key sequence*, or **scan**, defines the codes that are sent by function keys that are to switch you to a specific screen. There can be up to eight of these hot key sequences defined. In the commands they are represented by **scan0** for the first screen, **scan1** for the next, up to **scan7** for the last. If you define a *toggle* key (see the preceding section) you don't have to define any hot key sequences for specific screens, but you can.

The *output sequence*, or **out**, defines the command that the IntelliServer needs to send to your terminal to make it switch to its corresponding screen (or *page*, as sometimes it is called). In the commands, the codes for the eight screens are represented by the keywords **out0**, **out1**, through **out7**. You must define an output sequence for each screen you wish to support.

If there is no information about your terminal at the end of this chapter, consult your terminal manual for the appropriate key codes and command sequences. Refer to Table 5-4 on page 94 to see how you must enter control codes.

The *Hot Key Timeout* specifies a maximum time that the IntelliServer waits to receive an entire hot-key sequence, once it sees that a sequence might have started. The time is in tenth-seconds and a zero means that there is no timeout. This applies to the toggle key (if defined) as well as the hot keys for individual screens.

Menu:	**Hot Key Timeout [*0*]**
Command:	**set profile iview** *name* **timeout** *time*

If the start of a hot key sequence comes into the IntelliServer, but is not entirely received before the timeout period, then the raw data is sent upstream. When the rest of the sequence comes in, it is treated as ordinary data.

This option is provided because there are some terminals whose function keys send out sequences that begin with data you might like to enter. For example, suppose your terminal had a function key that sent **escape A**, and you wanted to use that as your toggle key. Fine. But now suppose your application uses the **escape** key (which presumably sends just an **escape** code) to perform some function. What happens when you hit the **escape** key? First, the IntelliServer recognizes that this *might* be the start of a sequence for the toggle key. So it does not send this data upstream. Not yet. It needs to see if the next data is an A. If it is, it will switch screens and send no data to the application. Suppose it's an X. Well, if **escape X** is not any other hot key, it will send the data upstream to the application.

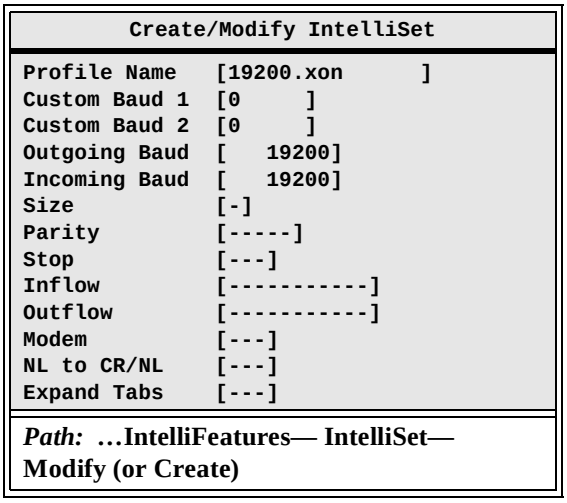
What if the user wanted just to type the escape key, however? If there were no timeout defined, and he didn't press any other keys after the escape key, the IntelliServer would still be waiting…hoping…that the next key might satisfy its craving for a hot-key.

With a timeout defined, the IntelliServer waits only so long for that **A** to come in. Then, it sends the escape code it had been saving, up to the application.

If you define a timeout value, the key is to choose one long enough so that under normal circumstances an entire hot key sequence would always arrive within that time, but short enough not to annoy a user waiting for the timeout to expire after pressing the escape key. When the hot key is a function key which automatically sends a string of data, a couple tenths of a second is usually enough. It is not necessary that your hot key be an actual key, however. Some administrators like to define an arbitrary key sequence (say, control-B, X or something like that) for a hot-key — particularly if the terminal does not have many function keys and they are all used for other purposes. In this case, the user would have to use multiple keystrokes to type the sequence, and it takes longer. If these sequences start with a code that is never used in another context, no timeout is needed. Otherwise, it needs to be quite long, perhaps as long as a second or two.

The IntelliSet Configuration Form is shown in Screen 13-7, and illustrates profile *19200.xon* which is supplied with the factory defaults. Except for the profile name and the two custom baud rates, all the other fields are *pick-lists*.

Screen 13-7: IntelliSet Configuration Form



In this example, many of the input areas contain dashes. These indicate that the corresponding parameter is not specified by this IntelliSet profile. When a value appears instead of dashes, then that parameter *is* specified. When a parameter is specified in an IntelliSet profile, that value overrides anything in the port configuration and it cannot be changed by whatever application (telnet, rlogin, etc.) might be running.

You can use these to specify a line speed that is not included in the standard table. These numbers have no effect unless *Outgoing Baud* or *Incoming Baud* is set to *Custom1* or *Custom2*.

Menu:	**Custom Baud 1 [*42000*]** **Custom Baud 2 [*0*]**
Command:	**set profile iset** *name* **custom1** *speed* **set profile iset** *name* **custom2** *speed*

Two custom rates are provided in case you want to set the incoming line speed to one custom rate and the outgoing speed to a different custom rate.

This allows you to set separate line speeds for the transmitter (*outgoing*) and the receiver (*incoming*). You may of course set them to be the same.

Menu:	**Outgoing Baud [*19200*]** **Incoming Baud [*19200*]**
Command:	**set profile iset** *name* **speed** *speed* **set profile iset** *name* **ispeed** *speed*

The speed must be one of the following:

-	150	1800	7200	57600	Custom1
50	200	2000	9600	64000	Custom2
75	300	2400	19200	76800	
110	600	3600	38400	115200	
134.5	1200	4800	56000		

The dash indicates that you don't want this IntelliSet profile to lock the baud rate, the port will use whatever rate is specified elsewhere. *Custom1* or *Custom2* indicates that the rates specified under *Custom Baud 1* or *Custom Baud 2* are to be used.

These parameters allow you to lock the character size, parity, and number of stop bits.

Menu:

```
Size      [8]
Parity    [even]
Stop      [1]
```

Command:

```
set profile iset name charsize character-size
set profile iset name parity parity
set profile iset name stopbits number
```

The possible choices for character size are:

-	5	6	7	8

The choices for parity are:

-	None	Odd	Even	Space	Mark

The choices for number of stop bits are:

-	1	1.5	2

The dash indicates that you don't want to specify the corresponding parameter here.

This allows you to specify and lock the input flow control. The IntelliServer uses input flow control when it is receiving data: if data comes in faster than the IntelliServer can process it, the IntelliServer needs to signal the sender to stop transmitting for a while. (See also "Flow Control" on page 60).

Menu:

```
Inflow   [IXOFF ]
```

Command:

```
set profile iset name inflow flow-control
```

The possible values are:

-	None	RTS	RTS+IXOFF
	IXOFF	DTR	DTR+IXOFF

The dash indicates that this IntelliSet profile will not affect input flow control. *IXOFF* indicates that the IntelliServer should send an XOFF character when its receive buffers become nearly full, and send an XON character when they again have room for more data. *RTS* indicates that the IntelliServer should drop the RTS signal when the buffers are nearly full, and raise it when they have room. *DTR* indicates that the IntelliServer should drop DTR when the buffers are nearly full, and raise it when there is room for more data. *RTS+IXOFF* and *DTR+IXOFF* indicate that a combination of actions are taken.

The *Outflow* parameter allows you to specify and lock the output flow control. When the IntelliServer is sending data to a device, and that device cannot process data quickly enough, it must signal the IntelliServer in some way to tell it to stop transmitting. Output flow control specifies what conditions will cause the IntelliServer to stop sending data.

Menu:	**Outflow [*IXON*]**
Command:	**set profile iset** *name* **outflow** *flow-control*

The possible values are:

-	None	IXON	IXON+CTS
	CTS	IXANY	IXANY+CTS

The dash indicates that this IntelliSet profile will not affect output flow control. *CTS* indicates that the IntelliServer will not transmit unless the CTS input is asserted. *IXON* indicates that the IntelliServer should stop transmitting when it receives an XOFF character and resume when it receives an XON. *IXANY* is the same, except that after the XOFF character has disabled transmission, receiving *any* character (not just an XON) will re-start it.

The *Modem* parameter specifies whether the port will be treated as a *modem* port. Modem ports are affected by the DCD (carrier detect) signal, while non-modem ports ignore it. The dash indicates that this IntelliSet profile will have no effect on whether the port is a modem or non-modem port.

Menu:	**Modem [*Yes*]**
Command:	**set profile iset** *name* **modem -|yes|no**

These parameters allow you to specify and lock output processing on this port. Carriage returns can be inserted before linefeeds to prevent *barber-pole* output as show on page 337. Tab expansion can be helpful if your output contains tab characters and your terminal doesn't understand tabs. The dash indicates that this IntelliSet profile should not affect output processing.

Menu:	**NL to CR/NL [*Yes*]** **Expand Tabs [*Yes*]**
Command:	**set profile iset** *name* **addcr -|yes|no** **set profile iset** *name* **tabs -|yes|no**

Popular IntelliFeatures Profiles

The tables in this section give the information you need to construct IntelliFeatures Profiles. Remember, command codes can often be entered in more than one way (see Table 5-4 on page 94 for more information).

IntelliPrint

Table 13-1 shows the terminal-dependent codes you need to enter to construct IntelliPrint profiles for different popular terminals.

Table 13-1: IntelliPrint Values

Terminal Type	Start Print	End Print
Ampex 230, 232	**\E\140**	**\E\141**
ANSI (like VT100)	**\EW**	**\EX**
IBM 3161, 3163, 3164	**^P^R**	**^P^T**
Televideo 925, 955	**\E\140**	**\E\141**
TI 931	**\EF1\E(**	**\E)**
VT52	**\EW**	**\EX**
VT100	**\E[5i**	**\E[4i**
Wyse 50	**^X**	**^T**
Wyse 60	**\Ed#**	**^T**
Wyse 75, 85 (like VT100)	**\EW**	**\EX**

IntelliView

To construct IntelliView profiles, Table 13-2 gives the screen-switch sequences you need, and Table 13-3 gives the codes sent by various function key and suggests at least one way of using them as hot keys. You may choose to assign different function keys for different purposes, in which case you would need to consult your terminal's reference guide for the function key codes not listed here. Table 13-3 suggest hot keys for switching directly to each screen as well as a toggle key.

You must define enough keys to be able to access each screen. If you assign a toggle key you don't need to assign keys for direct access. If you don't assign a toggle key, you need to assign a function key to access each screen. If you want, you can assign a function key for each screen, *and* a toggle key.

Table 13-2: IntelliView Output Sequences

Terminal Type	Number of Screens	Output Sequence, Screen 0	Output Sequence, Screen 1	Output Sequence, Screen 2	Output Sequence, Screen 3
Ampex 230	2	**\EJ**	**\EK**		
IBM 3151	2	**\E\040pA**	**\E\040pB**		
Relisys TR170	3	**\E[0z**	**\E[1z**	**\E[2z**	
Televideo 955	2	**\E1;0}**	**\E1;1}**		
Wyse 60	2	**\Ew0**	**\Ew1**		
Wyse 60 in Econ-80 mode	3	**\Ew0**	**\Ew1**	**\Ew2**	
Wyse 60 in Wyse 50 emulation, Econ-80 mode	7	**\Ew0**	**\Ew1**	**\Ew2**	**\Ew3** *and so on up to \Ew7*

Table 13-3: IntelliView Hot Key Sequences

Terminal Type	Toggle Key	Hot-Key Sequence, Screen 0	Hot-Key Sequence, Screen 1	Hot-Key Sequence, Screen 2	Hot-Key Sequence, Screen 3
Ampex 230	**^A\140^M** *shift-F1*	**^Aa^M** *shift-F2*	**^Ab^M** *shift-F3*		
IBM 3151	**\Ej^M** *F10*	**\Ek^M** *F11*	**\El^M** *F12*		
Televideo 955	**^Ak^M** *F12*	**^Ai^M** *F10*	**^Ak^M** *F11*		
Relisys TR170	**\E[X** *F12*	**\E[k** *ctrl-F1*	**\E[l** *ctrl-F2*	**\E[2** *ctrl-F3*	
Wy60 (in 8-bit mode)	**^AK^M** *F12*	**\200** *ctrl-F1*	**\201** *ctrl-F2*	**\202** *ctrl-F3*	**\203** *ctrl-F4*

CHAPTER 14

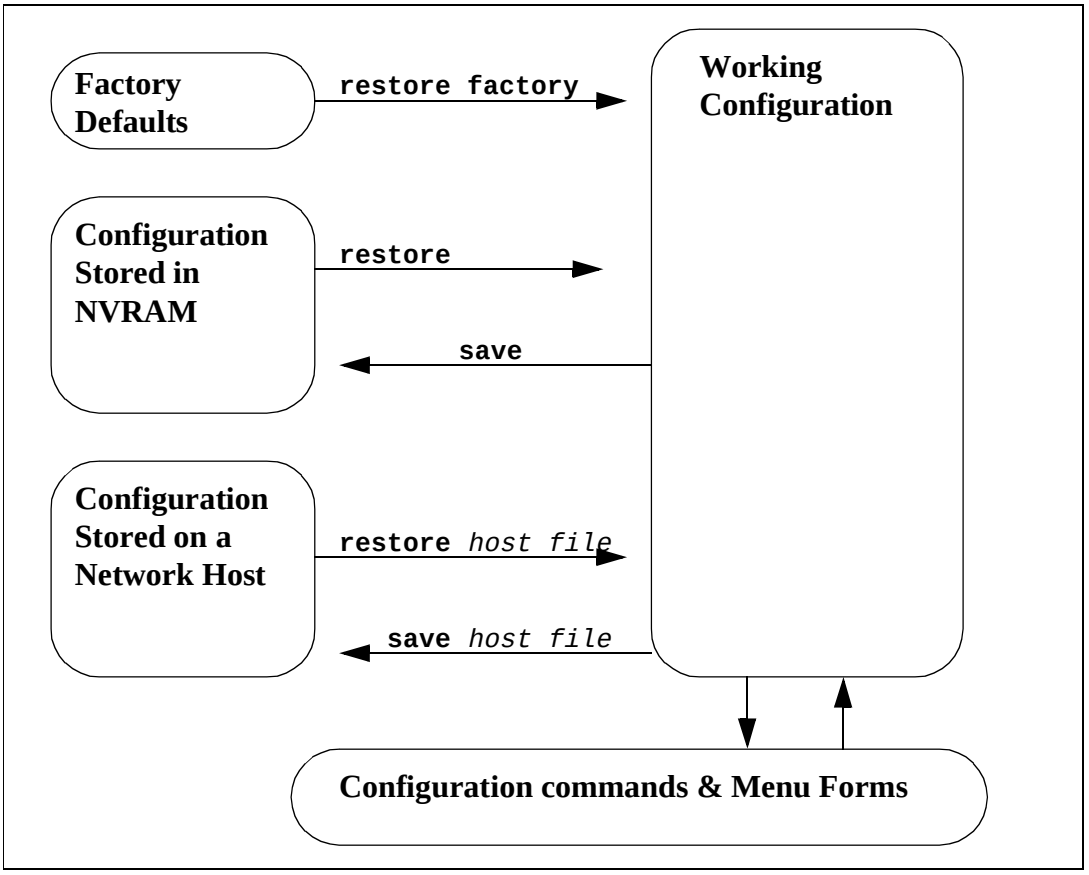
Saving and Restoring Configurations

In this chapter you learn how the IntelliServer's configuration may be stored and how stored configurations can be retrieved. In addition, discussed are how to:

- Store and retrieve configurations from local NVRAM.
- Store and retrieve configurations from host files.
- Configure your host to support configurations.
- Temporarily restore factory default configurations.
- Use BOOTP to completely configure an IntelliServer.

The following illustration summarizes the action of different **`save`** and **`restore`** commands:

Example 14-1: What the Save and Restore Commands Do



The **working configuration** contains all the settings currently in effect. When you change the configuration using menu forms and commands, it is the **working configuration** that changes. These changes are not stored in NVRAM or to an external host until you perform an explicit command to do so.

The **`save`** command is used to save the working configuration. If no host and file name are given, the working configuration is saved in NVRAM.

If a host and file name are specified, the IntelliServer uses TFTP to store the configuration on the specified host. This requires cooperation on the part of the host. In addition, TFTP must be enabled and usually a file by that name must already exist and have universal write permissions.

When a configuration is saved, it replaces any other configuration that has been saved in the same place. There is only one NVRAM, so whatever you save there replaces the last thing you saved. Network hosts can hold several files, so you can save different configurations under different file names if you wish. *It is a good idea to save a known working configuration to a network host as a backup, before embarking on new and unusual configuration experiences.* This has been known to save time in the past.

When configurations are restored they are always restored to the **working configuration**. For example, when you restore factory defaults, NVRAM is not affected. If you later **`save`** your configuration, *then* it is affected.

Start-Up

When the IntelliServer boots, it first reads its configuration from NVRAM. It may be that this configuration contains an option to load a different configuration from a host (see "Configuring Bootstrap Options" on page 217). In that case, the IntelliServer loads this new configuration from the appropriate file on the host. Here the process ends, even if this new configuration file had specified yet *another* boot configuration host and file.

The configuration files carry a checksum so that the IntelliServer can tell if one is corrupted. If the IntelliServer tries to load a configuration file from a host at start-up, and the file is bad (or unavailable), the IntelliServer uses the NVRAM configuration.

If the configuration stored in NVRAM is corrupt, the IntelliServer displays a warning message and uses its factory default settings instead. If this happens, the IntelliServer also may be unable to determine its Ethernet address, so it sends a message to the console screen asking you to type in the correct Ethernet address. How do you know the correct address? It is printed on a label found on the back of every IntelliServer. After you have entered the correct address, the IntelliServer automatically saves it in NVRAM, and reboots itself.

It is unusual for NVRAM to become corrupt, but this can happen if you power down the IntelliServer while you are in the process of saving a new configuration to NVRAM. The process of saving takes a few seconds and during that time the NVRAM does not contain an entire valid configuration.

Restoring From a Host Ignores IP Address

When you restore your configuration from a host file, the IntelliServer must have had an IP address already, otherwise it could not have TFTP'd the file over the network. Since the IP address is known to be good, the IP address in the down-loaded configuration file is ignored.

Forcing Factory Defaults

Suppose you accidently were to configure your IntelliServer in such a way as to render it unusable. For example, what if you configured all the ports for 56,700 baud and did not have any devices capable of that speed? Then, to top it off you had set all the networking parameters to something incorrect, so you are unable to telnet *into* the IntelliServer to do maintenance in that way. How are you going to manage to log into the IntelliServer so that you can correct things?

Because of the possibility of this accident occurring, the IntelliServer has a special provision for forcing it back to factory defaults. At power-up, just before it displays the console messages, it checks to see whether it has received an **`esc`** key on port 0, at 9600 baud. If it has, then it restores the working configuration to factory defaults. This does not affect the configuration that is stored in NVRAM, however. After you have booted successfully using the default configuration, you can then restore the configuration from NVRAM, make the necessary corrections to your configuration, and save back to NVRAM.

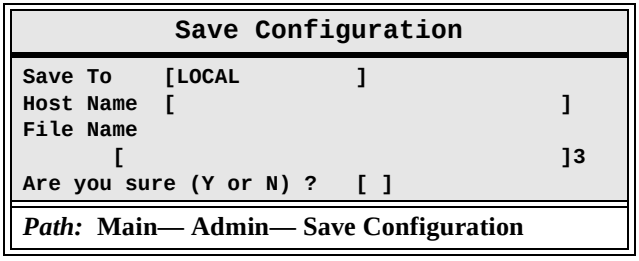
Use the following procedure to force the IntelliServer back to factory defaults:

1. Connect a terminal to port 0.
2. Configure the terminal for 9600 baud, 8 bits, no parity. Do this regardless of how Port 0 is configured on the IntelliServer.
3. Reboot the IntelliServer. As it is booting, press the terminal's **`esc`** key repeatedly (about twice a second will do) until you see a message "Restoring Factory Defaults" on the console screen. This indicates that the IntelliServer has recognized the escape key. When start-up is complete, there should be a command prompt on your console.
4. If you do not find yourself at a command prompt, the IntelliServer may have not seen the *escape* key at the proper time. You may need to repeat step 3 one or more times, once you have rechecked your terminal and cable for obvious problems.
5. To fix the configuration in NVRAM, first do a **`restore`** command to load the configuration from NVRAM into the working configuration. (The version in NVRAM was not restored to factory defaults). You will still be able to use Port 0, even if the port configuration in NVRAM is bad because this configuration wouldn't take effect until you log off port 0. So don't log out just now.

6. Run whatever commands and menus are necessary to fix the configuration problem.
7. Use the **save** command to save your corrected working configuration back to NVRAM.
8. Reboot to confirm everything is the way you want it.

Screen 14-1 shows the menu form for saving your working configuration.

Screen 14-1: Save Configuration Form



The first item, **Save To**, is a pick-list. Your choices are:

- **LOCAL**, to save to NVRAM.
- **HOST**, to save via TFTP to a host file.
- **HOST and LOCAL**, to save to a host file and to NVRAM.

If you are saving to a host, you must also supply the host and file names in the indicated areas. To confirm, enter **Y** in the space marked *Are you sure?*, and press **Enter**. The IntelliServer briefly displays the message *"Working…"*, and then a message indicating whether the save was successful.

Screen 14-2 shows the menu form for restoring your working configuration. The first item is a pick list. Your choices are:

- **LOCAL**, to restore the working configuration from NVRAM.
- **HOST**, to restore from a host file via TFTP.
- **FACTORY**, to restore the working configuration to factory defaults.

Screen 14-2: Restore Configuration Form

```
          Restore Configuration
Restore From [HOST   ]
Host Name    [                              ]
File Name
        [                                   ]
Are you sure (Y or N) ?     [ ]
```

Path: **Main— Admin— Restore Configuration**

If you choose to restore from a host file, supply the host and file names in the indicated areas. To confirm, enter **Y** in the space marked *Are you sure?*, and press **Enter**. The IntelliServer briefly displays the message *"Working…"*, and then a message indicating whether the restore was successful.

Saving and restoring through the menu has the same effect as saving and restoring using the **save** and **restore** commands. If you have been using the menu for configuration, it is sometimes handier to save from the menu as well.

Example 14-2: Save Command

```
save
save host file
```

```
# save
# save 160.77.99.3 /usr/lib/isconf.3
# save jeeves /tmp/j34
```

Example 14-3: Restore Command

```
restore
restore factory
restore host file
```

```
# restore
# restore host
# restore factory
# restore 160.77.99.3 /usr/lib/isconf.3
```

Example 14-2 and Example 14-3 show the syntax and some examples of the save and restore commands. Refer again to Example 14-1 on page 352 if you are unclear what they do.

Ethernet Address

Each IntelliServer's Ethernet address is stored in NVRAM and normally would never be changed. If, however, the NVRAM was corrupted and the Ethernet address was manually entered incorrectly, it would be necessary to correct it manually. To do so, first you would use the *set server ethernet…* command (see page 208) and then you would save it to NVRAM. You cannot simply do a *save* command, because the IntelliServer preserves the *existing* Ethernet address and does not use the one you just defined. It does this in order to make it extremely difficult for you to accidently change the Ethernet address. If you want the new Ethernet address to be saved as well, you must use the command, *save ethernet*.

Example 14-4: Save Ethernet

```
save ethernet
```

```
# set server ethernet 00:80:64:10:10:10
# save ethernet
```

Fun with RARP and BOOTP

When factory defaults are in effect, none of the IntelliServer's network parameters are configured. There is no IP address, netmask, or broadcast address. These parameters need to be supplied before you can access the IntelliServer over the network, and normally this is done by attaching a terminal to port 0 and using commands to configure the necessary items.

In some environments, it is not practical to configure the IntelliServer from a local port in this way. In widely distributed networks, it may be easier if an IntelliServer could be connected to one portion of the network and then configured over that network from a central location.

At first glance, this would seem to be a *Catch-22*. You can't configure the IntelliServer until its on the network and you can't talk to it over the network until you configure it. It is for this reason that the RARP and BOOTP protocols were devised.

RARP

RARP stands for *Reverse Address Resolution Protocol*. A network host, designated as a *RARP server*, has a file containing Ethernet Address and corresponding IP Addresses. If there is some host on the local network that does not know what its own IP Address is, it broadcasts a *RARP request* over the network. It supplies its Ethernet address (which it *must* know) as part of the request. The *RARP server*, seeing this request, looks for a matching Ethernet address in its file. If it finds it, it sends back a reply supplying the IP address.

On many UNIX systems, the RARP server is called *rarpd*, and the configuration file is */etc/ethers*. You should consult your own system's documentation for details because different versions of UNIX vary.

BOOTP

BOOTP works in much the same way as RARP, but is designed to supply much more information. A network host designated as a *BOOTP server* has a file containing the Ethernet Address of each host it is supporting. In addition to the IP address, this file can store other essential network information such as nameservers, gateways, domain name, and even the name of a configuration file to restore from.

A host that wants this information broadcasts a BOOTP request containing its Ethernet address. The *BOOTP server* looks up the Ethernet address in its file and sends back a reply containing all the information that was stored.

When does the IntelliServer use RARP and BOOTP

When the IntelliServer starts up, if its NVRAM configuration does not have a valid IP address, it attempts to get one by broadcasting BOOTP and RARP requests. If there is a reply, it adds the information to its working configuration and continues the start-up process. If there is no reply to these requests, the IntelliServer continues the start-up process bringing up the serial ports. Since there is still no IP address, it will not start up any of its network daemons.

Once the serial ports are active, you are able to configure an IP address and other settings manually if you have a terminal and wish to do so. But meanwhile, the IntelliServer is continuing to send out BOOTP and RARP requests, once every few seconds. It stops sending these requests when one of two things happens: if you enter an IP address manually, or if it receives a reply. In this way, the IntelliServer receives its configuration information from the network, or manually from a terminal, whichever is available first.

If you have no interest in using BOOTP to configure your IntelliServer, just attach a terminal and configure everything manually: if a valid IP address is stored in NVRAM the IntelliServer will no longer do BOOTP and RARP requests at start-up.

BOOTP— Host Configuration

On many UNIX systems, the BOOTP daemon is called **`bootpd`**, and the configuration information is stored in a file called **`bootptab`**. Consult your system's documentation for details, as always. Different UNIX flavors may store this file in different places, and there may be slight variation in the syntax.

Example 14-5 shows a sample entry from a **bootptab** file for an imaginary IntelliServer called *zeus*: The first line has the IntelliServer's host name followed by a colon and a backslash.

Example 14-5: Sample Bootptab Entry

```
zeus:\
        :ht=ethernet:\
        :ha=008069800997:\
        :ip=160.77.99.30:\
        :sa=160.77.99.1:\
        :sm=255.255.0.0:\
        :ds=160.77.99.2, 160.77.99.3:\
        :dn=computone.com:\
        :vm=rfc1084:\
        :bf=/usr/local/cnx131:\
        :df=/usr/local/cnx.cnf:\
        :gw=160.77.99.1:\
        :hn:\
        :lg=160.77.99.3:
```

Each item of information starts with a colon. Then there is a code (sometimes called a *tag*) that indicates what information is being supplied: **ht** for hardware type, **ha** for hardware (Ethernet) address, **ip** for Internet Address, and so on. These are standard abbreviations but may vary slightly from system to system. After the code there is an equal sign, followed by the value for that item, and a final colon. Each line except the final one ends with a backslash.

When the BOOTP server receives a BOOTP request, it looks for an entry in this file with a matching Ethernet address (00:80:69:80:09:97 in this example), then sends all the information back in a reply.

BOOTPTAB Parameters

Table 14-1 shows which BOOTP tags that are understood by the IntelliServer. These would be entered into your **bootptab** file as illustrated in the preceding section, or according to your UNIX system documentation.

The BOOTP reply that is sent over the network consists of two sections: a header of fixed format followed by a free-form *vendor specific* area. The first column of the table indicates where each value can be found. Items appearing in the vendor-specific area show the *tag value* for this item, and items in the fixed area show the byte offset into the header. You will not normally need this information, but it may be useful in reconciling any compatibility problems that may arise. For example, there was one version of **bootpd** that did not understand the **df** tag. The tag **T14** was able to be used instead, because this version of **bootpd** had a provision for entering otherwise undefined tags in this way.

The next columns give the corresponding tag name and an example of its use. The last column has a description of each item and the IntelliServer command you use to set each corresponding parameter. This should make it clear what each field in the **bootptab** file actually represents.

Table 14-1. BOOTPTAB Parameters

Value	Tag	Example	Description **Equivalent Configuration Command**
tag 1	**sm**	**:sm=255.255.0.0:**	IntelliServer's Subnet Mask **set server netmask** *value*
offset 16	**ip**	**:ip=160.77.99.2:**	IntelliServer's IP Address **set server address** *address*
The *sm* and *ip* tags supply the IntelliServer's Subnet Mask and IP address. There is no tag for setting the Broadcast address, but the IntelliServer creates a default one based on the class of the IP address.			
tag 15	**dn**	**:dn=synapse.net:**	Domain name **set server domain** *name*
tag 12	**hn**	**lightning:** **:hn:**	Supply IntelliServer's host name (from the first line of this bootptab entry). **set server name** *name*

Table 14-1. BOOTPTAB Parameters (Continued)

Value	Tag	Example	Description **Equivalent Configuration Command**
The next four tags, *gw, ds, lg,* and *sa* identify particular hosts on the network. You may supply either an IP address or a host name. If you supply a host name, the BOOTP server resolves it to an IP address before sending the BOOTP response. Either way, the IntelliServer receives an IP address.			
tag 3	**gw**	**:gw=160.77.99.1:** **:gw=harp:**	Gateway address for default route **set gateway default** *address*
tag 6	**ds**	**:ds=160.77.99.4:** **:ds=harp, shade:**	Domain name server (you can list up to three) **add nameserver** *address*
tag 7	**lg**	**:lg=160.77.99.5:** **:lg=bunyan:**	Syslog Host **set server sysloghost** *address*
offset 20	**sa**	**:sa=160.77.99.2:** **:sa=vanderbilt:**	Server Host Address **set boot primary** *host ... ...*
The next two tags, *bf* and *df*, identify the Primary TFTP Boot file and Configuration, respectively. The TFTP host is assumed to be the host that responded to the BOOTP request, unless the *sa* tag appears (see above). Taken together, this address and the value of the *bf* and *df* tags comprise what you need for a single **set boot primary** *host bootfile cfgfile* command.			
offset 108	**bf**	**:bf=/usr/is/cnx:**	TFTP Boot file **set boot primary** *... boot ...*
tag 14	**df**	**:df=/usr/is/cnf:**	TFTP Configuration file **set boot primary** *... ... cfg*
The last three tags do not correspond to specific configuration commands on the IntelliServer but are required for BOOTP to function properly.			
offset 172	**vm**	**:vm=/rfc1084:**	*Required: defines the format used in the vendor-specific area of the BOOTP reply.*
n/a	**ha**	**:ht=ethernet:**	*Hardware Address Type.*
n/a	**ha**	**:ha=008069800997:**	*Ethernet address: Defines which IntelliServer receives the information in this block.*

Booting a New IntelliServer

When the IntelliServer is configured to factory defaults, it has an Ethernet address but no IP address or other network information. Because there is no IP address, the IntelliServer sends out BOOTP requests.

The IntelliServer is also configured to TFTP boot, but there is no TFTP host, boot file, or configuration file defined. This is done so that the IntelliServer initially prepares itself to netboot a kernel, in case a boot file is named in a BOOTP reply. (see "When Net-booting Fails" on page 218 for an explanation of why this is so).

The following sequence is what happens when you boot the IntelliServer for the first time. This is similar to the list on page 220, but factors in the effect of information that might be received from BOOTP replies.

1. Because the *boot type* defaults to *TFTP*, the IntelliServer reconfigures its memory in preparation for possible net-booting.
2. Because there is no IP address defined, the IntelliServer broadcasts BOOTP and RARP requests. Normally, there will be *either* a RARP server *or* a BOOTP server on your network, but in case there are both, the BOOTP requests are sent first because the IntelliServer prefers the additional information that these replies can provide, compared to RARP replies which are somewhat stingy.
3. If the IntelliServer receives either a BOOTP or RARP reply, it stores the information in its working configuration and continues the booting process. If it does not receive a reply, skip to step 7.
4. If the BOOTP reply specified a configuration file, the IntelliServer attempts to download it via TFTP. The information in this file replaces information received from the BOOTP reply itself, except for the IP address which is never restored from remote configurations.
5. If there is now a TFTP boot file (defined either from the BOOTP reply or from the remote configuration file), the IntelliServer attempts to boot this new kernel. Skip to step 11.
6. If there was no TFTP boot file defined in the reply, then the IntelliServer reboots itself using the software in PROM. The rebooting allows the IntelliServer to re-configure its DRAM for normal operation. Skip to step 11.

7. If the IntelliServer receives no BOOTP or RARP reply at this point, it still has no IP address and there is no bootfile specified. It cannot possibly net-boot at this time so it re-boots using the kernel in PROM, allowing it to configure its DRAM for normal operation. When the kernel starts up, it sends BOOTP requests again, because it still does not have an IP address. It ignores any boot-file's however, because it is no longer configured to net-boot.
8. If it still receives no response to its BOOTP requests, it continues the start-up process of bringing up the serial ports. However, it continues to broadcast BOOTP requests every few seconds. This is done to support installations where an unattended IntelliServer is attached to a network that suddenly loses power. When power is restored, the IntelliServer is likely to start sending its BOOTP requests long before any host is ready to send a reply. By continuing to send the requests, it can receive a reply when the BOOTP host is ready to send one. Then, it will have its IP address, allowing it to be reached over the network. If it needs to be net-booting a newer version of software, a network host can now send it *shutdown* commands. This time when it comes up the BOOTP host is ready to send a reply immediately, and it can net-boot.
9. While the BOOTP requests are continued in the *background*, the IntelliServer continues to bring up the serial ports, issuing command prompts on ports 1, 7, 8, and 15. At factory defaults, these four ports are configured for 9600/8/no parity, and for *Autologin* as user root. This is why they issue a command prompt immediately, without requiring a login. All other ports default to *disabled*.
10. Once you have entered the IntelliServer's IP address manually, it brings up its network software and stops sending BOOTP requests. Skip to step 12.
11. When *this* kernel boots, it recognizes that there will be no further attempt at net-booting and so configures itself for normal operation. Like its ancestor, he broadcasts a BOOTP request for configuration information, but this time it ignores any boot-file's that might be specified.
12. Start-up is complete.

For Those Who Want To Use BOOTP Always

Since the IntelliServer will not send out BOOTP requests to learn the IP address if it has an IP address stored in NVRAM, and because your working configuration now has an IP address, *don't save your working configuration to NVRAM.* Instead, save it via TFTP to a configuration file on the host.

For Those Who Don't Want To Netboot

If you aren't using BOOTP and have no interest in net-booting, change the boot type to *None*, specify an IP address, and save your working configuration to NVRAM along with any other configuration changes you want to make. Since there is now an IP address, there will be no BOOTP or RARP requests and since the boot type is *None* the kernel comes up configured for normal operation. There is no need to re-boot because there is nothing different to net-boot.

When Factory Defaults Don't Net-boot

If you force factory defaults using the **esc** key as described on page 355, then the kernel configures itself for normal operation and does not attempt to net-boot. This way, if you had been *incorrectly* configured to retry netbooting forever from an imaginary host, you can stop this behavior before it starts.

When Boot Type Is BOOTP

When the *boot type* has been configured as BOOTP, then a BOOTP request is sent to learn the name of the boot and configuration files, even if there *is* a valid IP address. In this case, however, the other configuration information in the reply is ignored.

UNIX Host Configuration Tips

Different UNIX hosts may have implemented TFTP, BOOTP, and RARP differently, so you should always consult your system's documentation for the final word on these issues. The advice in this section applies to many UNIX systems, and even when it does not directly apply it may suggest where to begin.

Bootp

To use BOOTP protocol, one of your network hosts must be running a BOOTP server called *bootpd*. If it is not already running, you may need to edit the network configuration file, **`/etc/inetd.conf`**. Look for a line like the following, and insert one if it does not already exist:

```
bootps  dgram  udp  wait  root  /etc/bootpd  bootpd
```

After adding this, you may need to re-boot the UNIX host before the change takes effect. The BOOTP configuration file, **`/etc/bootptab`**, is configured as shown in the previous sections.

RARPD

While RARP is less useful than BOOTP, some of you may wish to use it instead. The RARP server software is usually called *rarpd*. If it does not already start up automatically, you need to consult your system's documentation to find out how to do this, because there is less uniformity than with BOOTP.

Example 14-2: Sample entries from /etc/ethers file

```
00:80:69:80:08:10    parasol
00:80:69:80:09:33    umbrella
```

Example 14-3: Sample entries from /etc/hosts file

```
160.77.99.44   parasol
160.77.99.45   umbrella
```

Once *rarpd* is running, it probably uses files called **`/etc/ethers`** and **`/etc/hosts`** to determine which IP address corresponds to which Ethernet address. In the examples above, Ethernet address **`00:80:69:80:08.10`** is associated with a host name of *parasol*, which is in turn associated with the IP address **`160.77.99.44`**.

TFTP

In order to net-boot a kernel or to save and restore configuration files to a host, a TFTP server must be running on that host. On most systems, it is called **`tftp`** and is activated by a line in **`/etc/inetd.conf`** like this:

```
tftp  dgram  udp  wait  root  /etc/tftpd  tftpd
```

If there is already a line there, but beginning with a **`#`** (*pound sign)* then the line has been commented out: remove the pound sign. There may be a **`-s`** on this line that indicates that TFTP is to run in *secure mode*. In this mode, all file transfers take place relative to a certain directory. Read your system documentation for details.

If you want to *restore* an existing configuration file or netboot a kernel, you have to store these files where *tftpd* can find them. Be sure to give the files *universal read permissions* using a command like **`chmod +r`** *`file-name`*.

Example 14-4: Host Commands

```
cd /tmp
mkdir ccc
cd ccc
tar xf /bin/cnx /dev/fd0135
mv bin/cnx /usr/boot
chmod +r /usr/boot/cnx
:> /usr/boot/mycfg1
chmod +rw /usr/boot/mycfg1
```

Example 14-5: IntelliServer Commands

```
save myhost /usr/boot/mycfg
set boot primary myhost /usr/boot/cnx /usr/boot/mycfg
save
```

If you want to *save* your configuration to a host, usually the file has to exist already and have universal write permissions.

Example 14-4 shows some host commands you might use to copy an IntelliServer from a distribution diskette, move it to a new directory, give it read permissions, and finally create a dummy configuration file.

Example 14-5 shows some IntelliServer commands you might use to save your configuration to that dummy file, and to configure the IntelliServer to netboot using this new kernel and the configuration you just saved. In this example, the file names are shown beginning with a slash to indicate that the paths are all relative to the root filesystem. On some systems, the *tftpd* daemon will not want to see these leading slashes so you need to leave them out.

CHAPTER 15

Other Administrative Commands

In this chapter you learn a variety of commands useful in administering the IntelliServer. These include:

- Commands to diagnose and reset serial ports.
- Commands to display system status.
- Advanced diagnostics commands.
- Miscellaneous commands.

Serial Port Commands

This section describes commands that are all related to serial ports. Some you use for diagnostic purposes, such as sending data to a port or putting the port into remote loopback mode. Others, you use for administrative purposes, like sending a message to all serial ports with users logged in, or being able to kill processes running on a particular port.

Port-List

Many of these commands apply to multiple ports as well as single ports The first command-line argument is *port-list* which is a series of one or more numbers from **0** to **63**, separated by commas or dashes. A single number selects a single port, of course. Numbers separated by a dash (like **0-7**) select ports 0 through 7 inclusive. To select several ports that aren't consecutive, separate them by commas (like **0,3,5**). If the port number is omitted, the current port (the one you are typing on) is assumed. The keyword **ALL** represents all ports.

To specify one of the *pseudo-tty* ports created when someone telnets *into* the IntelliServer, use port names **pts0** or **pts1**.

Output Port

With the **output port** command, you must specify a single port, not a port-list. The command comes in three variations.

Example 15-1: Output Port Command

```
output port port string text [forever]
output port port pattern barber [forever]
output port port pattern columns [forever]
```

```
# output port 3 string "Hello World"
# output port 4 string "message keeps going" forever
# output port 7 pattern barber
# output port pattern columns
```

The first (**`output port`**...**`string`***)* sends a string of text to the selected ports you specify the text on the command line. The second (**`output port`**...**`pattern barber`**) sends a fixed "barber-pole" pattern of data. The pattern is called a "barber pole" because the lines look something like this:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijkl
BCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklm
CDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmn
DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmno
EFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnop
```

The third form (**`output port`**...**`pattern columns`**) is similar to the second, but the data on each line is identical, not staggered as with the barber-pole pattern.

If the optional keyword **`forever`** is added at the end of the command, the output is repeated until you terminate it by pressing the *interrupt key*, usually **`DEL`** or **`ctrl-C.`** Otherwise, a single text string or 23 lines of pattern will be sent.

The **`output port`** command sends the data to the selected port regardless of whether the port is in use. If the port is currently enabled, the data is sent according to the ports present configuration. If the port is currently disabled, the port is opened and configured according to its stored settings and the data is sent and the port re-closed.

The **`output port`** command has many uses, including:

- Sending continuous data to a port to troubleshoot cabling or signal levels.
- Sending test data to a printer to confirm that flow control is configured properly.
- Sending test messages to a local terminal.

Echo Port

The **`echo port`** command places the selected port into remote loopback mode until you exit this command by pressing your *interrupt key*, usually **`DEL`** or **`ctrl-C`**.

Example 15-2: Echo Port Command

```
echo port port
# echo port 5
```

When the port is in remote loopback mode, all data received by the serial port is transmitted back out. The data is not seen by the IntelliServer and data that the IntelliServer sends to the port is ignored. This command does not allow you to put your *own* port into remote loopback mode because how would you tell the IntelliServer you were done?

This command is useful for troubleshooting if a terminal is connected to the port and you can see data echo back, then the terminal, cabling, and the port's physical configuration are all probably ok.

Kill, Hangup Port

The **`hangup port`** command sends a *hangup* signal to all processes running on the selected ports. The effect is the same as if the *carrier-detect* (**DCD**) signal on those ports had been dropped. Processes respond as they would when a call is disconnected. Generally, they perform some cleanup or logging operations and then close the port themselves.

Example 15-3: Kill Port, Hangup Port Commands

```
kill port port-list
hangup port port-list
# kill port 7-12
# hangup port 13
```

The **`kill port`** command sends a kill signal to all processes running on the selected ports. This causes the processes to stop immediately, without any opportunity for clean-up.

Because the **hangup port** command allows the process to clean up after itself before it exits, this is the better choice if you want to terminate a PPP, SLIP, or CSLIP session manually. If you use the **kill port** command, routes and ARP-table entries associated with that connection may not be deleted right away.

When you change the configuration of a port that is presently enabled, you may want to use one of these commands to terminate processes that are already using the port so that the new configuration can take effect. Otherwise, you need to wait for whoever is using the port to finish. If a user is presently logged into the port, the changes take effect when the user logs out.

Broadcast

The broadcast command sends the message to all specified, active ports. Nothing is sent to ports waiting for carrier or login, and the message is not sent to your own port.

Example 15-4: Broadcast Command

broadcast *port-list message*
 # broadcast 4-5 "So long and thanks for the fish" **# broadcast all "I will be re-booting the server at 4pm"**

The complete message looks like this:

Example 15-5: Format of a Broadcast Message

Broadcast message from *username* **(***port #***) on** *server-name: Text of the message*
Broadcast message from root (0) on jeeves: **So long and thanks for the fish**

Use broadcast with caution. The broadcast message is sent to all the ports you specify, if they are active. If the port is disabled or if it is a login port and no one is logged in, then the port is considered inactive and no message is sent there. Active printer ports, ports with PPP connections running on them, as well as login ports with users logged in and doing work are all active and receive a broadcast message if you include these ports in the list.

You usually would not want to send a broadcast message over a PPP line or to a Reverse TCP port so don't include these ports in your list.

Broadcast messages are sent without any regard for what a login user might be doing on his terminal. If he receives a broadcast message while viewing (or worse—displaying!) a menu screen, both the menu and the message are apt to be garbled. Because of these barbarisms, the broadcast command is generally used for matters of some urgency.

Shutdown Command

This command is certainly port-related. After sending warnings to every active port (omitting printer ports and outbound SLIP/PPP connections) it then kills all the processes and shuts down the IntelliServer.

Example 15-6: Shutdown Command

shutdown now|*minutes* [*message*]
shutdown now **shutdown 5 "Need to do maintenance"**

If you have made configuration changes that you have not yet saved in NVRAM, you are warned of this and asked for confirmation before the shutdown proceeds. As the shutdown time approaches, repeated warnings are sent to any users still logged on. These warnings include the optional message specified with the shutdown command. Any ports still active at zero hour are killed.

Note	When ports are killed as part of shutdown, syslog and RADIUS accounting records may not have time to be sent. The safest way to ensure these records are preserved is to do a **hangup port** command targeting all billable ports, before you run the shutdown command.

System Status Commands

The commands in this group allow the administrator to monitor the status of the IntelliServer.

Whodo

The **whodo** command lists the commands that are running on all active serial ports, as well as administrative sessions created when you telnet into the IntelliServer. These sessions are designated by the port names *pts0* and *pts1*, while the serial ports are designated by their port number. Active PPP, SLIP, and CSLIP connections are shown, as well as active connections to *Reverse-TCP* and *Printer* ports.

Example 15-7: Whodo Command

whodo [all]

```
# whodo
port session  owner     command
3    0        jake      telnet   160.77.99.23
pts0 0        root      shell    whodo
#
# whodo all
port session  owner     command
0    0        root      init     awaiting login
1    0        x1        init     awaiting DCD
2    0        x2        init     awaiting DCD
3    0        jake      telnet   160.77.99.23
pts0 0        root      shell    whodo all
```

The **whodo all** command lists everything **whodo** does, but also gives the status of any login ports which are enabled but not yet active; (i.e., those waiting for carrier or for a response to login). These commands are useful for providing an overview of what is happening on each port of the IntelliServer. For example, you might use this to see if there is anything important going on before shutting down the IntelliServer.

Systat

The **systat** command reports current memory and CPU usage. If you specify an optional number of seconds, then systat reports continuously at that interval until you exit this command by pressing your *interrupt key*, usually **DEL** or **ctrl-C.** In *syslog's* output, **user** is the percentage of total CPU time spent running applications such as telnet, rlogin, menu, and so on. **System** represents CPU time spent in the system kernel servicing interrupts and running the protocol stacks. IntelliServers running mainly PPP, SLIP and CSLIP connections have moderate **system** CPU usage and almost no **user** CPU usage. Systems running mostly other types of connections find the system and user time more nearly equal.

Example 15-8: Systat Command

systat [*seconds*]

```
# systat
sampling...   0% user,  0% system, 100% idle, 2172 KB mem
# systat 3
sampling...   0% user,  0% system,  99% idle, 2172 KB mem
sampling...   0% user,  0% system,  99% idle, 2172 KB mem
sampling...
```

The remaining percentage tells how much the CPU is **idle** and waiting for something to happen. If this number is unexpectedly low, it may be a warning sign that something has been mis-configured. The last number in the report is the amount of free memory available for user processes.

Advanced Diagnostics

The advanced diagnostics include three commands (**`ps`**, **`streams`**, and **`queues`**) that display different kinds of status information, and two commands (**`test1400`** and **`eloop`**) that perform diagnostic tests. You will not normally need any of these commands; and when you do use them it will probably be under the direction of someone in our technical support department. In order to run these diagnostics, you must place the IntelliServer into *development mode*.

Production Command

The IntelliServer can boot in one of two modes:

- In *Production Mode*, the advanced diagnostic tools are deleted in order to make more memory available for regular tasks.
- In *Development Mode*, the advanced diagnostic tools are left in place.

When factory defaults are in effect, production mode is enabled. To enable development mode, follow the three steps shown in Example 15-9:

Example 15-9: Enabling Development Mode

```
# set production disabled
# save
# shutdown now
```

First, enable development mode by *disabling* production mode. Then save the configuration in NVRAM (or to a host if you are restoring your configuration from a host at boot time). Finally, restart the IntelliServer. The shutdown is necessary because by the time you run this command the advanced diagnostics tools have been removed and you need to reboot in order to get them back. Also, because the reboot was needed, it is necessary to save the configuration.

When you have finished with the advanced diagnostic tools, repeat the above process except typing **`set production enabled`** instead of disabled.

ps

The **ps** command reports the status of all processes running on the IntelliServer. Sample output is shown in Example 15-10. A lot of this information is meaningful only to the IntelliServer's software engineers, so details are not provided in this manual. Four of the columns are interesting, however:

- **PRT** is the port number the processes is using. Port numbers 200 and 201 represent the sessions created when you telnet into the IntelliServer. A question mark under this column indicates *daemon* processes not associated with a particular port.
- **COMMAND** gives the name of the process or command that is running on that port.
- **TIME** indicates the number of seconds of CPU time this process has used since it started.
- **PID** stands for the Process ID number.

Example 15-10: PS Command

```
ps
SLOT and ADDR F S EID UID   PID  PPID  C PRI    WCHAN PRT SN   TIME COMMAND
   0/80001000 4 R   0   0     0     0 80  63        0   ?  ?  39.06 idle
   1/8000a000 0 S   0   0     1     0 63  32 f0000000   ?  ?   0.46 init
   2/8000e000 5 S   0   0   294     0 17  32 8013aaf5   ?  ?   0.00 lcd
   3/80011000 5 S   0   0     3     0 17  32 8013abad   ?  ?   0.00 lcd
   4/80010000 0 S   0   0   296     1 38  32 80149908   ?  ?   0.00 rcpd
   5/80016000 0 S   0   0   735     1 80  32 800828bc   ?  ?   0.63 ttyd
   6/80017000 0 S   0   0     6     1 42  32 80149878   ?  ?   0.00 fingerd
   7/8001b000 0 S   0   0     7     1 80  32 f0000000   ?  ?   0.44 ttyd
   8/8001c000 0 S   0   0     8     1 42  32 8014aa78   ?  ?   0.04 telnetd
   9/800fa000 0 S   0   0     9     1 48  32 800828bc   ?  ?   0.04 logger
  10/80102000 0 S   0   0 16070     1 65  32 8014ad48   0  0   0.02 init
  11/80104000 0 S   1   0    11     1 48  32 80158398   1  0   0.02 init
  12/80106000 0 S   2   0    12     1 48  32 801583d0   2  0   0.01 init
  13/80108000 0 S   3   0    13     1 48  32 80158408   3  0   0.01 init
  14/8010a000 0 S   4   0    14     1 48  32 80158440   4  0   0.01 init
  15/8010c000 0 S   5   0    15     1 49  32 80158478   5  0   0.01 init
  16/8010e000 0 S   6   0    16     1 49  32 801584b0   6  0   0.01 init
  17/80110000 0 S   7   0    17     1 51  32 801584e8   7  0   0.02 init
  18/80118000 0 O   0   0  1478   459 80  32        0 200  0   0.32 ps
  20/80114000 0 R   0   0   458     8 80  32        0   ?  ?   4.79 telnetd
  21/80116000 0 S   0   0   459   458 80  32 80116000 200  0   5.22 shell
```

Why are the port, command, pid, and time interesting? Because when there is trouble, this is where the red flags appear. For example:

- If the **PID** assigned to one of the init processes is changing rapidly, this is an indication (among others you will have) that login processes are exiting and restarting, exiting and restarting. If the associated port is supposed to be idle, this can indicate a problem such as modem configuration or cabling.
- If the **TIME** spent on a process is growing quickly, this is also a sign that something is wrong. Normally, these numbers do not grow very fast. Check the configuration and cabling of the affected port.

Streams

The **streams** command gives statistics on streams buffer usage. Streams buffers are used by the kernel to carry data between the serial port drivers, various protocol modules, the Ethernet driver, and applications. If you are experiencing a symptom that may be related to system load, sometimes the streams command helps you to locate the problem. Sample output is shown in Example 15-12 on page 384.

Example 15-11: Streams Command Definition

streams [seconds] [**syslog**]
 # streams **# streams 60 syslog** **# streams syslog**

If you include the keyword **syslog**, the status report is sent to the syslog service (LOG_INFO priority). Otherwise it is sent to your terminal. If you specify an optional interval (in seconds) the IntelliServer runs this command in the background, forever sending a report at the requested interval. This option is generally

used when you are syslogging the results. If no interval is given, the report is given just once.

Example 15-12: Streams Command

```
streams
ITEM                CONFIG     ALLOC     FREE          TOTAL       MAX      FAIL  AVG
streams                444        92      352           3376        98         0
queues                1720       203     1517          12360       217         0
message blocks        4144       125     4019         918943       174         0
data block totals     4144       125     4019         912254       173         0
data block size 8     1350         3     1347         108553        26         0    0
data block size 16     690        11      679          43812        18         0    0
data block size 36     480        13      467         181739        39         0    0
data block size 64     560         8      552         185247        16         0    0
data block size 128    432        64      368          85922        71         0    0
data block size 256    590         3      587          99642        11         0    0
data block size 512      2         0        2              0         0         0    0
data block size 1024     0         0        0              0         0         0    0
data block size 1536    40        23       17         207339        23         0    0
data block size 2048     0         0        0              0         0         0    0
data block size 4096     0         0        0              0         0         0    0
number of bufcalls: 0
```

Here again, there is much that is *un*important. The important columns to watch are:

- **FAIL**, which indicates how many times a buffer was needed and there wasn't one available. When a buffer of a desired size is not available, processing on that data can be delayed until one *is* available, or other fall-back methods are used depending on the circumstances. If a few failures have been counted, this does not mean that there was any problem, but it is normal for the failure count to be zero or very low and not to be growing very quickly. To see otherwise is to suspect that the IntelliServer is under heavy load. If you see this when the IntelliServer is *not* under heavy load, it would be worth mentioning to one of our tech support people while they are helping you with whatever problem prompted you to run this command in the first place.
- **FREE** indicates the number of streams buffers of various types that are available. These numbers rise and fall but should not decrease over time. If you haven't rebooted the IntelliServer for a year and it is completely idle, the number of **FREE** buffers should not be much different from when it was first booted.
- **Number of bufcalls** represents the number of times the IntelliServer responded to a buffer shortage by suspending processing until a suitable buffer was available. Like **FAIL**, do not expect this number to go very high.

Queues

The queues command indicates what streams buffers are currently used by different drivers and protocol modules within the IntelliServer. Sample output can be seen in Example 15-14.

Example 15-13: Queues Command Definition

```
queues [seconds] [syslog]
# queues
# queues 60 syslog
# queues syslog
```

If you include the keyword **syslog**, the status report is sent to the syslog service (LOG_INFO priority); otherwise, it is sent to your terminal. If you specify an optional interval (in seconds) the IntelliServer runs this command in the background, forever sending a report at the requested interval. This option is generally used when you are syslogging the results. If no interval is given, the report is given just once.

Example 15-14: Queues Command

```
queues
total queues 1720
erwfn>..............tty [  92/ 128] 1:0:1 36,1,24
erwfn>..............tty [  92/ 128] 1:0:1 36,1,24
erwfn>..............tty [  92/ 128] 1:0:1 36,1,24
erwfn>..............tty [  92/ 128] 1:0:1 36,1,24
erwfn>..............tty [  92/ 128] 1:0:1 36,1,24
erwfn>..............tty [  92/ 128] 1:0:1 36,1,24
erwfn>..............tty [  92/ 128] 1:0:1 36,1,24
eRwfn>...........tcp_uwq [ 912/2048] 7:1:1 8,1,2 64,4,219
128,2,146
eRwfn<.........strrhead [ 384/ 512] 6:1:1 8,6,9
```

Table 15-1: Queue Command Key

Line of output: *flags.....queue-name* **[** *cost* / *high-water* **]** *messages*:*blocks-in-last*:*max-dups* *size*,*count*,*bytes* *size*,*count*,*bytes…*		
flags:		Indicate the type of queue and its status.
	e E	**E — e**nabled: the service routine for this queue will run.
	r R	**R** — wants read: another queue wants to read from this one.
	w W	**W** — wants write: another queue wants to write to this one.
	f F	**F** — full: no more data can be added to this queue.
	n N	**N** — no enable: do not enable automatically.
	> <	> indicates an output queue, < indicates an input queue
queue-name		The name of the queue indicates its function
cost		Resource cost of the items on this queue. This is based on the number and type of data buffers it is using.
high-water		Resource budget for this queue. When the resource cost exceeds the high-water, the queue is marked full and no more data will be added.
messages		The total number of messages on this queue.
blocks-in-last		The number of data blocks in the last message on the queue.
max-dups		The maximum number of duplicated blocks in the queue.
size		Each entry ends summarizing the *count* of data blocks of each *size* assigned to this queue, and the *total bytes* of data contained in these blocks.
count		
total bytes		

This report is not likely to make much sense unless you are familiar with the IntelliServer's software. There are really no conditions which you can consider *red flags* in themselves, but this report sometimes provides useful information to our software engineers and support personnel.

Eloop

The **eloop** command runs a loopback test on the Ethernet interface. Options indicate whether the test should be run once or repeatedly, whether it is to be run in background mode, and whether the Ethernet controller will be tested in internal or external mode.

Example 15-15: Eloop Command

eloop *options*
 # eloop i **# eloop b f**

Caution: This test was intended for manufacturing testing. If it detects a failure, a message is sent to the IntelliServer's console port (usually port 0) and the IntelliServer halts with its LED's blinking an error code. Keep this in mind when considering whether to run this on a live system. Also, when you are testing in external loopback mode, a *great* amount of Ethernet traffic is generated, so it may not be a good idea to run this with the IntelliServer connected to your real network. The Ethernet interface must be properly terminated for the external test to work. If your IntelliServer has a BNC connector, you should at least attach a Tee-connector with a terminator on each end.

Table 15-2: Eloop Command Options

i	Run the test with the Ethernet interface configured for *internal* loopback mode. If this option is absent, the interface is configured for normal operation and the test data is sent to the attached local network.
f	Run the test forever. Otherwise, a single pass of the test is run and **eloop** exits.
b	Run the test in background mode. After the test starts up, the command line returns to a prompt where you can run another command. Generally, the **b** and **f** options are used in tandem.

TEST1400

The **test1400** command runs a loopback test on selected serial ports. Options indicate which ports to test, whether to test data-set signals, and how to report errors.

Example 15-16: Test1400 Command

```
test1400 options ports...
```

```
# test1400 -dtrdcd -rtscts -r 1 3 5 7 11
# test1400 -t -r1 -c 2
# test1400 -b
```

If no ports are specified, then all ports configured as *disabled* are tested. If you specify ports to test, you must list them separately. You cannot enter a range of ports as with some of the other commands.

Ports are tested using the physical line settings (baud rate, character size, etc.) for which they were configured, unless one of the **-r** options is specified. Spaces must always separate options from each other. For example, you cannot combine the **-i** and **-c** options by entering **-ic**, as is possible with some other commands. Table 15-3 describes the command-line options.

Table 15-3: Test1400 Command Options

Manufacturing Options:	
-b	Burn-in mode - Same as specifying the **-f**, **-r**, **-c**, and **-z** flags. Also, hardware is automatically sensed to determine which data-set signals are supported. Data and all supported data-set signals are looped back and tested. DTR is assumed looped back to DCD and DSR; RTS is assumed looped back to CTS and RI in hardware configurations where these exist.
-ft	Final-test mode - The same as the **-b** option, except that implied options are **-s**, **-r**, **-c**, and **-e**. The test runs fewer passes and the test stops after four errors, where in burn-in mode the IntelliServer halts with an LED code at the first error.
Line-Speed Options: If no line-speed options are given, each port is tested at the speed for which it is configured.	

Table 15-3: Test1400 Command Options (Continued)

-r **-r1** **-r9**	Configure the ports being tested with different random baud rates at each stage of the test. • **-r** selects rates from 300–38,400 baud (average rates) • **-r1** selects rates from 134–9600 baud (slower rates) • **-r9** selects rates from 19,200–64,000 baud (faster rates)
Data-set Signal Options: If no data-set signal options are given, data set signals are not tested.	
-dtrdcd	Ensure the DTR signal is looped back to DCD. Assert and de-assert DTR and make sure DCD is asserted only when DTR is.
-dtrdsr	Ensure the DTR signal is looped back to DSR. Assert and de-assert DTR and make sure DSR is asserted only when DTR is.
-dtrcts	Ensure the DTR signal is looped back to CTS. Assert and de-assert DTR and make sure CTS is asserted only when DTR is.
-rtscts	Ensure the RTS signal is looped back to CTS. Assert and de-assert RTS and make sure CTS is asserted only when RTS is.
-rtsri	Ensure the RTS signal is looped back to RI. Assert and de-assert RTS and make sure RI is asserted only when RTS is.
-rtsdcd	Ensure the RTS signal is looped back to DCD. Assert and de-assert RTS and make sure DCD is asserted only when RTS is.
-dcd0	Ensure the DCD signal is always negated.
-dcd1	Ensure the DCD signal is always asserted.
-cts0	Ensure the CTS signal is always negated.
-cts1	Ensure the CTS signal is always asserted.
-dsr0	Ensure the DSR signal is always negated.
-dsr1	Ensure the DSR signal is always asserted.
-ri0	Ensure the RI signal is always negated.
-ri1	Ensure the RI signal is always asserted.
-t	Toggle DTR and RTS during test. Normally, if no other data-set signal options are given, DTR and RTS are both asserted during the entire test. With the **-t** option, these signals are asserted and de-asserted throughout the test, but corresponding input signals are not checked.
Testing Options:	

Table 15-3: Test1400 Command Options (Continued)

-c	Compare data received from a port with the data that was sent to it. Otherwise, incoming data is still read, but not compared.
-n	When the **-c** option is used, data is normally expected to be received in a timely fashion after it is sent. This is designed to detect a port whose receive or transmit data pins are entirely disconnected. With the **-n** option there is no limit to how long this could take, so a completely disconnected port is not detected. (If data ever *were* received, it is compared and has to be correct).
-i	Internal loopback - The UART chip is put into internal loopback mode for the duration of the test. Data that the UART would have transmitted are instead looped back internally; nothing is sent to the TXD pin. Data appearing on the RXD pin are ignored. Without this option, the UART is configured for normal operation, so data will have to be looped back using an external plug. This option has no effect on data signals: they cannot be tested in "internal" mode.
-z	Stop after only 8000 passes. (If neither the **-z** nor **-e** options are given, the test continues forever).
-e	Stop after a mere 1000 passes.
Reporting Options: If no reporting options are given, the test continues through the specified number of passes, reporting each error encountered to your own terminal.	
-f	Any errors are "fatal". A message is sent to the console port (defaults to port 0), and the IntelliServer halts with a code blinking in the LED's.
-s	Errors are displayed to your terminal as they are discovered, but the test stops after the fourth error.
-q	Quiet option - test1400 will not generate any output unless there are errors.
-d	Enables debugging output.

Clear

This command clears the terminal screen you are typing on. If the terminal type is known (see page 84) then the appropriate escape sequence is sent to the terminal to do this. If the terminal type is unknown, a string of 25 line-feeds are sent. Crude, but effective.

Example 15-17: Clear Command

```
clear
```

```
# clear
```

Env

Displays environment information about this process. In Example 15-18, the first sample was run from the command line as a result of a telnet session into the IntelliServer, the second sample was run from a terminal on one of the serial ports.

Example 15-18: Env Command

```
env
```

```
# env
TERM=xterm
LOGNAME=root
```

```
# env
TERM=ansi
RTERM=ansi-48
LOGNAME=root
SHELL=shell
TZ=EST
```

The following defines the terms in Example 15-18:

- On a serial port, **TERM** is the local terminal type configured for this port (see page 84). From a telnet session, this is the terminal type passed from the client to the IntelliServer under telnet protocol.
- **RTERM** is the remote terminal type configured for this serial port, as shown on page 85.
- **LOGNAME** is the current user's login name.
- **SHELL and TZ** are reserved for future use; currently they are always assigned as shown.

Tty

The tty command tells you which port and session you are using. Port numbers 200 and 201 represent the sessions created when someone telnets into the IntelliServer. Session numbers other than 0 are created to run on the alternate terminal screens supported by IntelliView.

Example 15-19: tty command

```
tty
```

```
# tty
tty 200 session 0
```

```
# tty
tty 5 session 1
```

Udp

This command enables or disables UDP checksums, or shows whether they are currently enabled or disabled. They are disabled by default.

Example 15-20: Udp Commands

```
set udp checksum enabled
set udp checksum disabled
show udp
```

```
# show udp
udp checksums disabled
# set udp checksum on
# show udp
udp checksums enabled
```

Version

The version command displays the release and version numbers of the IntelliServer software that is currently running. It is important that you know the release number if you are needing help from Computone's technical support department.

Example 15-21: Version Command

```
version
```

```
# version
Computone IntelliServer
Release 1.3.0 Version 951001
```

CHAPTER 16

Connections

In this chapter you learn how to use the telnet and rlogin commands to log into hosts on your network. In this chapter you learn:

- Telnet and its command-line options.
- Rlogin and its command-line options.
- Starting connections from the command line.
- Starting connections from selected and global connections menus.
- Starting connections automatically at login.
- Starting connections automatically without a login.

Telnet

Telnet is traditionally used to log into a remote host in order to run interactive terminal sessions. In a more general sense, telnet communicates with a host over the network using the Internet *telnet* protocol. This implementation also allows you to communicate with arbitrary TCP service ports, and even create a TCP connection that bypasses *telnet* protocol entirely by sending the data directly between your terminal and the TCP connection. This allows the **`telnet`** command to be used for purposes beyond its original intention.

Here are some of the ways the telnet command might be used from the IntelliServer's command line:

Example 16-1: Telnet Command

`telnet` *`hostname|ip-address`* [*`port`*] *`options`*

```
# telnet 160.77.99.100 -t wy60
# telnet abercrombie.computone.com
# telnet feather -RC
# telnet jeeves 9003 -8
```

Telnet Arguments and Options

Command-line arguments are used to specify which host you want to access and which TCP port on that host, when you are using telnet for non-standard connections. Command-line options are used to supply terminal names and for choosing other defaults for telnet negotiations. Table 16-1 shows the telnet command-line agruments and options.

Table 16-1: Telnet Command-Line Arguments and Options

Arguments…	
none!	If the telnet command is given with no command-line arguments or options, it goes automatically into *telnet command* mode, indicated by the prompt, **`telnet>`**.
`hostname` *`ip-address`*	The name or IP address of the host. If a host name is used, the IntelliServer must resolve it using its host table or external nameservers. When a hostname or IP address is supplied, telnet immediately tries to open a connection to that host. If it fails to open a connection, the telnet command exits with an error message. (It does *not* drop into command mode, as do many implementations because this would constitute a security loophole for users configured to only telnet to certain hosts).
`port`	The TCP port number. The default is 23, which is the well-known port for telnet service. In some installations different services may be available on other TCP ports which telnet could access in this way.
Options…	
`-t` *`termtype`*	Allows you to specify a terminal name to the remote host. If you do not supply this option, *Remote Terminal Type (page 85)* configured for your port is used. If *that* is blank, then the *Local Terminal Type (page 84)* is sent. The host sets up the user's TERM environment variable based on the terminal type the IntelliServer sends. This enables screen-based applications to run properly. In order to be useful, the terminal name specified must correspond to an appropriate entry in your host's */etc/termcap* file or *terminfo* database.
`-E`	The telnet *escape* character is used for switching telnet into command mode. By default, this character is *ctrl-***`]`**. When the **`-E`** option is used, the regular escape key is treated like an ordinary character and the telnet *escape* function is performed by the terminal's **`break`** key instead.
`-8`	This disables the telnet *escape* key in the same way as the **`-E`** argument but also negotiates with the host to switch to *telnet binary* mode. If the **`-8`** option is used, the **`-E`** option is superfluous.

Table 16-1: Telnet Command-Line Arguments and Options (Continued)

-R **-RC**	These are usually used only when a TCP port (other than 23) has been specified and disables telnet protocol entirely. Raw data from the keyboard is sent to the TCP connection and raw data from the connection is sent to the terminal. When the **-RC** option is used, the keyboard is configured for canonical input; otherwise, the keyboard input is raw. Since *telnet* protocol is not used, there is no telnet escape key and no command mode. The **break** key from the terminal terminates the session. These options were intended to enable the IntelliServer to access simple custom applications you might write, but there are other uses: see "Using Raw TCP Connections" on page 405.	
-P1 -P2 -P3	These options are used to change the way *Carriage Return* (*hex 13)* characters are handled when you are not operating in telnet binary mode. This affects only characters that are *received* by the serial port (normally this would be keyboard data). This is provided for compatibility with hosts with archaic telnetd implementations.	
	default	According to *telnet* standard - *Carriage Returns* are padded with nulls (0).
	-P1	No null padding after *Carriage Returns.*
	-P2	*Newlines (0x10)* added after *Carriage Returns.*
	-P3	*Carriage Returns* replaced by *Newlines*.

Telnet Command-Mode

When you invoke **telnet** without any command-line arguments, or if you press the telnet escape character during an active connection, you enter telnet command mode which is indicated by the **telnet>** prompt.

When you are in command mode, your typed commands are interpreted and performed by the telnet program itself. When there is an open *telnet* connection to a host and you are not in command mode, your keyboard input is sent to the host and data received from the host is sent to your terminal.

The telnet commands are explained in Table 16-2 below:

Table 16-2: Telnet Command-Mode Commands

open *host* [*port*] **connect** host [port]	Attempts to open a connection to the specified host (or IP address) using the specified TCP port (or TCP port 23 if none is specified). If the connection fails, you are returned to command mode. *You cannot open a new connection if there is already one open.*
close	Closes an existing connection. If telnet was invoked with no command-line options, then this connection must have been created from the telnet command-line using the *open* command. So, in command mode you remain. If telnet was invoked with a host name or IP address on the command line, then the connection you are closing is *that* one. You should not be allowed to remain in command mode where you might open another session, so telnet exists now.
escape	Changes the telnet escape character when you enter this command. *Telnet* prompts you to enter the new escape character. Enter the new character you want to use, followed by the **enter** key. Then, *telnet* prints a confirmation. For example, if you wanted to use the escape key as the telnet escape character, you press the **esc** and **enter** key, and then *telnet* replies, "**Escape key is ^[**". Why would you do this? Perhaps after you telnet to this host, you want to use *its* telnet to reach another host. By changing *your* telnet escape, you can now use the default escape to enter *its* telnet's command mode, or *your* new escape code to enter *your* telnet's command mode. You could also use this command if the default telnet escape (**ctrl-]**) is used by the application you want to run on the host machine. The **-E** command-line argument can also be used to accomplish the same effect (see Table 16-1 on page 397).
bye **quit**	Closes any open *telnet* session and exits.

Table 16-2: Telnet Command-Mode Commands (Continued)

status	Prints status information about the telnet session, such as: • Whether you are connected to a remote host or not. • The telnet escape character, or whether **break** is used instead. • Whether binary telnet is enabled or not.	
?	Prints out a list of the telnet commands with short descriptions. (Remains in command mode afterward).	
? *command*	Prints a short description of this command. (Remains in command mode afterward).	
send *arguments*	This command is used to send special telnet messages or characters to the host. *Arguments* include the following:	
	ao	Output Abort (**IAC, AO**)
	ayt	Are you there? (**IAC, AYT**)
	brk	Break (**IAC, BREAK**) If you are telnetting to an IntelliServer port that is configured for reverse-TCP, this command causes the port to transmit a *break*. Otherwise, it sends a break signal to the process running on the host you are logged into.
	ec	Erase Character (**IAC, EC**)
	el	Erase Line (**IAC, EL**)
	escape	Send the telnet escape character to the host.
	ga	Go Ahead (**IAC, GA**)
	ip	Interrupt Process (**IAC,IP**)
	nop	No-op (**IAC, NOP**)
	sync	Synch (**IAC, DM**)
	?	*Display a list of commands*
	The sequences marked **IAC**… are special commands recognized by telnet protocol and are defined in the telnet RFC's.	

Table 16-2: Telnet Command-Mode Commands (Continued)

crmod	This command enables and disables whether a **NL** (newline) is to be added after any **CR** (carriage return) received from the network connection for display on the terminal. Normally, newlines are not added: use the **crmod** command once and they are added; use it again and they aren't. Note that this function does not overlap that of the **-P***n* command-line option. The latter affects data received from the serial port to send to the network and *crmod* affects data received from the network to send to the serial port.
options	This command enables and disables the display of telnet option negotiations. Normally these are not displayed: use the options command once and they are displayed. Use it again and they aren't. See "Telnet Option Negotiation" on page 404.

If a connection to a host is open, then telnet leaves command mode after performing the command (except for help commands). If there is no connection open, telnet remains in command mode.

Using Telnet Connections

When a telnet connection is open, data from the keyboard is sent to the remote host and data from the remote host is sent to your terminal. When you do not specify a TCP port number, you connect to the default telnet service port, 23. On the other end of your connection there will be something there to log you into the host system and offer you the appropriate services based on your login.

On UNIX hosts, this service is usually performed by a process called *telnetd* which creates a '"pseudo-tty" device on the host system and then sends data through this device to reach the application.

Pseudo-TTY's

What is a "pseudo-tty" port? A UNIX system might have a built-in console, or built-in serial ports, and can be configured so that you log in on those ports. Each built-in device has a device name such as /dev/tty1A associated specifically with it. For a user to be able to log in, the UNIX system must have opened and configured this device, and then sent login and password prompts, waiting for the replies. Then, the login process passes control to the appropriate application.

When UNIX *telnetd* establishes a connection, it creates a new device name which the login and application processes *think* is connected to some built-in serial port or console. In fact, it is connected to the *telnetd* program itself. *Telnetd* takes data that would have been sent to some serial port and sends it instead to the device on the other end of the telnet connection (the IntelliServer's telnet command, for example). Data coming from the telnet connection is sent so that applications on the host will read the data from the pseudo-tty device. In general, the processes on the UNIX box that do the real work do not realize that they are talking over a *telnet* connection because the telnet processes handle the protocols transparently.

Because of all of this, applications which can be run over dedicated built-in serial devices can usually run over telnet connections. There are two exceptions:

1. Host applications which rely on being able to control the serial port's physical protocols in real time (being able to raise and lower data-set signals under program control, or dynamically change line speeds and other physical link characteristics), will not operate over a *telnet* connection, because *telnet* connections by their nature do not allow this type of control.
2. Host applications which must be configured based on the specific tty device names may not be able to run over a *telnet* connection, or they may require specific configuration. The problem here is that pseudo-tty device names may be assigned randomly according to when the telnet connection is established. One day you telnet in from your office and the application thinks you are /dev/ttyp00, and the next day /etc/ttyp04.

It is important to note that these are limitations of network access through pseudo-tty ports in general, not a specific limitation of IntelliServers. Software designers who want their applications to be network-accessible will follow these rules:

1. Do not attempt to control the physical line settings of your tty device.
2. Do not attempt to "site configure" this session based on the terminal's device name. Use instead "environment variables" which *can* be used to identify the location of the terminal running the telnet session, as well as the terminal's characteristics. In networking environments, the device name will not indicate the physical location of the device.

Perhaps you are wondering how you would identify which particular terminal has started a telnet session? Easy, if you are using an IntelliServer:

- During port configuration, configure each port's *Remote Terminal Type* with a different terminal name: e.g., ISV0, ISV1, ISV2, ISV3… (see page 85, *Remote Terminal Type).*
- When you telnet from the IntelliServer, *telnet* protocol passes this terminal name to the host, which assigns it to the TERM environment variable.
- When you log in (as "evan", say) your UNIX host runs the **`.profile`** script in evan's home directory, /usr/evan, for example.
- What you must now do in order to let an application determine the site using environment variables is add appropriate code to the **`.profile`** script, similar to Example 16-2.

This example assumes that the application was designed to determine the terminal type from the environment variable TERM, and the site from a variable MY_PORT. Since different applications could be designed to use different environment variables for site-identification, you have to hand-craft the code in **`.profile`** to match your circumstances.

Example 16-2: Shell Script to interpret term variable

```
...
case $TERM in
ISV0)   TERM=wy60
        MY_PORT=0
        ;;
ISV1)   TERM=ansi
        MY_PORT=1
        ;;
esac
...
```

Telnet sends syslog messages at LOG_NOTICE level whenever a connection is established or disconnected.

Telnet Option Negotiation

Telnet options are special messages sent between a telnet client and server which allow them to pass information and agree on configuration options (Do not confuse these with the telnet command-line options discussed earlier). Normally, this negotiation takes place silently, but you can use the telnet command **`options`** to make telnet display the options messages as they are sent and received. This is sometimes useful for debugging compatibility problems. Since many options are negotiated as soon as a connection is established, you usually want to set this option before you open a connection.

To do this, you first invoke telnet without any command-line arguments. This puts you into telnet command mode. Then, you use the **`option`** command to show the option processing and, finally, bring up the connection using telnet's **`open`** command. Example 16-3 shows how this works and some of the output that results.

Example 16-3: Telnet, Showing Option Processing

```
# telnet
telnet> option
will show option processing
telnet> open aardvark
trying aardvark (160.77.99.203)...
SENT do SUPPRESS GO AHEAD (don't reply)
SENT will TERMINAL TYPE (don't reply)
Connected.
Escape character is ^]
RCVD do TERMINAL TYPE (don't reply)
RCVD will SUPPRESS GO AHEAD (don't reply)
RCVD will ECHO (reply)
SENT do ECHO (don't reply)
...
```

Using Raw TCP Connections

When you use the -R or -RC arguments, telnet protocol is bypassed entirely. Data from the keyboard is sent directly to the TCP connection and data from the connection is sent to the terminal. When would you use this? Here are three examples:

1. **Connecting to a custom application.** Perhaps you have written an application that listens on TCP port 8076 for connections from some client. Then, data from the connection is processed directly by your application and replies from your application are sent to the connection. This has been used for data collection from RS232 instrumentation. The instruments are connected to serial ports on the IntelliServer which runs **`telnet -RC`** sessions to send the data to the host application. This is especially useful because it allows the devices to initiate the connection, as when the application is designed to support a dozen modems into which dial hundreds of branch offices for data-collection purposes.
2. **Accessing standard TCP services for which there is not already a specific IntelliServer command.** For example, there is no IntelliServer *finger* command, although the IntelliServer does respond to finger requests from other hosts. Try the command **`telnet`** *`hostname`* **`79 -RC`**. TCP port 79 is the well-known port for *finger* service. After *telnet* reports that the connection

has come up, press **enter**. The connected host replies with host-specific status information. In Example 16-4, the IntelliServer has sent the finger request to itself.

Example 16-4: Impersonating The "Finger" Command

```
telnet jeeves 79 -RC
trying jeeves (Raw TCP Connection) (160.77.99.201)...
(hit enter again)
Welcome to the Computone IntelliServer "jeeves"
Running cnx kernel release 1.3.0, version 951031
port  session   owner   command
0     0         root    telnet 160.77.99.203
1     0         root    init awaiting DCD
Systat: 0% user, 3% system, 96% idle, 666K free
Up for 7 days, 7 hours, 7 minutes, 7.7 seconds
connection closed by remote host
#
```

3. **Connecting to a Reverse-TCP port on the IntelliServer**, when its TCP option is set to *Raw* (see page 87). If a different TCP option had been configured (*Normal* or *CRNL->CR)*, then the **-R** or **-RC** option would not be used.

Break Key

When the standard telnet escape key has been disabled (as when in telnet binary mode or when the **-E** option is used) your terminal's **break** key performs the telnet escape function. When a data line is inactive, it is said to be in the "mark" condition. When a break is sent, the data line goes into the *other*, or "space" condition for a quarter-second or longer. This duration is so long that it is not mistaken for a valid character being transmitted. It is a special condition, similar to what you would obtain if the data line were disconnected.

The break condition does not occur during normal data transmission, file transfers, and the like. Using it as the telnet escape key when in binary mode allows you to send arbitrary data across the connection without concern for whether the telnet escape code might be contained. It won't.

If you are logged into the IntelliServer through telnet (see "Telnet access to the IntelliServer" on page 158) rather than using a serial port, you cannot use the break key for this purpose. First, there might not *be* a break key where you are sitting. Secondly, even if there is one, your host's telnet will not propagate this condition to the host (IntelliServer) *it* is connected to.

Flow Control

Telnet uses whatever flow control has been configured for the serial port. If you are running in telnet binary mode and transferring data that might contain XON or XOFF characters, you want to use CTS/RTS flow control, rather than XON/XOFF (see "Flow Control" on page 60).

Rlogin

Rlogin is traditionally used to log into a remote host in order to run interactive terminal sessions. In a more general sense, rlogin communicates with a host over the network using the Internet *rlogin* protocol.

Example 16-5: Rlogin Command

`rlogin` *`hostname|ip-address options`*

```
# rlogin 160.77.99.100 -t wy60
# rlogin abercrombie.computone.com
# rlogin arrow.computone.com -l fletcher
# rlogin 160.77.99.101 -8 -l “”
```

Rlogin Arguments and Options

Command-line arguments are used to specify which host you want to access and are used to specify login names, terminal types, and other optional configurations.

Table 16-3: Rlogin Arguments and Options

Arguments…	
none!	If the rlogin command is given with no command-arguments or options, it prompts you: **`host name and arguments:`** You are to type the host name and arguments as you would have after the word *rlogin* on the command line.
`hostname`	The name or IP address of the host. If a host name is used, the IntelliServer must resolve it using its host table or external nameservers.
`ip-address`	
Options…	
`-t` *`termtype`*	Allows you to specify a terminal name to the remote host. If you do not supply this option, *Remote Terminal Type (page 85)* configured for your port is used; but if *that* is blank, then the *Local Terminal Type (page 84)* is sent. The host sets up its TERM environment variable based on the terminal type you send. This enables screen-based applications to run properly. The terminal name you send needs to correspond to an appropriate entry in your host’s */etc/termcap* or *terminfo* databases.

Table 16-3: Rlogin Arguments and Options (Continued)

-e*c*	Once you have established a connection to a host, typing the characters **~.** (tilde, dot) *at the start of a line* causes the connection to be closed and rlogin to exit. This option allows you to specify a different character, other than the *tilde*. For example, if you used the option **-e+**, then the sequence **+.** (plus, dot) would cause you to close and exit. This is intended to allow you to define a different escape character in case you needed to send *tilde* to the remote host, but suffers from the disadvantage that the *new* character is now compromised.
-8	Binary mode: This disables the rlogin escape sequence entirely. Keyboard input is no longer scanned for **~.** sequences or anything else. In binary mode, the **break** key closes the connection and exits. Refer to the section labelled "Break Key" on page 406 for more discussion. When the -8 option is not used, the break key has no effect.
-n	Reserved
-d	Reserved
-l *username*	This option allows you to specify a login user name. By default, the name by which you logged into the IntelliServer is used. When rlogin establishes a connection, it immediately provides a login name to the remote host. It can bypass prompting for a host name and immediately prompt for a password. (Should the login fail, many hosts prompt for a new login name).
-l ""	Sometimes it is desirable to start with a login prompt from the host, not a password prompt. When you specify a NULL login name (represented by a pair of double quotes), *some* hosts immediately prompt for a login name, realizing that there is not one already. This is not universal; some hosts' rlogin daemons start with a password prompt anyway.

Using Rlogin Connections

When an rlogin connection is open, data from the keyboard is sent to the remote host and data from the remote host is sent to your terminal.

Rlogin accesses the remote host using the standard TCP port 513. It communicates with a process (called *rlogind* on most UNIX systems) which logs you into the host system and offers you the appropriate services based on your login. As with *telnetd*, UNIX systems accomplish this using *pseudo-TTY* devices which are discussed in detail on page 402. The intrinsic capabilities and limitations of pseudo-tty devices are equally true when rlogin is used, as they are for telnet.

Flow Control

Rlogin uses whatever flow control has been configured for the serial port. If you are running in rlogin binary (-8) mode and transferring data that might contain XON or XOFF characters, you want to use CTS/RTS flow control, rather than XON/XOFF (see "Flow Control" on page 60).

Syslogging

Like telnet, rlogin sends syslog messages of LOG_NOTICE priority whenever a connection is established or disconnected.

Telnet and Rlogin Compared

This table summarizes some of the differences between the telnet and rlogin commands.

Table 16-4: Telnet and Rlogin Differences

Issue	`telnet`	`rlogin`
TCP Port	Default 23, can override	513, no override
Specify a terminal type on command line?	Yes	Yes
Specify a user name on the command line?	No - no user name is passed to the telnet daemon: the host prompts for a login name.	Yes - a user name is passed to the rlogin daemon; one given on the command line or the user's own login name. Presuming the login name, the host usually prompts for the password first.
Escape	Single telnet escape character puts you into telnet command mode. From command mode you can close the connection.	Sequence of two characters, and only at the beginning of a line, closes the rlogin connection and exits you from rlogin.
Command Mode	Telnet has a command mode.	Rlogin does not have a command mode.
Raw TCP mode	Supported	Not supported (rlogin without rlogin protocol is the same as telnet without telnet protocol)
Binary mode	Using -8 command-line option or command-mode **`binary`** command	Using -8 command-line option.

How To Start Connections

There are several ways to start a telnet or rlogin connection to a remote host:

- Type the telnet or rlogin command from the IntelliServer's command line.
- Enter a connection manually from the menu system.
- Select a connection from the global connection menu.
- Select a connection from this user's selected connection menu.
- Have the connection start automatically when the user logs in.
- Have the connection start automatically without requiring a user to log in.

To get the appropriate behavior, you need to properly configure the user that logs in, and the serial port that the user logs in on. These issues are discussed in chapter 5, *Configuring Serial Ports* and chapter 7, *Configuring Users*, but Table 16-5 on page 413 is a handy summary of what is important.

Notes to Table 16-5

- To enter connections from the command line, the user must get the "shell" when he logs in. Configure the user as *Direct Connect per Screen* and specify *shell* as the user's selected connection for each screen.
- To enter a connection directly from the main configuration menu, specify *menu* as this user's selected connection, or let him type **`menu`** from the shell. From the main menu he would select *Connections*, then either *Telnet to Host* or *Rlogin to Host.* Then, he is prompted to supply the host name and other command-line arguments.
- To start a connection as soon as the user logs in, configure the user as *Direct Connect per Screen* Instead of specifying *shell* as the selected connection, specify the command, (telnet or rlogin) host name, and the options you would have typed from the command line.
- Users configured as *Selected Connection Menu* are presented a menu of up to eight connections you have configured for that user. Users configured as *Global Connection Menu* are presented a menu of all connections configured for all users.

Table 16-5: Behavior Modification

Behavior	Connection Option (see page 124)	Selected Connections (see page 128)	Port Type (see page 67)
Enter connection from command line	Direct Connect per Screen	shell	Login by Port, wait Login by Port/TCP Login by virtual screen
Enter connection manually from the menus	Direct Connect per Screen	menu *Main— Connection— Telnet (Rlogin) to Host*	
Global Connection Menu	Full Connection Menu		
Selected Connection Menu	Selected Connection Menu	*Enter up to eight selected connections. Each connection consists of the command and arguments you would use to access this host.*	
Start connection at login	Direct Connect per Screen		
Start connection without login	*Configure the user as any of the above, and enter the user name into the port configuration. When an Auto-login port starts up, it proceeds as though that user had already logged in.*		Auto-login Auto-login, wait *Also set the user name (page 69)*

Example — Automatic Access To A Host

You have a modem on port 4, and you want to dedicate it solely to one of your network hosts. Anyone who dials in immediately gets a login message from that host. The user can either log in or disconnect but cannot do anything else on the IntelliServer. Suppose that host's IP address is 160.77.99.35

- Create a user (say, *leary*) and configure it as *Direct Connect per Screen.*
- Set its first (and only!) selected connection to *telnet 160.77.99.35.*
- Configure port 4 as a modem port, Auto-login, and set the port's user name to *leary.*

Because the port is Auto-login, the IntelliServer does not issue any login prompt; *leary's* connection is started automatically. Since the connection is a telnet one, the remote host prompts for a login name and password.

Example — Nice Menu

If anyone logs into the IntelliServer as "redactor", you want them to get a menu of two selections "Payroll" and "Accounts Receivable". Each choice causes the user to be logged into a host machine and start the appropriate application there. The following applies:

- The user needs to log into a port *somewhere*, so assume there is a terminal connected to serial port 5. Configure it as Login-by-Port, because you do not want to give just anyone this menu.
- Create a user, *redactor*, and configure it as *Selected Connection Menu*.
- Set up this user with two selected connections. The command will be *rlogin*, and you will specify a host and login user name as appropriate, for example:

```
rlogin 160.77.99.37 -l prman #Payroll

rlogin 160.77.99.38 -l arman #Accounts Receivable
```

Note that comments have been included after all the command-line arguments. In the selected connection menu, anything appearing after the pound sign (**#**) appears in the menu instead of the command and arguments themselves (see "Connection Comments" on page 126)

CHAPTER 17

User Authentication using RADIUS

In this chapter you learn how the IntelliServer uses RADIUS protocol to authenticate user logins and control the availability of services. You also learn how RADIUS accounting is used to keep track of user logins and logouts.

Introduction To RADIUS

RADIUS stands for *Remote Authentication Dial-In User Service*, and is an IETF draft standard. By supporting this standard, different manufacturers can produce products that are interoperable within a single system.

RADIUS protocol defines the communication between a RADIUS *client* and a RADIUS *server.* The RADIUS *client* is the IntelliServer or some equivalent product and is the device that users would be logging into. The RADIUS *server* is a computer somewhere on your network running *RADIUS* server software. This software listens for requests from RADIUS clients and sends replies.

RADIUS consists of two parts, Authentication and Accounting:

1. RADIUS Authentication occurs when a user tries to log into the RADIUS *client*. After prompting the user for login name and password, the client sends this information in an *authentication request* to the RADIUS server. The RADIUS server checks the validity of the request, then checks its database of user names and passwords. If they are bad it sends a *rejection* back to the client, which in turn rejects the login. If the login name and password are good, the RADIUS server sends back a packet containing information about this user and the client (i.e., the IntelliServer) uses this information to decide what type of service to supply for the user.
2. RADIUS Accounting occurs when a user logs into or out of a RADIUS *client* after approving the login (either through an internal database or through RADIUS authentication). The client sends notification to the accounting server that this particular user has logged in. When the user logs off or is disconnected, the client also sends notification including the number of seconds the user was connected. When the RADIUS Accounting server receives these notices, it stores the information and then sends an acknowledgment back to the *client*. If the client does not receive an acknowledgment for its notices, he assumes they were lost and sends out duplicates.

You can do RADIUS authentication without doing accounting, or accounting without authentication. If you are doing both, the accounting server can be the same host or a different one from the authentication server. Secondary authentication and accounting hosts can also be defined which the IntelliServer uses when there is no reply from the primary servers.

RADIUS Advantages

RADIUS protocol offers several advantages to the IntelliServer administrator:

- Up to 112 users can be stored in the IntelliServer's NVRAM, but your RADIUS server can store as many users as its disk storage permits.
- If you are running *lots* of IntelliServers, you do not have to configure all your users on each of the IntelliServers. Just configure them once on your RADIUS server and let all the IntelliServers authenticate their logins from the same place.
- The RADIUS user data base can store more user-specific information than the IntelliServer's NVRAM. This is particularly important with PPP/SLIP/CSLIP users, for whom it is often desirable to define user-specific IP addresses, routes, and IP filters. Without RADIUS, these settings cannot be assigned to specific users, only to specific Remote Profiles.

IntelliServer Configuration

In order to support RADIUS, you need to do some configuration on the IntelliServer to specify the IP addresses of any RADIUS authentication and accounting servers you want to use, and to specify a *secret* which is used to validate requests. The necessary configuration is discussed in the section, "RADIUS Configuration" on page 139 and following.

RADIUS Server Configuration

The details of installing and configuring the RADIUS server software depends on what version you use. There are now a number of versions available; some are free, some are not. An excellent *free* implementation is available, with source, via anonymous FTP from **`merit.edu`**. There is also a versions available on our Computone's FTP site (**`ftp.computone.com`**). These versions are distributed as compressed *tar* files. After you extract the contents there are one or more files that describe how the software is used.

In this section, what you need to configure on your RADIUS server is explained but no attempt is made to tell you *how* to do it under every implementation. The Merit implementation now available is used for some of the examples, but future implementations may do things differently and you'll have to allow for this as you read.

Regardless of the implementation, there are a few things you always have to configure:

1. **A list of authorized clients and their shared secrets**. The RADIUS server needs to know the IP addresses of all the authorized RADIUS clients. Along with each client's address is a *secret*. You can pick whatever you like, but this same secret has to be configured into the *client* (IntelliServer) as well (see page 143). The RADIUS client and server use the secret to encrypt parts of the packets they send each other, and to guarantee that the messages and replies are authentic. Your RADIUS server might store this list in a text file and in Merit's implementation this is a file called *clients*.

2. **A list of authorized users and their configuration information.** The RADIUS server needs to know which users have what passwords and what these users are authorized to do after they log in. In Merit's implementation, this is a text file called *users*. Each user is listed along with password (or an indication that the UNIX password file should be consulted), and any restrictions as to which IntelliServers or serial ports the user may be allowed to log in from. Information about the user is stored as a list of RADIUS protocol *attributes* and their associated *values*. These translate directly into the authentication reply the server sends back to the client.

RADIUS Protocol — Overview

You probably do not need to understand RADIUS protocol completely to get authentication and accounting up and running. The more you know, however, the better prepared you are to deal with the unexpected. The complete description of the protocol is found in the draft standard, available from the IETF archives. There is also a copy in Computone's FTP site —ftp.computone.com — but it is not guaranteed always to be the latest version.

RADIUS protocol is designed to support a wide variety of features. Not every RADIUS client and server support every feature defined in the RADIUS protocol. The intent is that features which *are* supported, are supported in much the same way from one manufacturer to another. This section is intended to provide enough information so that when you configure a RADIUS user that logs in, the right things happen.

Packets

RADIUS packets are sent using UDP protocol. The packet itself consists of a RADIUS header followed by a list of attributes.

- The **RADIUS header** includes a code telling what type of packet it is, a field giving the length of the entire packet, and an *authenticator*, which together with the shared secret is used to verify that the packet itself comes from an authorized source.
- Each attribute in the **attribute list** has an *attribute number* to identify it and an associated *value*. Depending on the attribute, the value can be a number or a string of characters. Each attribute also has a length associated with it, so that it is known where one attribute ends and the next one begins.

Attributes & Values

If you want the right thing to happen when your user logs in, the RADIUS server has to send the right set of attributes and their values as part of the *accept* packet returned to the client. In the simplest RADIUS server implementations, there is a *dictionary* file that translates the numerical values of attributes and values into keywords. In the *users* file you use these keywords as you list the attributes and values you want to have sent for each authorized user.

Radius Packet Types

Each RADIUS header contains a field that identifies the packet type. Typically, different packet types are handled by different agents within a RADIUS client or server. Table 17-1 shows the five types of packets the IntelliServer supports:

Table 17-1: RADIUS Packet Codes

1	**Access-Request**	Sent from the RADIUS *client* (IntelliServer) and contains the user's login name and encrypted password. Can this user log in?
2	**Access-Accept**	Sent from the RADIUS *server* in response to an Access-Request if the user is authorized to log in. This packet contains attributes that define what happens to this user next.
3	**Access-Reject**	Sent from the RADIUS *server* in response to an Access-Request if the user is not authorized to log in. This packet may contain *messages* for the user: ("Pay your bill!")
4	**Accounting-Request**	Sent from the RADIUS *client* when users log on and off or are disconnected.
5	**Accounting-Response**	Sent from the RADIUS *server* in reply to Accounting-Requests so that the client knows that its request was received and processed.

There is an important difference between Access requests and Accounting requests:

- After sending an Access-Request, the RADIUS client (e.g. IntelliServer) must wait for a reply before operations on that port can continue. Until the reply is received, the client does not know whether the user is authorized to log in, or what service to provide. If no reply is received, additional requests are sent (to a secondary RADIUS server as well, if defined) and if there is still no reply within a few seconds, access to this user is refused.

- After sending an Accounting-Request, the RADIUS client must wait for a reply confirming that the accounting server received this packet. But, in this case it continues to use the port while it is waiting for a reply. For example, if a user logs in and is properly authenticated, its sessions begin immediately. They don't wait for the accounting request to be sent and acknowledged. Accounting requests are also sent when a user disconnects. The port is immediately available for the next login and it doesn't wait for this request to be acknowledged, either. If Accounting-Requests go unanswered, the IntelliServer sends duplicate requests for several minutes until they are. If there is *still* no response, it sends a *syslog* error message to warn you.

Authentication

Special codes called *keys* or *secrets* are used by the RADIUS client and server to ensure that the packets it receives come from an authorized source. Configuring the IntelliServer's *Radius* and *Accounting Secrets* was described on page 143.

RADIUS clients check Access-Request packets to make sure they are from an authorized source. For example, suppose the IntelliServer sends an Access-Request packet to Merit's RADIUS server ("*Merit").* The sender's IP address is available for such packets, so *Merit* checks a *clients* file to see whether that IP address is listed there. If it is, then it uses the *key* (also stored in the *client's* file) to ensure that the request is authentic. If the IntelliServer which sent the packet did not have a matching key, *Merit* will know, and will not send any reply to the request. (It logs the event as a security violation, however.)

If the sender's IP address checks out and the request was generated with the proper key, the RADIUS server then checks the user against its *users* file. If the user's name and password check out, it sends an Access-Accept packet; otherwise, an Access-Reject.

The RADIUS client (e.g., IntelliServer) uses the key to authenticate any Access-Accept or Access-Reject packets it receives. Without knowing the keys, someone cannot set up a bogus RADIUS server to intercept authorization requests or send unauthorized approvals.

RADIUS Attributes

When you are ready to configure users, the most important thing to understand are the RADIUS attributes and their meanings. The body of each RADIUS packet consists of lists of attributes and their values. In an Access-Accept packet, it is the collection of attributes that define the type of service this user should receive.

Different RADIUS server implementations are going to differ in the way their *user* data-base is handled, but they define *their* implementation in terms of the RADIUS attributes and values. If your implementation is defined in terms of the same attributes, you should be able to sort out easily how to do what is needed.

Table 17-2 is a list of the RADIUS attributes and their related values that the IntelliServer supports. The *attribute names* are taken from Merit's *dictionary* file. Attribute names may vary a little from one implementation to another while the numerical attribute values are fixed by the protocol. The names are, however, more descriptive and easier to remember. The full RADIUS protocol may contain other attributes which correspond to features not supported by the IntelliServer. After you look through the table for an overview, some of the attributes are discussed in more depth.

Table 17-2: RADIUS Attributes

Value	Attribute Name	Value Name or type	Description
Attributes Sent in Access-Request packets:			
1	**User-Name**	string	The login name supplied by the user.
2	**User-Password**	string	The encrypted password supplied by the user.
4	**NAS-IP-Address**	address	The IP address of the RADIUS client (IntelliServer) that the user is trying to log into.
5	**NAS-Port**	integer	The serial port number the user is logging into.

Notes:

- Attribute and Value Names are as they appear in Merit's RADIUS dictionary file.
- The numerical values for each attribute and value are set by RADIUS protocol and remain constant regardless of the RADIUS server implementation.

Table 17-2: RADIUS Attributes (Continued)

Value	Attribute Name	Value Name or type	Description
Attributes sent in Access-Accept packets:			
6	**Service-Type**	What type of service should this user receive?	
		(1) Login	User is logged onto a remote host.
		(2) Framed	User brings up a PPP, SLIP, or CSLIP link.
		(6) Administrative-User	User gets the IntelliServer Command prompt.
		(7) Exec-User	User gets a menu of possible connections.
	The following attributes can be used when User-Service-Type is Framed-User:		
7	**Framed-Protocol**	What protocol to use?	
		(1) PPP	Bring up a PPP link.
		(2) SLIP	Brings up a SLIP link (for CSLIP, specify SLIP here, and VJ compression below).

Notes:

- Attribute and Value Names are as they appear in Merit's RADIUS dictionary file.
- The numerical values for each attribute and value are set by RADIUS protocol and remain constant regardless of the RADIUS server implementation.

Table 17-2: RADIUS Attributes (Continued)

Value	Attribute Name	Value Name or type	Description
8	**Framed-IP-Address**	address	The IP address of the *user's* side of the SLIP or PPP connection. (From the IntelliServer's perspective, the Remote Address). Two addresses have special meaning:
	255.255.255.255 — Allow the user to supply its address when the link is brought up. For PPP, this implies PPP address negotiation. For SLIP connections, the user is prompted to enter an address.		
	255.255.255.254 — The IntelliServer should assign its Remote IP address from a pool. The pool consists of all the Remote Profiles which were configured with valid Remote Addresses (see page 279).		
	Other IP addresses are assigned as the Remote IP Address for the PPP/SLIP connection this user is bringing up. Read the section, "Assigning Remote Profiles" on page 290 for details on how the Framed-Address specified here is related to the Remote Addresses you configure into the IntelliServer's Remote Profiles.		
10	**Framed-Routing**	How are RIP packets handled on this interface? (If this attribute is not specified, the RIP option in the IntelliServer's Remote Profile applies).	
		(0) None	Do not broadcast RIP packets to this interface, nor listen for them.
		(1) Broadcast	Broadcast RIP packets on this interface, but do not listen for them.
		(2) Listen	Listen for RIP packets, but do not send them.
		(3) Broadcast-Listen	Broadcast RIP packets and listen for them on this interface.
11	**Filter-Id**	string	The name of an IP Filter defined on the IntelliServer, to be attached to this interface. If one is not specified, then any IP filter defined in the Remote Profile applies.

Notes:

- Attribute and Value Names are as they appear in Merit's RADIUS dictionary file.
- The numerical values for each attribute and value are set by RADIUS protocol and remain constant regardless of the RADIUS server implementation.

Table 17-2: RADIUS Attributes (Continued)

Value	Attribute Name	Value Name or type	Description
12	**Framed-MTU**	integer	Maximum Transmit Unit for this interface. If not specified, the one in the Remote Profile applies.
13	**Framed-Compression**	**(0) None**	No IP header compression is used.
		(1) Van-Jacobson-TCP-IP	VJ header compression is used. If not specified, the setting in the Remote Profile is used.
22	**Framed-Route**	string	A list of any network or host addresses which should be routed through this link's remote address. Multiple destinations are separated by spaces and subnetted networks are indicated using **/nn** notation.
		For example *Framed-Route = "160.77.128.0/17 160.77.99.2"* adds a route to the first address, a subnetted network, and the second address, a host. The destination of each route is the link's Remote Address.	
	The following attributes may be used when the User-Service-Type is Login-User:		
14	**Login-IP-Host**	address	The IP address of the host this user wants to log into.
15	**Login-Service**	How will the user log into the remote host?	
		(0) Telnet	User will telnet to the host.
		(1) Rlogin	User will rlogin to the host.
16	**Login-Port**	integer	If specified, the TCP port to be used for a telnet connection for this host.
219	**CTON-Argument**	string	All command-line arguments that apply to the telnet or rlogin. If several arguments are needed, they are to be separated by spaces and the whole thing enclosed in double-quotes.

Notes:

- Attribute and Value Names are as they appear in Merit's RADIUS dictionary file.
- The numerical values for each attribute and value are set by RADIUS protocol and remain constant regardless of the RADIUS server implementation.

Table 17-2: RADIUS Attributes (Continued)

Value	Attribute Name	Value Name or type	Description
	The following attributes may be used in any replies:		
18	**Reply-Message**	string	A reply message is usually sent as part of an Access-Reject packet, but could be sent in an Access-Accept packet as well. Any message specified here is displayed to the user.
25	**Class**	string	The string specified here is remembered and sent to the accounting server in any Accounting-Request packets.
Attributes sent in Accounting-Request packets:			
1	**User-Name**	string	User's login name. (If accounting is enabled, accounting requests are sent for both NVRAM and RADIUS users).
25	**Class**	string	If this attribute was supplied by the RADIUS server when the user was authenticated, this information is sent to the accounting server in the Accounting-Request packets.
40	**Acct-Status-Type**	**(1) Start**	Indicates that the user has logged onto the IntelliServer.
		(2) Stop	Indicates that the user has logged off the IntelliServer. (Includes all forms of disconnection).
41	**Acct-Delay-Time**	integer	The elapsed time in seconds from when the user logged on or off and when this Accounting-Request was sent.
44	**Acct-Session-Id**	integer	An arbitrary number to help you match up Start and Stop packets. Start and Stop packets with the same Acct-Session-Id are presumed to refer to the same login session.

Notes:

- Attribute and Value Names are as they appear in Merit's RADIUS dictionary file.
- The numerical values for each attribute and value are set by RADIUS protocol and remain constant regardless of the RADIUS server implementation.

Table 17-2: RADIUS Attributes (Continued)

Value	Attribute Name	Value Name or type	Description
45	**Acct-Authentic**	**(0) None**	*Not used*
		(1) RADIUS	User was authenticated by a RADIUS server.
		(2) Local	User was an NVRAM user.
46	**Acct-Session-Time**	integer	Sent with accounting "Stop" notices. Gives the elapsed time the user has been on.

Notes:

- Attribute and Value Names are as they appear in Merit's RADIUS dictionary file.
- The numerical values for each attribute and value are set by RADIUS protocol and remain constant regardless of the RADIUS server implementation.

Access-Request Attributes

In addition to the *User-Name* and *User-Password* attributes, an Access-Request packet contains the *NAS-IP-Address* and the *NAS-Port.* The RADIUS server can use this information to authenticate users that have restricted access through only certain IntelliServers or through only certain serial ports.

Service-Type

Every Access-Accept reply should contain one instance of this attribute, as described below:

Table 17-3: Service Type Attributes

Service-Type	Description
Login User	This user will automatically be opened a telnet or rlogin connection to a host. The *Login-Service* attribute tells whether telnet or rlogin is used and the *Login-IP-Host* supplies the host's IP address. *Login-Port* specifies the TCP port to use for the connection (if the standard one is not used). This is equivalent to configuring an IntelliServer NVRAM user as *Direct-Connect-per-Screen* (page 124), and setting its connections to rlogin and telnet sessions. If there are multiple *Login-Service* and *Login-IP-Host* attributes defined, each set is assigned to a different virtual screen, the same way that connections are assigned for NVRAM users
Framed User	This user will automatically open a PPP or SLIP connection over the serial port on which he logged in. The characteristics of this connection are given by other attributes, e.g., *Framed-Protocol* to indicate whether it will be a PPP or SLIP link, *Framed-IP-Address* to specify the IP address at the user's end of the link, and so on. This is equivalent to configuring an NVRAM user as *Inbound-PPP, Inbound-SLIP,* or *Inbound-CSLIP* (page 126)
Administrative User	This user will be given an IntelliServer command prompt. This is equivalent to configuring an NVRAM user as *Direct-Connect-per-Screen* (page 123), and setting his connection to be a shell. From the IntelliServer's command line, a user can attempt to telnet or rlogin to any host, so this is intended for the more administrative of your users.
Exec-User	This user is handled in almost the same way as *Login* users. The goal is to open a telnet or rlogin session with another host. But this user is like an NVRAM user configured as *Selected Connection Menu* (page 124). If there are multiple *Login-Service* and *Login-IP-Host* attributes defined, they are not assigned to virtual screens one per each. Instead, the user is presented with them all as a menu of possible connections. This menu can only contain telnet or rlogin connections specified through *Login-Service* types; it cannot contain options to bring up PPP or SLIP connections.

For connections to a user which has its own IP address, you will probably provide that address here. If the user is known to have a network behind that address, you give the network address in the *Framed-Route* attribute. (If you are connected to the Internet, you and anyone you route to must have registered IP addresses.

For temporary Internet connections to single hosts that does not have registered IP addresses, you generally set aside some IP addresses from your own (registered) network and let these users borrow them while they are connected. To do this, configure a pool of Remote Profiles to use these addresses as their *Remote Addresses*. If you are using addresses from the IntelliServer's local network and keep proxy ARP enabled, other hosts route to these user's sites as though they were on your local network.

Generally, users who have to borrow one of your IP addresses are generally single hosts without any other network traffic to route through them. If there *were* a network behind this user's host, it would have to be using registered addresses. Otherwise, multiple hosts with the same (unregistered) IP address could simultaneously appear at different locations on the Internet. But if the user has a registered network, it could have used one of its *own* addresses as the *Framed-IP-Address*. If the user at the other end is an *IntelliServer* configured for an outbound-PPP/SLIP/CSLIP connection, it must use its own IP address. You cannot assign it a temporary address to use from a pool.

Framed-Routing

Usually, the problem is notifying your local network (and points beyond) how to reach networks behind your PPP/SLIP connections. If you are an ISP, then your users have *default* routes through their PPP/SLIP connections to you, so there is no need to provide them with any more detail. On the other hand, if they have multiple transient networks behind their connection, you need to keep the world informed as to where *those* are. In that case, you probably want to be listening, but not broadcasting, on the connections to your dial-up PPP/SLIP users. Contrariwise, you probably want to broadcast on your local (Ethernet) network, and on any PPP/SLIP connections heading *outbound*.

Login-IP-Service

This attribute can appear more than once. Each time it appears it represents a new connection definition. Each connection defines a telnet or rlogin session to a particular host (and to a particular TCP port, using particular command-line arguments). If the user is a *Login* user, each separate connection is assigned to a virtual IntelliView screen. If the user is an *Exec-User*, then the user can choose from a menu of all the connections defined for him. Table 17-4 shows how you might specify a telnet connection to each of two hosts.

Table 17-4: Defining Multiple Login Hosts

```
Service-Type = Login
Login-Service = Telnet
Login-IP-Host = 160.77.99.101
Login-Service = Telnet
Login-IP-Host = 160.77.99.102
```

CTON-Argument

This is not a standard attribute, so you may need to modify your RADIUS server to support it. For the *Merit* implementation, this involves making a change to the *dictionary* file. The *CTON-Argument* replaces the *Login-IP-Host* and *Login-Port* attributes, gaining you flexibility at the cost of interoperability.

The gain in flexibility comes in two ways. First, the IntelliServer's *telnet* and *rlogin* commands support numerous command-line options which can be accessed through RADIUS in no other way. For example, you can default a telnet or rlogin connection to *binary* mode by using the *-8* command-line option. You can specify terminal types to *telnet* using the *-t* option and an alternate login name to *rlogin* using the *-l* option. For a complete list of options, see Table 16-1, "Telnet Command-Line Arguments and Options" on page 397, and Table 16-3, "Rlogin Arguments and Options" on page 408.

The second gain in flexibility comes because of the way the *Login-IP-Host* is sent to the IntelliServer. The numerical IP address is already sent. If your RADIUS server implementation allows you to specify a host name, the RADIUS server uses name resolution to get its IP address and then sends the address. This is fine for *Login* users, since they won't be looking at the address anyway. For *Exec-Users*, however, the connection type and IP address would appear in a

selected connection menu. This might not be the most user-friendly presentation possible.

If you use the CTON-Argument, then whatever string you supply is sent to the IntelliServer, which treats it the same as an NVRAM user's connection host and arguments (page 135).

The loss of interoperability is on two fronts: first, other RADIUS clients besides the IntelliServer are not apt to support the CTON-Argument attribute (and, if they did, the interpretation of command-line arguments would likely be different). Second, you may need to configure your RADIUS server specifically to support it.

Class

This seems to be a trick for sending information from the RADIUS authentication server to the authorization server (which might be running on separate hosts), via the RADIUS client. When a user is logged in, the *class* may be supplied as one of the attributes in the Access-Accept packet sent from the RADIUS server. When the RADIUS client sends accounting requests, it includes this class as well.

Acct-Session-ID

This is to help you associate start records with stop records. Presently, it is an ascii string of the form `NN.P.S`, where *NN* is a sequence number that starts at 0 when the IntelliServer is rebooted and increases by one each time a user logs in. *P* is the port number, 0 through 63, and *S* is the session number, 0-7 (but almost always this will be 0).

Caveat: The sequence number is reset to zero any time an IntelliServer is rebooted.

Acct-Session-Time

When a user logs off, the Accounting-Request of status-type *Stop* contains one of these attributes, which gives the elapsed time in seconds that the user was logged on. Except for the users that are presently connected or that never disconnect, you could do billing based on the stop records alone.

Table 17-5: Attributes for a PPP User

```
pppu1   Password = “mypass”,
        Service-Type = Framed,
        Framed-Protocol = PPP,
        Framed-IP-Address = 255.255.255.254,
        Framed-Routing = Listen,
        Filter-Id = “myfilter”,
        Framed-Compression = None
pppu2   Password = “mypass2”
        Service-Type = Framed,
        Framed-Protocol = PPP,
        Framed-IP-Address = 160.77.99.22
        Framed-Routing= None,
        Framed-Route = “192.9.200.128/25”
```

In Table 17-5, the first user, *pppu1*, shows an IP Address of 255.255.255.254. When it logs in, the IntelliServer assigns it an IP Address taken from an unused Remote Profile’s *Remote Address.* The IP Filter named *myfilter*, defined on the IntelliServer, is attached to this interface. If RIP is running on the IntelliServer, it listens for RIP packets from this interface, but does not broadcast RIP packets there.

The second user, *pppu2*, shows an IP Address of 160.77.99.22, which is used as its IP address, overriding any that might be present in the IntelliServer’s Remote Profile. RIP is not be used on this interface, but it reports that the subnet 192.9.200.128/25 can be reached through that interface, so the IntelliServer adds the appropriate route when the link comes up.

Table 17-6: Attributes for a Login User

```
logu1   Password = “mxpass”,
        Service-Type = Login,
        Login-Service = Telnet,
        Login-IP-Host = 160.77.99.42
logu2   Authentication Type = Unix-PW
        Service-Type = Login,
        Login-Service = Rlogin,
        CTON-Arguments = “160.77.99.32 -8”
```

In Table 17-6, user *logu1* automatically starts up a telnet connection to 160.77.99.42 when it logs in. User *logu2* automatically starts up an rlogin connection to 160.77.99.32. Notice that the “experimental” CTON-Arguments attribute is used to force the 8-bit rlogin option. The Authentication Type of Unix-PW tells the Merit radiusd that it should consult the UNIX password file instead of expecting the password to be present in the users file.

CHAPTER 18

Reverse TCP and Printing

In this chapter you learn many things associated with serial ports configured for *Printer* and *Reverse TCP.* Some of the things you learn about are:

- Using the *telnet* command on your external host to communicate directly with an IntelliServer's serial port.
- Using *rcp* and *rsh cat* to send output to a serial port.
- Using *iservcat* to send output to a serial port.
- Using *iservd* to support a bidirectional link between a serial port and a process running on your host.
- Sending output to printers attached to terminals.

General Considerations

In earlier chapters, configuring the IntelliServer to perform two of the three most common uses of serial ports was discussed. These included:

1. **Login access to hosts on your local network -** A serial terminal or modem is attached to the IntelliServer to allow a user to log into network hosts using rlogin or telnet. The user then runs most applications as though it had been directly connected to that host via a serial port. Sometimes the "terminal" is really a PC running terminal emulation software. Sometimes the host application allows the user to download and upload files to their PC. Usually the login access is through the host's standard telnet and rlogin servers, but sometimes the IntelliServer is used to establish connections with special host software which emulates the action of rlogin or telnet and provides access to special software packages.
2. **Extending your network via PPP, SLIP, or CSLIP -** A serial port is configured to support dial-in or dial-out access to a similarly-configured host in a remote location, usually over modems or leased lines. When properly configured, hosts on the local and remote network can access each other as though they were on the same local network.

There is a third common use of serial ports which haven't been discussed yet:

3. **Access to serial devices from host software -** Your print spooler wants to send data to a serial printer. Your payroll system uses custom software to poll serial time-clocks. Local network users want to dial out over modems to access remote BBS systems.

Doesn't this last one sound like three more *different* uses? In a way they are, but they are connected by a common thread. In each case, software on the host initiates contact with the serial port on the IntelliServer. Using network terminology, the three classifications above can be restated as follows:

1. The IntelliServer runs a **client** (e.g. *telnet*) which accesses a server (e.g. *telnetd*) running on the host.
2. The IntelliServer acts as **neither** a client nor a server, but links two networks at the IP protocol layer.
3. The IntelliServer runs a **server** (e.g. *ttyd)* which is accessed by a host's client (e.g. *telnet).*

The mission of this chapter, then, is to explain how various client software on your host can access the services you can configure on your IntelliServer.

Standard Services

The IntelliServer provides services to network clients through two the rcp and telenet protocols. These are outlined in Table 18-1 below, from the perspective of the host's client software.

In the examples, **`isname`** is assumed to be the host name of an IntelliServer. The IntelliServer's IP address could be used in place of its host name, as well.

Table 18-1: IntelliServer Services — The Client's Perspective

Protocol	Service Port	Description	Examples of compatible client software on a UNIX host
rcp	TCP 514	Hosts can use the **`rcp`** or **`rsh cat`** command to send data to a selected serial port.	**`rcp`** *`file`* **`isname:12`**
			`cat` *`file`* **`| rsh isname cat 12`**

Table 18-1: IntelliServer Services — The Client's Perspective

Protocol	Service Port	Description	Examples of compatible client software on a UNIX host
telnet	TCP 23	Hosts can use telnet to log into the IntelliServer in order to do maintenance.	**`telnet isname`**
	TCP 9000…	Hosts can use telnet to access a serial port directly.	**`telnet isname 9012`**
			`iservcat isname 9012` *`file`*
			pseudo-tty's using iservd
	TCP 10000 …	Hosts can use telnet to access an available serial port from a group.	**`telnet isname 10005`**
			`iservcat isname 10005` *`file`*
			pseudo-tty's using iservd

NOTES:

1. The UNIX commands **`rcp`** and **`rsh`**…**`cat`** are generally used to copy files from one network host to another. Since the IntelliServer does not store files, the only destination file names are numbers which correspond to the serial ports.

2. If you *telnet* into an IntelliServer on the default port, you can log into the IntelliServer to perform maintenance (see "Telnet access to the IntelliServer" on page 139). If you specify a TCP port number from 9000-9063, your telnet connection is directly to a serial port. If you specify a TCP port number from 10000-10015, your telnet connection will be with a serial port from the designated group 0-15. (On page 86, you learned to assign an optional group number to a port; this is one of the reasons).

3. TCP connections designed for *telnet* protocol will of course communicate with your host's telnet command, but they also will communicate with any other process designed to use telnet protocol. We wrote two such examples, and ship them with the IntelliServers on our supplemental diskette: viz., **`iservd`** and **`iservcat`**. We supply C source to these two programs, and they are completely unencumbered: you may port them to any operating system you wish and you may use the source as an example to create derivative software for any purpose whatsoever. If you do make changes, we ask only that you make sure your users contact *you* for any related support. If they contact our support department and we don't realize they are using customized software, there could be confusion.

Port Configuration

Table 18-1 on page 437 summarizes the types of services that are available on the IntelliServer, but it does not explain how serial ports on the IntelliServer would be configured to supply these services. Telnet access through TCP port 23 does not require any special configuration, because it does not apply to any specific serial port. The remaining services are listed below in Table 18-2.

Table 18-2: Services — The IntelliServer Perspective

Service Port *Protocol*	Typical UNIX commands	Port Type: (see page 67)	Notes
TCP 514 *rcp*	**rcp** **rsh**…**cat**	**Printer** *IntelliPrint* **NOT:** **Reverse TCP** **Login by Port/TCP**	Ports configured as Reverse-TCP or Login-by-Port/TCP do not support connections via **rcp** and **rsh…cat.** Output only. File names **0-63** correspond to serial ports 0–63.
TCP 9000… *telnet*	**telnet** **iservcat** *iservd*	**Printer** *IntelliPrint* **Reverse TCP** **Login by Port/TCP**	TCP ports 9000–9063 correspond to serial ports 0–63.

Table 18-2: Services — The IntelliServer Perspective (Continued)

Service Port *Protocol*	Typical UNIX commands	Port Type: (see page 67)	Notes
TCP 10000… *telnet*	**`telnet`** **`iservcat`** *iservd*	**`Printer`** *IntelliPrint* **`Reverse TCP`** **`Login by Port/TCP`**	TCP ports 10000–10015 correspond to groups 0–15. A serial port may be assigned to group number 0–15, or to *none*.

Additional Notes:

1. In the table, *IntelliPrint* represents a port configured as either **`Login-by-Port`** or **`Login-by-virtual-screen`**, where an IntelliPrint profile (page 88) has been assigned to that port. A printer would be attached to the terminal's auxiliary port. Data from one of these connections to an *IntelliPrint* port would be sent to the printer through the terminal.

2. Telnet connections can specify a single IntelliServer port by using TCP port numbers 9000-9063, or the first available port from a group (see page 86) by specifying TCP port numbers 10000-10015.

3. Rcp connections can only specify a particular port, not a port group.

4. Ports configured as **`printer`** or *IntelliPrint* listen for **`rcp`** or **`rsh…cat`** connections, as well as TCP connections over ports 9000-9063 or 10000-10015. Ports configured as TCP or Login-by-Port/TCP do not support **`rcp`** and **`rsh`**…**`cat`** connections.

After studying Table 18-1 and Table 18-2, you should now have an idea which of the host's client software might be used with which IntelliServer port configurations.

Now by studying Table 18-3 you learn how the port configuration affects the behavior of the services:

Table 18-3: Port Configuration Controls Behavior

Where Does Output Go?	Supports Input And Output?	Port Type	Output Processing?	If port (or every port in the group) is already in use?
Terminal's Auxiliary Port	Output Only	**`Login by Port`** or **`Login by Virtual Screen`** using *IntelliPrint*	From Port Configuration and IntelliPrint Profile	Connection from client is accepted, but waits for port to be available.
Serial Port	Supports input and output if the client software does.	**`Printer`**	From Port Configuration (see page 82)	
		`Reverse TCP`	Port's output processing is disabled.	Connection from client is refused.
		`Login by Port/TCP`		

NOTES:

1. Since data for *IntelliPrint* ports is sent to the terminal's auxiliary port, the port is not considered busy just because a user is logged in and using the port. There would need to be already another *client* like this one accessing the port.

2. Since data for Login-by-Port/TCP ports is directed to the port itself, the port is considered busy if someone is currently logged in, or even if there has been a login prompt issued. If you want the port to ever be available for TCP connections you must configure it as a modem port. You wouldn't be configuring a port this way unless there was a modem connected, anyway. When the port is idle and there is no incoming call, carrier is low and it will accept TCP connections. When an incoming call comes in, carrier is asserted and a login prompt is sent to the caller. Subsequent TCP connections are refused until this incoming user is disconnected.

3. When a port is unavailable (busy or not configured) and an incoming connection is refused, the client making the connection may report it in various ways, depending on who it is. If you were using an interactive telnet session it would probably report, "connection refused".

4. Ports configured as TCP or Login-by-Port/TCP disable any output processing that was configured for that port but observes any processing implied by the port's TCP option (page 87). The other port types use any output processing you have configured. For example, when you are using the port to support a printer, you might want to configure it to automatically add carriage returns before linefeeds and to expand tabs, if this is not being done by the client software sending the data.

Examples:

If the light bulb has not yet turned on in your mind, let's consider some examples:

1. **From a host on your network, you want to dial out of a modem on the IntelliServer to reach someone's dial-in BBS.** Assume the modem has been correctly cabled to port 6 and that the port has been configured with the appropriate physical settings (line speed, etc.). Now configure the port as *Reverse-TCP*. Assume your IntelliServer's host name is *jeeves*. From your UNIX host, run **`telnet jeeves 9006`**.
2. **Like Example 18-1, but allowing users to dial into the modem and log into the IntelliServer when you aren't using the modem for dial-out.** The same configuration as the above, but configure the port as *Login-by-Port/TCP*, and make sure it is configured as a modem port. When the *carrier detect* (**DCD**) signal indicat**es** there is an incoming call, you cannot dial out. When you are dialing out, the modem won't accept incoming calls.
3. **You have a printer connected to port 8 and want to send data to be printed.** Configure port 8 with the correct physical settings for the printer. Flow control is especially important because printers can be slow. Configure the port as *Printer*. On the UNIX host, suppose the data you want to print is the output of some command; *slartibartfast*, for example—the name isn't important. You can use the host's *rsh…cat* command to send the data, thus: **`slartibartfast | rsh jeeves cat 8`**. The output of the command is piped to the *rsh* process, which then sends the data to port 8 of the IntelliServer called *jeeves*.
4. You have printers connected to ports 3 and 7. You want to print something to whichever printer is available first. Configure ports 3 and 7 as you did in Example 18-3. Configure each port as *Printer*, but now be sure to set the port's group number to 3, say. Now what client shall you use to send the data to this printer? Telnet is no good because it is interactive. Rcp and rsh…cat are no good either, because they use *rcp* connections, which don't support groups (see notes to Table 18-2). You need something that opens a TCP port using telnet protocol. There are some versions of UNIX whose print spoolers already support this type of interface, but assume yours doesn't. You will use the **`iservcat`** command described on page 451. This is a command which is shipped on the supplemental software diskette that accompanied your IntelliServer. Supposing what you want to print is in a file called **`/tmp/devil/twain`**. You would send the file using the command:
   `iservcat jeeves 10003 /tmp/devil/twain`.

5. **You are running lab software designed to communicate with RS232 interfaces on several pieces of test equipment.** The issue here is that the IntelliServer's serial ports have to correspond to specific devices on the network host. First, configure the ports as *Reverse-TCP*. Ensure the device is cabled correctly and the physical settings are correct. With some devices you might be able to "check the plumbing" at this point. Use *telnet* to send data to the device interactively and see what comes back. Next, set up a configuration file for the Computone command, **`iservd`**, described on page 455 and beyond. This file assigns device names on the host system that correspond to specific ports on the IntelliServer. Next, configure your system so that **`iservd`** runs automatically when your system comes up. Finally, configure your lab software to use the device names you added. This example is a bit more complicated than the others, and you aren't really expected to understand the details until after you have read the section, "Computone Clients: Iservd" on page 455.
6. **You want your UNIX system 's print spooler to send data to a printer attached to one of your terminals.** Configure the port as *Login-by-Port*. Create and assign an IntelliPrint profile based on your terminal type. Configure your UNIX spooler to send data to that port as you would for a printer connected directly to the IntelliServer's serial port.

If some of the hosts on your network are running UNIX, you have three commands which can act as clients to the IntelliServer; **`telnet`**, **`rcp`** and **`rsh`**.

Telnet

Since the IntelliServer itself has a **`telnet`** command, its features have been discussed at length (see "Telnet" on page 396). Other implementations of **`telnet`** may not have all the same command-line options, but most allow you to specify a host name and a service port on the command line.

You use **`telnet`** for two purposes:

1. To log into the IntelliServer to do maintenance.
2. To connect directly and interactively with an IntelliServer's serial port that is configured as Reverse-TCP.

When you do not supply a specific TCP port, **`telnet`** connects to the default telnet service on TCP port 23, as shown in Example 18-1. The IntelliServer sends a login prompt and after you log in correctly you are given a command prompt so that you can perform maintenance.

Example 18-1: Telnetting From Host To IntelliServer

```
telnet IServer-Name|IP-address
```

```
unix host# telnet jeeves
IntelliServer Release 1.3.0
jeeves login:
```

When you supply a TCP port number in the range 9000-9063, or 10000-10015, you are *not* logged on to the IntelliServer. Instead, what you type is sent to one of the IntelliServer's serial ports and anything that comes in the serial port is displayed on your screen.

In Example 18-2, TCP port 9003 is specified, which corresponds to serial port 3 on the IntelliServer. Presume that this port is configured as Reverse-TCP and that there is a modem attached. After logging in, there is no additional banner or login prompt. Knowing there is a modem there, the command **AT** is typed and the modem replies, **OK**. Then, it dials a certain number where it connects to a modem on the other end, and it reports with the message, **CONNECT**. Then, the remote system sends its banner message. Communications continues with the remote system in this way until finished and ready to log off.

Example 18-2: Telnetting from Host to IntelliServer Serial Port

telnet *IServer-Name|IP-address TCP-port*

```
unix host# telnet jeeves 9003
AT
OK
ATDT15551212
CONNECT
Welcome to the INFO-BBS...
```

Disconnection from the remote system can be done in one of two ways:

1. You can log off the remote site using the commands appropriate to that site. For example, if you were at a UNIX command line, you might type *exit* to tell that host you wanted to disconnect. Then, it would close the serial port connected to its modem, causing it to hang up and causing our modem to lose carrier. Assuming our IntelliServer's port is configured as a modem port (see page 78) it sees *carrier* drop and closees the TCP connection to the host it started with.
2. You could enter **telnet** command mode and type the **quit** command. When this host's telnet session exits, the TCP connection is closed. When the IntelliServer sees the TCP connection close, it shuts down the associated serial port and drops the DTR signal to the modem. This causes the modem to hang up, making the *remote site* lose carrier, shutting *it* down until the next incoming call.

The net result is the same: one side decides it must disconnect and the other side learns of it through the modem's data-set signals.

Rcp

The **rcp** *(remote copy)* command is used on UNIX systems to copy files from one system to the other. But, on UNIX systems character devices like printers and terminals have names just as files do and so it is natural to use **rcp** to send copy data to one of the remote system's devices, like a printer attached to a serial port. When you use your host's **rcp** command, it will try to connect to TCP port 514 on the IntelliServer.

Example 18-3: rcp Used To Send Print Data To IntelliServer

```
rcp filename IServer-Name:serial-port
```

```
unix host# ls -l > /tmp/is000
unix host# rcp /tmp/is000 jeeves:3
unix host# rm /tmp/is000
```

After connecting, it sends the name of the IntelliServer's file that it wants to copy to. In this case, the file name is the number of the serial port to which are data is being sent (see Table 18-2 on page 439).

Rcp can only send data to IntelliServer ports configured as Printer, or that have IntelliPrint profiles. Consequently, you cannot use **rcp** to send data to a port out of a group.

Rsh

Using **rcp** to send print data has one disadvantage: it is designed to copy files. Most implementations of **rcp** require the data to be first stored in a file on the local host, and then copied. In Example 18-3, the output of the **ls** command is dircted to a file, then copied to port 3 of IntelliServer *jeeves*, and finally the temporary file is deleted. This can be a problem if the thing you want to print is very long. What you need, then, is a way of sending the output of a process directly to the IntelliServer, without copying it to a file first.

Contrast Example 18-4 with Example 18-3 on page 446. Using **rsh**, it is not necessary to copy the data to a file first and then clean up by deleting the file afterwards.

Example 18-4: rsh Used To Send Print Data To IntelliServer

output-process **|** **rsh** *IServer-Name* **cat** *Serial-port*
unix host# ls -l | rsh jeeves cat 3

The **rsh** command stands for "remote shell" and was designed to allow one host to run commands on another host. Like **rcp**, it connects to the remote host on TCP port 514. By default, the only command the IntelliServer supports through the *remote shell* is the *cat* command, as shown. You may, however, use the IntelliServer's **rhosts** command to enable support for the other shell commands. This is explained under "Remote Shell Access" on page 159. On some hosts, this **rsh** command has a different name, possibly **remsh**, because **rsh** is also a common name for the "restricted shell", a completely different command.

Configuring Spoolers

A common question is, “How shall I configure my UNIX print spooler so it sends data to a printer connected to the IntelliServer?” By now, you have learned how you can configure your serial ports and you know a variety of methods for sending data to these ports. But how can you integrate these techniques into your existing print-spooling administration?

Because systems vary so greatly, there is no one complete answer. In this section a number of techniques are examined, but you have to decide which work best on your system.

Are You Spooling

The ultimate goal is to get printed output from applications you are running. These applications are sometimes designed to send data to your system’s print spooler, but other times they are designed to send data directly to some device, or pipe the data to a script you define, or to dispose of it some other way. The first step, if you have not already taken it, is to understand these application requirements. Another thing you should learn is how you specify (to the application) which particular printer should receive the data.

When you understand your application’s requirements, then you know whether the problem is one of configuring the system print spooler, or of configuring the application, or both.

By Any Other Name

Most print spoolers recognize a specific printer according to a *printer name* you have defined. This *printer name* is associated with a collection of resources, like some of these:

- A place to send the data - sometimes a physical device on the local host, sometimes a device on a remote host.
- A lock file to prevent two sets of output from being sent to the printer at once.
- One or more *filters* - programs and scripts designed to put the data in a form suitable to be sent to the printer. Sometimes there is no filter, at other times they can be quite elaborate.

When an application wants to send data to the spooler, it *pipes* it to the spooler process, supplying the printer name. The spooler uses the printer name to decide what to do with the data.

Regardless of the spooler, the proper configuration generally takes one of the following three forms:

1. Sometimes a system's print spooler already knows how to send data to a remote host using one of the mechanisms the IntelliServer supports. Then, it is merely a matter of configuring the spooler with IntelliServer's IP address or host name and supplying the proper TCP port number or device name.
2. Other spoolers do not understand remote devices or do not use any of the IntelliServer's mechanisms. The technique in this case is to adapt one of the *filter*s so that instead of translating the data and sending to a specified local device, it takes complete responsibility for sending the data to the IntelliServer. This technique is also used for applications which are designed to pipe output to arbitrary scripts and filters.
3. If your spooler understands neither remote devices nor filters, you can use the **`iservd`** daemon (see page 455) to create *pseudo-devices* on your host for each of the IntelliServer's printer ports. This is also useful for configuring applications that bypass the system spooler and only send output to specific device names.

Check The Diskette

Each IntelliServer includes a supplemental software diskette. This includes the latest version of IntelliServer software, source to Computone utilities like **iservcat** and **iservd**, sample spooler scripts, and documentation. Even if your specific operating system isn't mentioned, you should look at the examples for other operating systems and see what is available. This is among what you will find:

Table 18-4: Spooler Scripts and Other Documentation

File Name	Description
printers.doc	General instructions for configuring System V and BSD UNIX print spoolers using either the standard **rcp** command or Computone's **iservcat** utility. Explains how to use the spooler scripts listed below.
s5_rcp.sh	For System V UNIX - commented shell script using the **rcp** command to send data to the IntelliServer.
s5_tcp.sh	For System V UNIX - commented shell script using **iservcat** to send data to the IntelliServer.
bsd_rcp.sh	For BSD UNIX - commented shell script using the **rcp** command to send data to the IntelliServer.
bsd_tcp.sh	For BSD UNIX - commented shell script using **iservcat** to send data to the IntelliServer.

Computone Clients: Iservcat

The iservcat utility is designed to send data to one of the IntelliServer's serial ports by opening a connection through its designated TCP port.

- TCP ports 9000-9063 represent serial ports 0-63.
- TCP ports 10000-10015 represent groups 0-15. During port configuration, each serial port can be made a member of one of these groups, or of no group. Any request directed to a group is honored by the first available port in that group.

IntelliServer ports accept connections on these TCP ports if they are not already in use, and are configured in any of the following ways:

- Reverse-TCP
- Login-by-Port/TCP
- Printer
- *IntelliPrint* (any login type, with an IntelliPrint profile associated with this port. Printed output is directed to an attached terminal's Auxiliary port).

Since iservcat is intended to be run on many versions of UNIX, the source code is supplied and a *Makefile* containing the commands for compiling it under a number of different operating systems. Also supplied are ready-to-use binaries for a number of UNIX versions, for example:

- SCO UNIX
- UNIX SVR4 (Unixware, etc.)
- SunOS
- Linux

If a binary is not supplied for your operating system, see whether the *Makefile* has the instructions for generating it. If it is not covered there, you need to start with the versions whose UNIX is the closet to yours and start porting from there. As information become available, Computone puts the information on our FTP site, so don't forget to check these spots. Once you have a binary, you need to copy it to a suitable directory in the path.

Arguments & Options

The syntax of the iservcat command is shown in Example 18-5, Table 18-6, and Table 18-5.

Example 18-5: Iservcat Command

iservcat *options iserver port files*

```
iservcat -c -w60 jeeves 9005 /etc/termcap
ls -l | iservcat jeeves 10013 /tmp/hdr - /tmp/trail
iservcat -x -d20 jeeves 9002 /etc/termcap 2>/tmp/err
iservcat -ax jeeves 9002
ls -l | iservcat -d10 -r20 jeeves 10013
```

You can copy data from one or more files, pipe data to iservcat from another process, or do both from the same command.

Table 18-5: Iservcat Arguments

Option	Description
iserver	The IntelliServer's IP address or host name. (If a host name is supplied, **iservcat** resolves it to an IP address).
port	The TCP port number. TCP ports 9000 through 9063 represent serial ports 0 through 63, and ports 10000-10015 represent port groups 0-15. These port ranges can be re-assigned, however, by changing the values for **tcp_direct_base** and **tcp_group_base** in the IntelliServer's services table (see page 230).
files *no files = standard output only* *hyphen represents standard input, in order.*	A list of one or more files to send to the IntelliServer. Files are sent in the order listed. If there are no files listed, iservcat sends whatever is sent to its standard input, for example: **ls -l | iservcat jeeves 9003** If a minus sign (hyphen) appears as one of the files, it represents the standard input. Consider this example: **ls -l | iservcat jeeves 9003 /tmp/hdr - /tmp/trail** In this example, the data from /tmp/hdr is sent first, then the data piped from the **ls -l** command, and finally the data from /tmp/trail.

Table 18-6: Iservcat Options

Option	Default (if option were not present)	Description
-d [*seconds*]	5 seconds between retries.	When **iservcat** starts up, it tries to establish a TCP connection to the specified port on the IntelliServer. If the port is unavailable, the attempt fails and it tries again and again.
-r [*limit*]	Maximum of 128 retries.	
• Each retry does not occur immediately: there is a pause first — 5 seconds or as specified by **-d**. • If a connection is not made within the maximum retries, **iservcat** gives up and exits. • **-r0** means there is no retry limit: try forever. Limits greater than 30,000 are treated as 30,000.		

Table 18-6: Iservcat Options (Continued)

Option	Default (if option were not present)	Description
-w [*seconds*]	3 seconds, if -o arguments is used, otherwise 0 seconds.	Delay for the specified time after all the data is sent, before closing the TCP connection and exiting.
This should only be needed when the -o option is used to bypass telnet protocol. When telnet protocol is used, a *timing mark* is sent to the IntelliServer after all the data. When the IntelliServer receives this mark, it waits until all data has been sent to the serial port, then sends back a reply. This is to ensure the IntelliServer has received all the data before the connection is closed. When telnet protocol is not used, there is no way to guarantee the IntelliServer has processed all the data, but a large delay before closing the connection makes it more likely to succeed.		
-o	Normally, iservcat uses telnet protocol for its connection to the IntelliServer. This corresponds to a port's *normal* TCP option. The **-o** option allows you to bypass telnet protocol and send the data to a raw TCP connection. This corresponds to the *Raw* TCP option. Normally, telnet protocol is desirable, but the raw options have been provided for interoperability with similar products from other manufacturers.	
-a	Transfer data using telnet ASCII mode. By default, data is transferred using telnet binary mode. Has no effect when the **-o** option is used.	
-c	Cook output - Expand TAB's and insert carriage-returns before any line-feeds. By default, there is no output processing except what is done by the telnet protocol.	
-x	Enable debugging output and send it to the standard error. Normally, you redirect the standard error to a file, as with **iservcat -x jeeves 9004 testfile 2>/tmp/errfile**	
Options — Rules: • All the options begin with minus signs (hyphens) and they must precede any other command-line arguments. • If an option is followed by a parameter, the intermediate space is optional. For example, **-d 30** could also be entered as **-d30**. • Two or more options that don't take parameters can be combined, for example **-a -x** could be entered as **-ax**.		

Computone Clients: Iservd

The iservd daemon creates pseudo-tty devices on your UNIX host and links these devices to serial ports on the IntelliServer. These are similar to the pseudo-tty devices that are created when an IntelliServer user logs *into* your UNIX host, but with one important difference:

- When an IntelliServer user uses telnet or rlogin to log into a UNIX host, the telnetd or rlogind daemons create a pseudo-tty device to support that connection. *The name of this device is randomly assigned and varies from one connection to another.* A user logged into the IntelliServer's serial port 3 might log into a UNIX host and be assigned to /dev/ttyp10 — or /dev/ttyp04 — but something entirely different when it logs in the next time. That makes this type of connection entirely unsuited to printing, where a particular device name is expected to be associated with a particular physical device. You don't want to print invoices on the check forms. For the same reason, this type of connection is not suited to any application designed to open specific serial devices and use them in particular ways. For more discussion of this, see "Pseudo-TTY's" on page 402.
- Iservd also creates pseudo-tty devices, but here *the same pseudo-tty device always represents the same physical port of the same IntelliServer.* Applications running on your host can open and use a specific device name, and always be communicating with the same physical device. This is useful for printing as well as all sorts of data-acquisition applications involving RS232 time-clocks, scales, spectrometers, stock-tickers, and other devices.

Iservd connects to the serial ports in the same way as **`iservcat`**:

- TCP ports 9000-9063 represent serial ports 0-63.
- TCP ports 10000-10015 represent groups 0-15.

The **`iservd`** daemon differs from **`iservcat`** in an important way: **`iservcat`** can only send data to the port; **`iservd`** can transfer data in both directions.

IntelliServer ports accept connections on these TCP ports if they are not already in use, and are configured in any of the following ways:

- Reverse-TCP
- Login-by-Port/TCP
- Printer
- Ports with *IntelliPrint* (but for output only)

Since **`iservd`** is intended to be run on many versions of UNIX, the source code is supplied, and a *Makefile* containing the commands for compiling it under a number of different operating systems. Also supplied are ready-to-use binaries for a number of these UNIX versions, for example:

- SCO UNIX
- UNIX SVR4 (Unixware, etc.)
- SunOS
- Linux

If there is not a binary for your operating system, see whether the *Makefile* has the instructions for generating it. If it is not covered there, you need to start with the versions whose UNIX is the closet to yours, and start porting from there. Check Commputone's FTP site for the latest binaries available.

Starting Iservd

Iservd is called a *daemon* process because it works "beneath the surface" of your UNIX system, more or less invisible to ordinary users. Daemon processes are created before time (*when the UNIX system comes up*), are immortal (*do not end when particular users log off)*, and occupy a station between the *user applications* and the *kernel.*

Although you can start **`iservd`** manually, it is normally started automatically at boot time. How? On most UNIX systems there is a set of sub-directories, **`/etc/rc0.d`**, **`/etc/rc1.d`**, **`/etc/rc2.d`**… which contain startup scripts which are run automatically when your system comes up. You will want to add a new script to start your **`iservd`** daemon. Look at the scripts that are already there. You will find one that starts up TCP/IP. You will want your script to run just after that one. On most systems, the scripts in a particular startup directory are run in alphabetical order, so you will need to name your script appropriately. If there appears to be a naming convention for these files, follow it. For example, if the existing files were named **`S00grumpy, S05sneezy, S25tcp,`** and

S30printers, you might consider naming your new file something like **S27iservd**.

Some systems use a single startup file instead of a collection of files. In that case, you have to edit this file, inserting the **iservd** command at the appropriate place. If you are not certain how to proceed, you should consult your UNIX system's documentation for information that is peculiar to your installation.

What Does Iservd Do

When **iservd** starts, it begins by reading a configuration file and a lock file. If **iservd** were accidently started twice, the lock file prevents multiple daemons from trying to set up the same pseudo-tty ports. The configuration file tells **iservd** which pseudo-tty devices must correspond to which physical ports and gives other options for creating the connections.

Each line of the configuration file corresponds to one pseudo-tty device and one TCP connection to the IntelliServer. The original **iservd** forks a separate process to handle each separate device: the command-line arguments of the process correspond roughly to the data in the configuration file. This is done quite deliberately. When you use the UNIX **ps** command to view all the processes and their command-line arguments, it is easier to see which processes correspond to which ports.

Iservd Command-Line

The iservd command-line takes two optional arguments, as shown in Example 18-6.

Example 18-6: Starting Iservd

```
iservd [config-file [lock-file]]
```

```
iservd /etc/iservd.conf /etc/iservd.pids
iservd
iservd /usr/lib/is2/conf1
iservd /usr/lib/is2/conf2 /usr/lib/is2/lk2
```

The first argument is the name of a configuration file which associates particular device names with specific ports on an IntelliServer. If there are two arguments, the second argument is the name of a lock file. If the lock file does not already exist, it is created automatically. The lock file prevents multiple iservd daemons from being accidently started for the same TCP connections. If no arguments are listed, **`/etc/iservd.conf`** is used as the configuration file, and **`/etc/iservd.pids`** is used as the lock file. If only a single argument is listed, it is assumed to be the name of a configuration file and the default lock file is used.

Stopping Iservd

Suppose there are several **`iservd`** daemons running, and you want to make some changes to the configuration file. How do you make the changes take effect? You need to stop the iservd processes that are currently running and restart them all. There is an easy way to do this, because the lock file keeps track of the **`iservd`** processes that are currently running. If you run **`iservd`** with **`-K`** instead of a configuration file name, it kills all the **`iservd`** daemons associated with a given lock file. If no lock file is named, all daemons associated with the *default* lock file are killed. This is shown in Example 18-7.

Example 18-7: Stopping Iservd Daemons

`iservd -K` [*`lock-file`*]
`iservd -K` **`iservd -K /etc/iservd.pids`** **`iservd -K /usr/lib/is2/lk2`**

Example 18-8: Stopping Iservd Daemons

`iservd -K` [*`lock-file`*]
`iservd -K` **`iservd -K /etc/iservd.pids`** **`iservd -K /usr/lib/is2/lk2`**

Restarting Some Iservd Daemons

If you do not wish to kill all the iservd daemons, but need to restart some of them, you can use your system's **ps** command to determine the process numbers of the daemons associated with each connection. (There may be two: one to send data and one to receive it.) Once the process numbers are known, the UNIX **kill** command can be used to remove those daemons. To restart them, simply run iservd again, specifying the same configuration and lock files as originally. The lock file prevents duplicate daemons from starting on any connections you have left running.

Normally, you configure your system to run iservd once from startup using a single configuration file and a single lock file. There are some administrators who like to invoke iservd several times at startup, each using a separate configuration and lock file name. Why? Since **iservd -K** kills only the daemons associated with a particular lock file, the administrator can then kill and restart daemons more selectively, without looking up their processes numbers manually.

In Example 18-9, suppose there are three different configuration files and each one configures a single connection. At startup each daemon is started separately, using a different lock file for each.

Example 18-9: Iservd for the Meek

```
at system startup...
iservd /usr/lib/is2/cf1 /usr/lib/is2/lk1
iservd /usr/lib/is2/cf2 /usr/lib/is2/lk2
iservd /usr/lib/is2/cf3 /usr/lib/is2/lk3

later on...
(make changes to /usr/lib/is2/cf3)
iservd -K /usr/lib/is2/lk3
iservd /usr/lib/is2/cf3 /usr/lib/is2 lk3
```

Later on, a configuration change is made and wants to restart one of these daemons, but doesn't want to affect the others. Therefore, **iservd -K** is used to kill all the daemons associated with the lock file (which in this case are the daemons for just the one connection), and then both of them are started up by re-running **iservd** with the correct lock and configuration file. If all the iservd's had been run from a single script, you could have re-run the entire script because the lock files prevent multiple daemons from starting on the same connection.

Limitations

The pseudo-tty ports created by iservd are subject to two limitations imposed by UNIX itself:

1. The UNIX application cannot change the port's physical line settings directly (baud rate, character size, flow control, data-set signals). These can only be changed using the IntelliServers administration commands. Applications designed to manipulate a serial port in these ways must be run on true local devices. This is not a common problem except in special, highly-customized applications.
2. Applications which rely on the UNIX *ttyname*() system call or the *tty* shell command to return the name of the controlling tty device are not guaranteed to work. The **`iservd`** daemon works by linking the device name you specify to second device name supplied by the operating system. Nothing can control which of these two names are returned. This is sometimes a problem with older applications written before networking became prevalent. See also "Pseudo-TTY's" on page 402 for some additional discussion.

Iservd Configuration File

Each entry in the configuration file contains the device name, the IntelliServer name (or its IP address) and the TCP port, which specifies which serial port or group of ports is desired. After these parameters you can list different types of options, and comments can appear following a pound (#) sign. Example 18-10 summarizes this format and gives a fragment of a sample file.

Example 18-10: Iservd Configuration File

device-name server-name tcp-port options

```
# Sample iservd configuration file
#
# Anything after a pound sign is a comment. This may include entire
# lines like these, or comments added to the end of regular lines
# as shown below:

# In each of these examples, the IntelliServer's IP address is
# 160.77.99.103
#
is1 160.77.99.103 9004 # The device name is /dev/is1. 9004 is port 4
is2 160.77.99.103 9005 # And device name /dev/is2 will reach port 5
#
# You can use IntelliServer's IP address, or you could use its host name,
# provided your host is configured to resolve it to an IP address.
#
is3 jarvis 9007 -p -i # A permanent connection
is4 jarvis 9008 -p -h # If the TCP connection is dropped, the process
                      # running on /dev/is4 gets a hangup signal.
#
# Be nice: put lots of comments in your config file
# Options cheat sheet:
# -p              permanent connection
# -d time         retry delay
# -r limit        retry limit
# -i              ignore pseudo-tty close
# -t time         inactivity timeout
# -w time         delay on close
# -o              bypass telnet protocol
# -a              use telnet ascii mode
# -c -u           Pseudo-TTY device initialization
# -h              Hangup Pseudo-TTY on network disconnect
# -s              Enable Syslogging
# -l file         Port-specific log file
# -x              Enable debugging
# -vo -va -vh -vd Enable data trace
```

Table 18-7 explains the *device-name*, *server-name*, and *tcp-port* on more detail:

Table 18-7: Iservd Configuration File Parameters

Parameter	Description
device-name	The name of the pseud-tty device to be created. All are created in the **/dev/** directory. **Do not use the name of any existing file or device on your system and do not use the same name more than once.** For example, do not use **lp** or **ttyp2** if there is a **/dev/lp** or a **/dev/ttyp2** on your system already. When your application wants to access the serial port, this is the device it opens. If your device name were **is3**, the application opens **/dev/is3**.
server-name	The IntelliServer's host name or IP address. (If a host name is used, your host system has to be able to resolve it into an IP address, via a host table or through domain name servers).
tcp-port	The TCP port number. TCP ports 9000 through 9063 represent serial ports 0 through 63, and ports 10000-10015 represent port groups 0-15. These port ranges can be re-assigned, however, by changing the values for **tcp_direct_base** and **tcp_group_base** in the IntelliServer's services table (see page 230). Remember, the corresponding ports on the IntelliServer must be configured as *Reverse-TCP*, *Login-by-Port/TCP,* or *Printer*, or as a login port with an *IntelliPrint* profile assigned. See chapter 5, *Configuring Serial Ports*.

There are a large number of configuration file options. Table 18-8 lists them by logical groupings (rather than alphabetically) so they are easier to understand:

Table 18-8: Iservd Configuration File Options

Option	Description	Default (if option were not present)
-p	Permanent connection	Temporary connection
	There are two types of connections: temporary and permanent. The **-p** option selects a permanent connection; otherwise, a temporary one is used.	
For permanent connections, **iservd** attempts to connect to the IntelliServer as soon as it starts up. When the connection has been established, any data written to the pseudo-tty device is sent to the IntelliServer's serial port, and any data received by the serial port can be read from the device. Once the connection is established, it is not closed unless the corresponding **iservd** process is killed. If the IntelliServer drops the connection for any reason, **iservd** attempts to re-establish it.		
For temporary connections, **iservd** does not attempt to open the connection until someone writes data to the pseudo-tty device. Then, it makes a certain number of attempts to connect to the IntelliServer. If it fails, it temporarily "shuts down" the pseudo-tty device, and the process that was trying to write to the device sees this as an error condition. If a connection *is* established, iservd starts performing bidirectional data transfer as with permanent connections. A temporary connection remains established until terminated according to the **-i** and **-t** options described below.		
-d [*seconds*]	Retry delay and retry limit: These options determine how long **iservd** will attempt to establish a connection.	5 seconds between retries.
-r [*limit*]		Temporary: 128 Permanent: forever.
• Each retry does not occur immediately: there is a pause first — 5 seconds or as specified by **-d**. • For a temporary connection, if a connection is not made within the maximum retries, iservd temporarily shuts down the pseudo-tty port to send an error condition to the process trying to use the device. • If the retry limit is exceeded on a permanent connection, an additional minute of delay is added and it continues to retry. • **-r0** means there is no retry limit: try forever. Limits greater than 30,000 are treated as 30,000.		
-i	Ignore pseudo-tty closes:	Drop temporary connection when pseudo-tty is closed

Table 18-8: Iservd Configuration File Options (Continued)

Option	Description	Default (if option were not present)
This option has no effect on permanent connections (**-p** option used). Without this option, iservd drops the TCP connection to the IntelliServer when the last application has closed the pseudo-tty device. When the TCP connection is closed, the IntelliServer shuts down the serial port and drops DTR to hang up the line. When the **-i** option is used, iservd holds the pseudo-tty device open and the TCP connection stays up.		
-t *seconds*	Inactivity timeout for temporary connections. If no data is sent for this number of seconds, **iservd** drops the TCP connection to the IntelliServer.	No time-out.
-w [*seconds*]	Temporary Connection. Delay for the specified time after all the data is sent, before closing the TCP connection.	No delay.
This should only be needed when the -o option is used to bypass telnet protocol. When telnet protocol is used, a *timing mark* is sent to the IntelliServer after all the data. When the IntelliServer receives this mark, it waits until all data has been sent to the serial port, then sends back a reply. This is to ensure the IntelliServer has received all the data before the connection is closed. When telnet protocol is not used, there is no way to guarantee the IntelliServer has processed all the data, but a large delay before closing the connection makes it more likely to succeed.		
-o	Bypass telnet protocol; use raw TCP connection instead.	*Telnet protocol used.*
Normally, iservd uses telnet protocol for its connection to the IntelliServer. This corresponds to a port's *normal* TCP option (see page 87). The **-o** option allows you to bypass telnet protocol and send the data to a raw TCP connection. This corresponds to the *Raw* TCP option. Normally, telnet protocol is desirable, but the raw options have been provided for interoperability with similar products from other manufacturers.		
-a	Use telnet ASCII mode. (Has no effect if the **-o** option is also specified).	Use telnet Binary mode.
-c	Pseudo-tty device initialization:	*Default configuration is system-dependent.*
-u		

Table 18-8: Iservd Configuration File Options (Continued)

Option	Description	Default (if option were not present)
• **-u** initializes the pseudo-tty port for raw input and output: no output post-processing, no input pre-processing, and canonical input and echoes are disabled. • **-c** initializes the pseudo-tty for raw input and cooked output: as above, but tabs are expanded and Carriage-Returns are inserted before linefeeds. The application that is using the pseudo-tty might very well set the input and output processing options appropriately, in which case you won't need these options. They are provided to support applications that think the port is already properly configured. For example, you may want to print a file by simply re-directing the output of the **cat** command to the device.		
-h	Send hangup on disconnect; affects temporary or permanent connections.	*Do not notify if TCP connection is dropped.*
If the IntelliServer drops the TCP connection to **iservd**, it normally just tries to re-establish it. When the **-h** option is used, **iservd** sends a *hang-up* signal to applications running on that device (the same as if they had been connected to local serial ports and had lost carrier). This allows the application to be aware that the connection had been dropped, and to respond accordingly.		
-s	Enable syslog output. If your UNIX host supports syslogging, an activity log is sent to the syslog service.	*No syslogging.*
• Facility: LOG_DAEMON. • Priority: LOG_ERROR and LOG_NOTICE for errors and general activity log. • Priority: LOG_DEBUG for debugging output enabled by the **-x** option.		
-l *logfile*	Send activity log to the specified log file.	*No log file, or* **/tmp/***name***.debug**
The activity log indicates when pseudo-tty devices are opened and closed and when TCP connections are established and dropped. If both the **-l** and **-s** options are used, the activity log is sent to both the log file and the syslog service. If the -x or -v option is used and a log file has not been specified, one is created and its name is of the form: **/tmp/***name***.debug** , where *name* is the pseudo-tty name.		
-x	Enables debugging output to the log file and syslog. This includes information from various stages of connection establishment and indications of data transfer.	*No debugging output.*

Table 18-8: Iservd Configuration File Options (Continued)

Option	Description	Default (if option were not present)
If the **-s** option was used, then debugging output is sent to the syslog service at LOG_DEBUG priority. If the **-s** option was not used, no syslog data is sent. Debug output is always sent to a log file, even if the **-l** option was not used; a log file of the form **/tmp/***name***.debug** will be created. For example, consider this line from a configuration file: **is2 jeeves 9005 -p -x** Debugging output is desired and no log file was specified, so the debugging output is written to **/tmp/is2.debug**.		
-vo **-va** **-vh** **-vd**	Enables enhanced debugging: Debugging output (discussed under the **-x** option) now includes a dump of all data sent and received over the TCP connection.	*No enhanced debugging.*
• -vo Dumps the data in octal. • -vh Dumps the data in hexadecimal. • -vd Dumps the data in decimal. • -va Dumps the data in modified ASCII: Unambiguous printable characters appear as themselves, separated by spaces. Other characters (including control characters, whitespace, and unprintables) are shown as 2-digit hexadecimal numbers.		

Permanent and Temporary Connections

When should you use a temporary connection and when a temporary one? Here are some guidelines:

- Use permanent connections to support a local terminal or printer dedicated to a single host. There is no need to ever drop the connection to allow access by other hosts, nor must you react to data-set conditions like loss of carrier. The best policy is to use permanent connections unless there is a reason not to.

- Use temporary connections to support dial-out access to a modem. When the application closes the pseudo-tty port, the TCP connection is dropped (assuming the -i option is not used), the serial port is closed, and the modem hangs up the line. With a permanent connection, there is no way to hang up the phone from the local side.
- Use temporary connections for connections to Login-by-Port/TCP ports. If you used a permanent connection you can *never* use the port for dial-in. It is always be in use.
- Use temporary connections to share printers (at the TCP connection level) between two hosts. Each host's configuration file contains a line that accesses the *same* port on the *same* IntelliServer. One host can use it unless the other one is. This is an example where the **`-i`** and **`-t`** options might be appropriate. You do not want to relinquish the TCP connection until there has been no output for a certain period, even if applications *do* keep closing and re-opening the device.

Index

S

T