Installing Conserver

version 1.0

David K. Z. Harris
zonker@certaintysolutions.com
Bryan Stansell
bryan@certaintysolutions.com

http://www.certaintysolutions.com/consoles/LISA2K-2.zip
http://www.conserver.com/consoles/LISA2K-2.zip

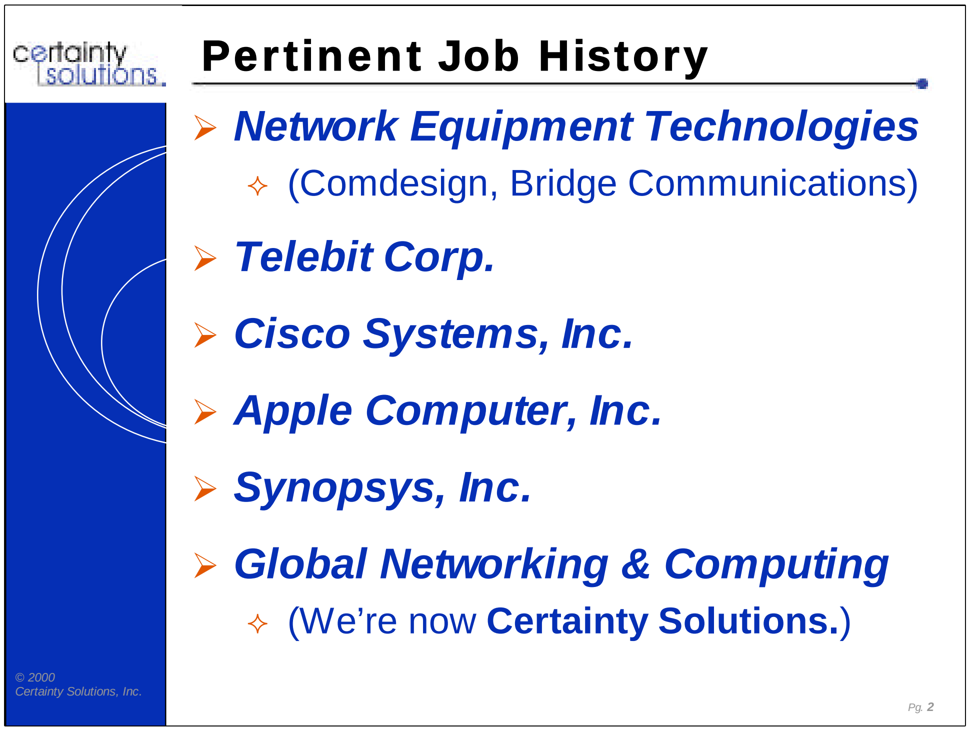
This presentation is a supplement to my console services web pages located at **http://www.certaintysolutions.com/consoles/LISA2K-2.zip**.

These pages have a substantial amount of information noted below each slide. We do this to help minimize the amount of note-taking that you need to do in class, and this should give you more time to listen to the instructors.

If you feel that you learn better by taking notes, please feel free to do so.

This presentation is meant to be a follow-up to a Basic Serial presentation. While this presentation can stand on its own, there is only a small amount of review of the earlier topic.

During this tutorial, we will be discussing the topic of Console Servers as a generic application, but our technical emphasis will be on the Conserver application, which is freely available from http://www.conserver.com/. For most purposes in this tutorial, “Console Server” and “Conserver” can be used interchangeably.



Pertinent Job History

- **Network Equipment Technologies**
  - (Comdesign, Bridge Communications)
- **Telebit Corp.**
- **Cisco Systems, Inc.**
- **Apple Computer, Inc.**
- **Synopsys, Inc.**
- **Global Networking & Computing**
  - (We're now **Certainty Solutions.**)

© 2000
Certainty Solutions, Inc.

Pg. 2

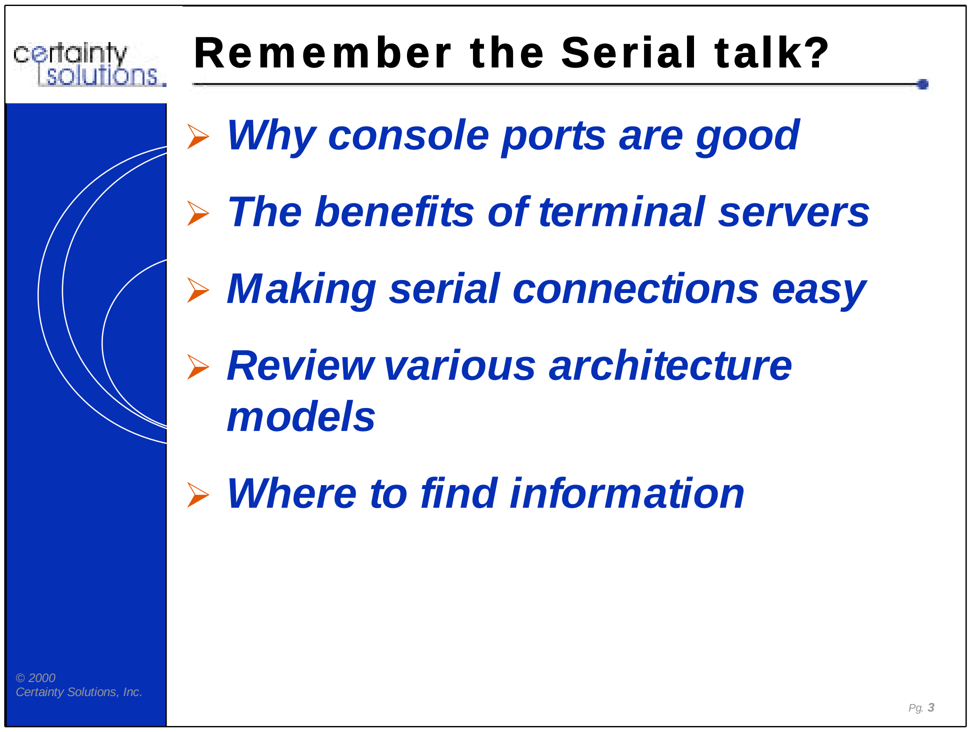
Before moving into networking, David Harris was a hardware hacker, working in repair and R&D roles. He has been tinkering with serial devices for more than a decade.

"My experience, plus reading plenty of manuals, has taught me that there are a few safe bets that you can make when working with unknown serial devices. I've also learned a few tricks that make the job of connecting serial devices to terminal servers easier. I'll share these tricks throughout this presentation."

Bryan Stansell is a Systems Administrator, and has been programming for many years, as an extension to his systems administration tasks, because he enjoys programming.

Bryan has been the keeper of the Conserver application for more than five years, having taken on the program while working for Arnold De Leon at Synopsys, Inc., and he'll help answer questions related to the Conserver application and client software.

Bryan was one of the original employees of Global Networking & Computing (GNAC, Inc.), more than 4 years ago. David Harris has been with the company for nearly three years. In August 2000, GNAC became Certainty Solutions.



Remember the Serial talk?

- Why console ports are good
- The benefits of terminal servers
- Making serial connections easy
- Review various architecture models
- Where to find information

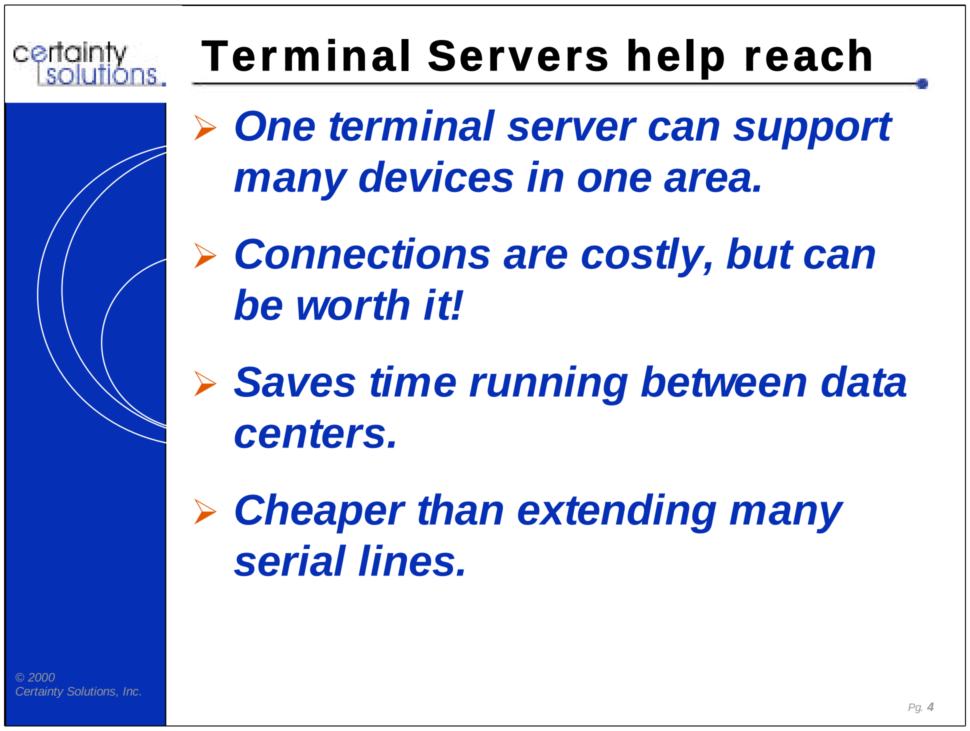
© 2000 Certainty Solutions, Inc.

Pg. 3

Terminal servers can allow us to provide remote access to console ports around the network, and this allows us to respond faster to outage/problem alerts, thus reducing the duration of outages/problems. While terminal servers can be costly to deploy, their advantages can offset those costs, especially when downtime has financial impact!

We took away a lot of the mystery surrounding serial connections, and discussed tools and accessories that can help make connecting various devices a lot easier. We also illustrated how to make and test those new connections, and explained why those methods work so well, and why they will save you time.

We looked at various architectures for deploying terminal servers in different areas of both large and small networks. (We'll review those architectures in this session, this time with console servers in mind.)



Terminal Servers help reach

- *One terminal server can support many devices in one area.*
- *Connections are costly, but can be worth it!*
- *Saves time running between data centers.*
- *Cheaper than extending many serial lines.*

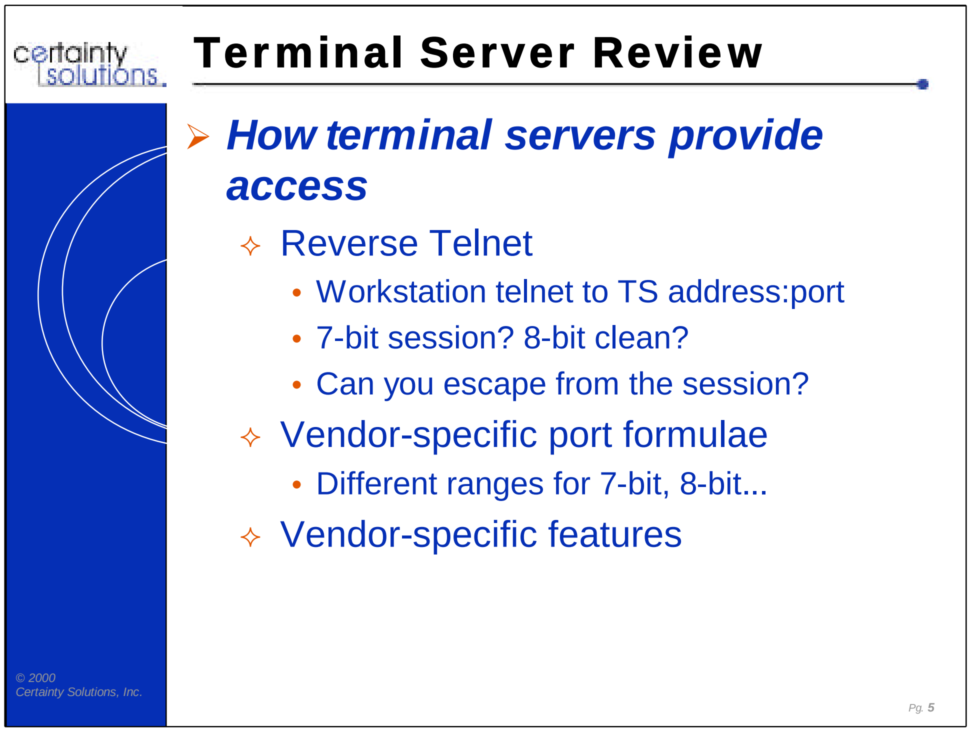
Pg. 4

If you have a bunch of devices at a remote location, you likely have a network already in place for the devices. Usually these are hosts in a remote data center, but they could include network gear in wiring closets, or even a remote site with telemetry and control equipment. In these cases, the terminal server becomes one more addressed device on the network, and it can connect to dozens of devices at one time.

The cost for a 10/1000 switch port is currently cheaper than a serial port some of the expensive terminal servers. However, the costs for the serial ports are beginning to drop. (And they are still cheaper than gigabit ethernet ports!)

The value of deploying terminal servers is from the reduced downtime because technical staff can fix problems more quickly when they can get to the consoles of ailing devices more quickly.

You can use serial extenders, if you have plenty of fiber between the main and remote console locations, but that fiber has value to you as well, which affects your cost equation.



Terminal Server Review

- **How terminal servers provide access**
  - Reverse Telnet
    - Workstation telnet to TS address:port
    - 7-bit session? 8-bit clean?
    - Can you escape from the session?
  - Vendor-specific port formulae
    - Different ranges for 7-bit, 8-bit...
  - Vendor-specific features

© 2000
Certainty Solutions, Inc.

Pg. 5

Originally, modems or 'dumb terminals' were connected to terminal servers, and users would telnet from the terminal server to other points around the network.

Today, most terminal servers allow you to open a socket-based connection to the IP address of the terminal server, but at a high TCP port number, to connect to a particular serial port.

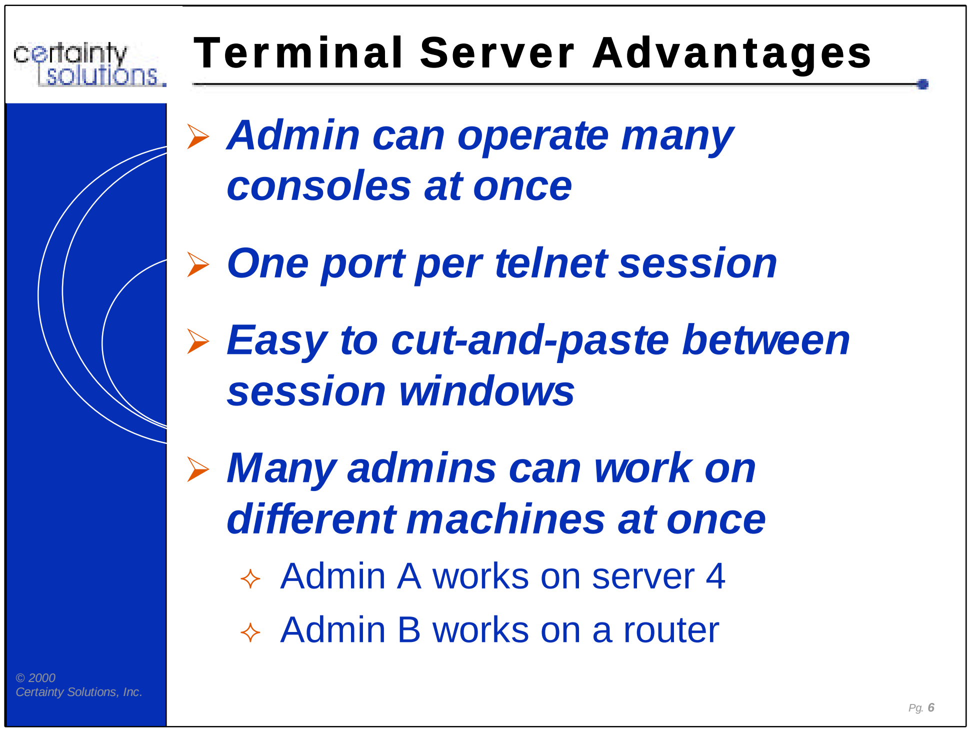
Some vendors allow only 7-bit sessions, while others provide the option for full 8-bit sessions, and even "non-escapable" sessions (where the attached device needs to drop the DCD or hardware handshake lead to disconnect you session).

The list below tells you the formulae to determine the TCP port number for two of the more popular terminal servers (where 'n' is the line (serial port) number you wish to connect).

Cisco: 2000 + n

Xyplex: 2000 + (100 * n)

IOLAN: 10000 + n



Terminal Server Advantages

- Admin can operate many consoles at once
- One port per telnet session
- Easy to cut-and-paste between session windows
- Many admins can work on different machines at once
  - Admin A works on server 4
  - Admin B works on a router

© 2000
Certainty Solutions, Inc.

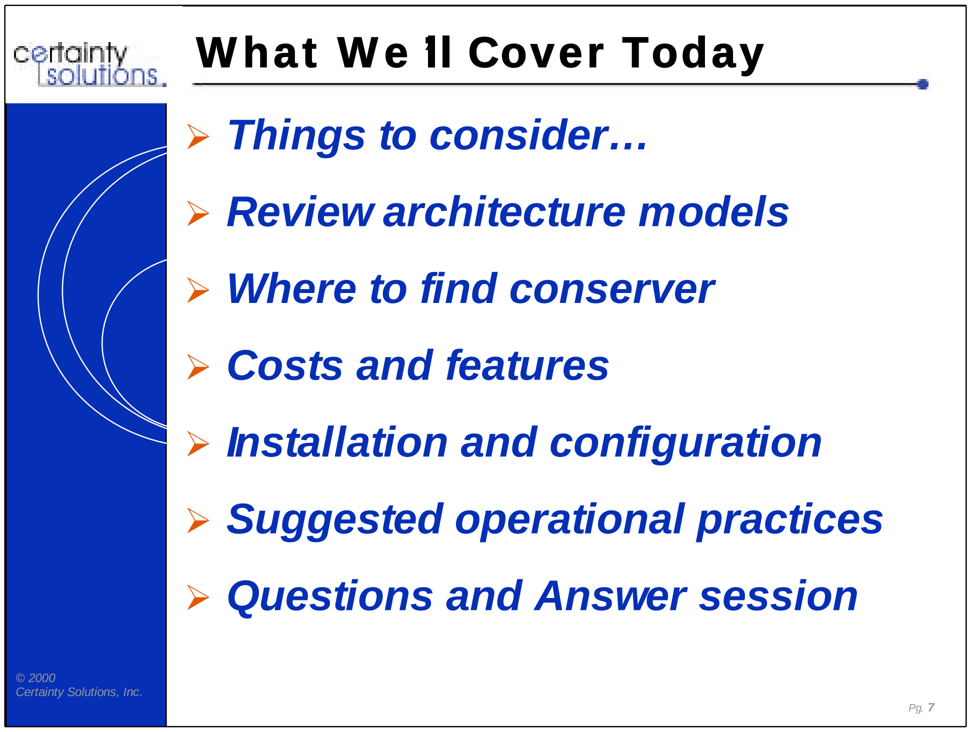
Pg. 6

It's common to open many windows to different ports, so you can copy and paste between them.

Many administrators can work with different hosts simultaneously, from different workstations, rather then waiting in line to use a shared terminal or laptop.

Only one person can be connected to each port at any given time. This is normally a good thing, as it can prevent two administrators from making configuration changes at the same time, providing that you make it a policy to make changes from the serial console. If another person tries to connect to a busy port, the connection is refused, with a message to the user.

There is a downside to this, however, in the case that one administrator connects to a port, and then goes to lunch, or home. With the idle session still connected, nobody else can connect to that same port. (Someone with administrator privileges on the terminal server needs to log onto the terminal server, and reset that serial port to break the connection.)



What We'll Cover Today

- Things to consider…
- Review architecture models
- Where to find conserver
- Costs and features
- Installation and configuration
- Suggested operational practices
- Questions and Answer session

© 2000
Certainty Solutions, Inc.

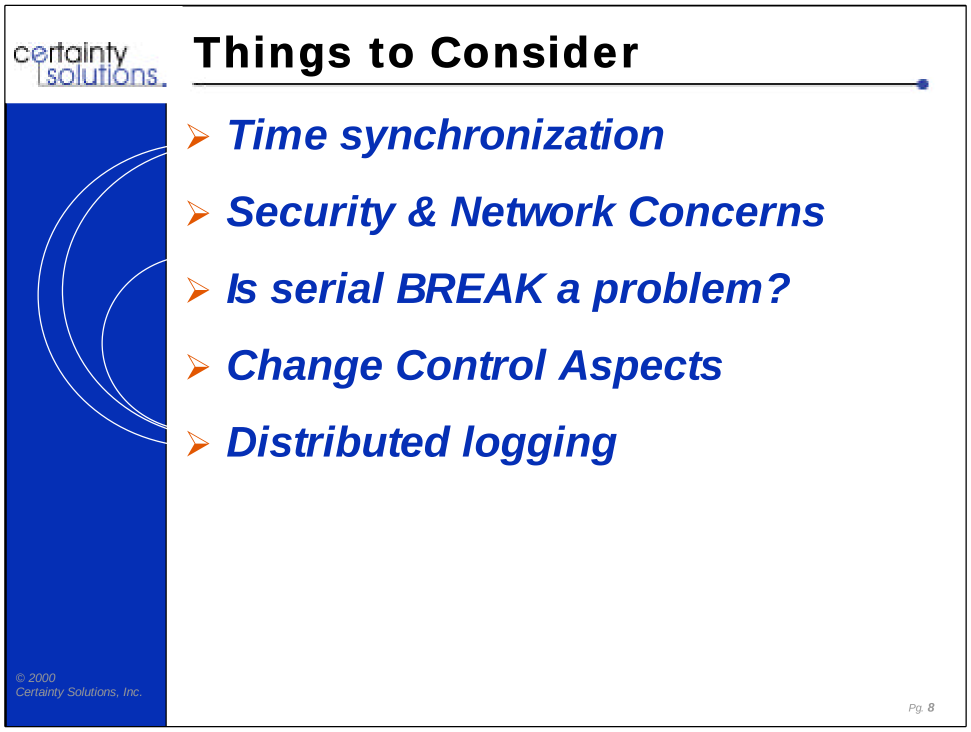
Pg. 7

This tutorial usually follows a Basic Serial tutorial, and so the next few slides would be a basic review of material covered before the break. As a result, we'll probably cover them pretty quickly.

If this tutorial is being offered as a stand-alone session, we'll spend more time covering security concerns, and explaining the architectures.

We also want to know what you expect from the class, and we'll explain how we think we can cover the topics.

During the tutorial, we'll also invoke a demo of conserver to help illustrate certain aspects of the application and user interface.



Things to Consider

- Time synchronization
- Security & Network Concerns
- Is serial BREAK a problem?
- Change Control Aspects
- Distributed logging

© 2000
Certainty Solutions, Inc.

Pg. 8

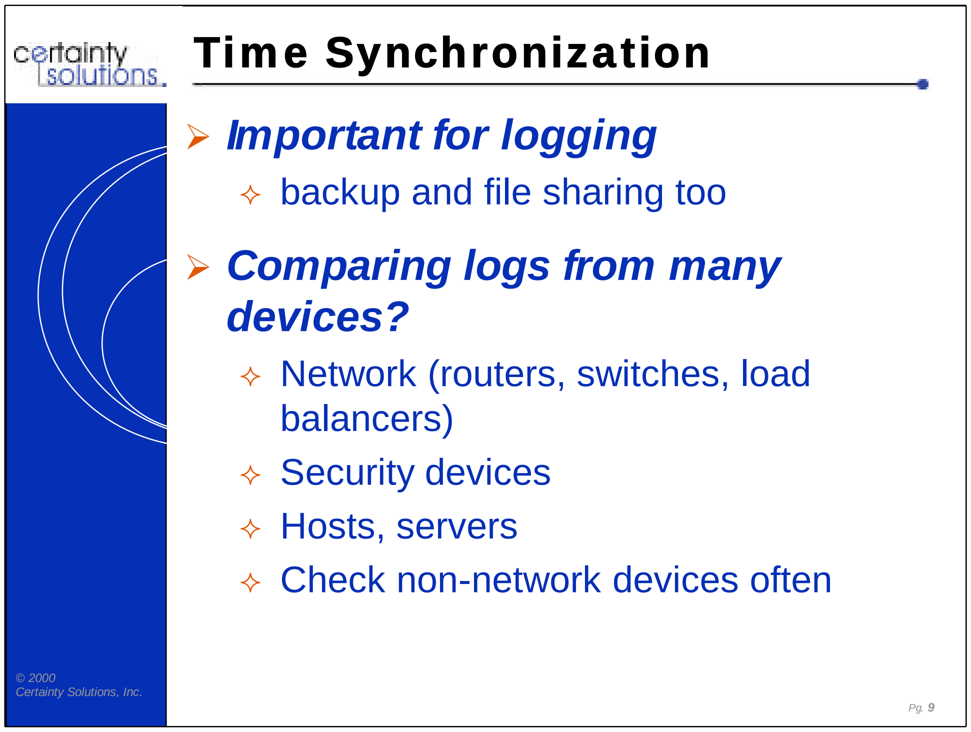
If you will be using your logs to troubleshoot, you should ensure that all of the machines have synchronized clocks.

Determine which network(s) will host the terminal servers.

Is your deployment in compliance with any existing security policies? (You may not be able to connect to certain networks, or you may not be allowed to extend those networks into any other non-secure areas.)

You may plan to put the terminal servers on the management network for security reasons. But find out if the networks are currently in the areas where you want to put the terminal servers. (You may need to use media converters to let you use spare fiber between data centers, or add memory to routers to allow you to use encrypted tunnels, etc.)

Do you have enough open network connections in those places, on those networks, reserved for your terminal servers (or do you need to order more hubs or switches)?



Time Synchronization

- ***Important for logging***
  - backup and file sharing too
- ***Comparing logs from many devices?***
  - Network (routers, switches, load balancers)
  - Security devices
  - Hosts, servers
  - Check non-network devices often

In a distributed file system, time synchronization is an important issue. It is also important if you are going to be comparing logs between distributed machines.

NTP is a practical solution for synchronizing the time between various machines and the logging server.

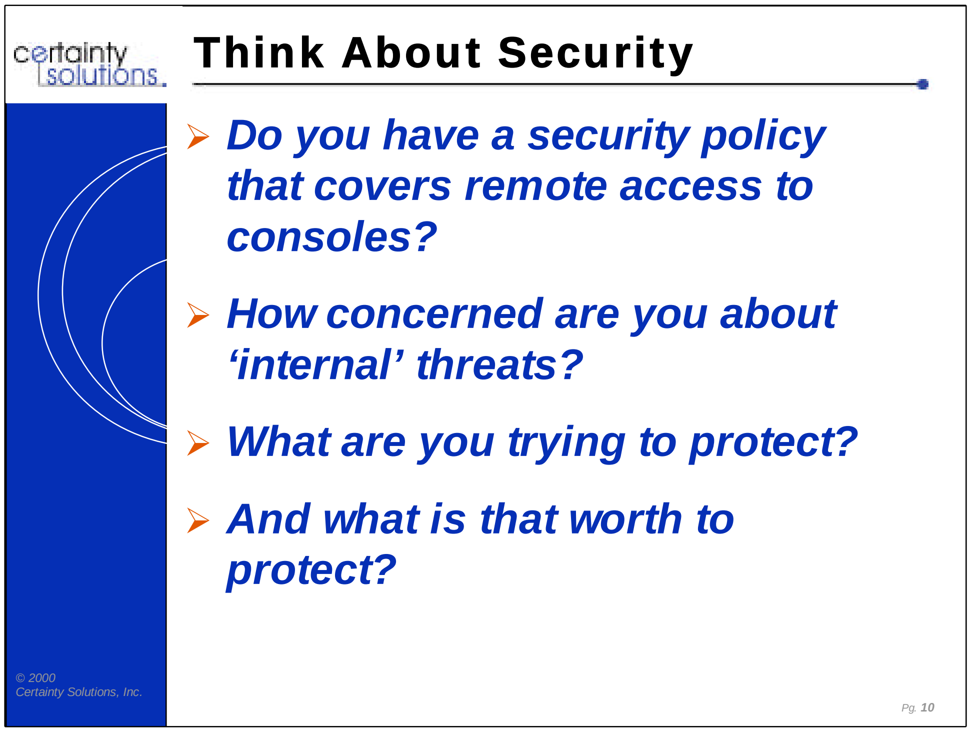
Network devices and servers should be configured to include date/time information in the messages sent to the serial console.

Consider the issues of comparing logs from machines across multiple time zones. If the machines keep their time in the ‘local’ time zone, the administrators will have to mentally compensate for the time differences when searching through the log files.

However, if you decide to sync all devices to a single time zone (GMT perhaps?), you then have the problem that admins still need to “do the math” and convert to their local time zone for resolving single-machine issues.

The bottom line here: Someone will have to convert timestamps some of the time. You should decide when it would be better; during a crisis, or day-to-day.

If you are logging devices which cannot include timestamps in the messages, you can ask conserver to include periodic timestamps from the server in specific log files, which can provide a basic reference timestamp.



Think About Security

- Do you have a security policy that covers remote access to consoles?
- How concerned are you about ‘internal’ threats?
- What are you trying to protect?
- And what is that worth to protect?

© 2000
Certainty Solutions, Inc.

Pg. 10

We will discuss some security issues in this talk, but security is a touchy subject, and most cases are unique in some aspects.

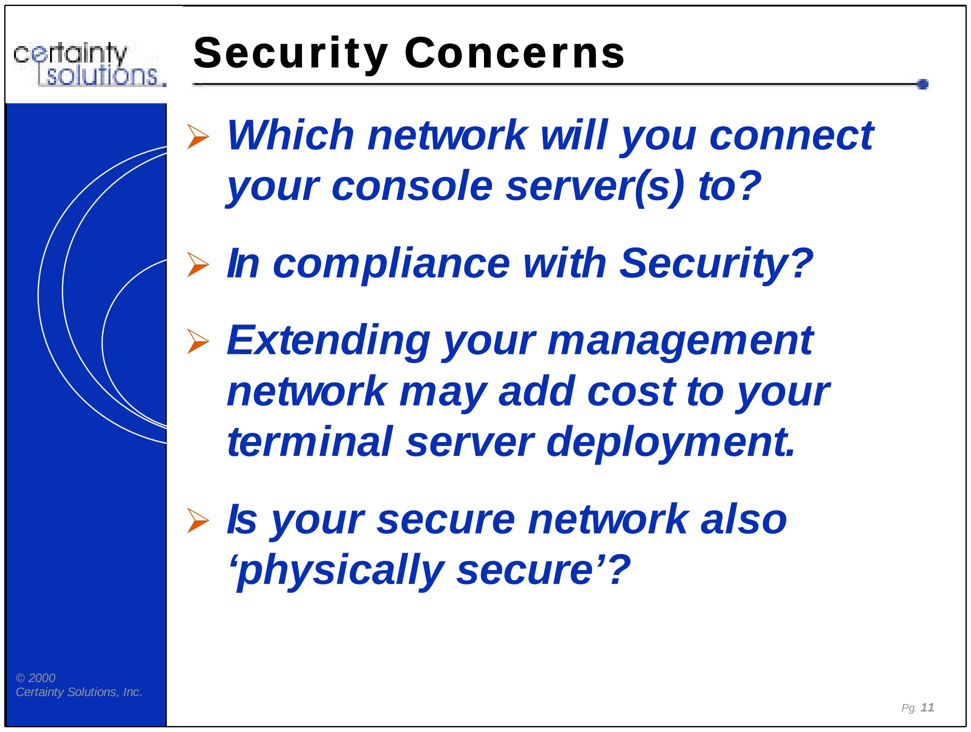
Due to this, we will discuss general points during the class, and the materials will give you some questions to think about and discuss.

During the conference, we will be holding a BOF, and we’d welcome any additional questions there, if you are comfortable asking them in that forum.

The biggest worry when implementing remote access to your consoles is whether you are concerned with the console traffic being monitored within your network. (Most companies use a “jelly bean” security model…hard on the outside, soft on the inside…meaning that they are not worried about folks on the inside sniffing on the wires.

In a switched Ethernet environment, it’s harder for folks to see the packets to and from the terminal servers but it’s not impossible.

If you have a console server host, you should consider if it is worth making that host single-purpose machine, and limiting the login accounts.



Security Concerns

- Which network will you connect your console server(s) to?
- In compliance with Security?
- Extending your management network may add cost to your terminal server deployment.
- Is your secure network also ‘physically secure’?

© 2000
Certainty Solutions, Inc.

Pg. 11

Do you have an existing security policy? Does it cover access to serial consoles?

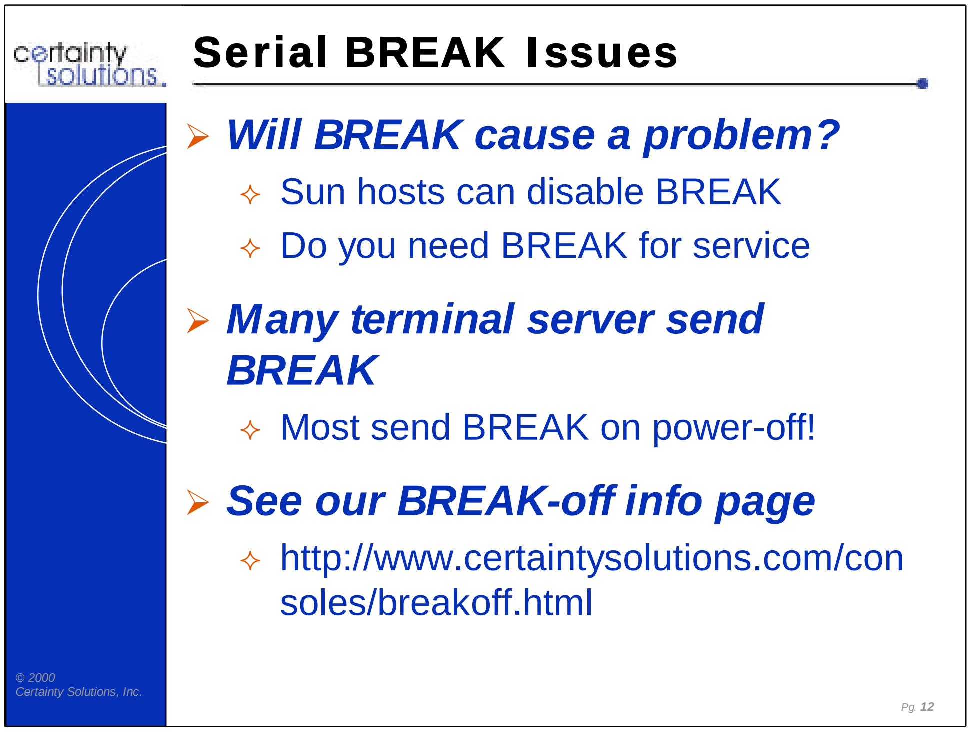
Do you have an existing management network? Where is it physically deployed? How hard would it be to extend it to other locations? (And is that expansion covered under your security policy?)

Physical access to the “secure” networks is part of your security scope. If you make it hard for someone to sniff your packets over the network, but don’t prevent them from connecting to the network itself, you still have an exposure.

You can lock some ports down on certain Ethernet switches. That is, once locked down, the switch only recognizes the MAC addresses of existing devices. If someone connects without authorization, they will not get link, or see packets.

There is still the exposure that someone could bring a laptop into the data center and plug into the console, but then the logs in your console server would stop accruing. It would take more sophistication to splice into the serial line and monitor both sides of the serial conversation.

Again, you need to consider the value of the information (when you have the secrets, and ‘they’ don’t), and the nature of the perceived threats. Only then can you look into solutions to the threats, and see if they are cost-effective.



Serial BREAK Issues

- **Will BREAK cause a problem?**
  - Sun hosts can disable BREAK
  - Do you need BREAK for service
- **Many terminal server send BREAK**
  - Most send BREAK on power-off!
- **See our BREAK-off info page**
  - http://www.certaintysolutions.com/consoles/breakoff.html

© 2000 Certainty Solutions, Inc.

Pg. 12

Serial BREAK:

1) An inversion (“MARK” state) on the data lead, for more than 1 complete character, including start and stop and parity bits.

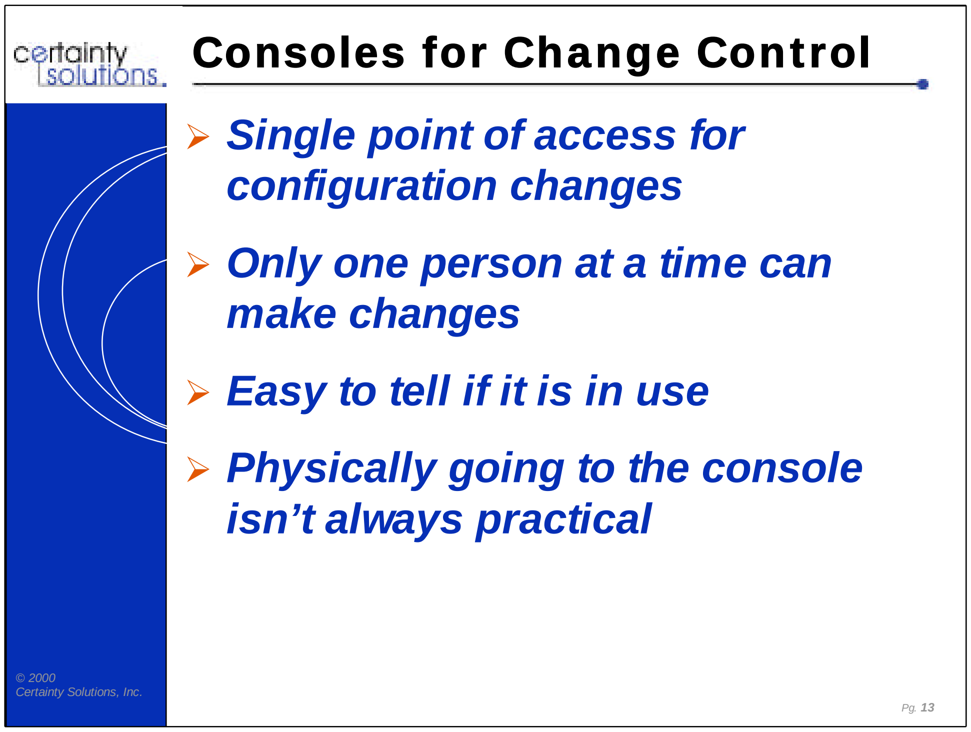
2) The logical equivalent to the Telnet BREAK signal, operationally.

Sun systems are the most common machines to have an operational problem when they see a serial BREAK signal. When they see a serial BREAK on their console port, they will stop all normal operations, and drop to the “ok>” prompt.

There are ways to prevent the Sun host from receiving, or acting on this signal. However, there are practical applications where you would WANT to use this signal to bring the machine to the OK prompt.

Recent versions of Solaris (2.6+) allow you to ignore this signal (see /etc/default/kbd). Solaris 8 can also replace the signal with a 3-character sequence of: CR, ~, CTRL-B. You can patch Solaris 2.6 and 7 (105924 and 107589 respectively) to provide the same functionality. SRDB ID 20427 has details about the patches.

Most terminal servers send a BREAK when you turn their power off. Imagine a large data center, with 96 Sun machines, all on UPS, and the power mains fail. Imagine that you find out that your terminal servers were NOT on the UPS, and all of your servers are sitting at the OK prompt until you can re-power the terminal servers, and type “go” to each machine. Now you understand why we care which terminal servers send BREAK at power-off.



Consoles for Change Control

- Single point of access for configuration changes
- Only one person at a time can make changes
- Easy to tell if it is in use
- Physically going to the console isn't always practical

© 2000
Certainty Solutions, Inc.

Pg. 13

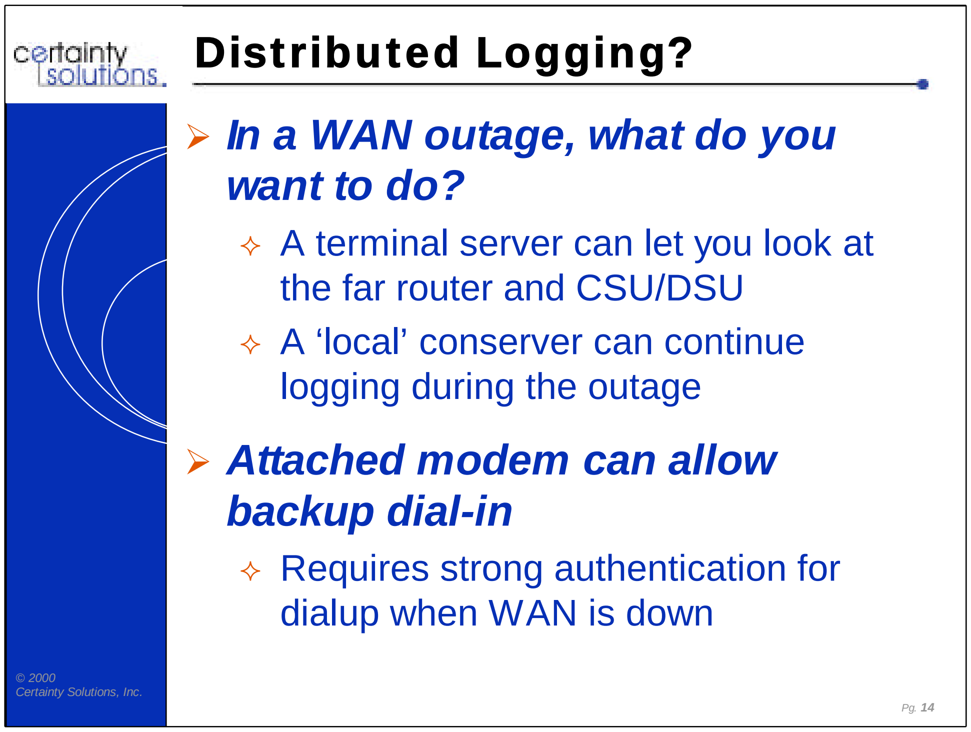
This is more of a process issue and a discipline issue. Even if you have remote access to the serial console, it is possible to have someone else making changes via a remote control session through the network.

When you have many administrators supporting any given device, you need to consider a change control process, to prevent one person from clobbering changes made by another person.

You can assign just one, or two, administrator(s) to manage each device, but that method doesn't scale well, and this method can suffer when an administrator goes on vacation, gets sick, has to serve on jury duty, etc.

Terminal servers will tell you if someone else is already attached to a particular port, but it may not tell you who is attached without the user taking extra steps to find out.

Does it snow where your office is? Does it get beastly hot in the summer? How long does it take to get around between the different data centers and wiring closets that you support? How often do you need to visit each?



Distributed Logging?

- ***In a WAN outage, what do you want to do?***
  - A terminal server can let you look at the far router and CSU/DSU
  - A 'local' conserver can continue logging during the outage
- ***Attached modem can allow backup dial-in***
  - Requires strong authentication for dialup when WAN is down

© 2000
Certainty Solutions, Inc.

Pg. 14

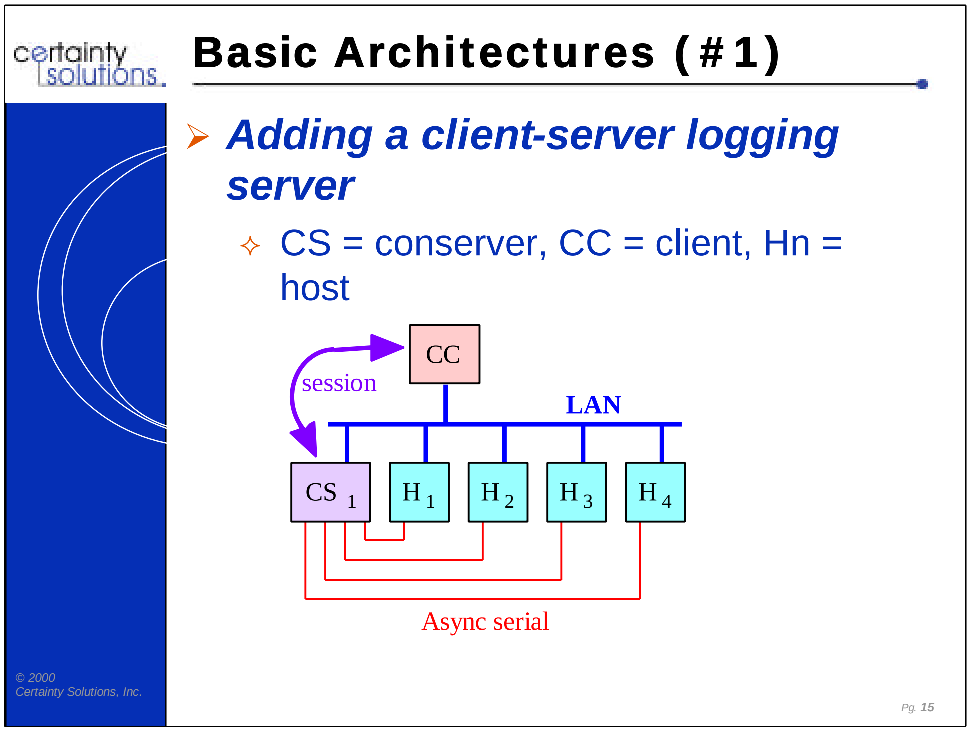
For the network administrator, when a WAN link fails, there are limited tests you can make from one end of a failed link, to try to determine the location of the failure.

With a modem on the 'far end' of a WAN link, the administrator could dial into the remote modem, and look at both ends at once, aiding in the triangulation of the problem.

When the WAN link fails, packets from the terminal server are lost. If you have a centralized console server, the reverse-telnet sessions are lost, and logging is stopped. If the hosts at the far site are complaining about anything, nobody will have heard it.

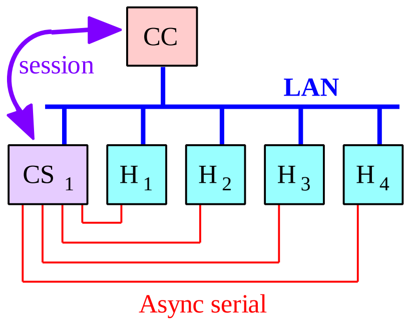
For the system administrator, a logging server at the far end means that the log messages during a failure can be captured. While you will see the expected NFS failure messages, you may find other dependencies that you hadn't expected, and then you have the ability to fix those.

You should strongly consider putting an authentication server in the far office, to provide for greater security against unauthorized access to the modem when you have it connected to a terminal server.



Basic Architectures (#1)

- **Adding a client-server logging server**
  - CS = conserver, CC = client, Hn = host



© 2000 Certainty Solutions, Inc.

Pg. 15

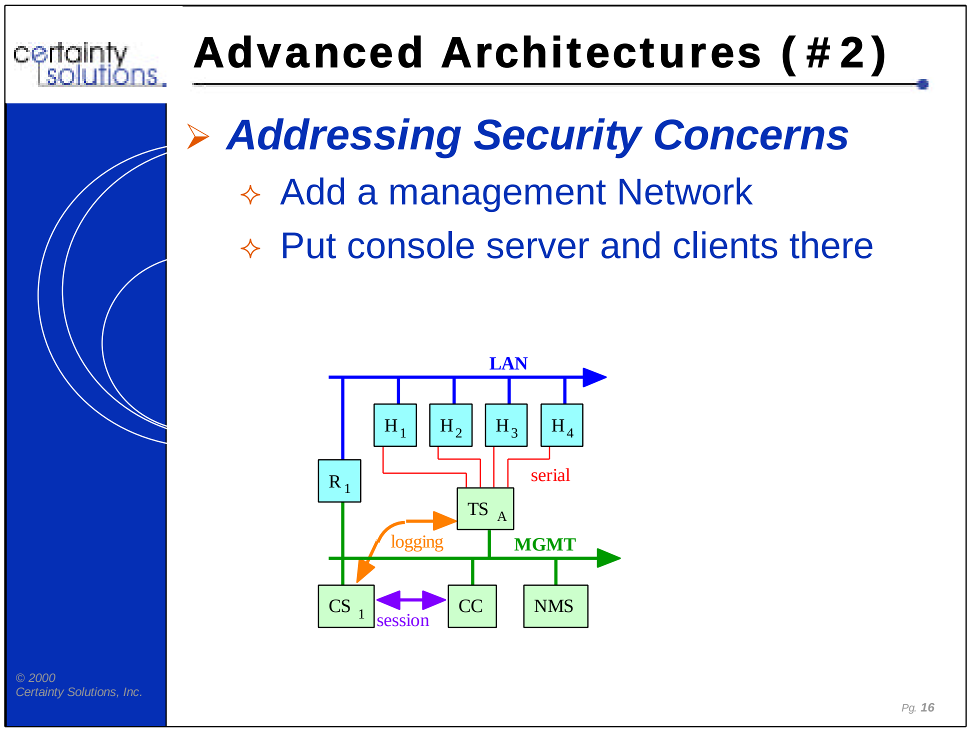
In this configuration, we presume we don’t worry about internal security.

In this case, the console server (CS) has multiple serial ports installed, connected to the various consoles that you care about.

Any data coming in from each attached device is appended to a log file for that specific device. (As an administrator, you may need to watch the size of the log files, and rotate them occasionally.)

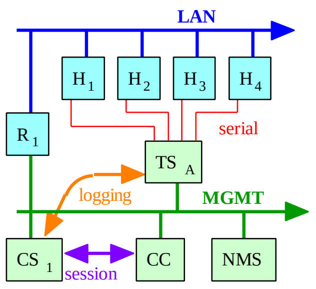
In our example, you would use a client (CC) to connect to the console server (CS), and attach to the console logging session that you want to talk to. Anything you type goes to the serial console you are talking to, while anything coming back from that console is logged, and then sent to the client(s) attached to that session. The conserver software watches the data stream from the user, looking for a specific escape sequence. The escape sequence and following action are not passed to the serial console (so they do not show up in the logs either).

Some console servers allow more than one client to attach to a single session at once. All clients see the data coming from the attached device, but only one client can type to the session. (Conserver functions in this way.)



Advanced Architectures (#2)

- **Addressing Security Concerns**
  - Add a management Network
  - Put console server and clients there

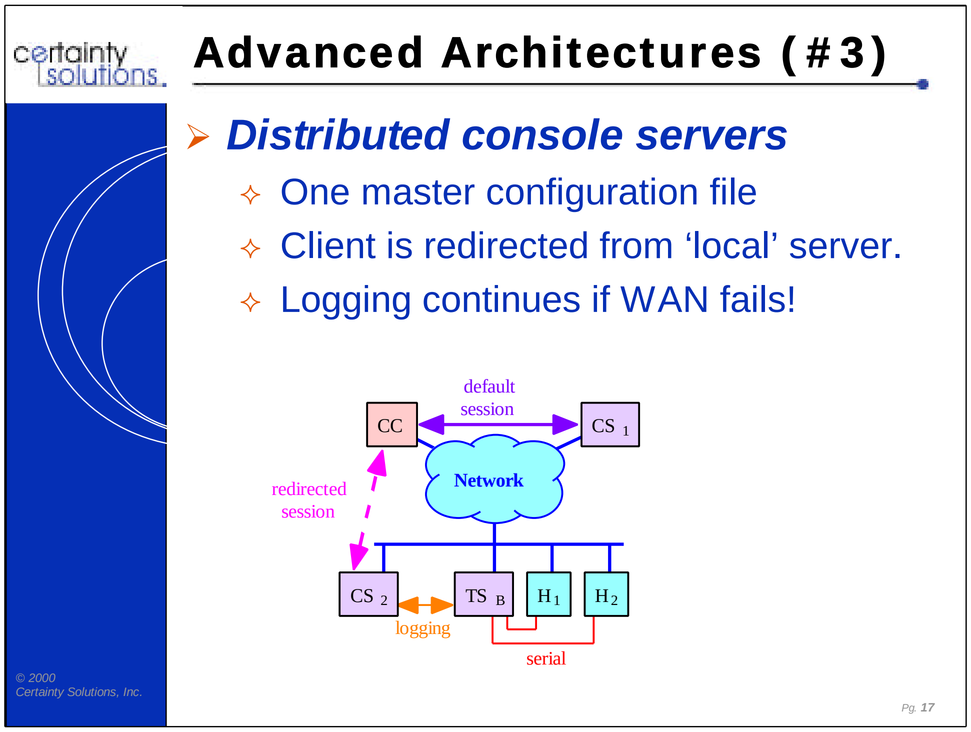


If you are concerned about someone sniffing the client-to-server connections, or the logging streams, then you probably already have a control/management network in place, where your monitoring and control activities take place.

In this model, the console server, and the client(s) all live on the management network, so that the client sessions, and the logging activity, only happen on the management network.

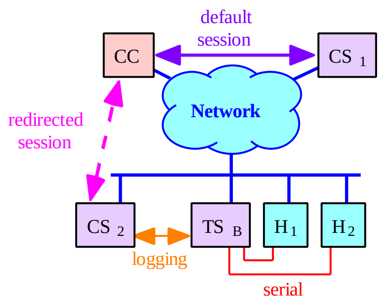
A good practice is to ensure that your management network is connected using switches, rather than hubs…this reduces the risk of someone sniffing the client-to-server console session traffic. (On a hub, anyone connected has the chance to hear whatever traffic comes through the hub, including the console traffic.)

Few console servers use SSL or SSH for the client-to-server connections, so these sessions travel in clear text. For this reason, if you are located outside of the management (or security) perimeter, you should consider making an SSH connection to a client host that is on the management net, and then making the client connection from there.



Advanced Architectures (#3)

- **Distributed console servers**
  - One master configuration file
  - Client is redirected from 'local' server.
  - Logging continues if WAN fails!



© 2000 Certainty Solutions, Inc.

Pg. 17

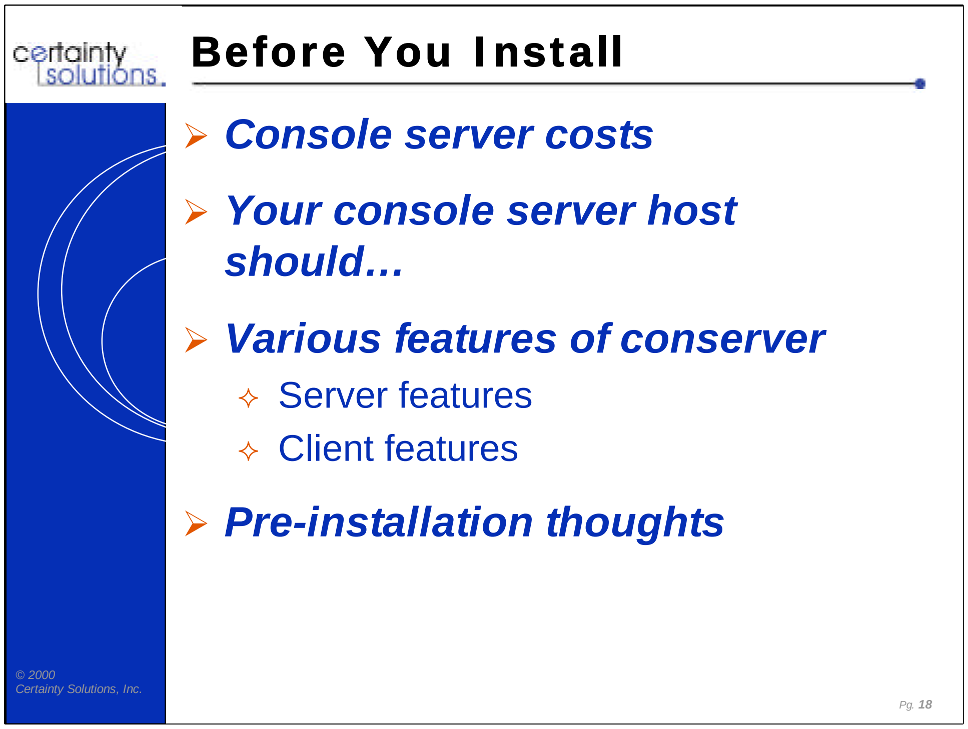
In the case of conserver, you can indicate in the configuration file which of a number of distributed conserver hosts is maintaining the logs for any particular device, as well as indicating which remote terminal server is directly connected to the device.

You can then push this master configuration file to all of the distributed conserver hosts.

Each client should try to connect with the conserver host that is closest to the client for all connections. (You can do this via hostnames, or even with a local host file entry.)

If a client (CC) connects to its local conserver host (CS1), but is trying to reach a device that is connected to another conserver host (CS2), the client is redirected to the other conserver host. The client session is then connected with the session of the desired device.

Whenever you have a distributed conserver implementation, you must be sure to append the "@server" tag to each console entry, otherwise each console server will try and manage the same console. We'll illustrate this feature in the slides to come.



Before You Install

- **Console server costs**
- **Your console server host should…**
- **Various features of conserver**
  - Server features
  - Client features
- **Pre-installation thoughts**

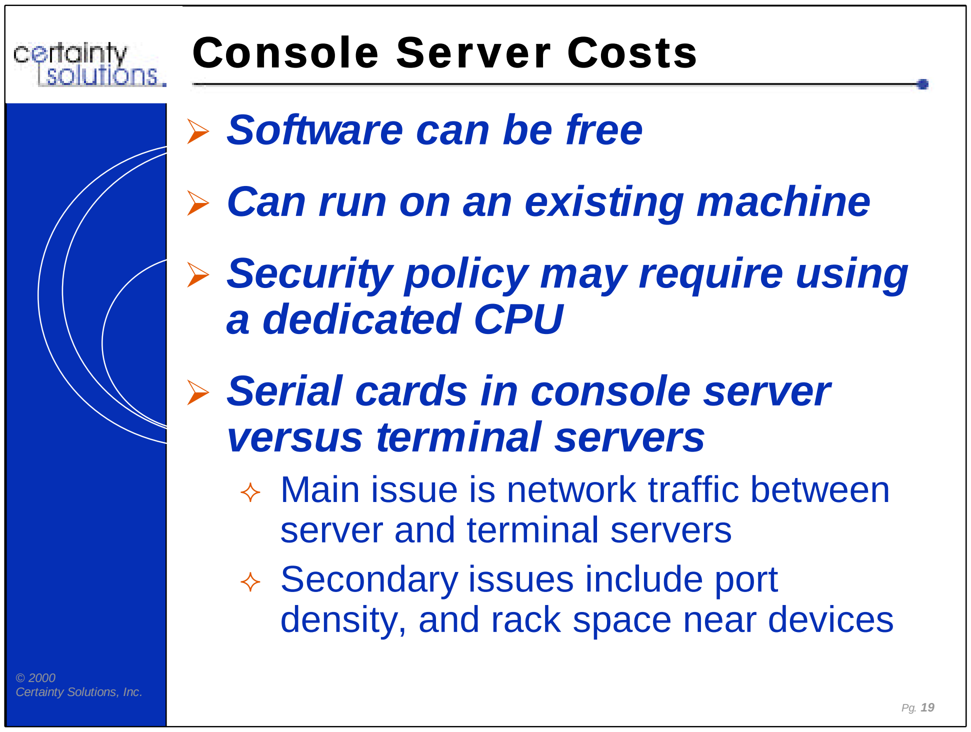
© 2000
Certainty Solutions, Inc.

Pg. 18

There are various things you should consider and know before installing conserver. There are costs associated with console management: software costs (conserver is free), host resources, and serial ports.

There are also general guidelines you should follow to make sure that your console servers are able to come up before any of the rest of your equipment (how else will you get to all those consoles).

Knowing the capabilities of conserver will enable you to configure the software and design a system such that you can take advantage of everything is has to offer.



Console Server Costs

- Software can be free
- Can run on an existing machine
- Security policy may require using a dedicated CPU
- Serial cards in console server versus terminal servers
  - Main issue is network traffic between server and terminal servers
  - Secondary issues include port density, and rack space near devices

© 2000 Certainty Solutions, Inc.

Pg. 19

There are many free solutions available, depending on the features that you want. Most notably, free software often lacks a strong support offering, which means you need to be able to support your site with minimal help.

Our favorite free applications is the conserver application, from http://www.conserver.com/.

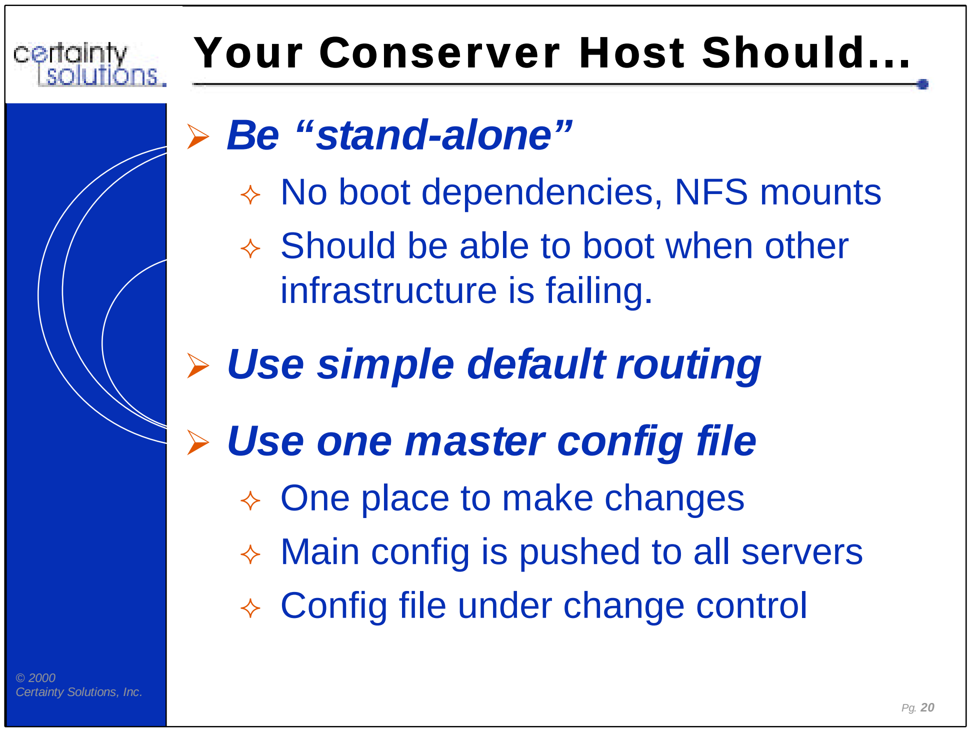
There are at least two commercial applications available to you, if you feel the need to have a pay-for support option.

You can often use free software on an existing server, as the needs of the software themselves do not require much memory or CPU. This can help offset the costs of a new terminal server deployment.

However, shared machines often mean security vulnerabilities in the other applications, which could allow unauthorized folks to see the logs (or tamper with them). A shared system often means more users that have legitimate access to the machine, which can be another vulnerability.

If security and integrity of the logs is a concern, you should consider purchasing a dedicated machine, installing a hardened OS, and limiting the number of folks who have login access to the CPU.

Adding multi-port serial cards to a machine increases power consumption, heat output, and (depending on the card) can also increase CPU load.



Your Conserver Host Should…

- ***Be "stand-alone"***
  - No boot dependencies, NFS mounts
  - Should be able to boot when other infrastructure is failing.
- ***Use simple default routing***
- ***Use one master config file***
  - One place to make changes
  - Main config is pushed to all servers
  - Config file under change control

© 2000
Certainty Solutions, Inc.

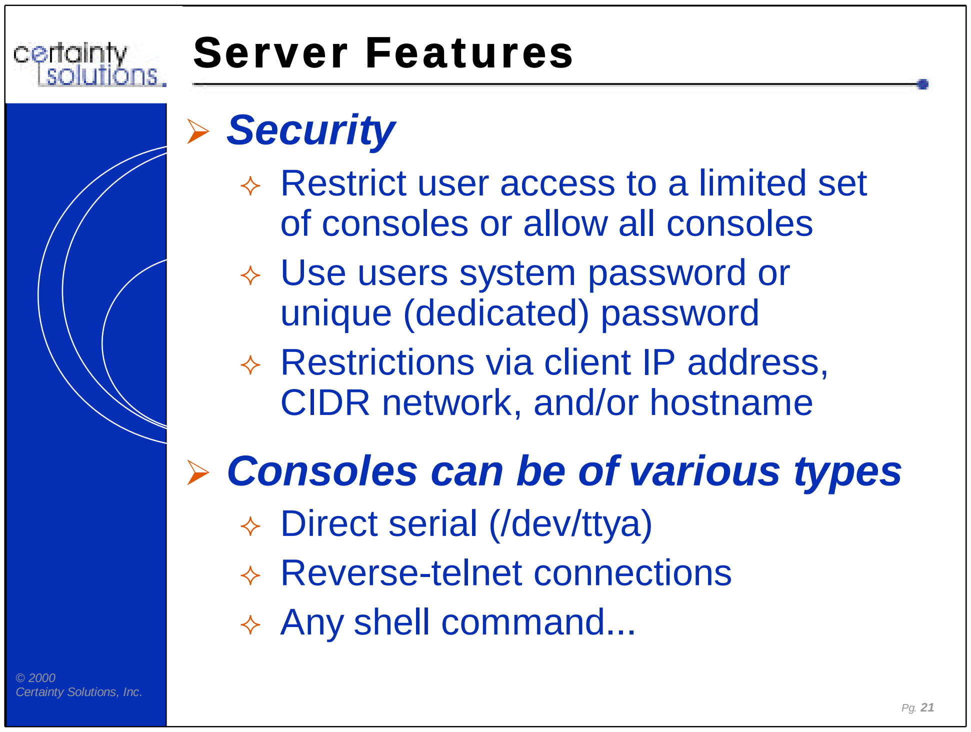
Pg. 20

In a complete power failure, the console server should be one of the first hosts turned back on (just after you turn all of the terminal servers, and basic network gear). You want the console server to boot quickly, and to try to establish the reverse telnet sessions to the terminal servers before you start bringing up your main infrastructure hosts, and peripheral network gear.

The server should not depend on any other machines in order to log. In the event that some of your network or infrastructure servers fail, you want the console server to be capturing log data for as long as possible during a failure.

If you are planning to use distributed console servers (putting some in remote offices, or distributed data centers), you should create one central console server file, which would include data for which servers will log for which ports. It's a great idea to keep the master file(s) under a change control system.

By keeping one file, under change control, you will have a single place to make changes, and only one place to check to see if a device is listed. Adding a new server later becomes easy, since you can edit one file to make the additions, and then push that file to all of the console servers, including the new one.



Server Features

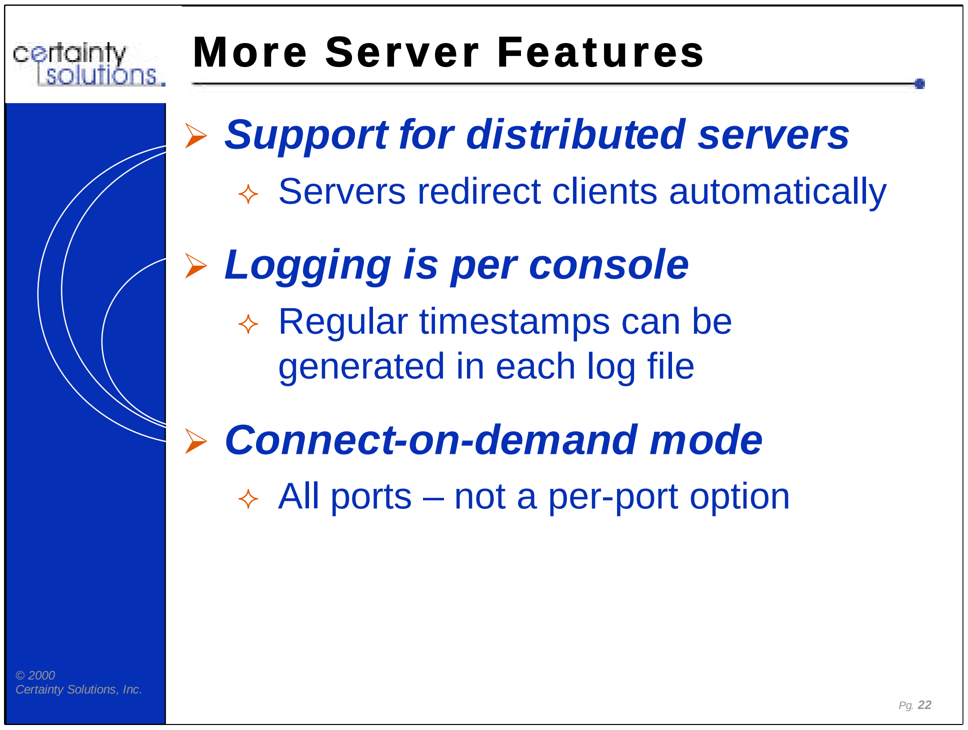
- **Security**
  - Restrict user access to a limited set of consoles or allow all consoles
  - Use users system password or unique (dedicated) password
  - Restrictions via client IP address, CIDR network, and/or hostname
- **Consoles can be of various types**
  - Direct serial (/dev/ttya)
  - Reverse-telnet connections
  - Any shell command...

© 2000
Certainty Solutions, Inc.

Pg. 21

Security (or lack thereof) is one of the greatest problems. While conserver can restrict users to certain consoles, require them to come from certain networks or hosts, and provide for dedicated passwords, there is no support for encrypted communication between the conserver host and the terminal servers, and no special protection for "unattached" consoles. If strong security is an issue, you must try and compensate for this by using one or more of the techniques we've described (like a private network where all console management resides with limited access).

Many things can be treated as a console. Local I/O ports, reverse-telnet enabled terminal servers, or any command ("/bin/sh –i" is fun!) is possible. The ability to make any command a "console" should enable conserver to integrate into most environments. Be warned that since conserver runs as root, all commands will be running as root as well. This can be quite dangerous and a very large security issue.



More Server Features

- **_Support for distributed servers_**
  - Servers redirect clients automatically
- **_Logging is per console_**
  - Regular timestamps can be generated in each log file
- **_Connect-on-demand mode_**
  - All ports – not a per-port option

© 2000
Certainty Solutions, Inc.

Pg. 22

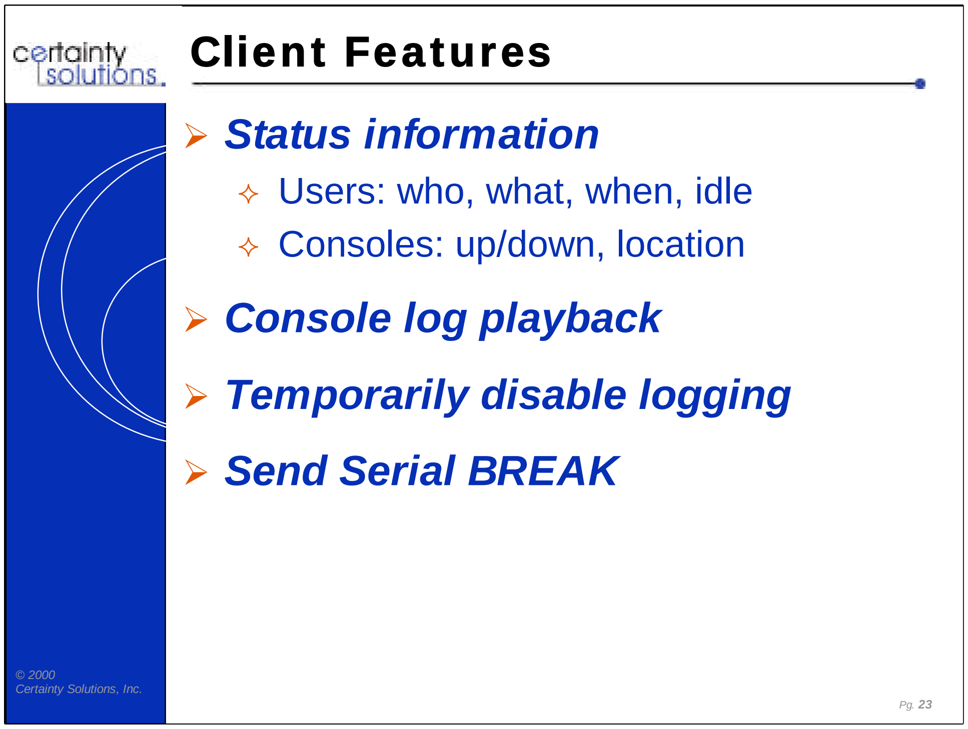
Multiple distributed console servers are supported. If there are no security zones restricting access, each server should use the same conserver.cf file. The conserver software will automatically redirect the clients to the appropriate console server.

Logging and regular timestamps ("marks") can be enabled on a per-console basis, through the entries in the conserver.cf file.

You may not want to log certain ports, and that's just fine. You can determine which files are logged by the entries for each host in the conserver.cf file.

Regular timestamps are useful for providing time bounds on console output for consoles that don't automatically timestamp their messages. You can set any port to provide a date/timestamp into the log file for that port by including the desired interval for each device that you want to have conserver timestamps.

Connect-on-demand mode will open a connection to a serial port when a user makes a connection to the console server. Connections are closed when all users have disconnected from that console. All logging outside of when connections are established are lost in this case. (This is a 'global option'. That is, if you enable this feature by invoking conserver with the "-I" flag, all of the consoles will work this way…if nobody is watching, then no logging occurs either.)



Client Features

- *Status information*
  - Users: who, what, when, idle
  - Consoles: up/down, location
- *Console log playback*
- *Temporarily disable logging*
- *Send Serial BREAK*

© 2000
Certainty Solutions, Inc.

Pg. 23

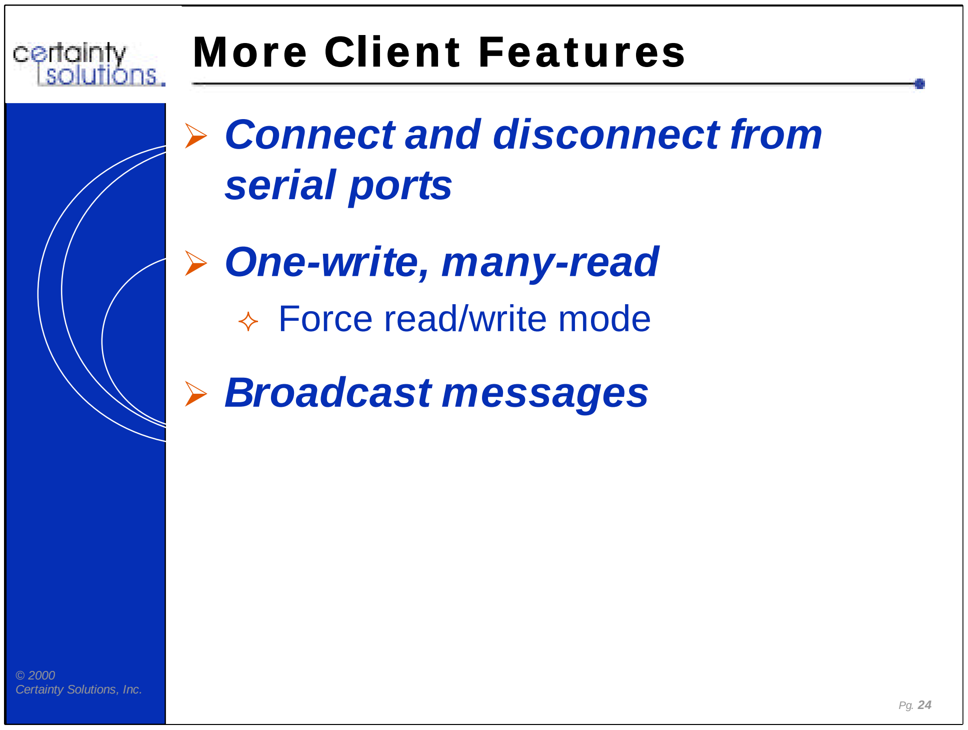
These are really implemented within the server, but they are seen as the user experience.

Various status information regarding users and consoles can be retrieved. Things such as who's connected to a particular console, what mode (read/write or read/only) the user is in, when they connected, and how long they've been idle are available. Console status, such as whether they port is connected or not and where the port is located (terminal server, local port, or command) can be retrieved using the client as well.

Temporarily disabling log files is a nice feature for when you know you'll be displaying unencrypted passwords or other sensitive information. When the active read/write user disconnects, logging is automatically re-enabled.

Issuing the command `console -W` will display which users are connected to which consoles, and which machine they are connecting from.

Issuing the command `console -V` will display the version of Conserver running, the initial master server (for that client), and the default escape sequence for that client.



More Client Features

- Connect and disconnect from serial ports
- One-write, many-read
  - Force read/write mode
- Broadcast messages

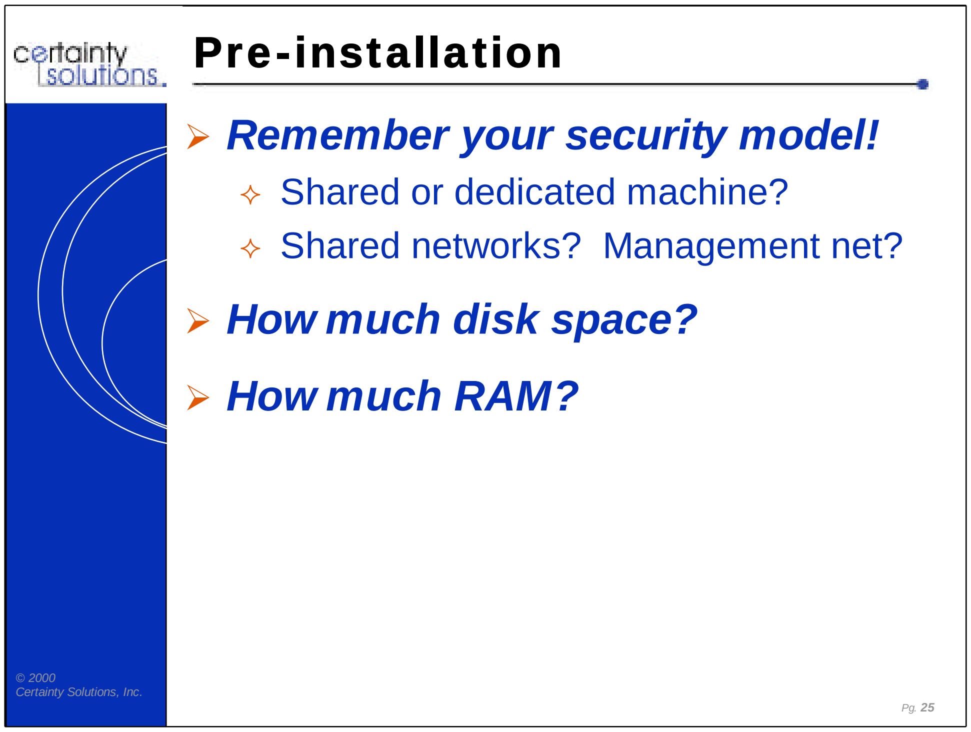
© 2000
Certainty Solutions, Inc.

Pg. 24

Connecting and disconnecting from serial ports helps reset things to a known state. Sometimes you can run into a software flow-control problem or general port lockup and dropping/closing the connection and re-establishing it can usually clear things.

The server enforces a policy of at most one client being in read/write mode for each console at any one time. Other clients attached to the console are in read-only (or "spy") mode. A user can "bump" others into "spy" mode and promote themselves to read-write at any time. Being able to forcibly acquire read/write mode is useful in emergencies and for mentoring purposes. If a less experienced admin is about to do something dangerous or an admin has left his keyboard but is still connected to the console, you can "steal" the connection and work on the issue.

Broadcast messages can be sent via the client "`console -b`" option. It's similar to a " wall" (write to all) message, but it is directed only clients currently connected to a console. The message is sent to users connected to all conservers, in a distributed conserver implementation.



Pre-installation

- **Remember your security model!**
  - Shared or dedicated machine?
  - Shared networks? Management net?
- **How much disk space?**
- **How much RAM?**

© 2000
Certainty Solutions, Inc.

Pg. 25

You need to distribute and connect the terminal server(s) and your conserver host(s) to your network based on your security needs.

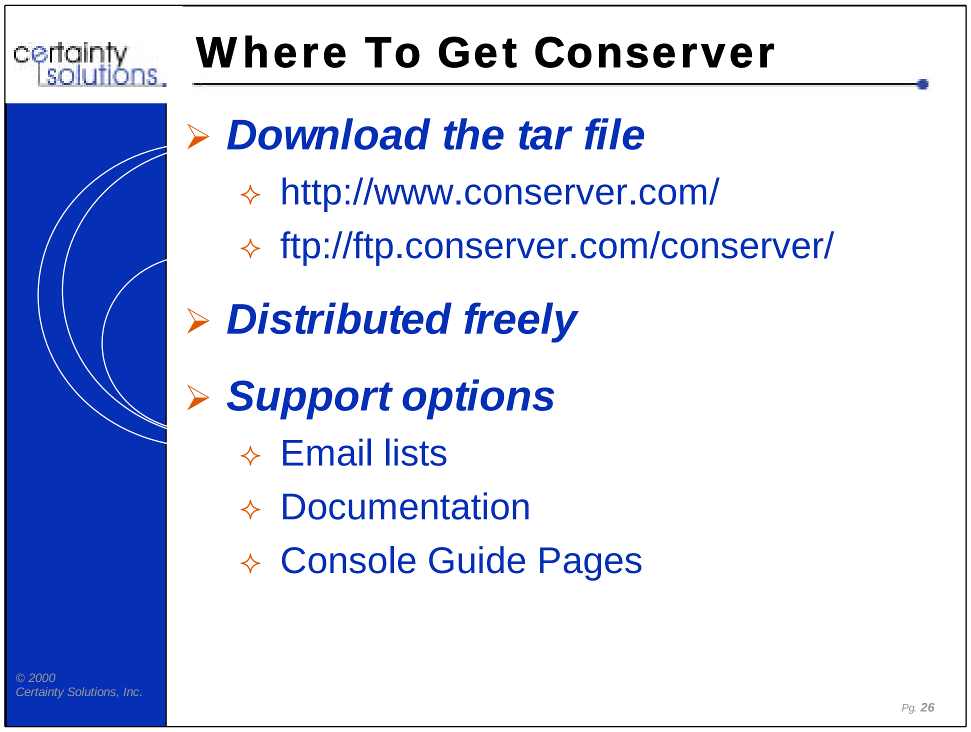
Disk space requirements are dependent upon the amount of data you expect to be logging. Some sites find a 1GB partition sufficient for long-term logging, others may find it too small. Disk is relatively cheap and the benefits of logging is great, so allow for a worst-case scenario of multiple hosts logging constant streams of 9600 baud data for a long time.

RAM requirements are mainly determined by how many consoles you plan to manage with the server. The calculation is based on the values that you give for the `MAXMEMB` and `MAXGRP` options in conserver/cons.h file.

`MAXMEMB` is the maximum number of consoles a process will manage.

`MAXGRP` is the maximum number of processes that will be spawned.

A default installation with ~120 consoles uses ~35MB under Solaris.



Where To Get Conserver

- ***Download the tar file***
  - http://www.conserver.com/
  - ftp://ftp.conserver.com/conserver/
- ***Distributed freely***
- ***Support options***
  - Email lists
  - Documentation
  - Console Guide Pages

© 2000
Certainty Solutions, Inc.

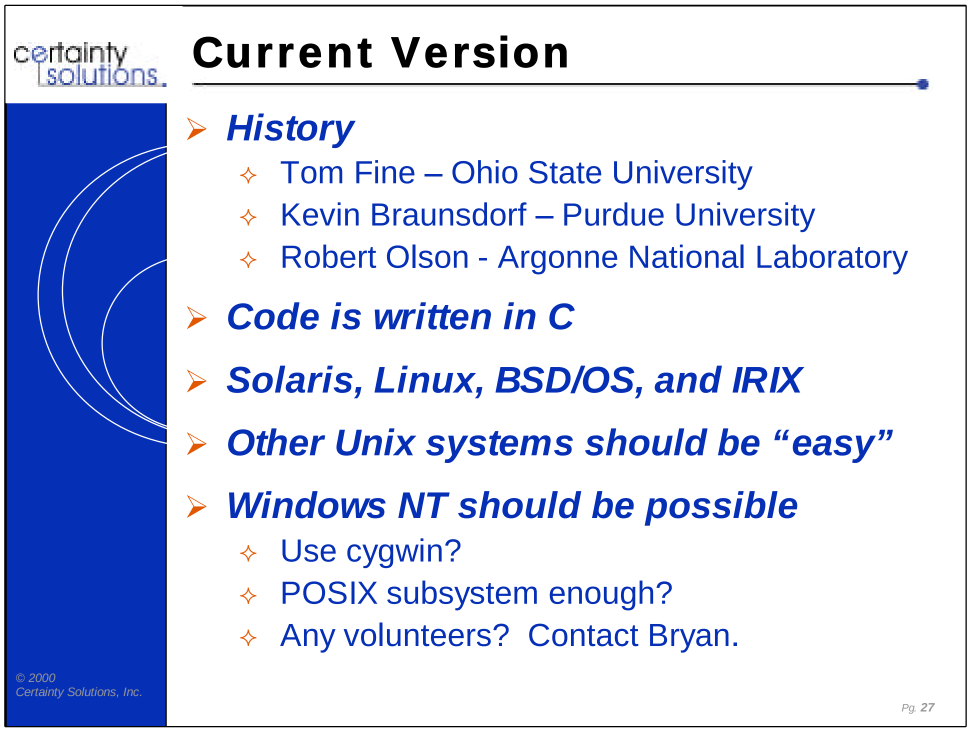
Pg. 26

The primary distribution site for conserver is http://www.conserver.com/.

Currently, there is no commercial support offering for this application. We’re trying to improve the documentation, and we are always interested in hearing comments from users, including suggestions for improvements or new features.

You can find additional supporting information about connecting devices to terminal servers at http://www.conserver.com/consoles/. There is a mirror of these pages at http://www.certaintysolutions.com/consoles/.

We have established email lists for new-version announcements, as well as a BOF-like discussion group. Subscription information can be found at the conserver.com web site (http://www.conserver.com/).



Current Version

- **History**
  - Tom Fine – Ohio State University
  - Kevin Braunsdorf – Purdue University
  - Robert Olson - Argonne National Laboratory
- **Code is written in C**
- **Solaris, Linux, BSD/OS, and IRIX**
- **Other Unix systems should be "easy"**
- **Windows NT should be possible**
  - Use cygwin?
  - POSIX subsystem enough?
  - Any volunteers? Contact Bryan.

© 2000
Certainty Solutions, Inc.

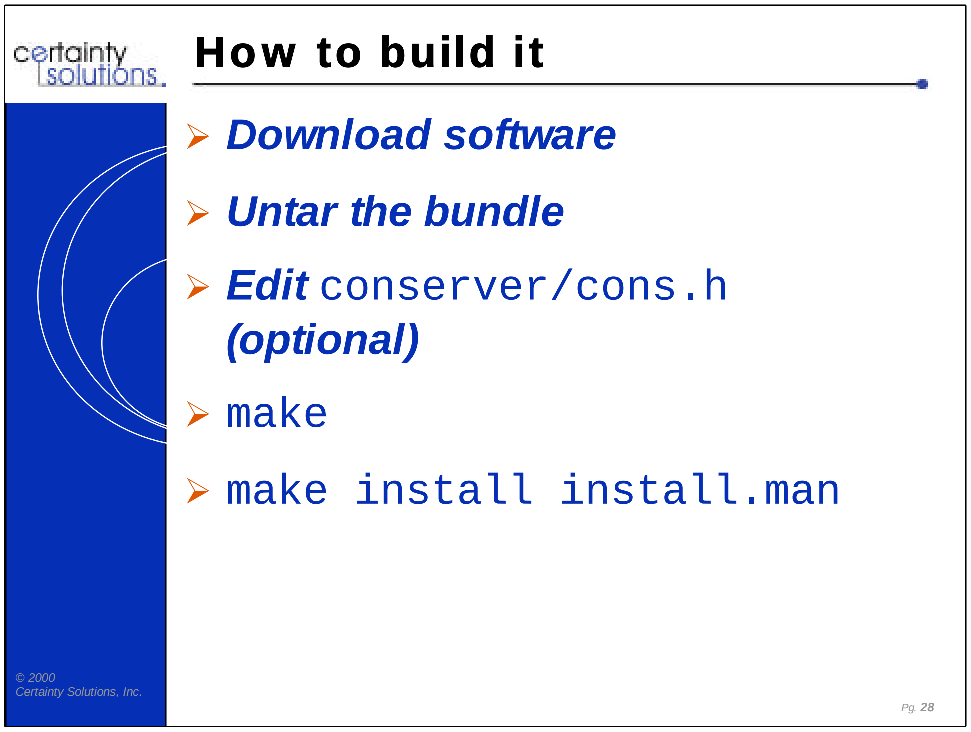
Pg. 27

This version of conserver is a combination of many people's work. Use of this code is not restricted, but credit must be given to the authors. Please check the conserver code for full copyright information.

Although only four operating systems are listed as official ports, compilation under others should be relatively straightforward. What you need to do is define the appropriate set of tags in conserver/cons.h and educate the build scripts. If anyone succeeds in porting to HP/UX, *BSD, or others, please contribute the differences. Details for porting to other operating systems can be found in the port/README file.

Windows NT should be a possibility. We don't know of anyone who has tried, but the simplest approach might be the cygwin environment. If someone succeeds, using anything, please let us know.

http://www.cygwin.com



How to build it

- *Download software*
- *Untar the bundle*
- *Edit* `conserver/cons.h` *(optional)*
- `make`
- `make install install.man`

© 2000
Certainty Solutions, Inc.

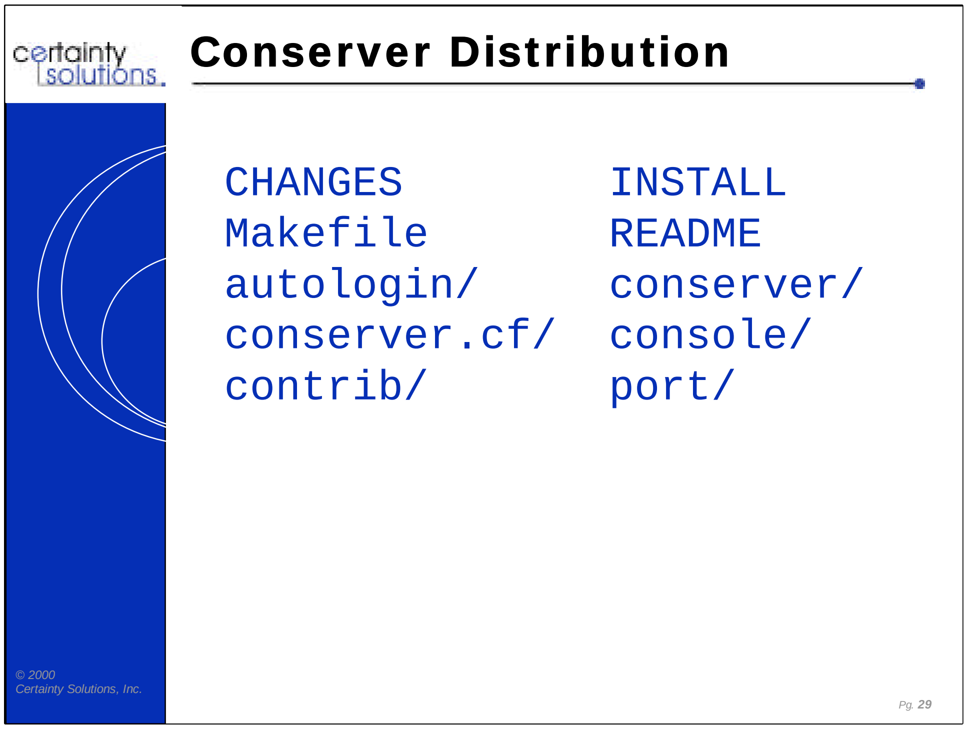
Pg. 28

If conserver has been ported to your platform, the steps are simple. Editing the conserver/cons.h file is optional, but larger or more complex sites might want to consider adjusting defaults to suit their environment.

Specific steps:

```
gunzip conserver-GNAC-6.16.tar.gz
tar xf conserver-GNAC-6.16.tar
cd conserver-GNAC-6.16
vi conserver/cons.h (optional)
make
make install install.man
```

To change the installation prefix of /usr/local, override the `PREFIX` variable when you use make install: `env PREFIX=/tools/conserver-6.16 make install install.man`



Conserver Distribution

```
CHANGES          INSTALL
Makefile         README
autologin/       conserver/
conserver.cf/    console/
contrib/         port/
```

© 2000
Certainty Solutions, Inc.

Pg. 29

When you untar the tarfile, you will find a directory named the same name as the tarfile (without the .tar extension). The slide above shows you the directories and files you can expect to find inside that new directory.

The README file at the top level will guide you through the distribution. Most things should be obvious. Here are some of the things that may not be.

The `autologin` directory is full of code that I haven't even tried to compile or use. Check out the `README` files if you're interested, but this code is designed to provide a shell on a host instead of a login prompt – so you don't have to log into your consoles.

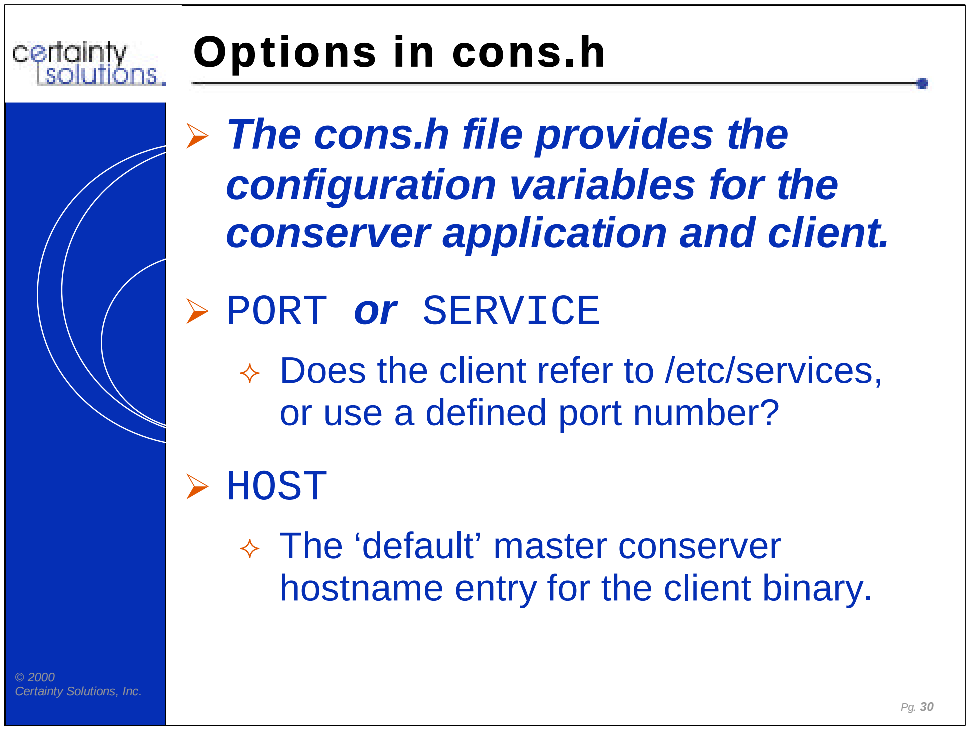
The `contrib` directory will contain various contributions from the community. Currently it has instructions and support files for building a Solaris package.

The `port` directory contains the build scripts for detecting the operating system type and setting the appropriate flags during compilation. Again, a README in that directory will guide you through the files as well as the basic process of "porting" the software to another system.

The `conserver` directory contains the source code for the application.

The `console` directory contains the sample configuration files.

The `conserver.cf` directory contains the documentation.



Options in cons.h

- ***The cons.h file provides the configuration variables for the conserver application and client.***
- PORT ***or*** SERVICE
  - Does the client refer to /etc/services, or use a defined port number?
- HOST
  - The 'default' master conserver hostname entry for the client binary.

© 2000
Certainty Solutions, Inc.

Pg. 30

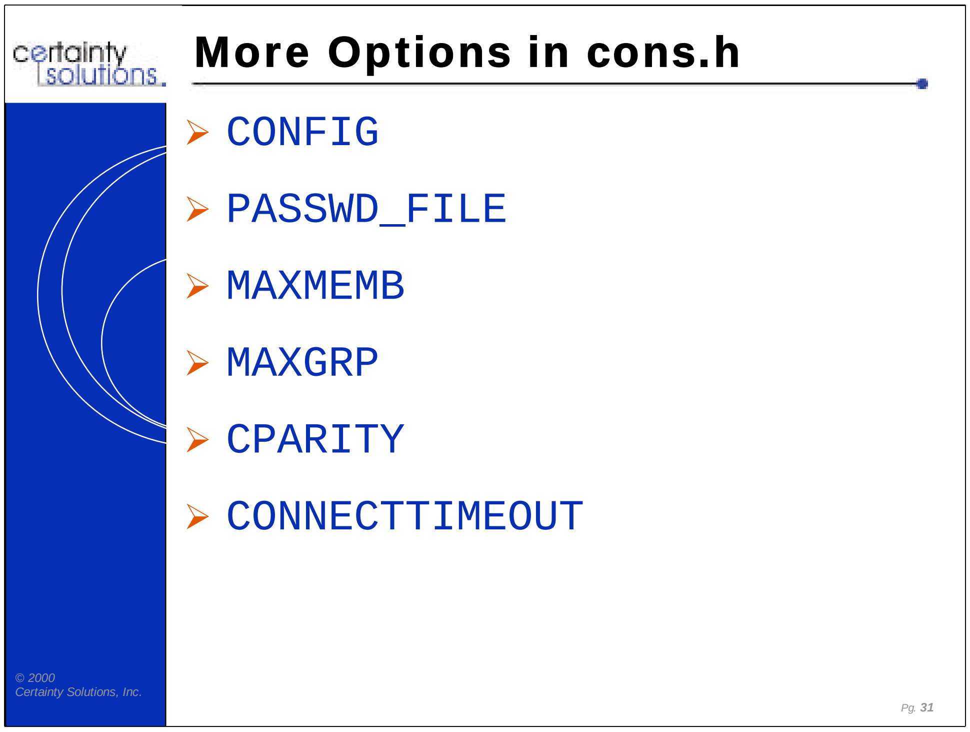
The default options in `conserver/cons.h` are appropriate for general installations of conserver. There are various reasons for changing the defaults. The format used is:

`VARIABLE`: Description [`default value`]

`PORT` or `SERVICE`: The default is to use a TCP /etc/services entry of "conserver". Clients machines as well as the server need this service defined. To hard-code a port number, comment out `SERVICE` and define `PORT` to a number.

Example: [`conserver|782`]

`HOST`: The default master conserver hostname. [`console`]
If you have one console server and multiple namespaces, you'll probably want to fully-qualify the name so you don't have to have aliases defined in each namespace. With distributed console servers and distributed clients, an unqualified name allows for "localization". This way clients initially connect to the local server and don't rely upon WAN connections and other possible hazards for local console services.



More Options in cons.h

- CONFIG
- PASSWD_FILE
- MAXMEMB
- MAXGRP
- CPARITY
- CONNECTTIMEOUT

© 2000
Certainty Solutions, Inc.

Pg. 31

Below are more options you can adjust.

`CONFIG`: Path to the configuration file. `[/etc/conserver.cf]`

`PASSWD_FILE`: Path to the password file. `[/etc/conserver.passwd]`

`MAXMEMB`: Maximum number of consoles per process. `[8]`

`MAXGRP`: Maximum number of conserver processes. `[32]`

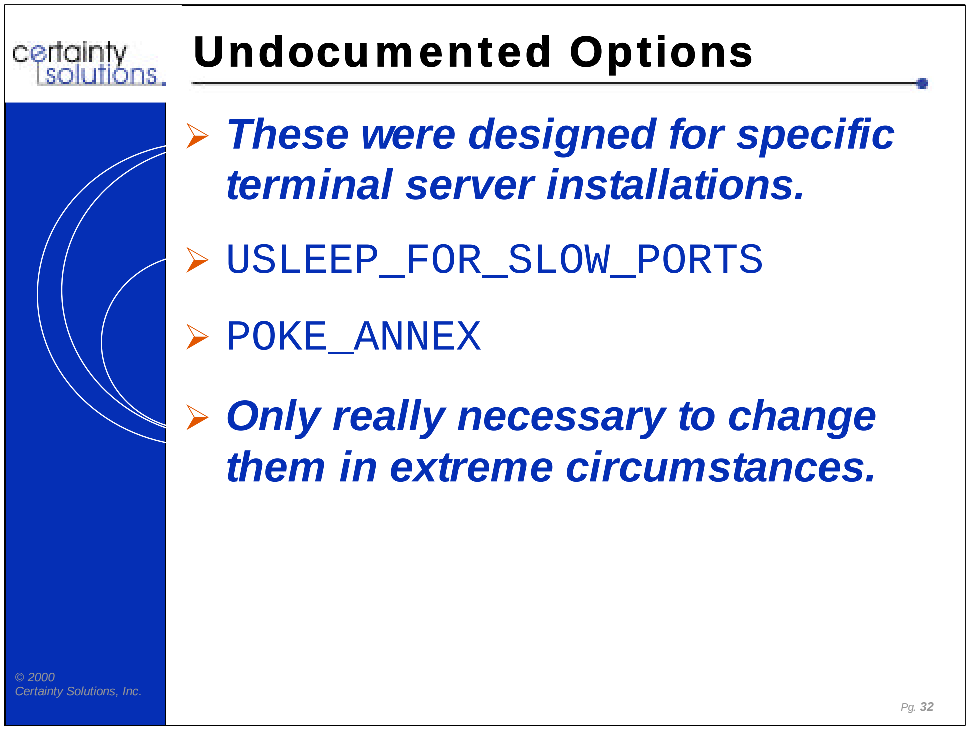
> `MAXMEMB*MAXGRP` is the maximum number of consoles conserver will handle. We've seen 16*96 work pretty well. Increasing `MAXMEMB` beyond 16, however, may not be wise. Increasing `MAXMEMB` reduces memory usage, however it also reduces parallelism. If you have a terminal server (or more) off-line, your startup time will increase.

`CPARITY`: Clear the high bit on data from the console `[1]`

> (You can set this to 0 if you want an 8-bit clean path through conserver, but this could get messy if a 7-bit session somehow trips the high-order bit, and starts displaying as graphics characters…)

`CONNECTTIMEOUT`: The time to connect to a TCP port. `[60]`

> The default could very well seem like an eternity if a terminal server is down.



Undocumented Options

- *These were designed for specific terminal server installations.*
- `USLEEP_FOR_SLOW_PORTS`
- `POKE_ANNEX`
- *Only really necessary to change them in extreme circumstances.*

© 2000
Certainty Solutions, Inc.

Pg. 32

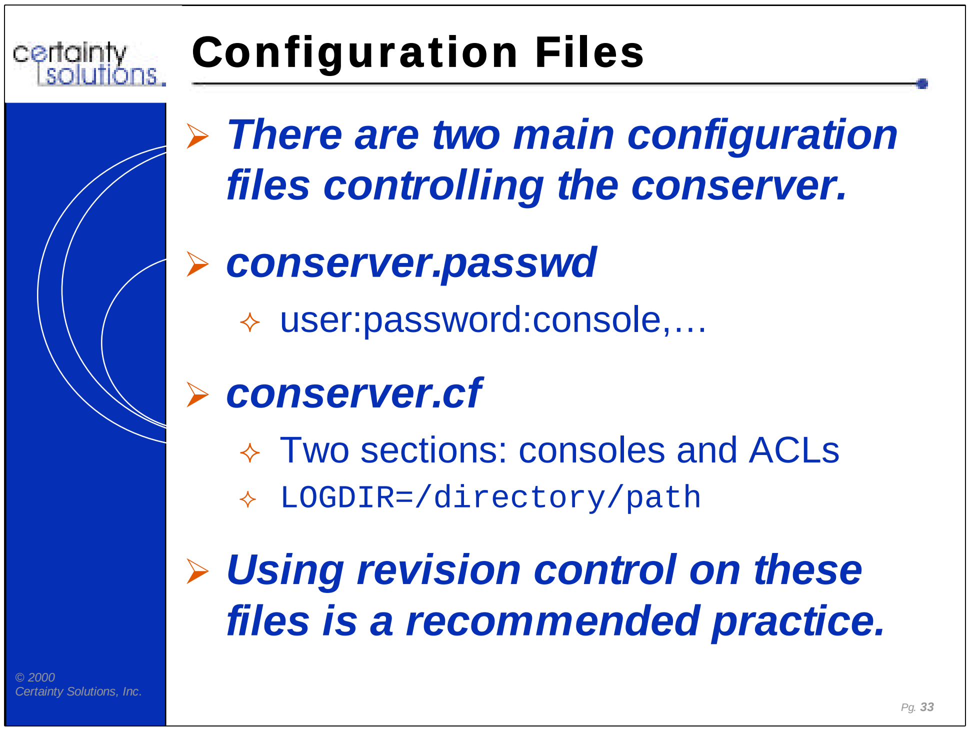
The following can be added to conserver/cons.h if you feel they are useful or need changing.

`USLEEP_FOR_SLOW_PORTS`: Number of microseconds to wait between bringing up consoles. [`100000`]

> Different terminal servers have different connection rates – try and connecting to too many ports too quickly and the terminal server will reject your connection. The default appears to be a good compromise between speed and functionality.

`POKE_ANNEX`: If defined, sends a carriage-return to each console upon connection. [`undefined`]

> A long, long time ago we used this option to because, for whatever reason, the Annex terminal servers needed to be "poked" to actually start communicating. They may even still do this – if so, define this option.



Configuration Files

- *There are two main configuration files controlling the conserver.*
- *conserver.passwd*
  - user:password:console,…
- *conserver.cf*
  - Two sections: consoles and ACLs
  - `LOGDIR=/directory/path`
- *Using revision control on these files is a recommended practice.*

© 2000
Certainty Solutions, Inc.

Pg. 33

Use revision control for editing these files, and push copies of the files into place only when they've been validated. (This can include looking at an RCSDIFF to determine that the changes are what you thought they were.) There is currently no automated tool for checking the syntax of your conserver files.

The `conserver.passwd` file has lines of the form

“`user:password:console-1, console-n,...`”

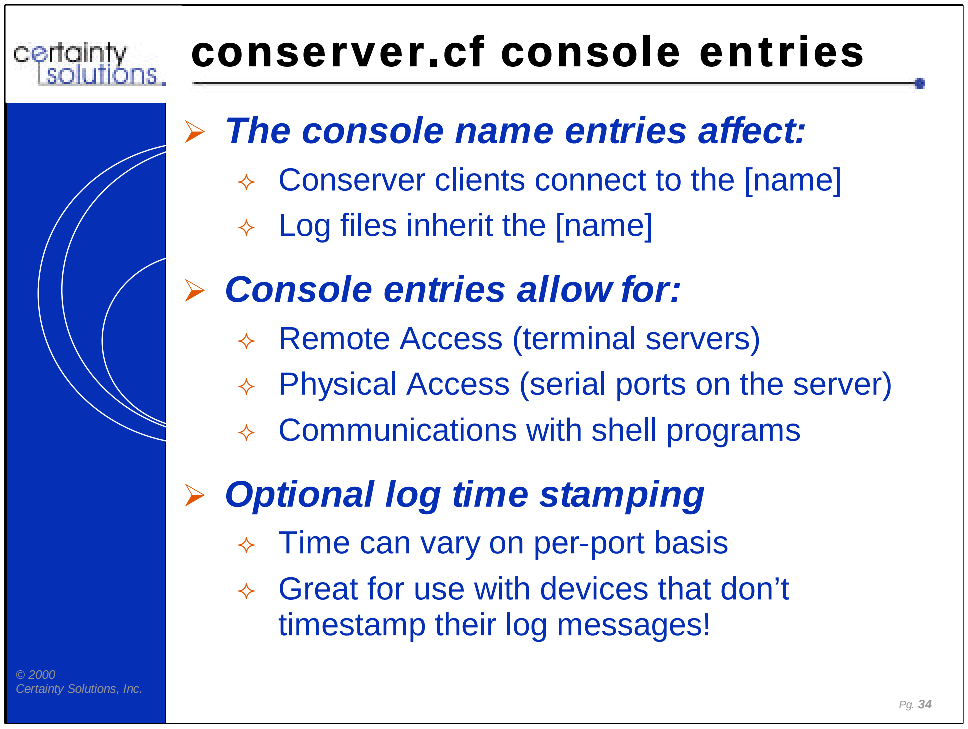
If the password is “`*passwd*`”, then conserver will use the system password for that user. Otherwise, the password entry is expected to be a valid crypt() string (just cut-and-paste out of your password/shadow file).

The console entries are names of consoles (listed in the `conserver.cf` file) that user is allowed to access. If the special entry of “`any`” is used, that user will be able to connect to any console.

The `conserver.cf` file is in two sections, separated by the line “%%”. The first half has console entries and directives, the second half has ACLs.

Using the “`LOGDIR=/some/path`” option, you can root all unqualified logfile entries. `LOGDIR` can be reset throughout the configuration file to alter the logfile root.

The next slides will provide more detail.



conserver.cf console entries

- **The console name entries affect:**
  - Conserver clients connect to the [name]
  - Log files inherit the [name]
- **Console entries allow for:**
  - Remote Access (terminal servers)
  - Physical Access (serial ports on the server)
  - Communications with shell programs
- **Optional log time stamping**
  - Time can vary on per-port basis
  - Great for use with devices that don't timestamp their log messages!

© 2000
Certainty Solutions, Inc.

Pg. 34

The first half (before the "%%" line) of a conserver.cf file contains console entries and, optionally, special keyword assignments.

Console entries are made up of 5 fields.

The console name

the location of the serial port

a secondary location value

a logfile name

and an optional mark time interval

If the logfile name contains the '&' character, it will be replaced with the console name.

If the logfile has a relative path, the value of `LOGDIR` will be prefixed.

Mark intervals, if used, are a numeric value followed by 'm', 'h', or 'd', for minute, hour, and day (4h = 4 hours).

For terminal server-connected ports, the syntax looks like this:

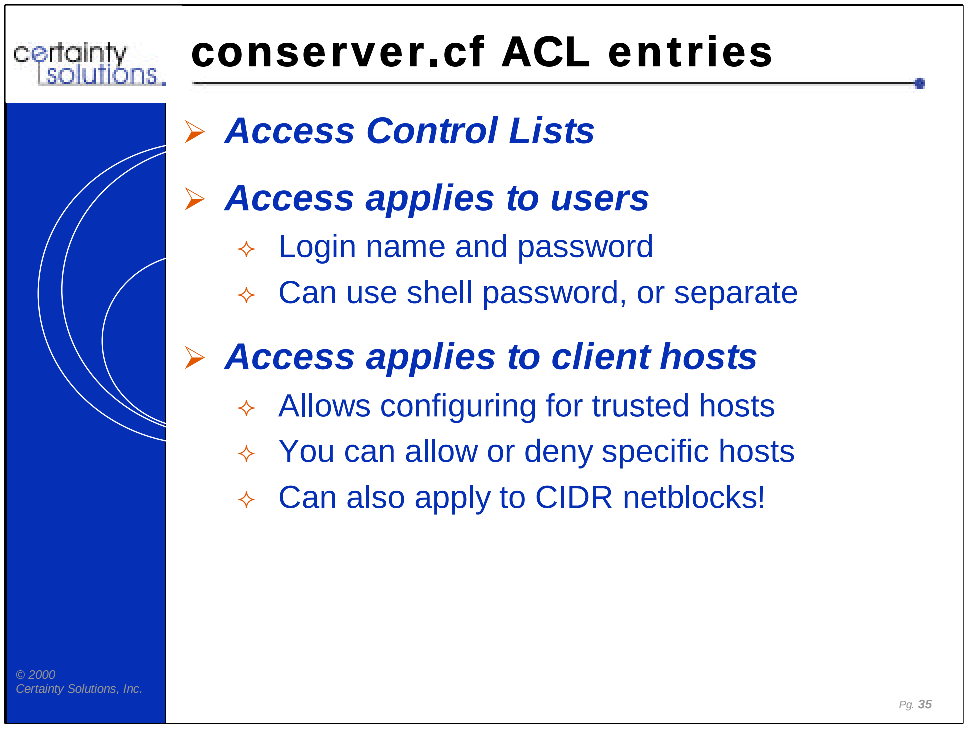
name : !host[@host] : port : logfile : mark-interval[m|h|d]

For connections on a built-in serial port, the syntax looks like this:

name : tty[@host] : baud[parity] : logfile : mark-interval[m|h|d]

For invoking a shell program, the entry looks like this:

name : |command : : logfile : mark-interval[m|h|d]



conserver.cf ACL entries

- *Access Control Lists*
- *Access applies to users*
  - Login name and password
  - Can use shell password, or separate
- *Access applies to client hosts*
  - Allows configuring for trusted hosts
  - You can allow or deny specific hosts
  - Can also apply to CIDR netblocks!

Pg. 35

The second half of a `conserver.cf` file (before the "%%" line) contains access control list (ACL) entries. The basic syntax of the command is;

```
{trusted|allowed|rejected} : host ip cidr
```

The ACL entries are two fields; the keyword "trusted", "allowed", or "rejected" separated by a colon from the hostname, ip address, or CIDR address block.

Access will be granted (or rejected) on a first-match basis, so the order of the entries is important.

The `trusted` tag means that no authentication is needed.

The `rejected` tag means that no connections are allowed from that host, address, or network.

The `allowed` tag means that connections are allowed, but still require username and password authentication.

For the second half of the list, you can use either of the following;

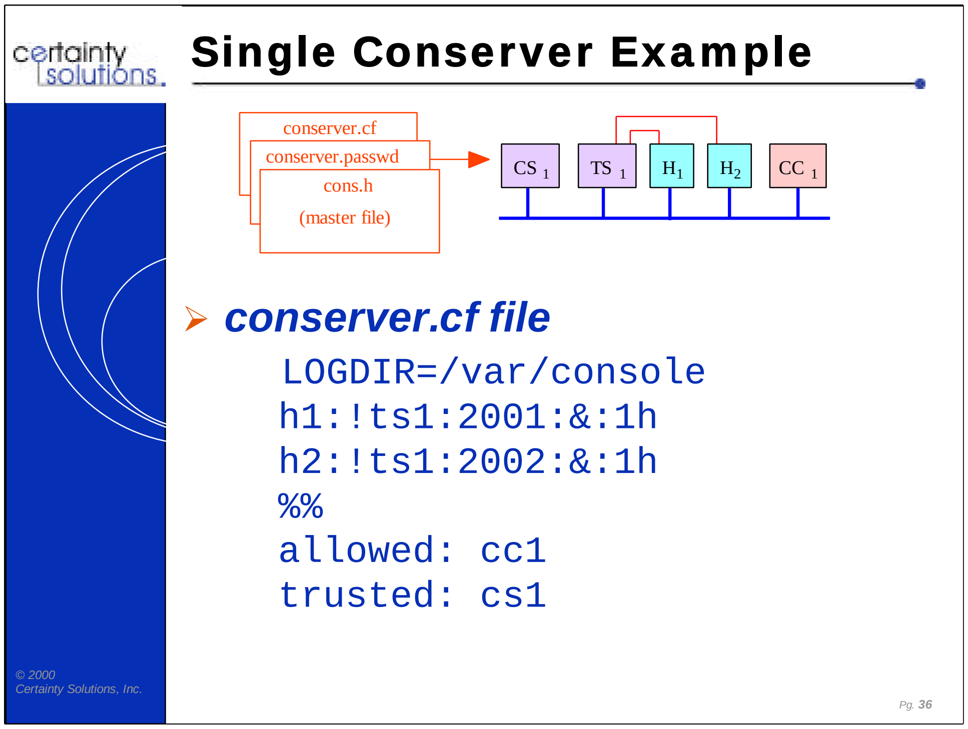
A fully-qualified hostname (example: `merlot.conserver.com`)

A TCP/IP host address (example: `192.168.33.42`)

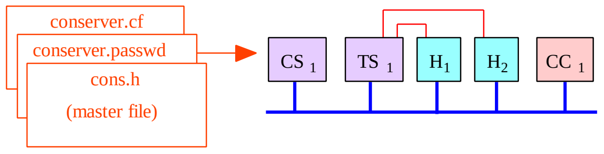
A CIDR network block (example: `172.17.56.0/22`)

You can actually have a list of host names (or host addresses, or CIDR blocks) on the same line, separated by spaces, if you prefer. You can decide which way is easier to read and understand.

Example: `allowed : 192.168.9.0/24 192.168.42.0/23`



Single Conserver Example



➢ ***conserver.cf file***

```
LOGDIR=/var/console
h1:!ts1:2001:&:1h
h2:!ts1:2002:&:1h
%%
allowed: cc1
trusted: cs1
```

© 2000
Certainty Solutions, Inc.

Pg. 36

The configuration file is very small, but it's specifying a lot of information.

First of all, the default location for log files is being set to /var/console.

Next, two consoles are being defined: h1 and h2. Each of the consoles are hosted on the terminal server named ts1. h1 is associated with TCP port 2001 and h2 is associated with TCP port 2002. Each console log file will be time-stamped hourly.

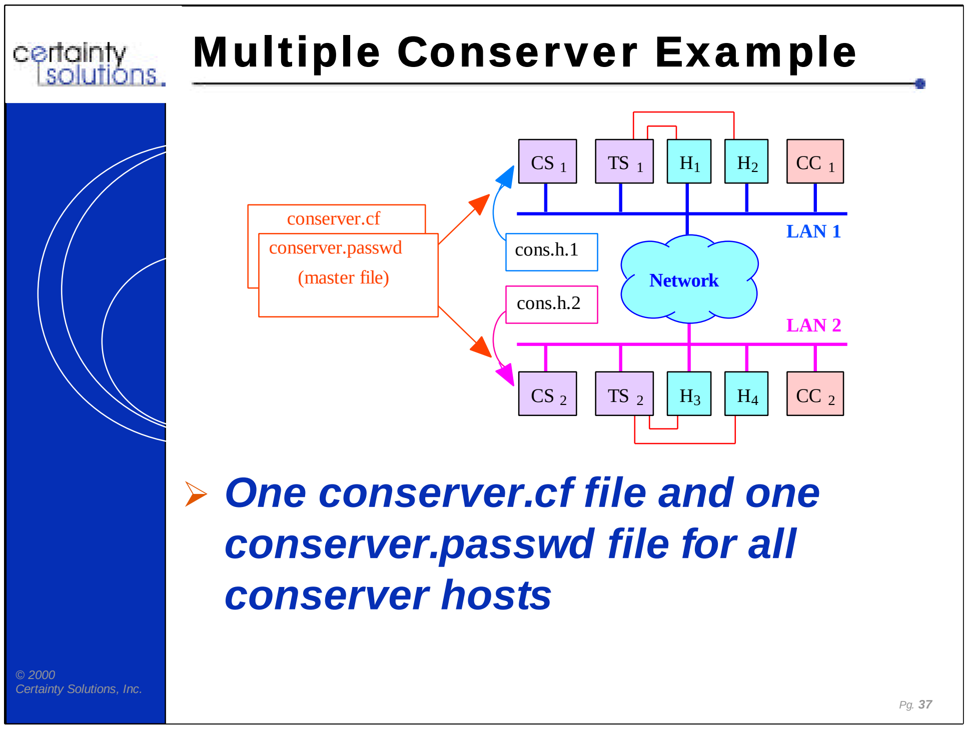
The log file itself is specified as "&". This special character is replaced with the console name ('h1' or 'h2' respectively). Since that is a relative path, the `LOGDIR` prefix is applied and you end up with "/var/console/h1" as the full log file name.

Then we see the "%%" line. This separates the console names from the ACLs.

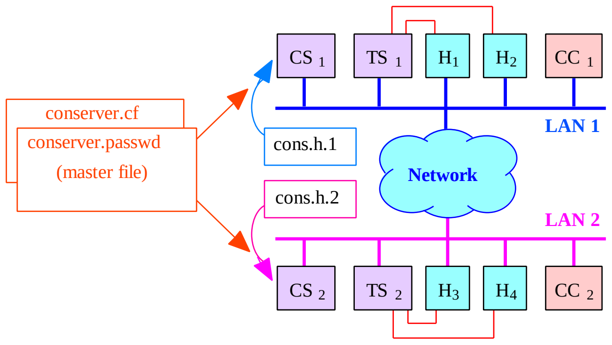
The Access Control List is stating that client connections from cc1 are allowed (must succeed with password authentication) and connections from cs1 are trusted (no authentication is necessary).

Additionally, you can use the "#" (pound sign) to add comments to your configuration files, to help keep things clear. (Consider adding a line before each terminal servers entries, giving the hostname, ip address, and physical location of the device. You can also consider noting what type of hardware it is.)

```
# ts-1 (c3620)
# Data center 1, bldg. B, top of rack 3
```



Multiple Conserver Example



- One conserver.cf file and one conserver.passwd file for all conserver hosts

© 2000 Certainty Solutions, Inc.

Pg. 37

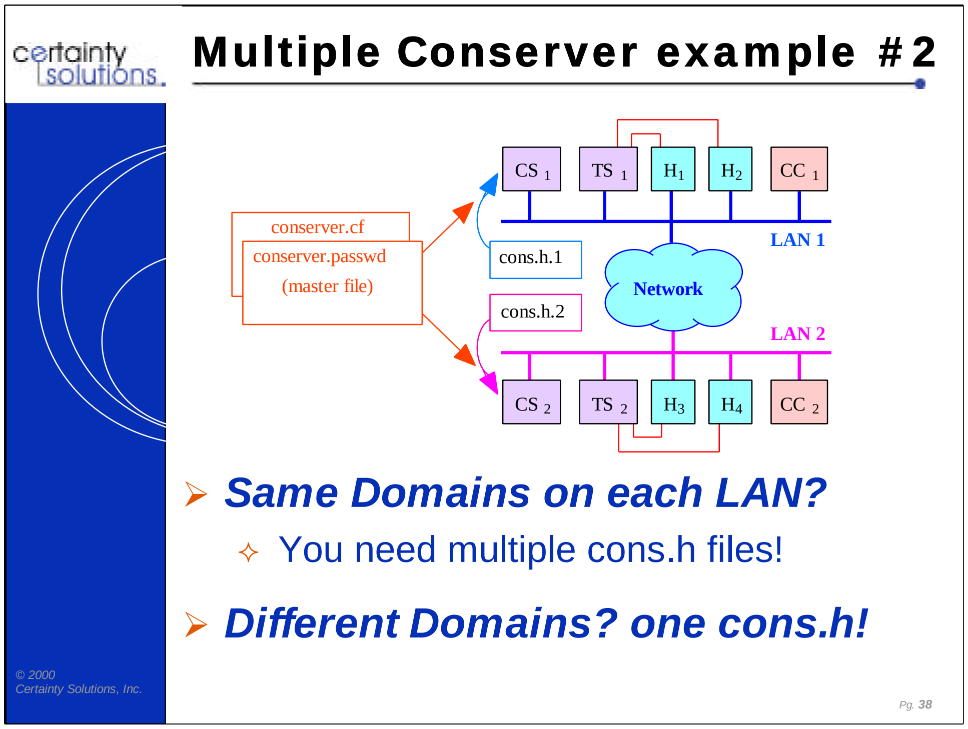
This example is a basic representation of a multi-conserver setup. Here is a sample configuration file for the diagram:

```
LOGDIR=/var/console
h1:!ts1@cs1:2001:&:1h
h2:!ts1@cs1:2002:&:1h
h3:!ts2@cs2:2001:&:1h
h4:!ts2@cs2:2002:&:1h
%%
allowed: cc1 cs1
allowed: cc2 cs2
```

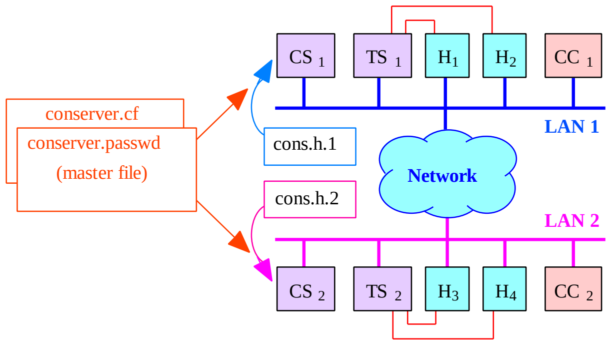
Notice the "`@cs*`" tags used in the second field. This specifies which console server host should manage which console. Any time you have multiple console servers, every line in the configuration file should contain a "`@cs`" tag.

You should create one `conserver.cf` file for use with all of your distributed conserver hosts, putting the @ tag in every host entry.

The result of this file is that cs1 will make two TCP connections to ts1, and cs2 will make two TCP connections to ts2. If either of them is contacted by a client and asked to be connected to a console they aren't managing, they will redirect the client to the appropriate console server. If you did not distribute one master configuration file using this syntax, the various servers would not know about the others and the redirection would not occur.



Multiple Conserver example #2



- ***Same Domains on each LAN?***
  - You need multiple cons.h files!
- ***Different Domains? one cons.h!***

© 2000
Certainty Solutions, Inc.

Pg. 38

This part can sometimes be tricky. We're hoping that this page will make the problem a bit easier to understand.

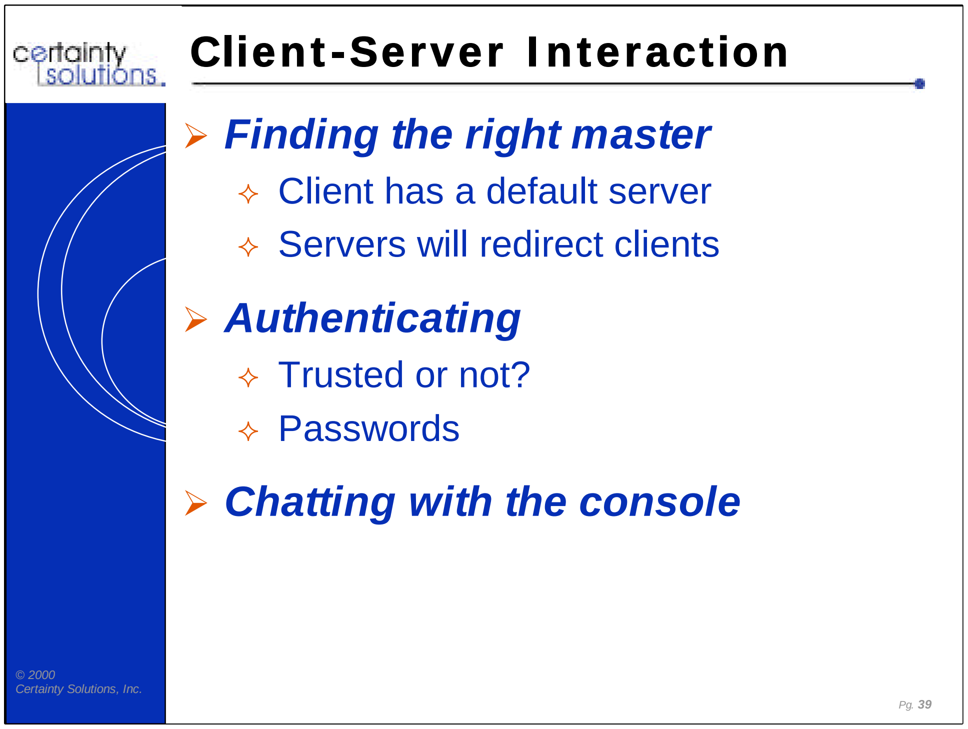
The "`HOST`" variable in the `cons.h` file is what tells the binary what the name of the default conserver host will be. That is, when you are on a client, and you type "`console h1`", the client will try to open a connection with the hostname that you specified in cons.h prior to making the files. (The make process also creates the "console" client binary application…)

In a distributed environment, you want the clients to try to contact the conserver that is 'local' to them, rather than relying on another server across the network.

If you have one big, flat namespace (one domain), then you can only have one host named "console"…so you would need separate cons.h files for each LAN segment. (Clients on LAN 1 would want to default to using the CS1 host. Clients on LAN 2 would want to connect to CS2).

When you have different domains (or sub-domains) on each LAN, you can just use the relative name "console" in each domain, assigned to the conserver hosts. Then you can use one cons.h, and create one client binary (which defaults to "console"), and you push that client binary to all the clients.

When you implement distributed conserver hosts, you don't want to have to ask a remote conserver to redirect you back to your conserver host so you can talk to your local hosts connected to your local terminal servers. Getting the hostnames and the `HOST` variable in `cons.h` are the key to success!



Client-Server Interaction

- ***Finding the right master***
  - Client has a default server
  - Servers will redirect clients
- ***Authenticating***
  - Trusted or not?
  - Passwords
- ***Chatting with the console***

© 2000
Certainty Solutions, Inc.

Pg. 39

The first step a client takes when connecting to a console is to find the right server. The client is compiled with a default server name (defined by the `HOST` variable in `cons.h` file) and it will default to that server, unless you override it with the –M flag.

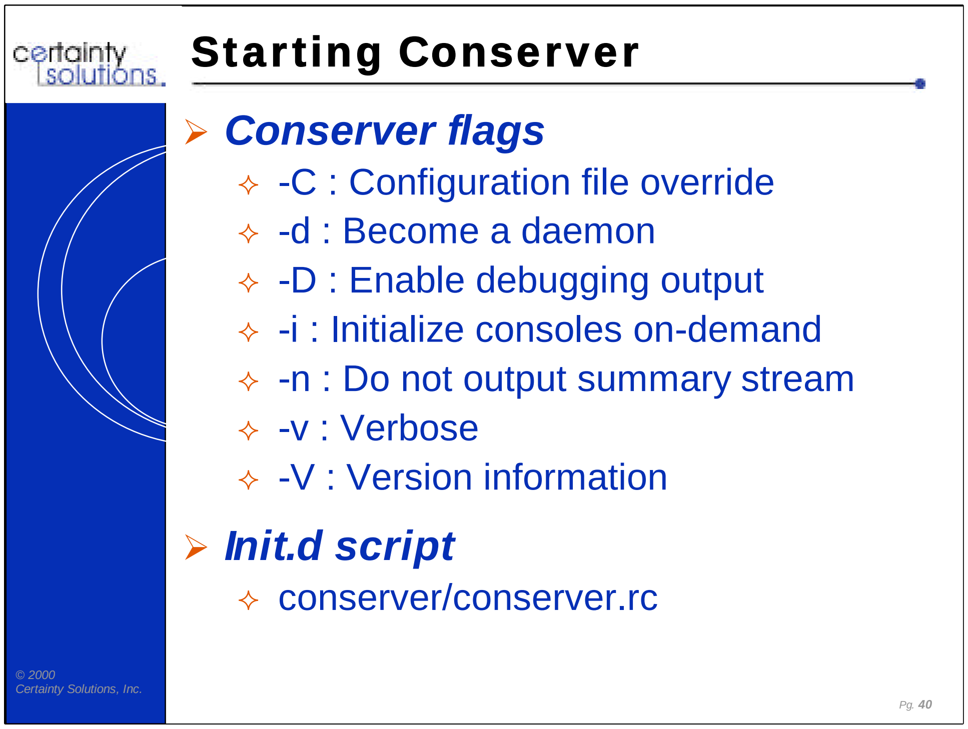
Example: `console -M cs3`

This can be useful if you're testing a new conserver host, or moving your conserver application to another machine during an emergency and don't have time to rebuild the client, or you're default console server is down in a distributed setup and you need access to a console managed by another server.

If you have a distributed console server setup, you can connect to any server and it will redirect the client to the server managing the console requested.

Once the proper server is found, access is either granted immediately (to "trusted" hosts), granted after authentication (for "allowed" hosts), or immediately rejected (for "rejected" hosts), based on the ACL entries in the `conserver.h` file.

If access is granted (immediately or after a password check), the connection is made to the requested console. Anything typed on the console will be delivered to the attached console, and the client will begin seeing the console output. (The traffic coming from the attached console is logged, as well as being passed to any connected conserver clients.



Starting Conserver

- **Conserver flags**
  - -C : Configuration file override
  - -d : Become a daemon
  - -D : Enable debugging output
  - -i : Initialize consoles on-demand
  - -n : Do not output summary stream
  - -v : Verbose
  - -V : Version information
- **Init.d script**
  - conserver/conserver.rc

Pg. 40

Conserver has many flags for modifying it's default behavior. Some of these flags are crucial, others are not. For normal operations you want to use both the –d and –n flags. This will put conserver in the background, disable the "summary stream" function, and produce reasonable output. The summary stream that is produced (without using the –n option) is all the activity on consoles without a user in read-write mode. So, if you have 5 consoles producing output and no one is connected to those consoles, conserver will echo those messages to stdout. Ideally, all this output is already being logged to individual log files, so having it here as well is relatively useless.

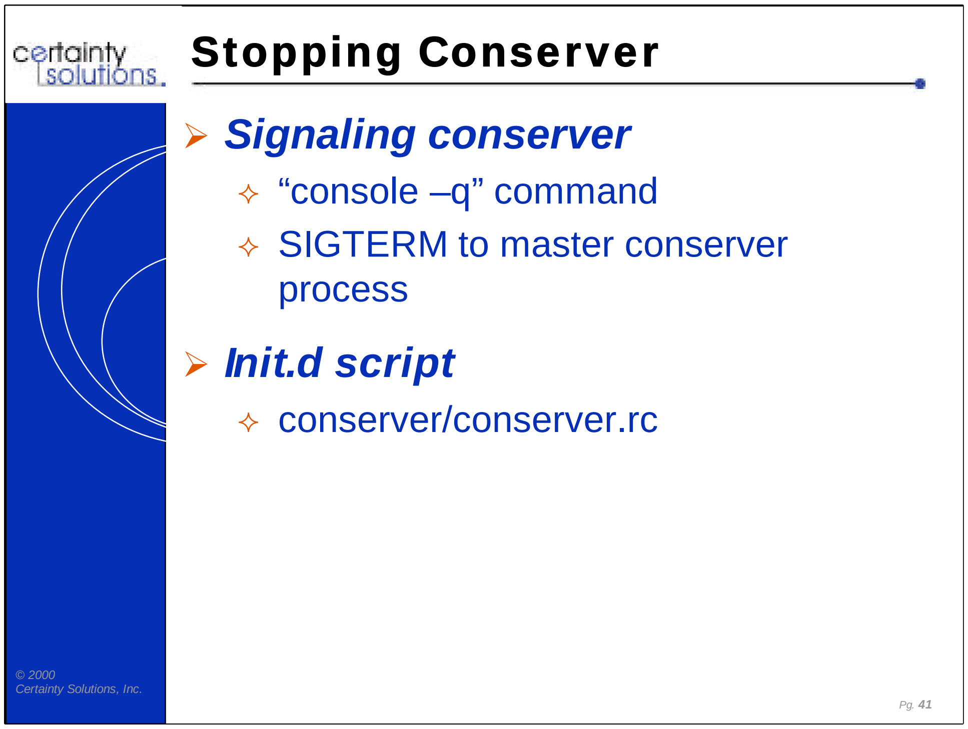
The –D and –v options are useful if you want to troubleshoot startup problems or other aspects of the server.

The –C flag allows you to override the location of the conserver.cf file. There is no flag to override the conserver.passwd file.

The –i flag enables the "initialize-on-demand" mode, which makes conserver connect to console ports only when a client connects, and disconnect from the console ports when all users have disconnected.

The –V flag provides version information as well as conserver/cons.h settings.

The conserver/conserver.rc startup script is a working startup/shutdown script written for a Solaris environment. Use it as a guide for generating your own script or /etc/rc.local entry.



Stopping Conserver

- **Signaling conserver**
  - "console –q" command
  - SIGTERM to master conserver process
- **Init.d script**
  - conserver/conserver.rc

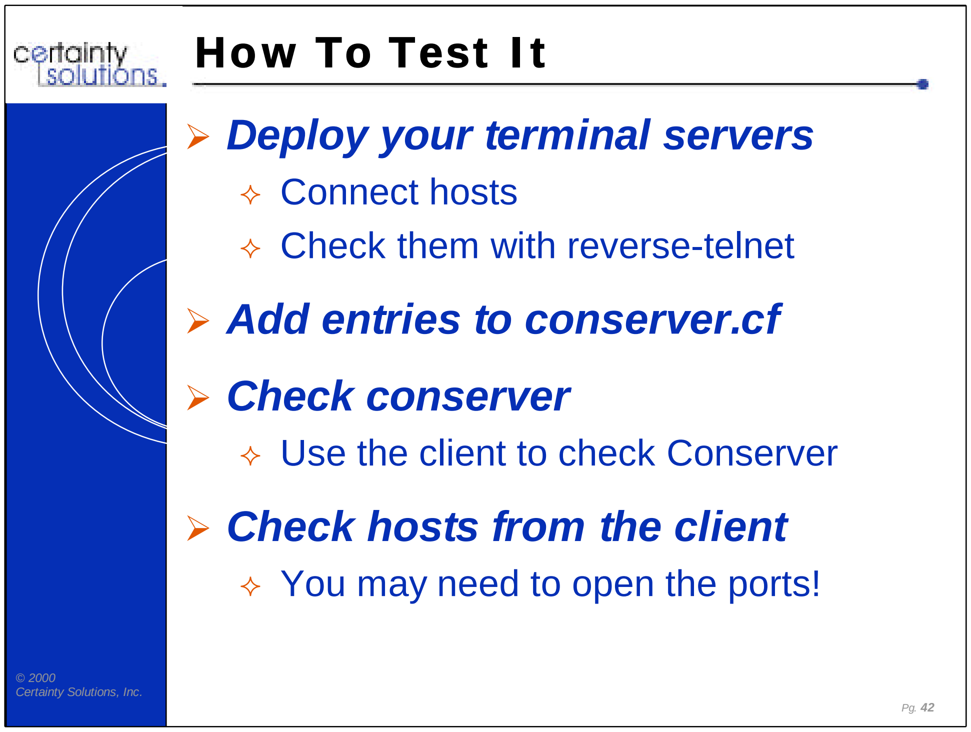
© 2000 Certainty Solutions, Inc.

Pg. 41

There are two ways to gracefully signal conserver to shut itself down. The first method is using the console client with the –q flag. If you're asked for a password when you use this method, you must provide the root password of the console server host. If no authentication is done (you're coming from a "trusted" host), conserver will shut down without a password.

The second method is to send a SIGTERM to the master conserver process. When it receives this signal, it signals all the child processes to exit and then exits itself.

The distribution comes with an init.d script in conserver/conserver.rc. This script (modeled for a Solaris environment) uses the SIGTERM method of shutting down the software.



How To Test It

- *Deploy your terminal servers*
  - Connect hosts
  - Check them with reverse-telnet
- *Add entries to conserver.cf*
- *Check conserver*
  - Use the client to check Conserver
- *Check hosts from the client*
  - You may need to open the ports!

© 2000
Certainty Solutions, Inc.

Pg. 42

Terminal server setup was covered briefly in the Basic Serial Tutorial.

After basic serial setup, try connecting to the consoles you have attached to terminal servers by using the reverse-telnet capability of the terminal server.

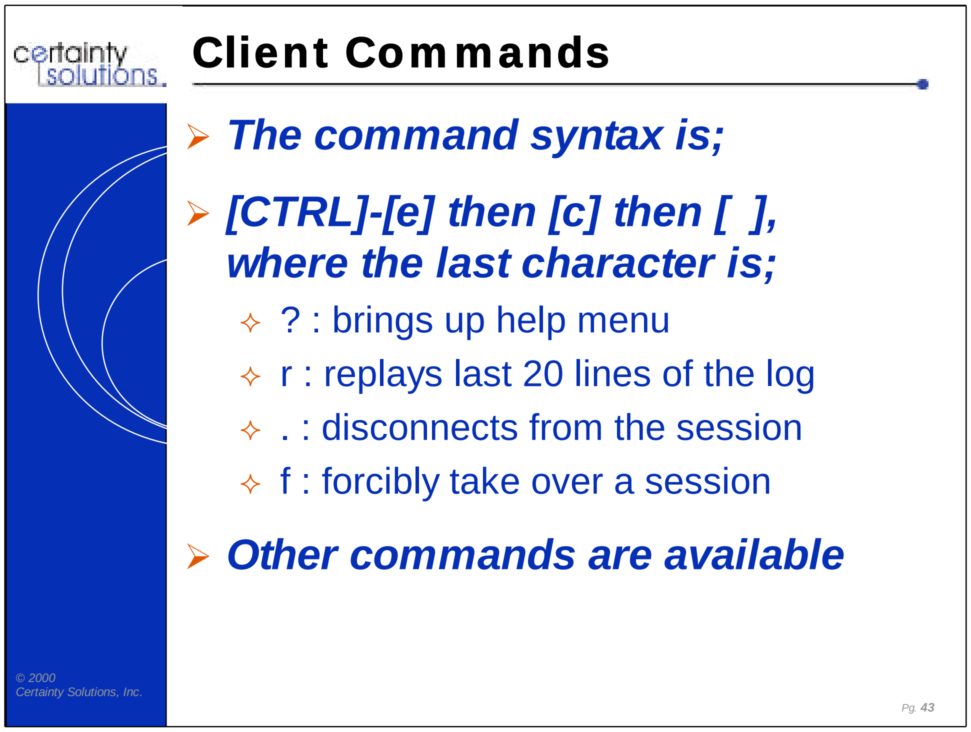
You should test consoles connected to local serial ports via “tip” or other tools.

Once you’ve verified connectivity, populate the conserver.cf file with the appropriate entries.

For consoles on devices that have DNS or NIS hostnames, we recommend that you use the same hostname for the name entries in `conserver.cf`. This usually makes it easier to remember the name to use to reach those consoles.

When running conserver, verify that connections are made to the various ports and/or terminal servers, and that no errors are being produced during startup. Typing ‘`conserver –V`’ on the conserver host will produce compile-time options and information that conserver is looking for. When debugging the client, most errors are due to invalid hostnames or an undefined service port. Using ‘`conserver –V`’ on the client will show you what the client application is looking for.

Here’s a tip to save you some typing: Conserver uses a ‘least-ambiguous’ rule for name entries. That is, you only need to type as much of a name such that conserver is sure that it knows which name you mean; this may only be a few characters for some names.



Client Commands

- *The command syntax is;*
- *[CTRL]-[e] then [c] then [ ], where the last character is;*
  - ? : brings up help menu
  - r : replays last 20 lines of the log
  - . : disconnects from the session
  - f : forcibly take over a session
- *Other commands are available*

All interaction is done from the client. Most interaction is done when connected to a console.

When you are connected, the default escape sequence is “CTRL-e” then “c”.

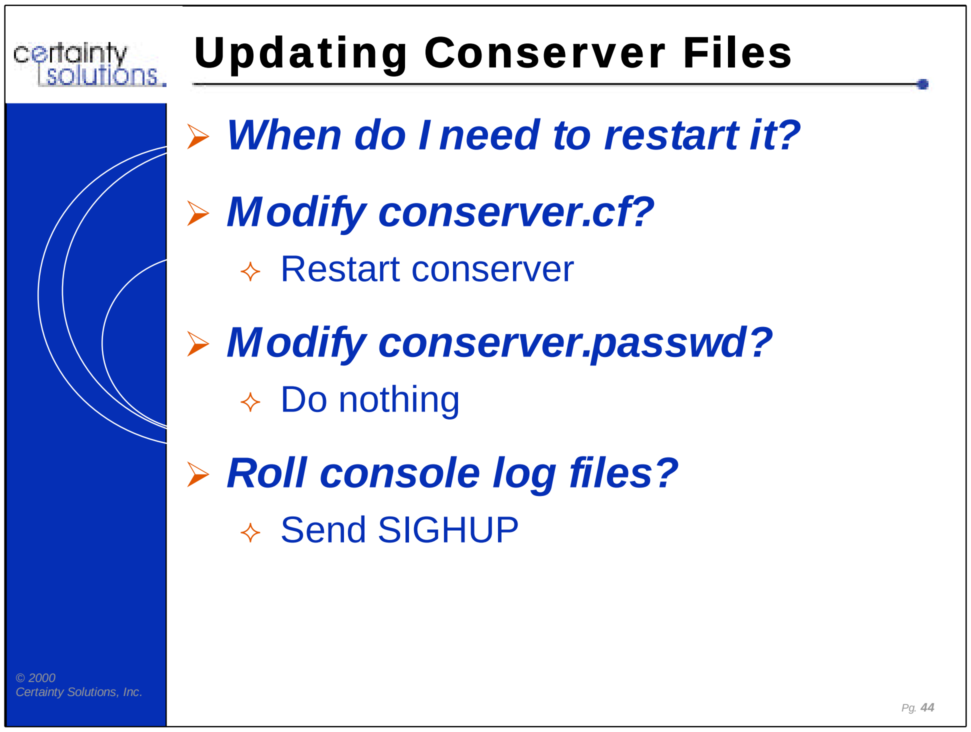
After the escape sequence, you use one or more characters that implement various functions.

A “`?`” will bring display a list of all available functions. The most popular are “`.`” to disconnect, “`z`” to suspend, “`f`” to bump someone into “spy” mode and force you to be read-write, “`r`” to replay the last 20 lines, and “`l1`” to produce a BREAK signal.

You can use “w” to see who else is also attached to the console that you are on (handy to see who is spying on you ;-).

You can even send any character you like to the host using a 3-digit octal code (useful for sending control characters and other strange bytes).

Depending on what mode you’re in, you’ll see different lists of functions. When you are in Spy mode (read-only), you have no write capabilities, so some of the normal functions will be missing.



Updating Conserver Files

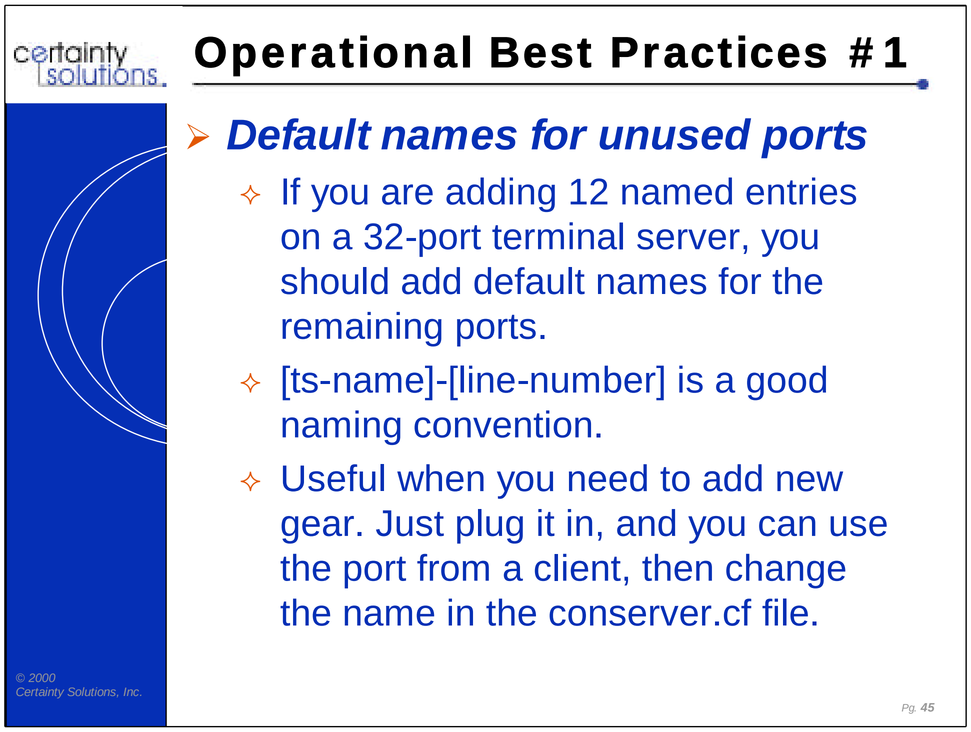
- ***When do I need to restart it?***
- ***Modify conserver.cf?***
  - Restart conserver
- ***Modify conserver.passwd?***
  - Do nothing
- ***Roll console log files?***
  - Send SIGHUP

Restarting the conserver host means a short bit of downtime. The time will vary, but the more console names you have, the longer a restart will take, as the processes are spun up, and each reverse-telnet session is opened, and the associated logging is restarted.

Updating the conserver.cf file will always require a full restart of conserver. The underlying architecture of the conserver code does not lend itself to dynamic reconfiguration, so you must restart the conserver process, in order to read in the changes to the file.

Any changes to the conserver.passwd file will be used immediately. The file is opened and read every time someone authenticates with conserver. If you have a large password file, this could be an issue...

Sending the conserver a SIGHUP will cause it to close all console connections and all log files, then reopen them as if it was restarting. This is useful if you roll your console logs or have some need for a “semi-restart”.


Operational Best Practices #1

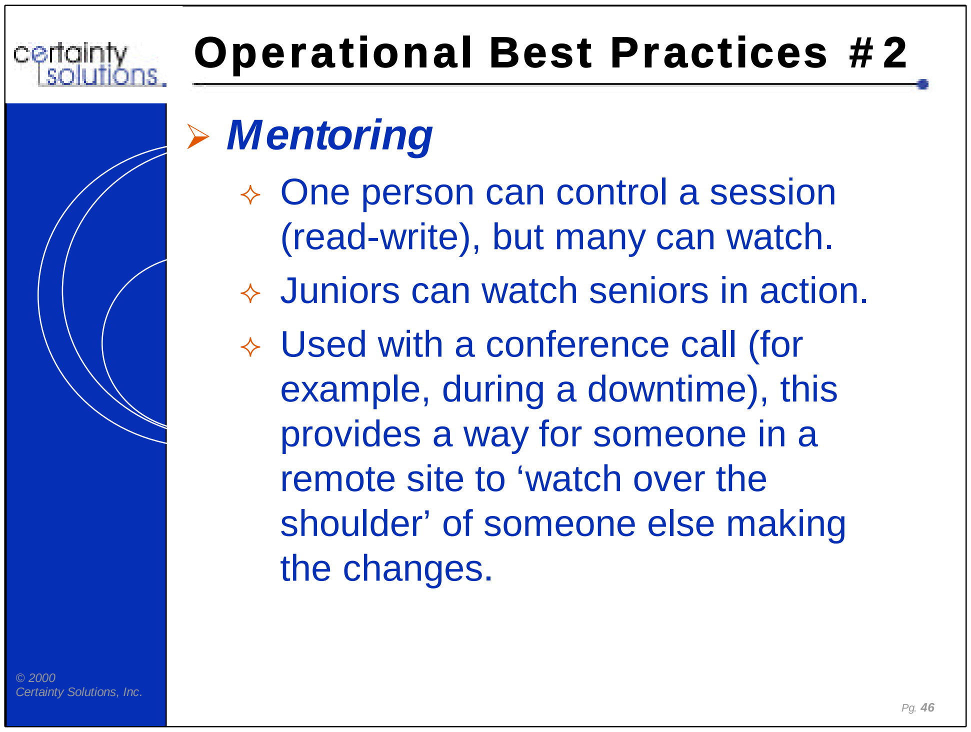
- ***Default names for unused ports***
  - If you are adding 12 named entries on a 32-port terminal server, you should add default names for the remaining ports.
  - [ts-name]-[line-number] is a good naming convention.
  - Useful when you need to add new gear. Just plug it in, and you can use the port from a client, then change the name in the conserver.cf file.

© 2000
Certainty Solutions, Inc.

Pg. 45

Here's a sample config file for a 16-port terminal server with 7 port in use;

```
LOGDIR=/var/consoles
#
# ts-3 (c3620, with NM-16A module)
# broadband video head-end
he-ubr:!ts-3@cs-1:2001:&:
he-cmon:!ts-3@cs-1:2002:&:
# Scientific Atlanta Consoles speed set to 38,400
sa-qam-1:!ts-3@cs-1:2003:&:1h
sa-qam-2:!ts-3@cs-1:2004:&:1h
sa-qpskm:!ts-3@cs-1:2005:&:1h
sa-qpskd:!ts-3@cs-1:2006:&:1h
3com-1:!ts-3@cs-1:2007:&:
# Ports prepared for future use
#ts-3-8:!ts-3@cs-1:2008:&:
#ts-3-9:!ts-3@cs-1:2009:&:
```



Operational Best Practices #2

- ***Mentoring***
  - One person can control a session (read-write), but many can watch.
  - Juniors can watch seniors in action.
  - Used with a conference call (for example, during a downtime), this provides a way for someone in a remote site to ‘watch over the shoulder’ of someone else making the changes.

© 2000
Certainty Solutions, Inc.

Pg. 46

This is an important feature. The simple concept is that only one person can control a port (that is, have read-write access) at one time, but remember these two other things;

1) many folks can be watching in “spy” mode at the same time.

2) control can be passed between all of the session attached.

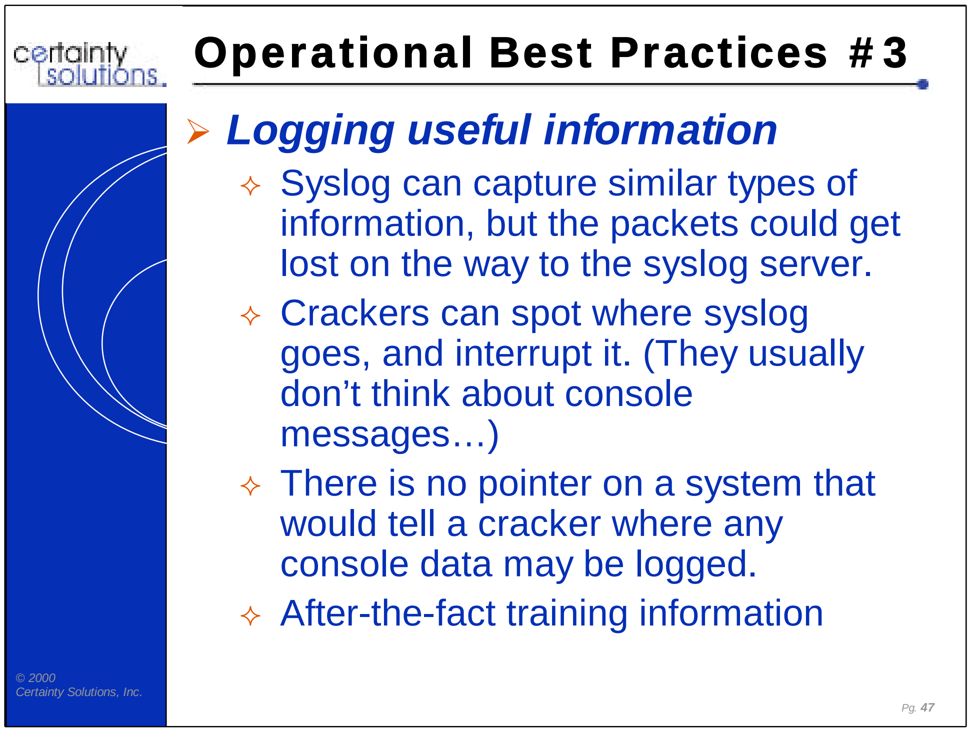
Scenario:

The tech in a lab has shown some promise, so you have granted read-only access to the local router, to reduce the number of calls about the network being down. Now, when there is a problem, the tech can investigate from the router, and perform limited testing.

One evening, the tech finds a problem, probably a routing entry, and you get the call. From home, you connect back to the network, and connect to the console.

Even though the tech is connected (in read-write mode), you can still attach in spy mode, and replay the last 20 or 60 lines of the log to see what has been viewed lately.

You’re still on the phone, and the tech can show you what they have done, while you watch as well. When you agree that a routing entry needs to be modified, you take control of the session (while the tech watches), and you enter the admin password, and make the change. You exit command mode, and let the tech take the session back, and see that it all works now!



Operational Best Practices #3

- ***Logging useful information***
  - Syslog can capture similar types of information, but the packets could get lost on the way to the syslog server.
  - Crackers can spot where syslog goes, and interrupt it. (They usually don't think about console messages…)
  - There is no pointer on a system that would tell a cracker where any console data may be logged.
  - After-the-fact training information

© 2000
Certainty Solutions, Inc.

Pg. 47

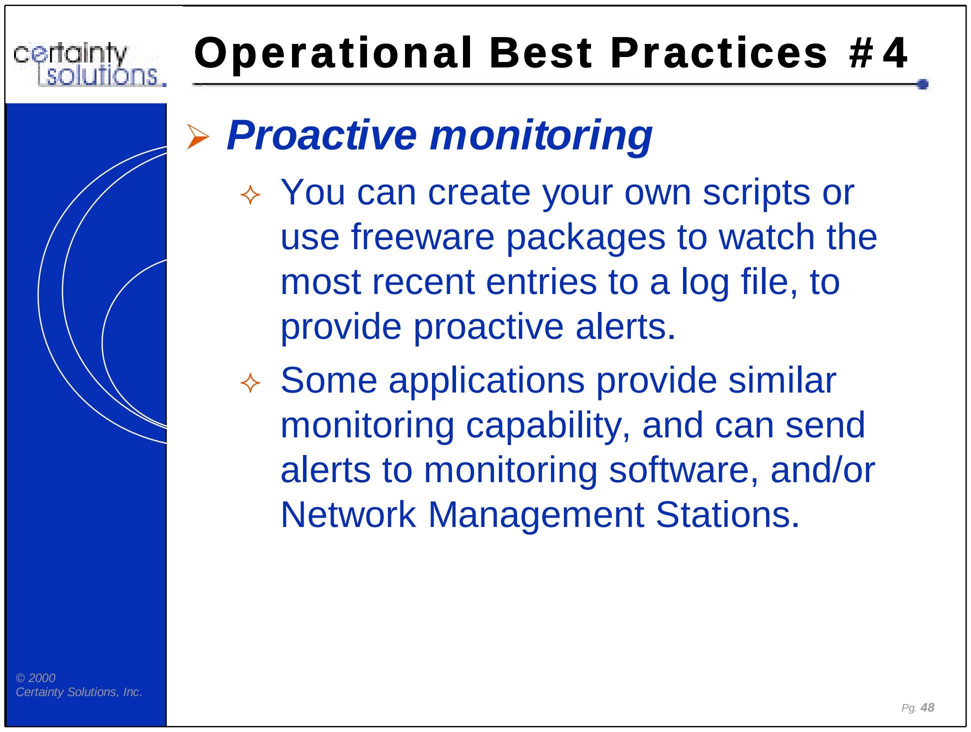
Logging is a valuable tool, as long as you don't get overwhelmed by the amount of data that is produced.

You can tune your logging, but you always walk that fine line between data overload, and making sure that you capture the important clues you need to monitor the systems, or detect failures (or attacks).

It's important to coordinate the clocks on your hosts, so that the times are useful when comparing events between different devices. (Can you correlate host failures with network events? Can failure on one host be traced to an event on another host?)

In many shops, the console is not connected to anything, or it is connected to a device that has little (or no) scrollback ability. Because of this, crackers tend to ignore that it may be attached to a write-only device (such as a printer). If they do think about it, they may try to deplete it by sending a tone of data to the console, hoping to use up the paper or ribbon.

With a conserver attached, all of that activity is logged, and `grep` will make it easier to sift through the extra noise to find your needle in that haystack!



Operational Best Practices #4

- ***Proactive monitoring***
  - You can create your own scripts or use freeware packages to watch the most recent entries to a log file, to provide proactive alerts.
  - Some applications provide similar monitoring capability, and can send alerts to monitoring software, and/or Network Management Stations.

© 2000
Certainty Solutions, Inc.

Pg. 48

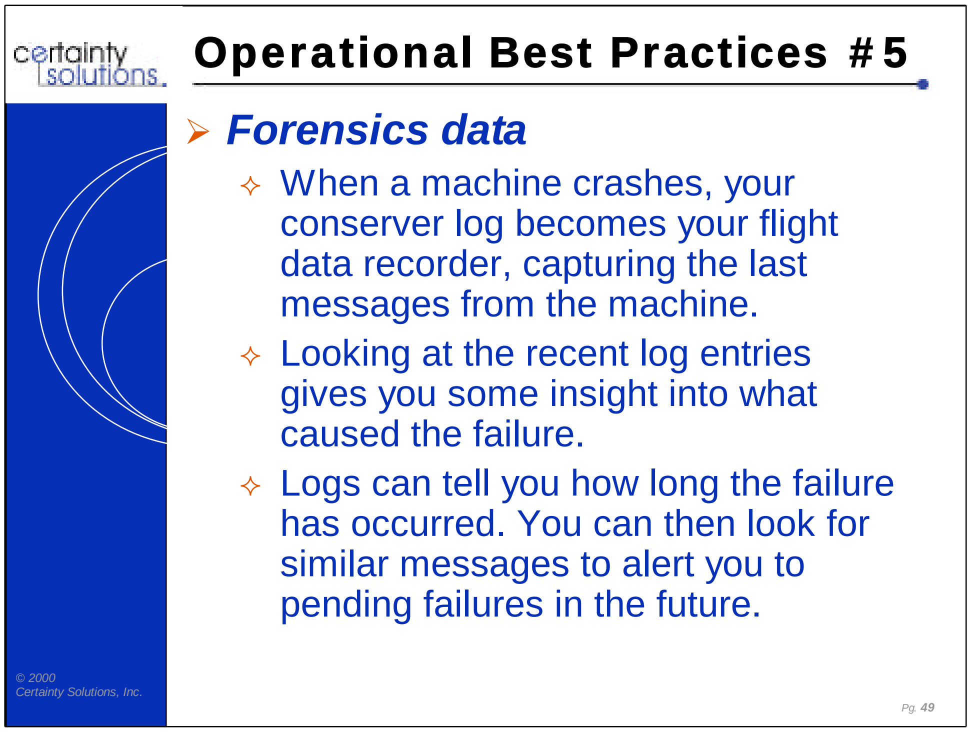
There are many packages available, both free and for sale, that will monitor log files and trigger alerts when things exceed pre-defined boundaries. Utilizing the log files that conserver provides can expand your view into health of your environment.

Netcool can be used to monitor the traffic in some of your logs, and incorporate that data into their system and network monitoring tools.

Scripts can be used to sift through the logs, searching for data.

You could also use scripts to watch for signs of trouble, by monitoring the log files themselves for unusual amounts of growth over a period of time. While not a perfect answer to monitoring, when a log file increases dramatically in size overnight, chance are that a sysadmin should be looking at the recent logs, looking for bad RAM, a failing disk, or just a lack of space in some partition.

Of course, it could also be important to note when a log file hasn't had any entries over a period of time. While normal for some devices, silence can sometimes be a bad thing for some systems or devices. Script monitoring can check to see when the last log entry was made. (For an active host, it could mean that a cracker has turned off logging to the console…)



Operational Best Practices #5

- **_Forensics data_**
  - When a machine crashes, your conserver log becomes your flight data recorder, capturing the last messages from the machine.
  - Looking at the recent log entries gives you some insight into what caused the failure.
  - Logs can tell you how long the failure has occurred. You can then look for similar messages to alert you to pending failures in the future.

© 2000
Certainty Solutions, Inc.

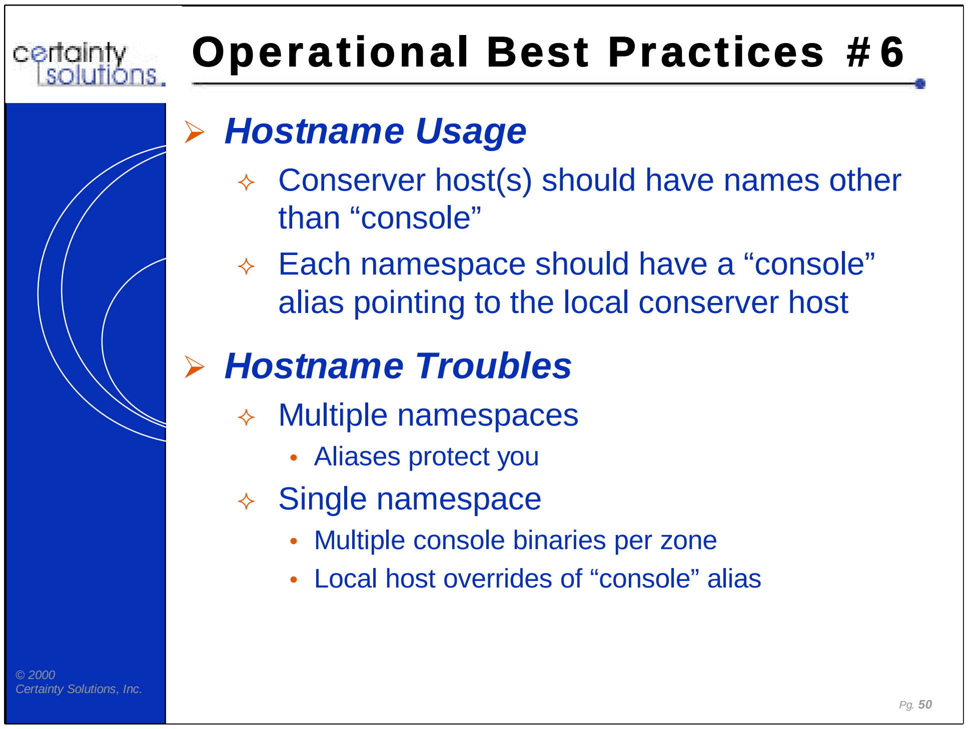
Pg. 49

Sometimes, devices fail. When you get the call that a device isn't responding, you may go to the console to check for a pulse. In the worst cases, you get on the console, hit the carriage return a couple times, and get nothing. Control characters don't help. "He's dead, Jim."

You could try to restart the machine immediately, but you still might not find out what killed it. (What do you tell the Director of Engineering when you are asked "What happened?" and "Well, how do you know it won't happen again?"

With conserver, after you tried hitting the return key, you could have played back the last 20 or 60 lines of the log. Think of this as the flight recorder, capturing the last statements of the pilot of a doomed flight. The log could tell you that this was a RAM problem, a full disk, or a faulty CPU. Each of these cases would require you to take different steps to get the machine back up to normal, and reduce the chances of another failure.

After the repair, you can use `grep`, or other tools, to search the logs for previous problems on this particular device, to see how many times it happened before, and over what period of time. You could then check the logs for similar hardware, to see if you find signs of a similar failure starting to show up on those devices!



Operational Best Practices #6

- **_Hostname Usage_**
  - Conserver host(s) should have names other than "console"
  - Each namespace should have a "console" alias pointing to the local conserver host
- **_Hostname Troubles_**
  - Multiple namespaces
    - Aliases protect you
  - Single namespace
    - Multiple console binaries per zone
    - Local host overrides of "console" alias

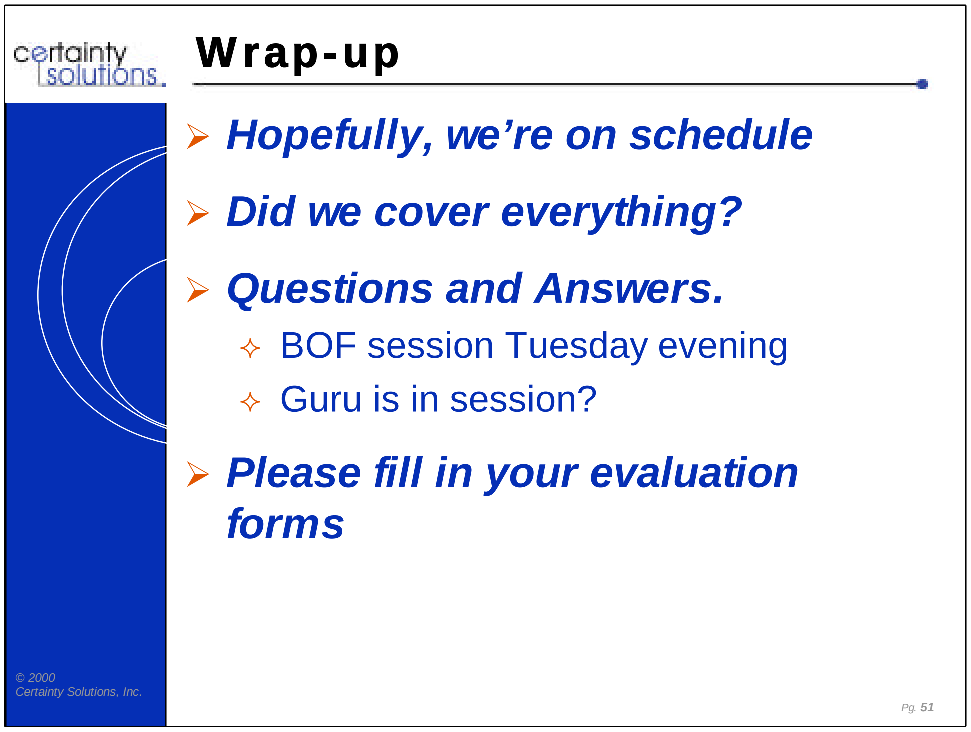
© 2000
Certainty Solutions, Inc.

Pg. 50

There are a few naming conventions that you can use to ease the operation of conserver. The following assumes the default installation of "console" being the hostname used by the clients to connect to the server.

First, none of your console server hosts should have a hostname of "console". "console" should always be an alias for the true name. This allows you to start with a single console server and add others in the future. In a multi-namespace environment, the alias "console" should exist anywhere a client might be located. For example, there should be a "console.eng.crtnty.com" and a "console.corp.crtnty.com". They both should point to their closest console server.

Where things can get tricky is when you have a single namespace and multiple console servers. You have two options for having clients find their local server: multiple console binaries (each with a different default hostname) and local alias overrides (local /etc/hosts entries for "console" can usually be made to override global definitions). Either solution will work, but you must decide on which is more difficult to maintain across your enterprise.


certainty solutions

Wrap-up

- Hopefully, we're on schedule
- Did we cover everything?
- Questions and Answers.
  - BOF session Tuesday evening
  - Guru is in session?
- Please fill in your evaluation forms

© 2000
Certainty Solutions, Inc.

Pg. 51

Hopefully we've done what we came to do, which is to teach you enough about conserver to be able to deploy it at your site. And we hope that we were able to include some of the questions and concerns that the students had coming into the class.

We'll have a Birds of a Feather session at LISA 2000, probably on Tuesday evening, and you are all invited to attend. (Check the BOF board for time and location information.)

We may be on the Guru is In roster. If you are interested, please check it out.

While the LISA staff are interested in getting an evaluation form from all of you, we'd also appreciate you filling in the class evaluation forms, since we'll get some feedback from the LISA folks about how well we did what we came to do.

Thanks for attending.

David K. Z. Harris   Bryan Stansell

Certainty Solutions, Inc.

California - Virginia - Toronto