BLACK BOX
NETWORK SERVICES

1101 and 1102 Secure Device Servers

Securely monitor, access, and control the computers, networking devices, telecommunications equipment, and power supplies in your data room or communications centers.

Manage your servers:
• Locally across your management LAN or through the local serial console port.
• Remotely across the Internet or private network.

Customer Support Information

Order toll-free in the U.S.: Call 877-877-BBOX (outside U.S. call 724-746-5500) •
FREE technical support 24 hours a day, 7 days a week: Call 724-746-5500 or fax 724-746-0746 •
Mailing address: Black Box Corporation, 1000 Park Drive, Lawrence, PA 15055-1018 •
Web site: www.blackbox.com • E-mail: info@blackbox.com

1101 and 1102 Secure Device Servers

Federal Communications Commission and Industry Canada Radio Frequency Interference Statements

This equipment generates, uses, and can radiate radio-frequency energy, and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio communication. It has been tested and found to comply with the limits for a Class A computing device in accordance with the specifications in Subpart J of Part 15 of FCC rules, which are designed to provide reasonable protection against such interference when the equipment is operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference, in which case the user at his own expense will be required to take whatever measures may be necessary to correct the interference. Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

This digital apparatus does not exceed the Class A limits for radio noise emission from digital apparatus set out in the Radio Interference Regulation of Industry Canada.

Le présent appareil numérique n'émet pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de la classe A prescrites dans le Règlement sur le brouillage radioélectrique publié par Industrie Canada.

724-746-5500 | blackbox.com

Normas Oficiales Mexicanas (NOM) Electrical Safety Statement
INSTRUCCIONES DE SEGURIDAD

1. Todas las instrucciones de seguridad y operación deberán ser leídas antes de que el aparato eléctrico sea operado.
2. Las instrucciones de seguridad y operación deberán ser guardadas para referencia futura.
3. Todas las advertencias en el aparato eléctrico y en sus instrucciones de operación deben ser respetadas.
4. Todas las instrucciones de operación y uso deben ser seguidas.
5. El aparato eléctrico no deberá ser usado cerca del agua—por ejemplo, cerca de la tina de baño, lavabo, sótano mojado o cerca de una alberca, etc..
6. El aparato eléctrico debe ser usado únicamente con carritos o pedestales que sean recomendados por el fabricante.
7. El aparato eléctrico debe ser montado a la pared o al techo sólo como sea recomendado por el fabricante.
8. Servicio—El usuario no debe intentar dar servicio al equipo eléctrico más allá a lo descrito en las instrucciones de operación. Todo otro servicio deberá ser referido a personal de servicio calificado.
9. El aparato eléctrico debe ser situado de tal manera que su posición no interfiera su uso. La colocación del aparato eléctrico sobre una cama, sofá, alfombra o superficie similar puede bloquea la ventilación, no se debe colocar en libreros o gabinetes que impidan el flujo de aire por los orificios de ventilación.
10. El equipo eléctrico deber ser situado fuera del alcance de fuentes de calor como radiadores, registros de calor, estufas u otros aparatos (incluyendo amplificadores) que producen calor.
11. El aparato eléctrico deberá ser connectado a una fuente de poder sólo del tipo descrito en el instructivo de operación, o como se indique en el aparato.
12. Precaución debe ser tomada de tal manera que la tierra fisica y la polarización del equipo no sea eliminada.
13. Los cables de la fuente de poder deben ser guiados de tal manera que no sean pisados ni pellizcados por objetos colocados sobre o contra ellos, poniendo particular atención a los contactos y receptáculos donde salen del aparato.
14. El equipo eléctrico debe ser limpiado únicamente de acuerdo a las recomendaciones del fabricante.
15. En caso de existir, una antena externa deberá ser localizada lejos de las lineas de energia.
16. El cable de corriente deberá ser desconectado del cuando el equipo no sea usado por un largo periodo de tiempo.
17. Cuidado debe ser tomado de tal manera que objectos liquidos no sean derramados sobre la cubierta u orificios de ventilación.
18. Servicio por personal calificado deberá ser provisto cuando:
    A: El cable de poder o el contacto ha sido dañado; u
    B: Objectos han caído o líquido ha sido derramado dentro del aparato; o
    C: El aparato ha sido expuesto a la lluvia; o
    D: El aparato parece no operar normalmente o muestra un cambio en su desempeño; o
    E: El aparato ha sido tirado o su cubierta ha sido dañada.

European Community (CE) Electromagnetic Compatibility Directive
This equipment has been tested and found to comply with the protection requirements of European Emission Standard EN55022/EN61000-3 and the Generic European Immunity Standard EN55024.
EMC:
  EN55022 (2003)/CISPR-2 (2002): class A
  IEC61000-4-2 (2001): 4KV CD, 8KV AD
  IEC61000-4-3 (2002): 3V/m
  IEC61000-4-4 (2001):1KV (power line), 0.5KV (signal line)

Trademarks Used in this Manual

Black Box and the Double Diamond logo are registered trademarks of BB Technologies, Inc.

Mac is a registered trademark of Apple Computers, Inc.

Linux is a registered trademark of Linus Torvalds.

Internet Explorer, Windows, Windows Me, Windows NT, and Windows Vista are a registered trademarks of Microsoft Corporation.

Nagios is a registered trademark of Nagios Enterprises LLC.

Java and Solaris are trademarks of Sun Microsystems, Inc.

Unix is a registered trademark of X/Open Company Ltd.

Any other trademarks mentioned in this manual are acknowledged to be the property of the trademark owners.

Table of Contents

1. Specifications

CPU: MIcrel KS8695P controller

Memory: 16 MB SDRAM, 8 MB Flash

Serial Baud Rates: 2400 to 115,200 bps

Connectors: LES1101A: (1) DB9 RS-232 serial, (1) RJ-45 10/100BASE-T Ethernet;
   LES1102A: (2) DB9 RS-232 serial, (1) RJ-45 10/100BASE-T Ethernet

Indicators: LES1101A: (4) LEDs: Power, Activity, On RJ-45: Connectivity, Activity
   LES1102A: (5) LEDs: Power, Serial 1, Serial 2, On RJ-45: Connectivity, Activity

Temperature Tolerance: Operating: 41 to 122° F (5 to 50° C);
   Storage: -20 to +140° F (-30 to +60° C)

Humidity Tolerance: 5 to 90%

Power: (1) 12-VDC universal input external wallmount power supply,100–240 VAC, 50/60 Hz

Size: LES1101A: 4"H x 1.75"W x 1"D (10.2 x 4.5 x 2.5 cm);
   LES1102A: 3.9"H x 2.8"W x 1"D (10 x 7.2 x 2.5 cm)

Weight: LES1101A: 0.25 lb. (0.11 kg);
   LES1102A: 1 lb. (0.45 kg)

2. Overview

2.1 Introduction

This User's Manual walks you through installing and configuring your Black Box Secure Device Servers (LES1101A or LES1102A). Each of these products is referred to generically in this manual as a "*console server.*"

Once configured, you will be able to use your *console server* to securely monitor access and control the computers, networking devices, telecommunications equipment, power supplies, and operating environments in your data room or communications centers. This manual guides you in managing this infrastructure locally (across your operations or management LAN or through the local serial console port), and remotely (across the Internet or private network).

2.2 Manual Organization

This manual contains the following chapters:

• Chapter 1. Specifications: Lists the general specifications for the console servers.

• Chapter 2, Overview: An overview of the features of *console server* and information on this manual.

• Chapter 3, Installation: Physical installation of the *console server* and how to interconnect controlled devices.

• Chapter 4, System Configuration: Describes the initial installation and configuration using the Management Console. Covers configuration of the *console server* on the network and the services that will be supported.

• Chapter 5, Serial Port and Network Host: Covers configuring serial ports and connected network hosts, and setting up Users and Groups.

• Chapter 6, Secure Tunneling (SDT): Covers secure remote access using SSH and configuring for RDP, VNC, HTTP, HTTPS, etc. access to network and serially connected devices.

• Chapter 7, Alerts and Logging: Explains how to set up local and remote event/data logs and how to trigger SNMP and email alerts.

• Chapter 8, Power Management: Describes how to manage USB, serial, and network attached power strips and UPS supplies including Network UPS Tool (NUT) operation and IPMI power control.

• Chapter 9, Authentication: Access to the *console server* requires usernames and passwords that are locally or externally authenticated.

• Chapter 10, Nagios Integration: Describes how to set Nagios central management with SDT extensions and configure the *console server* as a distributed Nagios server.

• Chapter 11, System Management: Covers access to and configuration of services that will run on the *console server*
.
• Chapter 12, Status Reports: View a dashboard summary and detailed status and logs of serial and network connected devices (ports, hosts, power, and environment)

• Chapter 13, Management: Includes port controls that *Users* can access.

• Chapter 14, Basic Configuration: Command line installation and configuration using the *config* command.

• Chapter 15, Advanced Config: More advanced command line configuration activities where you will need to use Linux commands. The latest update of this manual can be found online at *www.Black Box.com/download.html*

• Appendix: Linux Commands and Source Code.

2.3 Types of Users

The *console server* supports two classes of users:

First, there are the administrative users who will be authorized to configure and control the *console server*; and to access and control all the connected devices. These administrative users will be set up as members of the admin user group and any user in this class is referred to generically in this manual as the *Administrator*. An *Administrator* can access and control the *console server* using the *config* utility, the Linux command line, or the browser-based Management Console. By default, the *Administrator* has access to all services and ports to control all the serial connected devices and network connected devices (*hosts*).

The second class of users are those who have been set up by the *Administrator* with specific limits of their access and control authority. These users are set up as members of the users user group (or some other user groups the *Administrator* may have added). They are only authorized to perform specified controls on specific connected devices and are referred to as *Users*. These *Users* (when authorized) can access serial or network connected

devices; and control these devices using the specified services (for example, Telnet, HHTPS, RDP, IPMI, Serial over LAN, Power Control). An authorized *User* also has a limited view of the Management Console and can only access authorized configured devices and review port logs.

In this manual, when the term *user* (lower case) is used, it refers to both the above classes of users. This document also uses the term *remote users* to describe users who are not on the same LAN segment as the *console server*. These remote users may be *Users,* who are on the road connecting to managed devices over the public Internet, or it may be an *Administrator* in another office connecting to the *console server* itself over the enterprise VPN, or the remote user may be in the same room or the same office but connected on a separate VLAN than the *console server*.

2.4 Management Console

The Management Console provides a view of the *console server* and all the connected devices.

Administrators can use any browser to log into the Management Console either locally or from a remote location. They can then use Management Console to manage the *console server*, the users, the serial ports and serially connected devices, network connected hosts, and connected power devices; and to view associated logs and configure alerts.
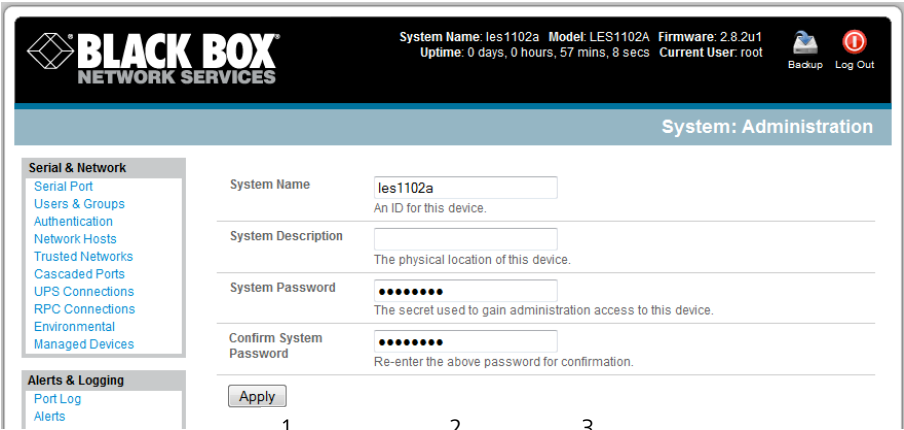
Figure 2-1. ...sole.

A *User* can also use the Management Console, but has limited menu access to control select devices, review their logs, and access them using the built-in java terminal or control power to them.

The *console server* runs an embedded Linux® operating system, and experienced Linux and UNIX® users may prefer to configure it at the command line. To get command line access, connect through a terminal emulator or communications program to the console serial port; connect via ssh or telnet through the LAN; or connect through an SSH tunneling to the *console server*.

2.5 Hardware Description

2.5.1 LES1101A Front Panel

Figure 2-2 shows the front panel of the LES1101A. Table 2-1 describes its components.
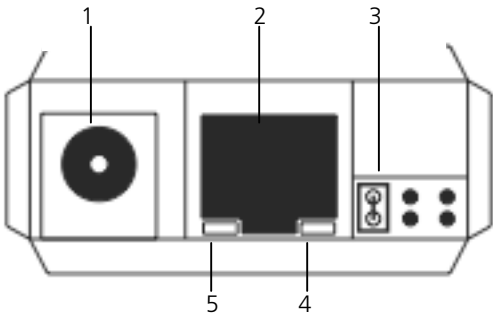
Figure 2-2. LES1101A front panel.

Table 2-1. LES1101A front-panel components.

Number	Component	Description
1	Barrel connector	Power
2	RJ-45 connector	Links to 10/100 Mbps Ethernet
3	J1 jumper	Selects RS-232, RS-485, RS-422
4	RJ-45 LED	Ethernet Connectivity LED
5	RJ-45	Ethernet Activity

2.5.2 LES1101A Back Panel

Figure 2-3 shows the LES1101A back panel. Table 2-2 describes its components.
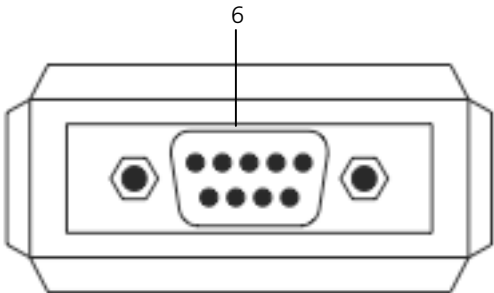
Figure 2-3. LES1101A back panel.

Table 2-2. LES1101A back-panel components.

Number	Component	Description
6	DB9 connector	Serial connector (RS-232, RS-485, RS-422)

2.5.3 LES1102A Front Panel

Figure 2-4 shows the front panel of the LES1102A. Table 2-3 describes its components.
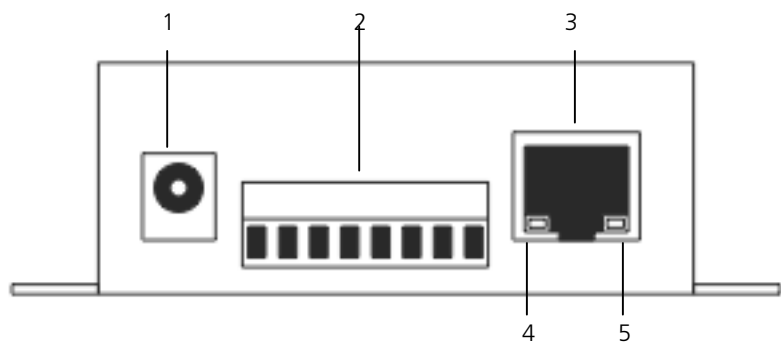
Figure 2-4. LES1102A front panel.

Table 2-3. LES1102A front-panel components.

Number	Component	Description
1	Barrel connector	Power
2	8-position Phoenix connector Port 2 (RS-422/485)	
3	RJ-45 connector	Links to 10/100 Mbps Ethernet
4	RJ-45 LED (left side of connector)	Ethernet Connectivity LED
5	RJ-45 LED (right side of connector)	Ethernet Activity LED

2.5.4 LES1102A Back Panel

Figure 2-5 shows the LES1102A back panel. Table 2-4 describes its components.
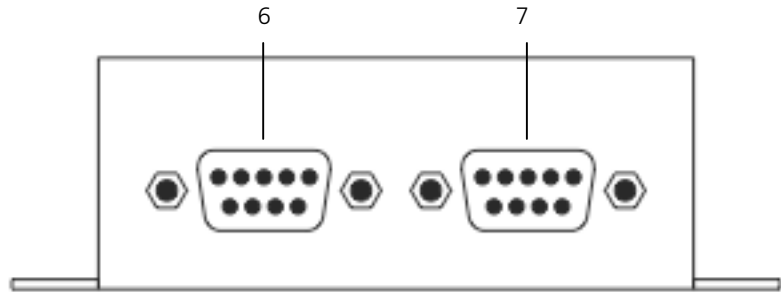
Figure 2-5. LES1102A back panel.

Table 2-4. LES1102A back-panel components.

Number	Component	Description
6	DB9 connector	Port 1 RS-232
7	DB9 connector	Port 2 RS-232

2.6 What's Included

Your package should include the following items. If anything is missing or damaged, contact Black Box Technical Support at 724-746-5500 or info@blackbox.com.

2.6.1 LES1101A

• 1101 Secure Device Server
• Universal input 12-VDC wallmount power supply
• Printed Quick Start Guide
• CD-ROM containing this user's manual

2.6.2 LES1102A

• 1101 Secure Device Server
• (2) UTP cables
• (2) DB9 F to RJ-45 S adapters
• Universal input 12-VDC wallmount power supply
• Printed Quick Start Guide
• CD-ROM containing this user's manual

3. Installation

Make sure you have everything listed in **Chapter 2, Section 2.6** for your 1101 or 1102 Secure Device Server.

3.1 Power Connection

The LES1101A or LES1102A models are each supplied with an external DC wall mount power supply. This power supply comes with a selection of wall socket adapters for each geographic region (North American, Europe, UK, Japan or Australia) and will operate with 100-240 VAC, 50/60 Hz input, 7.2 watts maximum.

Plug in the DC power cable. The 12V DC connector from the power supply unit plugs into the DC power socket on the side of the *console server* casing.

Plug in the power supply AC power cable and turn on the AC power. Confirm that the *console server* Power LED (*PWR*) is lit.

NOTE: When you first apply power to the LES1101A or LES1102A the Local and Serial LEDs will flash alternately.

The LES1102A can also be powered directly from any +9V DC to +48V DC power source by connecting the DC power lines to the IN-GND and VIN+ screw jacks.

Power connector

+9 to +48 VDC   1   GND
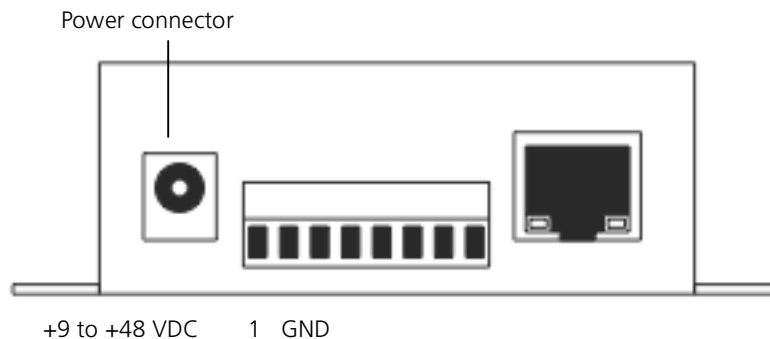
Figure 3-1. Power connector.

3.2 Network Connection

The RJ-45 LAN ports are located on the side of the LES1101A and LES1102A units.

All physical connections are made using industry standard CAT5 cabling and connectors. Make sure you only connect the LAN port to an Ethernet network that supports 10BASE-T/100BASE-T.

The first time you configure the *console server*, you must connect a PC or workstation to the *console server*'s network port.

3.3 Serial Port Connection

The LES1102A has two DB9 serial ports (Ports 1–2). By default, Port 1 is configured in Local Console (modem) mode. The LES1101A also has one DB9 serial port that's configured by default in Local Console (modem) mode.

The serial ports are all set by default in RS-232 mode. The RS-232 pinout standards for the DB9 connector are described in Table 3-1.

Table 3-1. RS-232 DB9 connector pinouts.

Signal	Pin	Definition
CD	1	Received Line Signal Detector
RXD	2	Received Data
TXD	3	Transmitted Data
DTR	4	Data Terminal Ready
GND	5	Signal Ground
DSR	6	Data Set Ready
RTS	7	Request To Send
CTS	8	Clear To Send
RI	9	Ring Indicator

3.3.1 Non RS-232 Serial Port Pinouts— LES1102A

Port 2 on the LES1102A can also be software selected to be an RS-485 or RS-422 port connected through the screw terminal block (pinout shown in Table 3-2.

Table 3-2. Non RS-232 serial port pinout for the LES1102A.

1	+V DC IN
2	GND
3	RX+
4	RX-
5	TX+
6	TX-
7	+3.3V DC OUT
8	GND

Figure 3-2. Front panel of the LES1102A showing pinout connections on the left side.

RS-422 uses a full-duplex transmit on TX+/TX- pair, receive on RX+/RX- pair.

RS-485 uses half-duplex over single pair. The LES1102A supports half duplex "party-line" communications over a 2-wire RS-485 bus (D+/D-). This is enabled by choosing the RS-485 option (instead of RS-232 or RS-422) for "Signaling Protocol" from the "Serial Port: Configuration" link on the

Web management console. Two short cable loops are also required between the RX+/TX+ pins and RX-/TX- pins. This is because the LES1102A uses universal differential transceivers that support 4-wire (RS-422) and 2-wire (RS-485) operation.

In RS-485 mode, Port 2 on the LES1102A listens on the 2-wire bus for receive data until it is required to send data. In RS-485 *send mode,* it stops receiving, enables its transmitters when there is data to be sent, transmits the data, and returns to receive mode. This eliminates the possibility of collisions with other devices that share the RS-485 bus and avoids receiving stale echoed data.

```
1 +3V
2 GND
3 RX+
4 RX-

        D+
        D-

5 TX+
6 TX-
7 +VIN
8 GND
```

Figure 3-3. RS-485 wiring diagram for LES1102A.

3.3.2 Non RS-232 Serial Port Pinouts— LES1101A

The one DB9 serial port on the LES1101A can be used as an RS-232, RS-485 or RS-422 port. By default, the LES1101A is configured in RS-232 mode (with a vertical jumper in place on the left hand J1 pins).
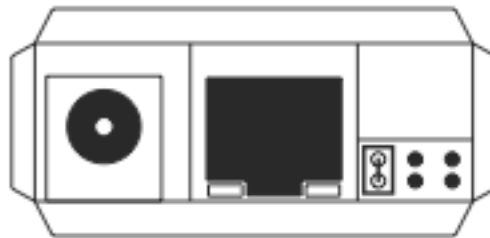
Figure 3-4. RS-232 configuration for the DB9 port on the LES1101A.

To set the port in RS-422 or RS-485 mode, you must remove the J1 jumper and then configure the *Signaling Protocol* using the Management Console.

The DB9 pinout is shown in Table 3-3.

Table 3-3. DB9 pinout for LES1101A.

Pin:	Mode:	RS232	RS422	RS485
1		DCD	DCD+	-
2		RXD	RX -	-
3		TXD	TX +	D+
4		DTR	DTR+	-
5		GND	GND	GND
6		DSR	RX +	-
7		RTS	TX -	D-
8		CTS	DCD-	-
9		-	DTR-	-

4. System Configuration

This chapter provides step-by-step instructions for the console server's initial configuration, and for connecting it to the Management or Operational LAN. The *Administrator* must:

• Activate the Management Console.

• Change the *Administrator* password.

• Set the IP address *console server*'s principal LAN port.

• Select the network services that will be supported.

This chapter also discusses the communications software tools that the *Administrator* may use to access the *console server*.

4.1 Management Console Connection

Your *console server* is configured with a default IP Address 192.168.0.1 Subnet Mask 255.255.255.0

• Directly connect a PC or workstation to the *console server*.

NOTE: For initial configuration we recommend that you connect the *console server* directly to a single PC or workstation. However, if you choose to connect your LAN before completing the initial setup steps:
   • make sure that there are no other devices on the LAN with an address of 192.168.0.1
   • make sure that the *console server* and the PC/workstation are on the same LAN segment, with no interposed router appliances.

4.1.1 Connected PC/Workstation Setup

To configure the *console server* with a browser, the connected PC/workstation should have an IP address in the same range as the *console server* (for example, 192.168.0.100):

• To configure the IP Address of your Linux or Unix PC/workstation, simply run *ifconfig*

• For Windows PCs (Win9x/Me/2000/XP, Windows NT, Windows Vista, Windows 7):

  - Click Start -> (Settings ->) Control Panel and double click Network Connections (for 95/98/Me, double click Network).

  - Right click on Local Area Connection and select Properties.

  - Select Internet Protocol (TCP/IP) and click Properties.

  - Select Use the following IP address and enter the following details:

      IP address: 192.168.0.100

      Subnet mask: 255.255.255.0

If you want to retain your existing IP settings for this network connection, click Advanced and Add the above as a secondary IP connection.
If it is not convenient to change your PC/workstation network address, you can use the *ARP-Ping* command to reset the *console server* IP address. To do this from a Windows PC:

• Click Start -> Run (or select All Programs then Accessories then Run).

• Type *cmd* and click OK to bring up the command line.

• Type *arp –d* to flush the ARP cache.

• Type *arp –a* to view the current ARP cache (this should be empty).
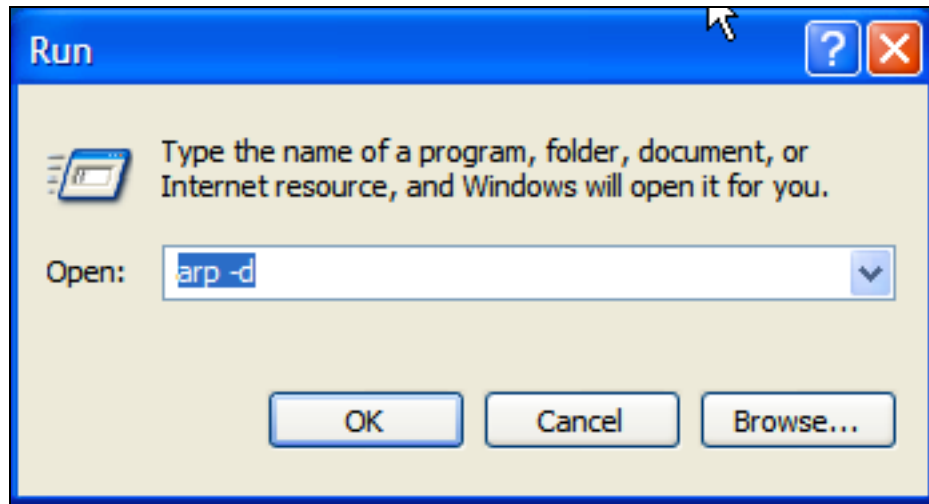
Figure 4-1. Run screen.

Now add a static entry to the ARP table and *ping* the *console server* to assign the IP address to the console server. In the example below, a *console server* has a MAC Address 00:13:C6:00:02:0F (designated on the label on the bottom of the unit) and we are setting its IP address to 192.168.100.23. Also the PC/workstation issuing the *arp* command must be on the same network segment as the *console server* (that is, have an IP address of 192.168.100.xxx)

• Type *arp -s 192.168.100.23 00-13-C6-00-02-0F* (Note for UNIX the syntax is: *arp -s 192.168.100.23 00:13:C6:00:02:0F*).

• Type *ping -t 192.18.100.23* to start a continuous ping to the new IP Address.

• Turn on the *console server* and wait for it to configure itself with the new IP address. It will start replying to the ping at this point.

• Type *arp –d* to flush the ARP cache again.

4.1.2 Browser connection

Activate your preferred browser on the connected PC/workstation and enter https://192.168.0.1 The Management Console supports all current versions of the popular browsers (Internet Explorer, Mozilla Firefox, Chrome, and more).

You will be prompted to log in. Enter the default administration username and administration password:

Username: root

Password: default
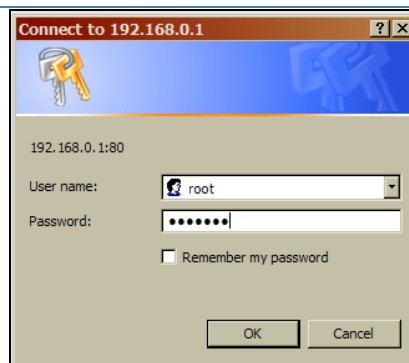
Figure 4-2. Login screen.

NOTE: *Console server*s are factory configured with HTTPS access enabled and HTTP access disabled.
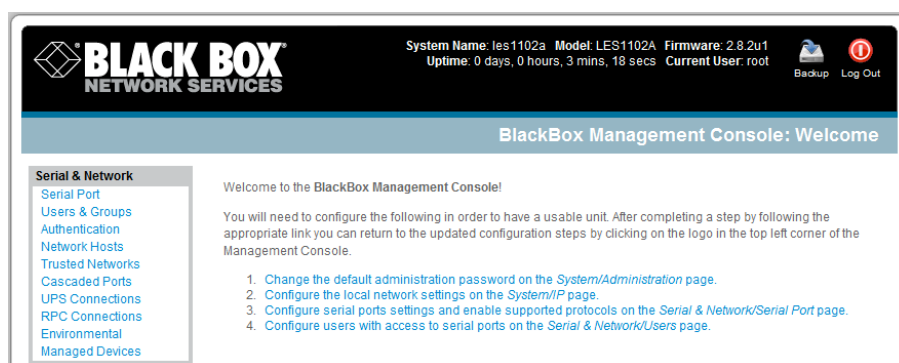
Figure 4-3. Management console welcome screen.

A Welcome screen, which lists four initial installation configuration steps, will be displayed:

1. Change the default administration password on the System/Administration page (*Chapter 4*).
2. Configure the local network settings on the System/IP page (*Chapter 4*).
3. Configure port settings and enable them on the Serial & Network/Serial Port page (*Chapter 5*).
4. Configure users with access to serial ports on the Serial & Network/Users page (*Chapter 4*).

After completing each of the above steps, you can return to the configuration list by clicking in the top left corner of the screen on the Black Box logo.

NOTE: If you are not able to connect to the Management Console at 192.168.0.1 or if the default Username/Password were not accepted, then reset your *console server* (refer to *Chapter 11*).

4.2 *Administrator* Password

For security reasons, only the administrator user named root can initially log into your *console server*. Only people who know the root password can access and reconfigure the *console server* itself. However, anyone who correctly guesses the root password could gain access (and the default root password is default). To avoid this, enter and confirm a new root password before giving the *console server* any access to, or control of, your computers and network appliances.

NOTE: We recommend that you set up a new *Administrator* user as soon as convenient and log in as this new user for all ongoing administration functions (rather than *root*). This *Administrator* can be configured in the *admin* group with full access privileges through the Serial & Network: Users & Groups menu as detailed in Ch*apter 5*.

Figure 4-4. System: Administration screen.

1. Select System: Administration.

2. Enter a new System Password then re-enter it in Confirm System Password. This is the new password for root, the main administrative user account, so choose a complex password, and keep it safe.

3. At this stage, you may also wish to enter a System Name and System Description for the *console server* to give it a unique ID and make it simple to identify.

NOTE: The System Name can contain from 1 to 64 alphanumeric characters (however you can also use the special characters "-", "_", and ".") There are no restrictions on the characters that can be used in the System Description or the System Password (each can contain up to 254 characters). However, only the first eight System Password characters are used to make the *password hash*.

4. Click Apply. Since you have changed the password, you will be prompted to log in again. This time, use the new password.

NOTE: If you are not confident that your *console server* has the current firmware release, you can upgrade. Refer to *Upgrade Firmware—Chapter 11*.

4.3 Network IP Address

The next step is to enter an IP address for the principal Ethernet (*LAN/Network/Network1*) port on the *console server*, or enable its DHCP client so that it automatically obtains an IP address from a DHCP server on the network it will connect to.

On the System: IP menu, select the Network Interface page, then check dhcp or static for the Configuration Method.

If you selected Static, you must manually enter the new IP Address, Subnet Mask, Gateway, and DNS server details. This selection automatically disables the DHCP client.

Figure 4-5. IP Settings screen.

If you selected DHCP, the *console server* will look for configuration details from a DHCP server on your management LAN. This selection automatically disables any static address. The *console server* MAC address is printed on a label on the base plate.

NOTE: In its factory default state (with no Configuration Method selected) the *console server* has its DHCP client enabled, so it automatically accepts any network IP address assigned by a DHCP server on your network. In this initial state, the *console server* will then respond to both its Static address (192.168.0.1) and its newly assigned DHCP address.

By default, the *console server* LAN port auto-detects the Ethernet connection speed. You can use the Media menu to lock the Ethernet to 10 Mbps or 100 Mbps, and to Full Duplex (FD) or Half Duplex (HD).

NOTE: If you changed the *console server* IP address, you may need to reconfigure your PC/workstation so it has an IP address that is in the same network range as this new address.

Click Apply.

Enter http://new IP address to reconnect the browser on the PC/workstation that is connected to the *console server*.

IPv6 configuration

You can also configure the *console server* management LAN for IPv6 operation:

• On the System: IP menu, select the General Settings page and check Enable IPv6.

• Then, configure the IPv6 parameters on the Network Interface page.

4.4 System Services

The *Administrator* can access and configure the *console server* and connect to the managed devices using a range of access protocols (services). The factory default enables HTTPS and SSH access to the *console server* and disables HTTP and Telnet.

A *User* or *Administrator* can also use nominated enabled services to connect through the *console server* to attached serial and network connected managed devices.

The *Administrator* can simply disable any of the services, or enable others.

Figure 4-6. System: Services screen.

Select the System: Services option, then select/deselect for the service to be enabled/disabled. The following access protocol options are available:

- HTTPS: This ensures secure browser access to all the Management Console menus. It also allows appropriately configured *Users* secure browser access to selected Management Console *Manage* menus. If you enable HTTPS, the *Administrator* will be able to use a secure browser connection to the *Console server*'s Management Console. For information on certificate and user client software configuration, refer to *Chapter 9—Authentication*. By default, HTTPS is enabled, and we recommend that that you only use HTTPS access if the *console server* will be managed over any public network (for example, the Internet).

- HTTP: By default HTTP is disabled. We recommend that the HTTP service remain disabled if the *console server* will be remotely accessed over the Internet.

- Telnet: This gives the *Administrator* Telnet access to the system command line shell (Linux commands). This may be suitable for a local direct connection over a management LAN. By default, Telnet is disabled. We recommend that this service remain disabled if you will remotely administer the *console server*.

- SSH: This service provides secure SSH access to the Linux command line shell. We recommend that you choose SSH as the protocol where the *Administrator* connects to the *console server* over the Internet or any other public network. This will provide authenticated communications between the SSH client program on the remote PC/workstation and the SSH sever in the *console server*. By default SSH is enabled. For more information on SSH configuration refer *Chapter 9—Authentication*.

You can configure related service options at this stage:

- Ping: This allows the *console server* to respond to incoming ICMP echo requests. Ping is enabled by default. For security reasons, you should disable this service after initial configuration.

And there are some serial port access parameters that you can configure on this menu:

• Base: The *console server* uses specific default ranges for the TCP/IP ports for the various access services that *Users* and *Administrators* can use to access devices attached to serial ports (as covered in *Chapter 4—Configuring Serial Ports*). The *Administrator* can also set alternate ranges for these services, and these secondary ports will then be used in addition to the defaults.

The default TCP/IP base port address for *telnet* access is 2000, and the range for *telnet* is IP Address: Port (2000 + serial port #), *that is,* 2001–2002. If the *Administrator* sets 8000 as a secondary base for telnet, then serial port #2 on the *console server* can be accessed via telnet at IP Address:2002 and at IP Address:8002.

The default base for SSH is 3000; for Raw TCP is 4000; and for RFC2217 it is 5000.

Click Apply. As you apply your services selections, the screen will be updated with a confirmation message:

Message Changes to configuration succeeded.

4.5 Communications Software

You have configured access protocols for the *Administrator* client to use when connecting to the *console server*. *User* clients (who you may set up later) will also use these protocols when accessing *console server* serial attached devices and network attached hosts. You will need to have appropriate communications software tools set up on the *Administrator* (and *User*) PC/workstation.

Black Box provides the *SDT Connector* Java applet as the recommended client software tool. You can use other generic tools such as PuTTY and SSHTerm. These tools are all described below as well.

4.5.1 SDT Connector

Each *console server* has an unlimited number of *SDT Connector* licenses to use with that *console server*.
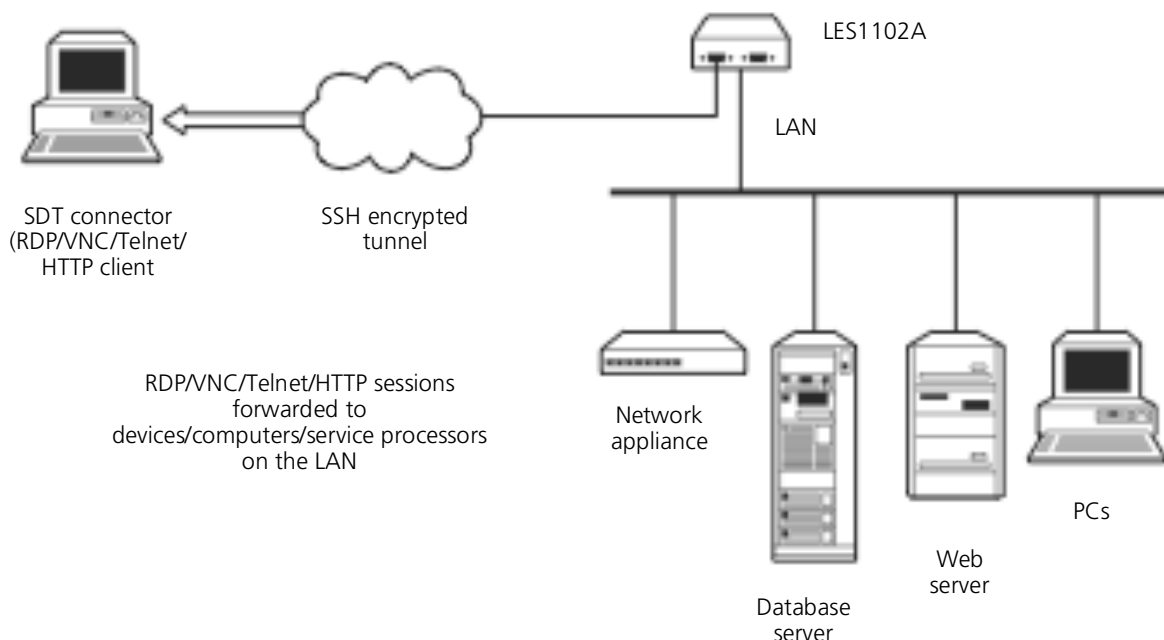
Figure 4-7. SDT connector application.

SDT Connector is a lightweight tool that enables *Users* and *Administrators* to securely access the *console server* and the various computers, network devices, and appliances that may be serially or network connected to the *console server*.

SDT Connector is a Java applet that couples the trusted SSH tunneling protocol with popular access tools such as Telnet, SSH, HTTP, HTTPS, VNC, and RDP to provide point-and-click secure remote management access to all the systems and devices being managed.

Information on using *SDT Connector* for browser access to the *console server*'s Management Console, Telnet/SSH access to the *console server* command line, and TCP/UDP connecting to hosts that are network connected to the *console server* is in *Chapter 6—Secure Tunneling*.

SDT Connector can be installed on Windows 2000, XP, 2003, Windows Vista, Windows NT PCs, and on most Linux, UNIX, and Solaris computers.

4.5.2 PuTTY

You can also use communications packages like *PuTTY* to connect to the *console server* command line (and to connect serially attached devices as covered in *Chapter 5*). *PuTTY* is a freeware implementation of Telnet and SSH for Windows and UNIX platforms. It runs as an executable application without needing to be installed onto your system. *PuTTY* (the Telnet and SSH client itself) can be downloaded from *http://www.tucows.com/preview/195286.html*
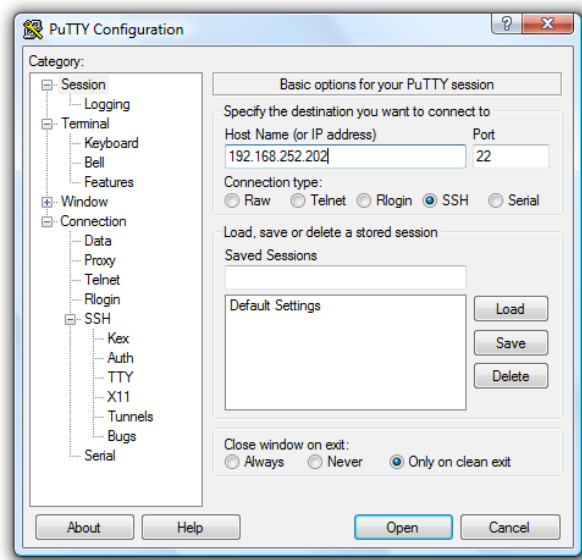
To use PuTTY for an SSH terminal session from a Windows client, enter the *console server*'s IP address as the "Host Name (or IP address)."
To access the *console server* command line, select "SSH" as the protocol, and use the default IP Port 22.
Click "Open" and the *console server* login prompt will appear. (You may also receive a "Security Alert" that the host's key is not cached. Choose "yes" to continue.)
Using the Telnet protocol is similarly simple, but you use the default port 23.

Figure 4-8. PuTTY screen.

4.5.3 SSHTerm

Another popular communications package you can use is *SSHTerm,* an open source package that you can download from *http://sourceforge.net/projects/sshtools*

To use *SSHTerm* for an SSH terminal session from a Windows Client, simply Select the "File" option and click on "New Connection."
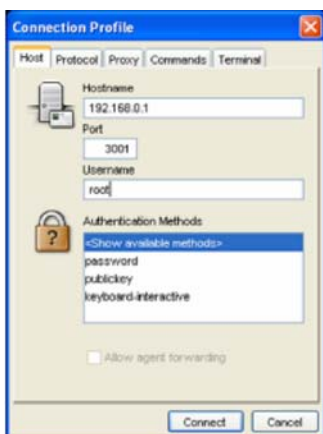
A new dialog box will appear for your "Connection Profile." Type in the host name or IP address (for the *console server* unit) and the TCP port that the SSH session will use (port 22). Then type in your username, choose password authentication, and click connect.

You may receive a message about the host key fingerprint. Select "yes" or "always" to continue.

The next step is password authentication. The system prompts you for your username and password from the remote system. This logs you on to the *console server*

Figure 4-9. Connection Profile screen.

5. Serial Port, Host, Device, and User Configuration

The Black Box LES1101A and LES1102A *console server* enables access and control of serially attached devices and network attached devices (*hosts*). The *Administrator* must configure access privileges for each of these devices, and specify the services that can be used to control the devices. The *Administrator* can also set up new users and specify each user's individual access and control privileges.
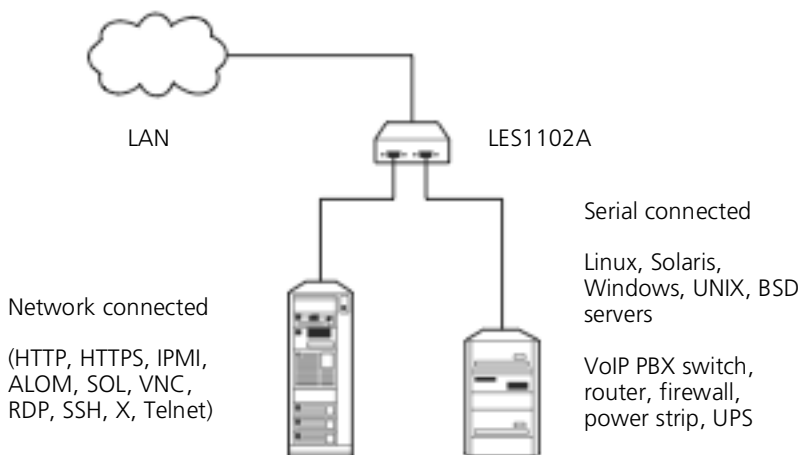
Figure 5-1. Configuration.

This chapter covers each of the steps in configuring hosts and serially attached devices:

Configure Serial Ports—setting up the protocols to be used in accessing serially-connected devices.

Users & Groups—setting up users and defining the access permissions for each of these users.

*Authentication—*covered in more detail in Chapter 9.

Network Hosts—configuring access to network connected devices (referred to as hosts).

Configuring Trusted Networks—nominate user IP addresses.

Cascading and Redirection of Serial Console Ports.

Connecting to Power (UPS PDU and IPMI) and Environmental Monitoring (EMD) devices.

Managed Devices—presents a consolidated view of all the connections.

5.1 Configure Serial Ports

To configure a serial port, you must first set the Common Settings (the protocols and the RS-232 parameters (such as baud rate) that will be used for the data connection to that port.

Select what mode the port is to operate in. You can set each port to support one of five operating modes:

1. Console Server Mode is the default and this enables general access to serial console port on the serially attached devices.

2. Device Mode sets the serial port up to communicate with an intelligent serial controlled PDU or UPS.

3. SDT Mode enables graphical console access (with RDP, VNC, HTTPS, etc.) to hosts that are serially connected.

4. Terminal Server Mode sets the serial port to wait for an incoming terminal login session.

5. Serial Bridge Mode enables transparently interconnects two serial port devices over a network.
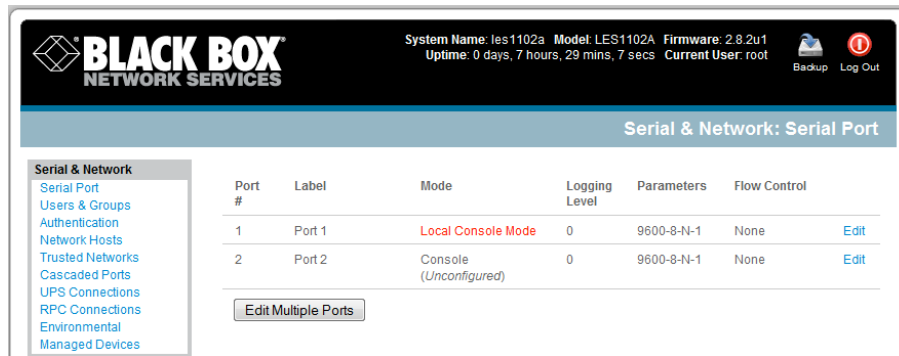
Figure 5-2. Serial port screen.

Select Serial & Network: Serial Port and you will see the current labels, modes, logging levels, and RS-232 protocol options that are currently set up for each serial port.

By default, each serial port is set in Console Server mode. To reconfigure the port, click Edit.

When you have reconfigured the common settings (*Chapter 5.1.1*) and the mode (*Chapters 5.1.2–5.1.6*) for each port, you can set up any remote syslog (*Chapter 5.1.7*), then click Apply.

NOTE: If you want to set the same protocol options for multiple serial ports at once, click Edit Multiple Ports and select which ports you want to configure as a group.

If the *console server* has been configured with distributed Nagios monitoring enabled, then you will also be presented with Nagios Settings options to enable nominated services on the Host to be monitored (refer *Chapter 10—Nagios Integration*).

5.1.1 Common Settings

There are a number of common settings that you can set for each serial port. These are independent of the mode in which the port is being used. Set these serial port parameters to match the serial port parameters on the device you attach to that port.
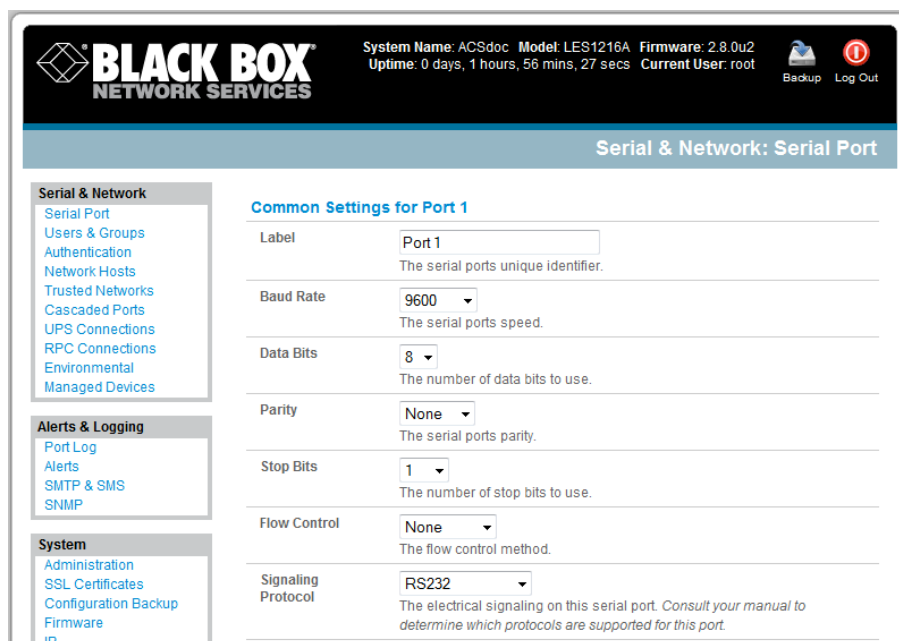
Figure 5-3. Edit multiple ports screen.

Specify a label for the port.

Select the appropriate Baud Rate, Parity, Data Bits, Stop Bits, and Flow Control for each port. (Note: The RS-485/RS-422 option is not relevant for *console server*s.)

1101 and 1102 Secure Device Servers

Before proceeding with further serial port configuration, connect the ports to the serial devices they will be controlling, and make sure they have matching settings.

NOTE: The serial ports are all set at the factory to RS-232: 9600 baud, no parity, 8 data bits, 1 stop bit, and *Console server* Mode. You can change the baud rate to 2400–230400 baud using the management console. You can configure lower baud rates (50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800 baud) from the command line. Refer to *Chapter 14— Basic Configuration (Linux Commands).*

5.1.2 Console Server Mode

Select Console Server Mode to enable remote management access to the serial console that is attached to this serial port:

Figure 5-4. Console Server settings screen.

Logging Level: This specifies the level of information to be logged and monitored (referto *Chapter 7— Alerts and Logging).*

Telnet: When the Telnet service is enabled on the *console server*, a Telnet client on a *User* or *Administrator's* computer can connect to a serial device attached to this serial port on the *console server*. The Telnet communications are unencrypted, so this protocol is generally recommended only for local connections.

With Win2000/XP/NT you can run *telnet* from the command prompt (*cmd.exe*). Vista and Windows 7 include a Telnet client and server, but they are not enabled by default. To enable Telnet:

• Log in as *Admin* and go to *Start/Control Panel/Programs and Features.*

• Select *Turn Windows features on or off,* check the *Telnet Client,* and click *OK.*
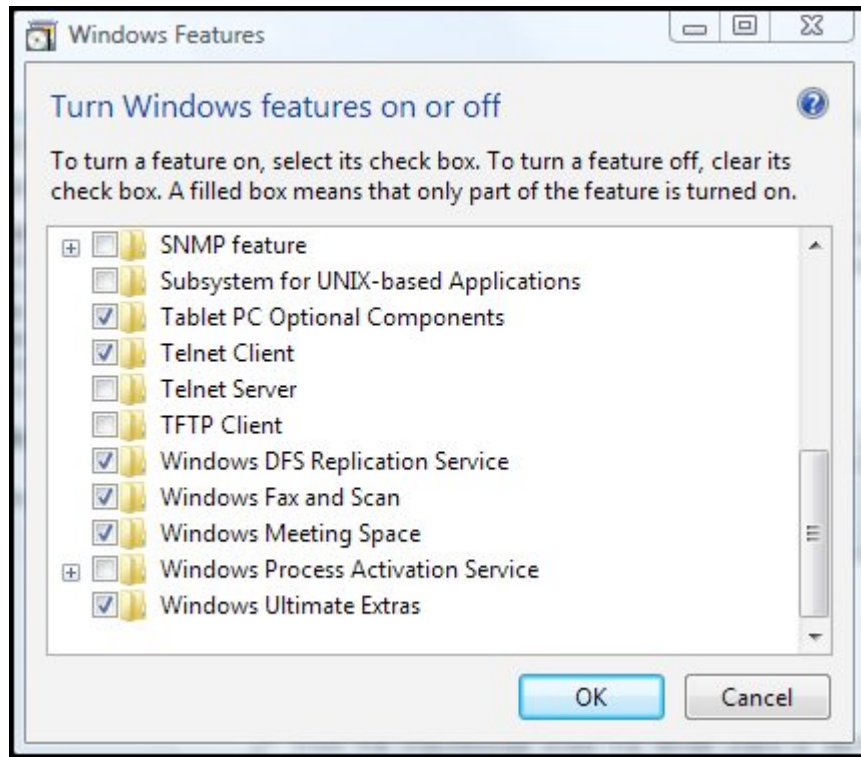
Figure 5-5. Windows features screen.

If the remote communications are tunneled with *SDT Connector*, then you can use Telnet to securely access these attached devices (refer to the Note below).

NOTE: In Console Server mode, *Users* and *Administrators* can use *SDT Connector* to set up secure Telnet connections that are SSH tunneled from their client PC/workstations to the serial port on the *console server*. *SDT Connector* can be installed on Windows 2000, XP, 2003, Vista, and Windows 7 PCs and on most Linux platforms. You can also set up secure Telnet connections with a simple point-and-click.

To use *SDT Connector* to access consoles on the *console server* serial ports, you configure *SDT Connector* with the *console server* as a *gateway*, then configure it as a *host*. Next, you enable Telnet service on Port (2000 + serial port #) *i.e.* 2001–2002. Refer to *Chapter 6* for more details on using *SDT Connector* for Telnet and SSH access to devices that are attached to the *console server* serial ports.

You can also use standard communications packages like *PuTTY* to set a direct Telnet (or SSH) connection to the serial ports (refer to the Note below).

NOTE: *PuTTY* also supports Telnet (and SSH), and the procedure to set up a Telnet session is simple. Enter the *console server's* IP address as the "Host Name (or IP address)." Select "Telnet" as the protocol and set the "TCP port" to 2000 plus the physical serial port number (*that is,* 2001 to 2002).

Click the "Open" button. You may then receive a "Security Alert" that the host's key is not cached. Choose "yes" to continue. You will then be presented with the login prompt of the remote system connected to the serial port chosen on the *console server*. Login as normal and use the host serial console screen.
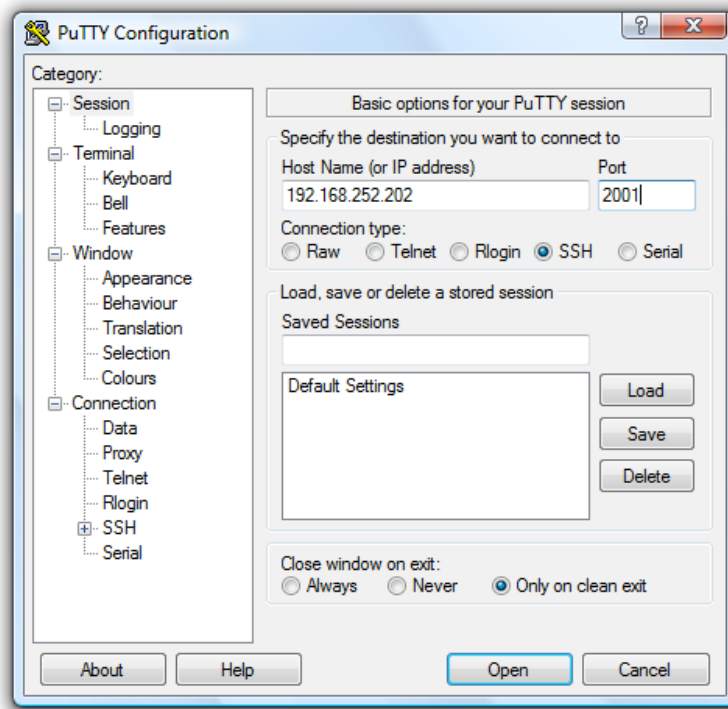
Figure 5-6. PuTTY Configuration screen.

PuTTY can be downloaded at http://www.tucows.com/preview/195286.html

SSH: We recommend that you use SSH as the protocol where the *User* or *Administrator* connects to the *console server* (or connects through the *console server* to the attached serial consoles) over the Internet or any other public network. This will provide authenticated SSH communications between the SSH client program on the remote user's computer and the *console server*, so the user's communication with the serial device attached to the *console server* is secure.

For SSH access to the consoles on devices attached to the *console server* serial ports, you can use *SDT Connector*. Configure *SDT Connector* with the *console server* as a *gateway*, then as a *host*, and enable SSH service on Port (3000 + serial port #) *i.e.* 3001-3002. *Chapter 6—Secure Tunneling* has more information on using *SDT Connector* for SSH access to devices that are attached to the *console server* serial ports.
You can also use common communications packages, like *PuTTY* or *SSHTerm* to SSH connect directly to port address IP Address _ Port (3000 + serial port #)*, for example,* 3001–3002.

SSH connections can be configured using the standard SSH port 22. Identify the the serial port that's accessed by appending a descriptor to the username. This syntax supports:

> *<username>:<portXX>*
> *<username>:<port label>*
> *<username>:<ttySX>*
> *<username>:<serial>*

For a *User* named "fred" to access serial port 2, when setting up the SSHTerm or the PuTTY SSH client, instead of typing *username = fred* and *ssh port = 3002*, the alternate is to type *username = fred:port02 (*or *username = fred:ttyS1)* and *ssh port = 22.*

Or, by typing *username=fred:serial* and *ssh port = 22.* A port selection option appears to the *User*:
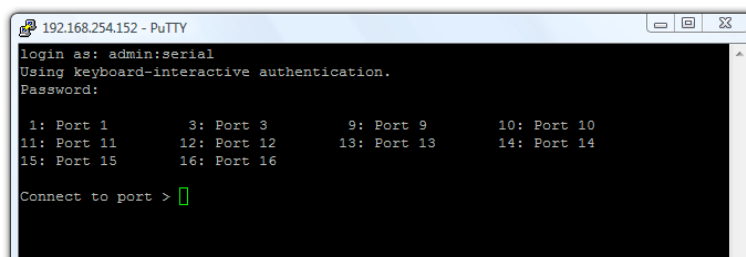
Figure 5-7. Port Selection option.
This syntax enables *Users* to set up SSH tunnels to all serial ports with only opening a single IP port 22 in their firewall/gateway.

TCP: RAW TCP allows connections directly to a TCP socket. Communications programs like *PuTTY* also support RAW TCP. You would usually access this protocol via a custom application.

For RAW TCP, the default port address is IP Address _ Port (4000 + serial port #) *for example,* 4001–4002.

RAW TCP also enables the serial port to be tunneled to a remote *console server*, so two serial port devices can transparently interconnect over a network (see *Chapter 5.1.6—Serial Bridging*).

RFC2217 Selecting *RFC2217* enables serial port redirection on that port. For RFC2217, the default port address is IP Address _ Port (5000 + serial port #), that is, 5001– 5002.

Special client software is available for Windows UNIX and Linux that supports RFC2217 virtual com ports, so a remote host can monitor and manage remote serially attached devices, as though they were connected to the local serial port (see *Chapter 5.6—Serial Port Redirection* for details).

RFC2217 also enables the serial port to be tunneled to a remote *console server*, so two serial port devices can transparently interconnect over a network (see *Chapter 5.1.6—Serial Bridging*).

Unauthenticated Telnet: Selecting *Unauthenticated Telnet* enables telnet access to the serial port without requiring the user to provide credentials. When a user accesses the *console server* to telnet to a serial port, he normally is given a login prompt. With unauthenticated telnet, the user connects directly through to a port with any *console server* login. This mode is mainly used when you have an external system (such as *conserver*) managing user authentication and access privileges at the serial device level.

For Unauthenticated Telnet, the default port address is IP Address _ Port (6000 + serial port #) i.e. 6001 – 6002.

Accumulation Period: By default, once a connection is established for a particular serial port (such as a RFC2217 redirection or Telnet connection to a remote computer) then any incoming characters on that port are forwarded over the network on a character by character basis. The accumulation period changes this by specifying a period of time that incoming characters will be collected before then being sent as a packet over the network.

Escape Character: This enables you to change the character used for sending escape characters. The default is ~.

Power Menu: This setting enables the shell power command. A user can control the power connection to a Managed Device from command line when he is connected to the device via telnet or ssh. To operate, the Managed Device must be set up with both its Serial port connection and Power connection configured. The command to bring up the power menu is ~p
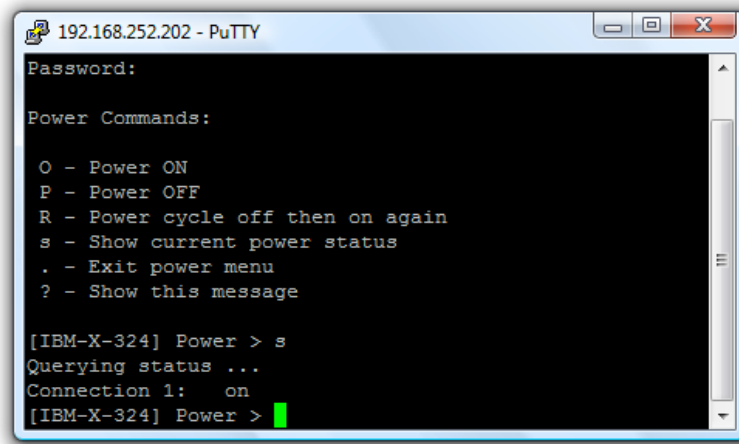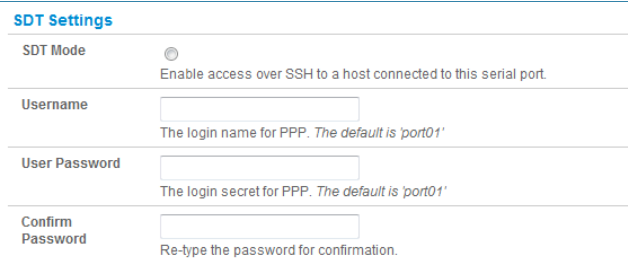
Figure 5-8. PuTTY screen.

Single Connection: This setting limits the port to a single connection> If multiple users have access privileges for a particular port, only one user at a time can access that port (that is, port "snooping" is not permitted).

5.1.3 SDT Mode

This setting allows port forwarding of RDP, VNC, HTPP, HTTPS, SSH, Telnet, and other LAN protocols through to computers that are locally connected to the *console server* by their serial COM port. Port forwarding requires that you set up a PPP link over this serial port.

Figure 5-9. SDT settings.

For configuration details, refer to Chapter *6.4—Using SDT Connector to Telnet or SSH connect to devices that are serially attached to the console server*.

5.1.4 Device (RPC, UPS, EMD) Mode

This mode configures the selected serial port to communicate with a serial controlled Uninterruptable Power Supply (UPS), Remote Power Controller/Power Distribution Unit (RPC) or Environmental Monitoring Device (EMD).
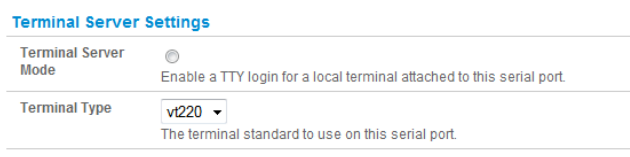
Figure 5-10. Device settings screen.

Select the desired Device Type (UPS, RPC or EMD)

Proceed to the appropriate device configuration page (Serial & Network: UPS Connections, RPC Connection or Environmental) as detailed in *Chapter 8—Power Management*.

5.1.5 Terminal Server Mode

Select Terminal Server Mode and the Terminal Type (vt220, vt102, vt100, Linux, or ANSI) to enable a *getty* on the selected serial port.

Figure 5-11. Terminal server settings screen.

The *getty* will then configure the port and wait for a connection to be made. An active connection on a serial device is usually indicated by the Data Carrier Detect (DCD) pin on the serial device being raised. When a connection is detected, the *getty* program issues a login: prompt, and then invokes the login program to handle the actual system login.

NOTE: Selecting Terminal Server mode will disable Port Manager for that serial port, so data is no longer logged for alerts, etc.
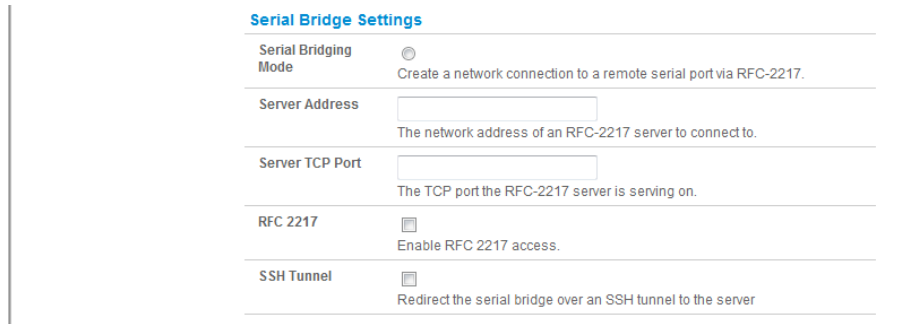
5.1.6 Serial Bridging Mode

With serial bridging, the serial data on a nominated serial port on one *console server* is encapsulated into network packets and then transported over a network to a second *console server*. It is then represented on its serial port again as serial data. The two *console server*s effectively act as a virtual serial cable over an IP network.

One *console server* is configured as the *Server*. Set the *Server* serial port to be bridged in Console Server mode with either RFC2217 or RAW enabled (as described in *Chapter 5.1.2—Console Server Mode*).

For the *Client console server*, the serial port to bridge must be set in Bridging Mode:

Serial Bridge Settings

Serial Bridging Mode	○ Create a network connection to a remote serial port via RFC-2217.
Server Address	[ ] The network address of an RFC-2217 server to connect to.
Server TCP Port	[ ] The TCP port the RFC-2217 server is serving on.
RFC 2217	☐ Enable RFC 2217 access.
SSH Tunnel	☐ Redirect the serial bridge over an SSH tunnel to the server

Figure 5-12. Serial bridge settings.

Select Serial Bridging Mode and specify the IP address of the *Server console server* and the TCP port address of the remote serial port (for RFC2217 bridging this will be 5001–5002).

By default, the bridging client will use RAW TCP. Select RFC2217 if this is the *console server* mode you have specified on the server *console server.*
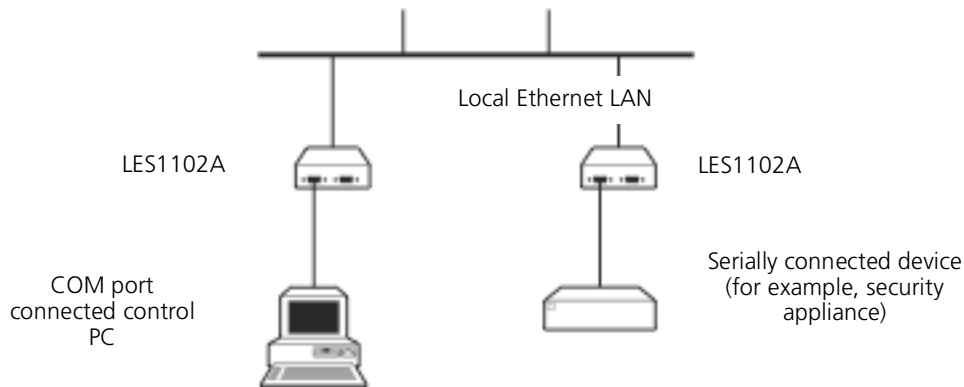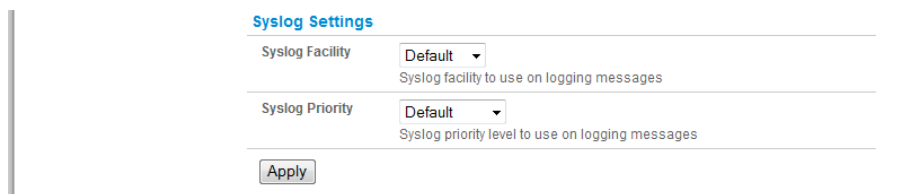
Figure 5-13. Serial bridging mode application.

You may secure the communications over the local Ethernet by enabling SSH. You will need to generate and upload keys (refer to *Chapter 15— Advanced Configuration*).

5.1.7 Syslog

In addition to built-in logging and monitoring (which can be applied to serial-attached and network-attached management accesses, as covered in *Chapter 7—Alerts and Logging*), you can also configure the *console server* to support the remote syslog protocol on a per serial port basis: Select the Syslog Facility/Priority fields to enable logging of traffic on the selected serial port to a syslog server; and to appropriately sort and action those logged messages (that is, redirect them/send alert email etc.).

Syslog Settings

Syslog Facility	[Default ▾] Syslog facility to use on logging messages
Syslog Priority	[Default ▾] Syslog priority level to use on logging messages
[Apply]	

Figure 5-14. Syslog setting screen.

For example, if the computer attached to serial port 3 should never send anything out on its serial console port, the *Administrator* can set the Facility for that port to *local0* (*local0 .. local1* are for site local values), and the Priority to *critical*. At this priority, if the *console server* syslog server does receive a message, it will automatically raise an alert. Refer to *Chapter 7—Alerts and Logging.*

1101 and 1102 Secure Device Servers

5.2 Add/ Edit Users

The *Administrator* uses this menu selection to set up, edit, and delete users, and to define the access permissions for each of these users.
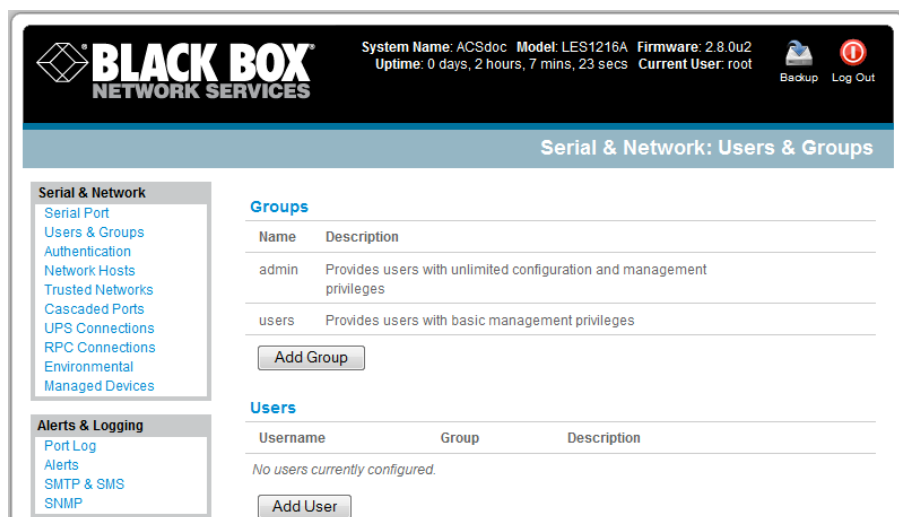
Figure 5-15. Users and Groups screen.

Users can be authorized to access specified *console server* serial ports and specified network-attached hosts. These users can also be given full *Administrator* status (with full configuration and management and access privileges).

To simplify user set up, they can be configured as members of Groups. There are two Groups set up by default (*admin* and *user*).

1. Members of the *admin* group have full *Administrator* privileges. The *admin* user (*Administrator*) can access the *console server* using any of the services that are enabled in *System: Services*. For example, if only HTTPS has been enabled, then the *Administrator* can only access the *console server* using HTTPS. Once logged in, they can reconfigure the *console server* settings (for example, to enabled HTTP/Telnet for future access). They can also access any of the connected Hosts or serial port devices using any of the services that have been enabled for these connections. The *Administrator* can reconfigure the access services for any Host or serial port. Only trusted users should have *Administrator* access.

NOTE: For convenience, the SDT Connector "Retrieve Hosts" function retrieves and auto-configures checked serial ports and checked hosts only, even for admin group users.

2. Members of the user group have limited access to the *console server* and connected Hosts and serial devices. These *Users* can access only the Management section of the Management Console menu and they have no command line access to the *console server*. They also can only access those Hosts and serial devices that are checked for them, using services that are enabled.

3. The *Administrator* can also set up additional Groups with specific serial port and host access permissions (same as *Users*). However, users in these additional groups don't have any access to the Management Console menu or any command line access to the *console server* itself. Finally, the *Administrator* can also set up users who are not a member of any Groups. They will have the same access as users in the additional groups.

To set up new Groups and new users, and to classify users as members of particular Groups:

Select Serial & Network: Users & Groups to display the configured Groups and Users.

Click Add Group to add a new Group.

Add a Group name and Description for each new Group, then nominate the Accessible Hosts, Accessible Ports, and Accessible RPC Outlets(s) that you want any users in this new Group to be able to access.

Click Apply.

Figure 5-16. Add a new user screen.

Click Add User to add a new user.

Add a Username and a confirmed Password for each new user. You may also include information related to the user (for example, contact details) in the Description field.

NOTE: The User Name can contain from 1 to 127 alphanumeric characters (you can also use the special characters "-", "_", and ".").

There are no restrictions on the characters that you can use in the user Password (each can contain up to 254 characters). Only the first eight Password characters are used to make the *password hash*.

Specify which Group (or Groups) you want the user to join.

Check specific Accessible Hosts and/or Accessible Ports to nominate the serial ports and network connected hosts you want the user to have access privileges to.

If there are configured RPCs, you can check Accessible RPC Outlets to specify which outlets the user is able to control (that is, Power On/Off).

Click Apply. The new user can now access the Network Devices, Ports, and RPC Outlets you nominated as accessible. Plus, if the user is a Group member, he can also access any other device/port/outlet that was set up as accessible to the Group.

NOTE: There are no specific limits on the number of users you can set up, nor on the number of users per serial port or host. Multiple users (*Users* and *Administrators*) can control/monitor one port or host.

There are no specific limits on the number of Groups. Each user can be a member of a number of Groups (they take on the cumulative access privileges of each of those Groups). A user does not have to be a member of any Groups (but if the *User* is not even a member of the default *user* group. then he will not be able to use the Management Console to manage ports).

The time allowed to re-configure increases as the number and complexity increases. We recommend that you keep the aggregate number of users and groups under 250.

The *Administrator* can also edit the access settings for any existing users:

Select Serial & Network: Users & Groups and click Edit for the *User* to be modified.

NOTE: For more information on enabling the SDT Connector so each user has secure tunneled remote RPD/VNC/Telnet/HHTP/HTTPS/SoL access to the network connected hosts, refer to *Chapter 6*.

1101 and 1102 Secure Device Servers

5.3 Authentication

Refer to *Chapter 9.1—Authentication Configuration* for authentication configuration details.

5.4 Network Hosts

To access a locally networked computer or device (referred to as a *Host*), you must identify the Host and specify the TCP or UDP ports/services that will be used to control that Host.

Selecting Serial & Network: Network Hosts presents all the network connected Hosts that have been enabled for access, and the related access TCP ports/services.

Click Add Host to enable access to a new Host (or select Edit to update the settings for an existing Host).

Figure 5-17. Network hosts screen.

Enter the IP Address or DNS Name and a Host Name (up to 254 alphanumeric characters) for the new network connected Host (and optionally enter a Description).

Add or edit the Permitted Services (or TCP/UDP port numbers) that are authorized to be used in controlling this host. Only these *permitted services* will be forwarded through by SDT to the Host. All other services (TCP/UDP ports) will be blocked.

The Logging Level specifies the level of information to be logged and monitored for each Host access (refer to *Chapter 7—Alerts and Logging*).

If the Host is a PDU or UPS power device or a server with IPMI power control, then specify RPC (for IPMI and PDU) or UPS and the Device Type. The *Administrator* can then configure these devices and enable which users have permission to remotely cycle power, etc. (refer to *Chapter 8*).

Otherwise, leave the Device Type set to None.

If the *console server* has been configured with distributed Nagios monitoring enabled, then you will also be presented with Nagios Settings options to enable nominated services on the Host to be monitored (refer to *Chapter 10—Nagios Integration*).

Click Apply. This will create the new Host and also create a new Managed Device (with the same name).

5.5 Trusted Networks

The Trusted Networks facility gives you an option to nominate specific IP addresses where users (*Administrators* and *Users*) must be located to access *console server* serial ports.

Select Serial & Network: Trusted Networks. To add a new trusted network, select Add Rule.
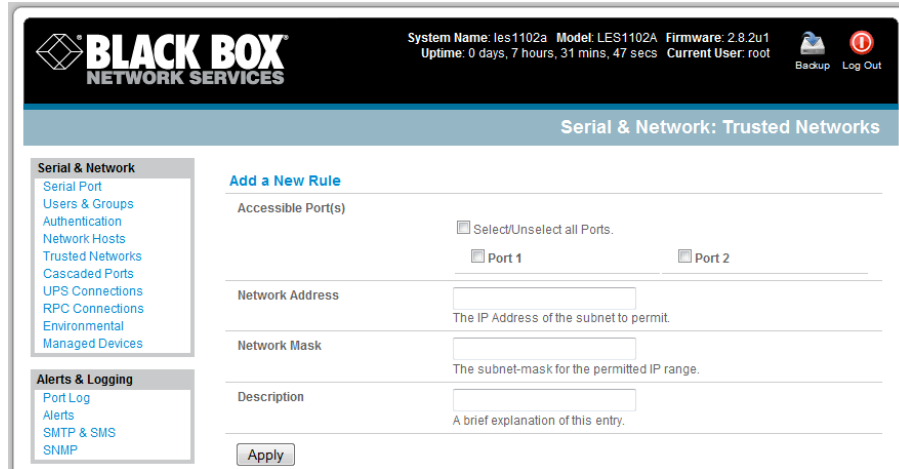
Figure 5-18. Trusted networks screen.

Select the Accessible Port(s) that the new rule is to be applied to.

Then, enter the Network Address of the subnet to be permitted access.

Then, specify the range of addresses that are to be permitted by entering a Network Mask for that permitted IP range, *for example:*

To permit all the users located with a particular Class C network (for example, 204.15.5.0) connection to the nominated port then you would add the following Trusted Network New Rule:

Network Address	204.15.5.0
Network Mask	255.255.255.0

If you want to permit only the one user who is located at a specific IP address (for example, 204.15.5.13 say) to connect:

Network Address	204.15.5.0
Network Mask	255.255.255.255

If, however, you want to allow all the users operating from within a specific range of IP addresses (for example, any of the thirty addresses from 204.15.5.129 to 204.15.5.158) to be permitted connection to the nominated port:

Host /Subnet Address	204.15.5.128
Subnet Mask	255.255.255.224

Click Apply.
.

NOTE: The above Trusted Networks will limit *Users* and *Administrators* access to the console serial ports. They do not restrict access to the *console server* itself or to attached hosts. To change the default settings for this access, you will to need to edit the *IPtables* rules as described in *Chapter 15—Advanced Configuration*.

5.6 Serial Port Redirection

To allow an application on a client PC to access the virtual serial ports on the console server, you need to run client software (to redirect the local serial port traffic to remote *console server* serial port).

There's a selection of commercial software available including *Serial to Ethernet* from Eltima (www.eltima.com) and *Serial/IP™ COM Port Redirector* from Tactical Software (www.tacticalsoftware.com/products/serialip.htm).
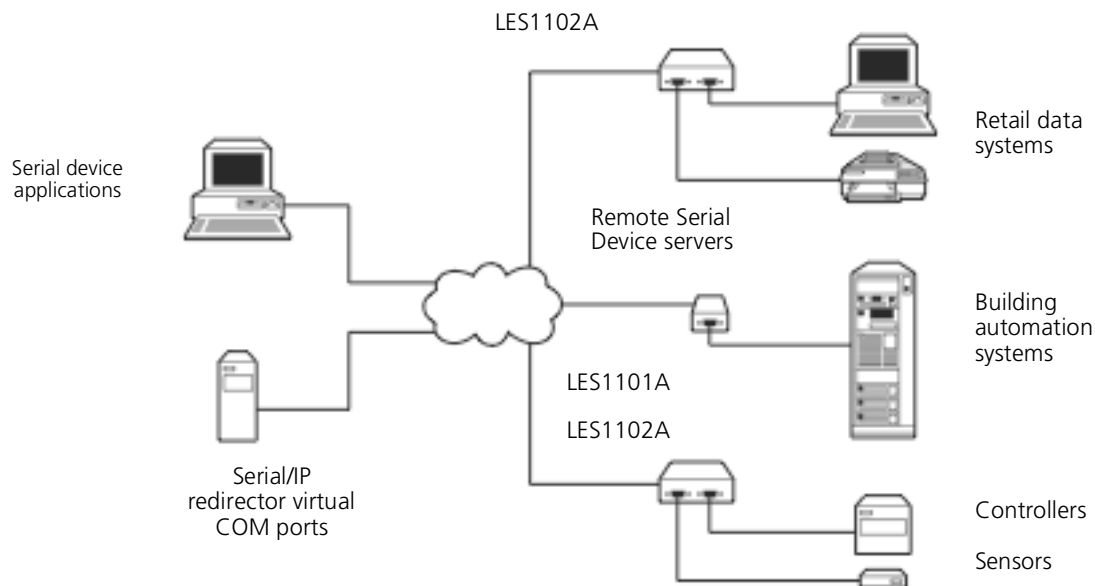
Figure 5-19. Serial Port redirection.

This serial port redirector software is loaded in your desktop PC, and it allows you to use a serial device that's connected to the remote *console server* as if it were connected to your local serial port.

5.7 Managed Devices

Managed Devices presents a consolidated view of all the connections to a device that you can access and monitor through the *console server*. To view the connections to the devices:

Select Serial & Network: Managed Devices.
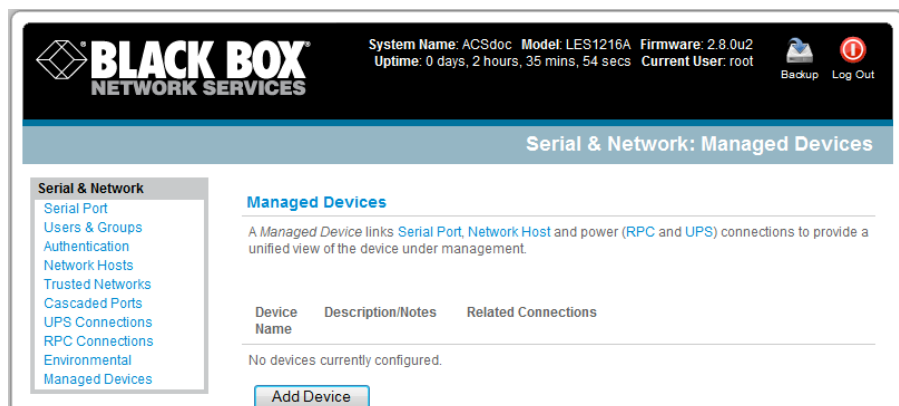
Figure 5-20. Managed Devices screen.

This screen displays all the Managed Devices with their Description/Notes. It also lists all the configured Connections, that is, Serial Port # (if serially connected) or USB if USB connected; IP Address (if network connected); Power PDU/outlet details (if applicable), and any UPS connections. Devices such as servers will commonly have more than one power connection (for example, dual power supplied) and more than one network connection (for example, for BMC/service processor).

All *Users* can view (but not edit) these Managed Device connections by selecting Manage: Devices. The *Administrator* user can edit and add/delete these Managed Devices and their connections.

To edit an existing device and add a new connection:

• Select Edit on the Serial & Network: Managed Devices and click Add Connection.

• Select the connection type for the new connection (Serial, Network Host, UPS, or RPC) and then select the specific connection from the presented list of configured unallocated hosts/ports/outlets.

To add a new network-connected Managed Device:

The *Administrator* adds a new network-connected Managed Device using Add Host on the Serial & Network: Network Host menu. This automatically creates a corresponding new Managed Device (as covered in *Section 5.4—Network Hosts).*

When adding a new network-connected RPC or UPS power device, you set up a Network Host, designate it as RPC or UPS, then go to RPC Connections (or UPS Connections) to configure the relevant connection. A corresponding new Managed Device (with the same Name /Description as the RPC/UPS Host) is not created until you complete this connection step (refer *Chapter 8—Power Management).*

NOTE: The outlet names on this newly created PDU will by default be "Outlet 1" and "Outlet 2." When you connect a particular Managed Device (that draws power from the outlet), then the outlet will take the powered Managed Device's name.

To add a new serially connected Managed Device:

• Configure the serial port using the Serial & Network: Serial Port menu (refer to *Section 5.1—Configure Serial Port).*

• Select Serial & Network: Managed Devices and click Add Device.

• Enter a Device Name and Description for the Managed Device.

Figure 5-21. Add a new device screen.

• Click Add Connection and select Serial and the Port that connects to the Managed Device.

• To add a UPS/RPC power connection or network connection or another serial connection, click Add Connection.

• Click Apply.

NOTE: To set up a new serially connected RPC UPS or EMD device, configure the serial port, designate it as a Device, then enter a Name and Description for that device in the Serial & Network: RPC Connections (or UPS Connections or Environmental). When applied, this will automatically create a corresponding new Managed Device with the same Name /Description as the RPC/UPS Host (refer to *Chapter 8—Power Management).*

All the outlet names on the PDU will by default be "Outlet 1" and "Outlet 2." When you connect a particular Managed Device (that draws power from the outlet), then the outlet will take the name of the powered Managed Device.

6. Secure SSH Tunneling and SDT Connector

Each Black Box console server has an embedded SSH server and uses SSH tunneling so remote users can securely connect through the console server to Managed Devices—using text-based console tools (such as SSH, telnet, SoL) or graphical tools (such VNC, RDP, HTTPS, HTTP, X11, VMware, DRAC, iLO).

The Managed Devices you access can be located on the same local network as the console server or they can be attached to the console server via a serial port. The remote User/Administrator connects to the console server thru an SSH tunnel via dial-up, wireless or ISDN modem; a broadband Internet connection; the enterprise VPN network; or the local network.

Secure Local
Management

Secure Remote
Management

LES1102A

Serial
connected
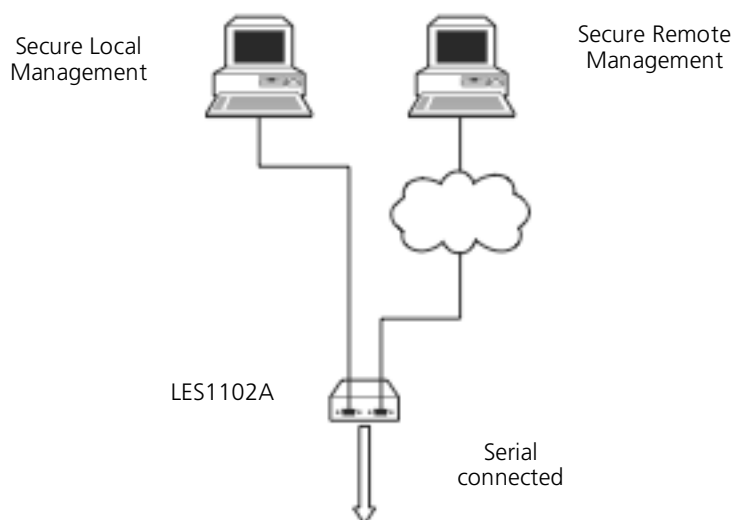
Figure 6-1. Secure network.

To set up the secure SSH tunnel from the client PC to the console server, install and launch SSH client software on the User/Administrator's PC, Black Box recommends that you use the SDT Connector client software supplied with the console server. SDT Connector is simple to install and auto-configure and it provides all your users with point-and-click access to all the systems and devices in the secure network. With one click, SDT Connector sets up a secure SSH tunnel from the client to the selected console server, then establishes a port forward connection to the target network connected host or serial connected device. Next, it executes the client application that it uses in communicating with the host.

This chapter details the basic SDT Connector operations:

• Configuring the console server for SSH tunneled access to network attached hosts and setting up permitted Services and user access (Section 6.1).

• Setting up the SDT Connector client with gateway, host, service, and client application details, and making connections between the Client PC and hosts connected to the console server (Section 6.2).

• Using SDT Connector to access the Management Console via a browser (Section 6.3).

• Using SDT Connector to Telnet or SSH connect to devices that are serially attached to the console server (Section 6.4).

The chapter then covers more advanced SDT Connector and SSH tunneling topics:

• Using SDT Connector for out-of-band access (Section 6.5).

• Automatic importing and exporting configurations (Section 6.6).

• Configuring Public Key Authentication (Section 6.7).

• Setting up a SDT Secure Tunnel for Remote Desktop (Section 6.8).

• Setting up a SDT Secure Tunnel for VNC (Section 6.9).

• Using SDT to IP connect to hosts that are serially attached to the console server (Section 6.10).

6.1 Configuring for SSH Tunneling to Hosts

To set up the console server to SSH tunnel to access a network attached host:

Add the new host and the permitted services using the Serial & Network: Network Hosts menu as detailed in Network Hosts (Chapter 5.4). Only these permitted services will be forwarded through by SSH to the host. All other services (TCP/UDP ports) will be blocked.

NOTE: Following are some of the TCP Ports used by SDT in the console server:
   22   SSH (All SDT Tunneled connections)
   23   Telnet on local LAN (forwarded inside tunnel)
   80   HTTP on local LAN (forwarded inside tunnel)
   3389 RDP on local LAN (forwarded inside tunnel)
   5900 VNC on local LAN (forwarded inside tunnel)
   73XX RDP over serial from local LAN – where XX is the serial port number (that is, 7301 to 7302 on a 2-port console server)
   79XX VNC over serial from local LAN – where XX is the serial port number

Add the new Users using Serial & Network: Users & Groups menu as detailed in Network Hosts (Chapter 5.4). Users can be authorized to access the console server ports and specified network attached hosts. To simplify configuration, the Administrator can first set up Groups with group access permissions, then Users can be classified as members of particular Groups.

6.2 SDT Connector Client Configuration

The SDT Connector client works with all Black Box console servers. Each of these remote console servers has an embedded OpenSSH based server that you can configure to port forward connections from the SDT Connector client to hosts on their local network (as detailed in the previous chapter). You can also pre-configure the SDT Connector with the access tools and applications that are available to run when you've established access to a particular host.

SDT Connector can connect to the console server using an alternate OoB access. It can also access the console server itself and access devices connected to serial ports on the console server.

6.2.1 SDT Connector installation

The SDT Connector set up program (SDTConnector Setup-1.n.exe or sdtcon-1.n.tar.gz) is included on the CD supplied with your Black Box console server.

Run the set-up program.

Figure 6-2. SDT connector setup window.

NOTE: For Windows clients, the SDTConnectorSetup-1.n.exe application will install the SDT Connector 1.n.exe and the config file defaults.xml. If there is already a config file on the Windows PC, then it will not be overwritten. To remove an earlier config file, run the regedit command and search for "SDT Connector," then remove the directory with this name.

 For Linux and other Unix clients, SDTConnector.tar.gz application will install the sdtcon-1.n.jar and the config file defaults.xml.

1101 and 1102 Secure Device Servers

Once the installer completes you will have a working SDT Connector client installed on your machine and an icon on your desktop:

Figure 6-3. SDT connector icon.

Click the SDT Connector icon on your desktop to start the client.

NOTE: SDT Connector is a Java application, so it must have a Java Runtime Environment (JRE) installed. You can download this for free from http://java.sun.com/j2se/. It installs on Windows 2000, XP, 2003, Vista, and 7 PCs and on most Linux platforms. Solaris platforms are also supported, but they must have Firefox installed. SDT Connector can run on any system with Java 1.4.2 and above installed, but it assumes the web browser is Firefox, and that xterm -e telnet opens a telnet window.

To operate SDT Connector, you first need to add new gateways to the client software by entering the access details for each console server (refer to Section 6.2.2). Then, let the client auto-configure all host and serial port connections from each console server (refer to Section 6.2.3). Finally, point-and-click to connect to the Hosts and serial devices (refer to Section 6.2.4).

Or, you can manually add network connected hosts (refer to Section 6.2.5) and manually configure new services to use to access the console server and the hosts (refer to Section 6.2.6). Then, manually configure clients to run on the PC that will use the service to connect to the hosts and serial port devices (refer to Section 6.2.).

6.2.2 Configuring a New Console Server Gateway in the SDT Connector Client

To create a secure SSH tunnel to a new console server:

Click the New Gateway icon or select the File: New Gateway menu option.
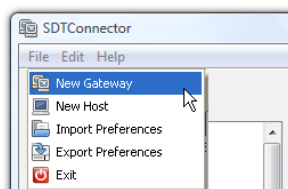
Figure 6-4. New Gateway menu option.

Enter the IP or DNS Address of the console server and the SSH port that you will use (typically 22).

NOTE: If SDT Connector is connecting to a remote console server through the public Internet or routed network you will need to:

• Determine the public IP address of the console server (or of the router/ firewall that connects the console server to the Internet) as assigned by the ISP. One way to find the public IP address is to access http://checkip.dyndns.org/ or http://www.whatismyip.com/ from a computer on the same network as the console server and note the reported IP address.

• Set port forwarding for TCP port 22 through any firewall/NAT/router that is located between SDT Connector and the console server so it points to the console server. http://www.portforward.com has port forwarding instructions for a range of routers. Also, you can use the Open Port Check tool from http://www.canyouseeme.org to check if port forwarding through local firewall/NAT/router devices has been properly configured.

Enter the Username and Password of a user on the gateway that is enabled to connect via SSH and/or create SSH port redirections.

724-746-5500 | blackbox.com
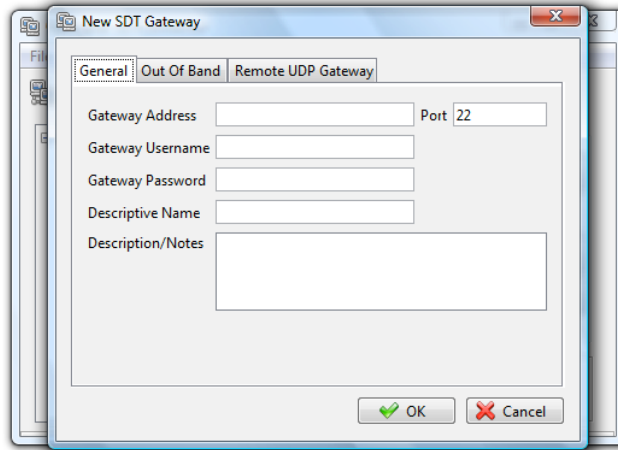
Figure 6-5. New SDT Gateway screen.

Or, enter a Descriptive Name to display instead of the IP or DNS address, and any Notes or a Description of this gateway (such as its firmware version, site location, or anything special about its network configuration).

Click OK and an icon for the new gateway will now appear in the SDT Connector home page.

NOTE: For an SDT Connector user to access a console server (and then access specific hosts or serial devices connected to that console server), that user must first be setup on the console server, and must be authorized to access the specific ports/hosts (refer to Chapter 6). Only these permitted services will be forwarded through by SSH to the Host. All other services (TCP/UDP ports) will be blocked.

6.2.3 Auto-configure SDT Connector Client with the User's Access Privileges

Each user on the console server has an access profile that was configured with those specific connected hosts and serial port devices the user has authority to access, and a specific set of the enabled services for each of these. You can upload this configuration automatically into the SDT Connector client:
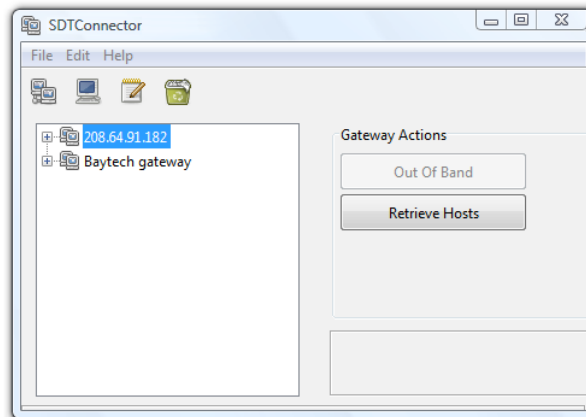
Figure 6-6. Gateway actions.

Click on the new gateway icon and select Retrieve Hosts. This will:

• configure access to network connected Hosts that the user is authorized to access and set up (for each of these Hosts) the services (for example, HTTPS, IPMI2.0) and the related IP ports being redirected.

• configure access to the console server itself (this is shown as a Local Services host).

• configure access with the enabled services for the serial port devices connected to the console server.
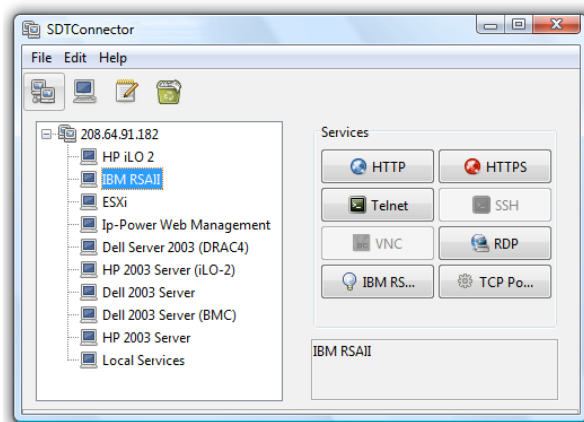
Figure 6-7. Hosts.

NOTE: The Retrieve Hosts function will auto-configure all user classes (that is, they can be members of user or admin or some other group or no group. SDT Connector will not auto-configure the root (and we recommend that you only use this account for initial config and to add an initial admin account to the console server).

6.2.4 Make an SDT Connection through the Gateway to a Host

Simply point at the host to be accessed and click on the service to use to access that host. The SSH tunnel to the gateway is then automatically established, the appropriate ports redirected through to the host, and the appropriate local client application is launched pointing at the local endpoint of the redirection.
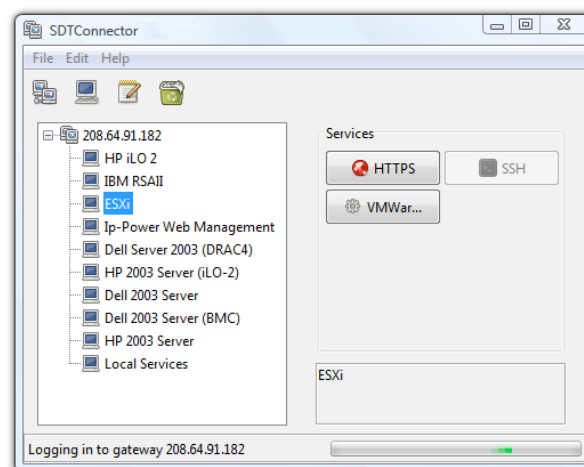
Figure 6-8. Services.

NOTE: You can configure the SDT Connector client with an unlimited number of Gateways (that is, console servers). You can configure each Gateway to port forward to an unlimited number of locally networked Hosts. There is no limit on the number of SDT Connector clients that can be configured to access the one Gateway. Nor are there limits on the number of Host connections that an SDT Connector client can concurrently have open through the one Gateway tunnel.

There is a limit on the number of SDT Connector SSH tunnels that can be open at the same time on a particular Gateway (console server). Each Gateway (console server) can support at least 50 such concurrent connections. At any time, you could have up to 50 users securely controlling an unlimited number of Managed Devices at a remote site through the on-site console server Gateway.

6.2.5 Manually Adding Hosts to the SDT Connector Gateway

For each gateway, you can manually specify the network connected hosts that you will access through that console server; and for each host, specify the services that you will use to communicate with the host.

Select the newly added gateway and click the Host icon to create a host that will be accessible via this gateway. (Alternatively, select File: New Host).
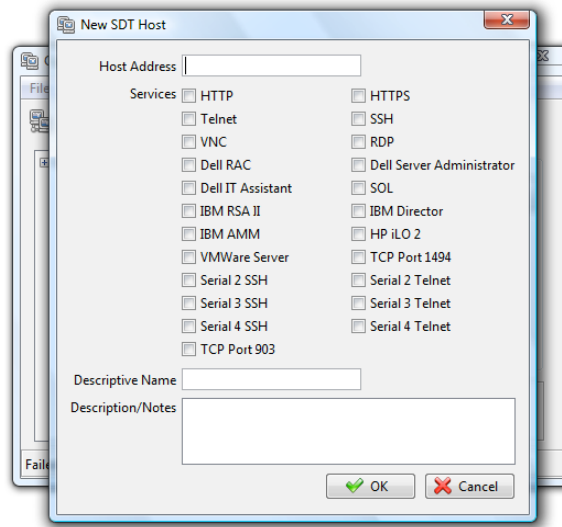
Figure 6-9. New SDT Host screen.

Enter the IP or DNS Host Address of the host (if this is a DNS address, it must be able to be resolved by the gateway).

Select which Services to use to access the new host. A range of service options are pre-configured in the default SDT Connector client (RDP, VNC, HTTP, HTTPS, Dell RAC, VMware, etc.). If you want to add new services to the range, then proceed to the next section (Adding a new service) then return here.

Or, enter a Descriptive Name for the host to display instead of the IP or DNS address, and any Notes or a Description of this host (such as its operating system/release, or anything special about its configuration).

Click OK.

6.2.6 Manually Adding New Services to the New Hosts

To extend the range of services that you can use when accessing hosts with SDT Connector:

Select Edit: Preferences and click the Services tab. Click Add.

Enter a Service Name and click Add.

Under the General tab, enter the TCP Port that this service runs on (for example, 80 for HTTP). Or, select the client to use to access the local endpoint of the redirection.
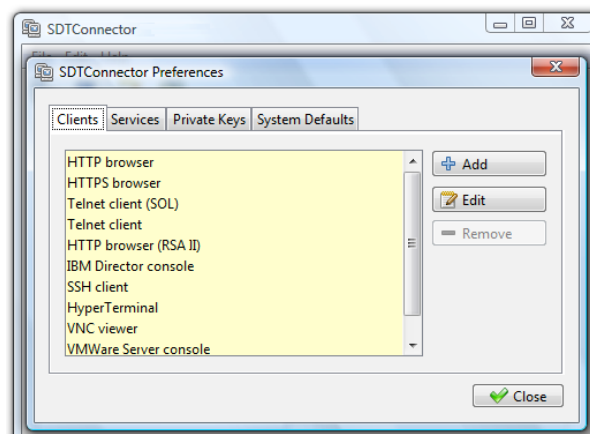
Figure 6-10. Clients.

1101 and 1102 Secure Device Servers

Select which Client application is associated with the new service. A range of client application options are pre-configured in the default SDT Connector (RDP client, VNC client, HTTP browser, HTTPS browser, Telnet client, etc.). If you want to add new client applications to this range, proceed to the next section (Adding a new client), then return here.
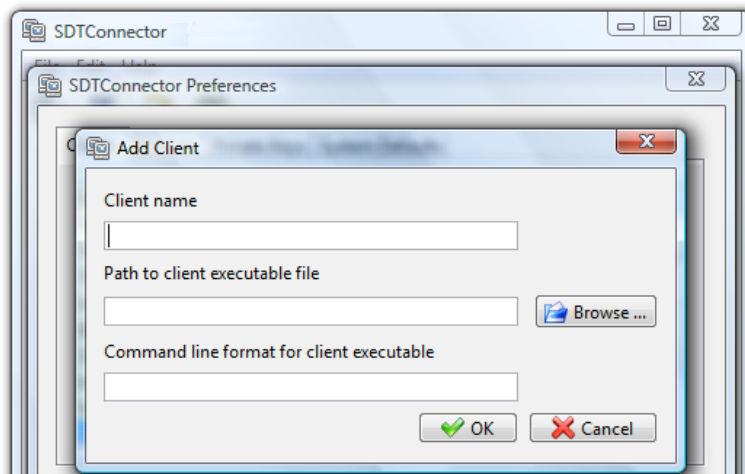
Figure 6-11. Select client.

Click OK, then Close.

A service typically consists of a single SSH port redirection and a local client to access it. It may consist of several redirections, and some or all may have clients associated with them.

An example is the Dell RAC service. The first redirection is for the HTTPS connection to the RAC server— it has a client associated with it (web browser) that it launches immediately when you click the button for this service.

The second redirection is for the VNC service that you may choose to later launch from the RAC web console. It automatically loads in a Java client served through the web browser, so it does not need to have a local client associated with it.
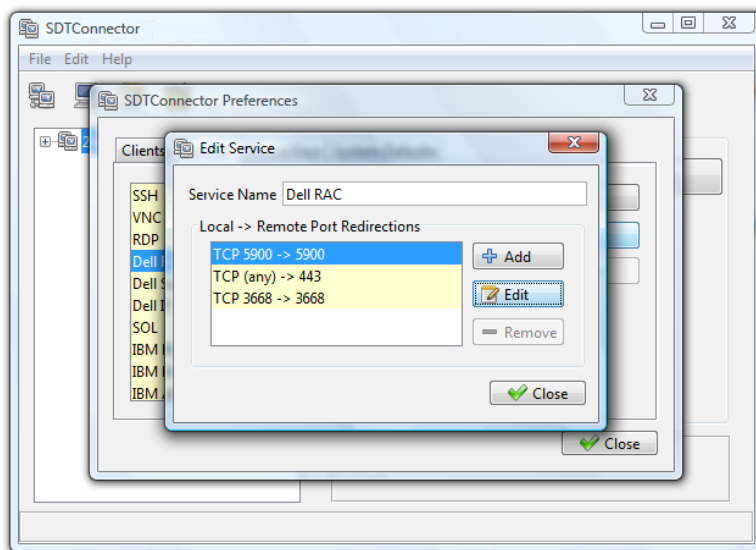
Figure 6-12. Edit Service.

On the Add Service screen, you can click Add as many times as needed to add multiple new port redirections and associated clients. You may also specify Advanced port redirection options:

Enter the local address to bind to when creating the local endpoint of the redirection. It is not usually necessary to change this from "localhost."

Enter a local TCP port to bind to when creating the local endpoint of the redirection. If you leave this blank, a random port is selected.

724-746-5500 | blackbox.com
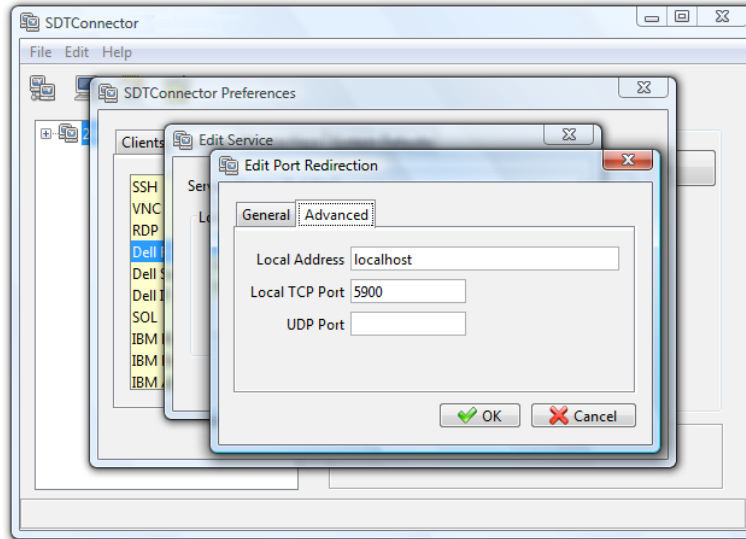
Figure 6-13. Edit port redirection.

NOTES: SDT Connector can also tunnel UDP services. SDT Connector tunnels the UDP traffic through the TCP SSH redirection, so it is a "tunnel within a tunnel."

Enter the UDP port where the service is running on the host. This will also be the local UDP port that SDT Connector binds as the local endpoint of the tunnel.

For UDP services, you still need to specify a TCP port under General. This will be an arbitrary TCP port that is not in use on the gateway. An example of this is the SOL Proxy service. It redirects local UDP port 623 to remote UDP port 623 over the arbitrary TCP port 6667.

6.2.7 Adding a Client Program to be Started for the New Service

Clients are local applications that you may launch when a related service is clicked. To add to the pool of client programs:
Select Edit: Preferences and click the Client tab. Click Add.
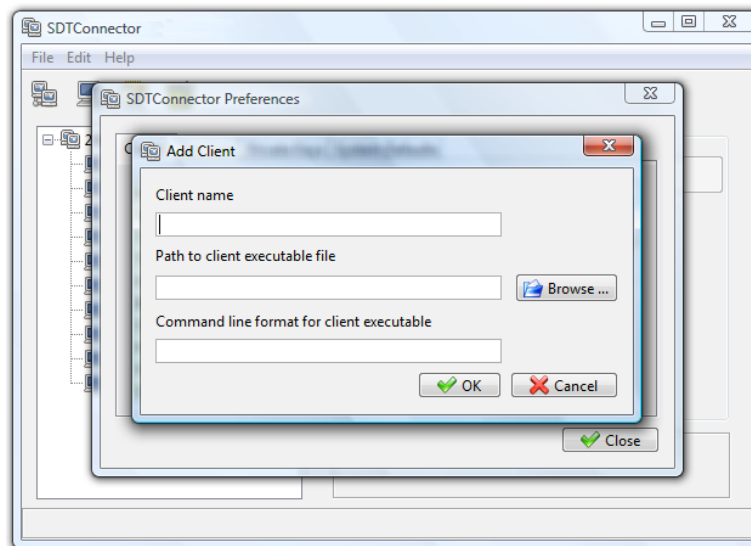
Figure 6-14. Add Client screen.

Enter a Name for the client. Enter the Path to the executable file for the client (or click Browse to locate the executable).

1101 and 1102 Secure Device Servers

Enter a Command Line associated with launching the client application. SDT Connector typically launches a client using command line arguments to point it at the local endpoint of the redirection. There are three special keywords for specifying the command line format. When launching the client, SDT Connector substitutes these keywords with the appropriate values:

%path% is path to the executable file, that is, the previous field.

%host% is the local address to which the local endpoint of the redirection is bound, that is, the Local Address field for the Service redirection Advanced options.

%port% is the local port to which the local endpoint of the redirection is bound, that is, the Local TCP Port field for the Service redirection Advanced options. If this port is unspecified (that is, "Any"), the appropriate randomly selected port will be substituted.

For example SDT Connector is preconfigured for Windows installations with a HTTP service client that will connect with the local browser that the local Windows user has configured as the default. Otherwise, the default browser used is Firefox.
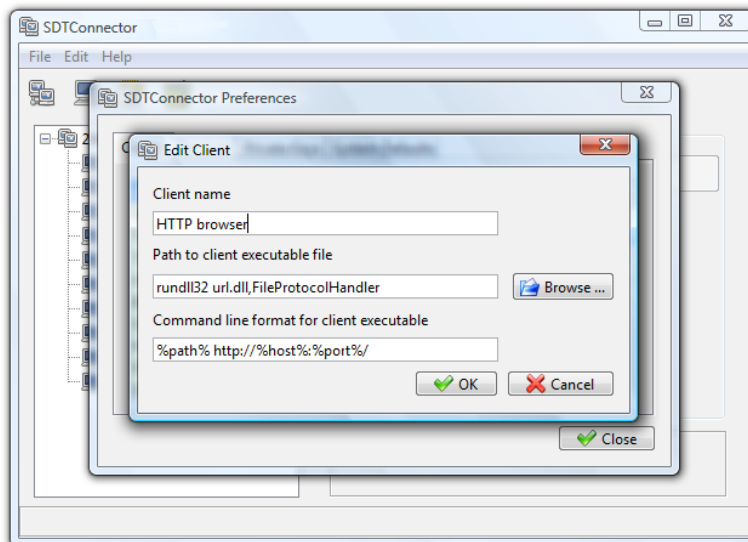
Figure 6-15. HTTP browser client.

Also some clients are launched in a command line or terminal window. The Telnet client is an example of this so the "Path to client executable file" is telnet and the "Command line format for client executable" is cmd /c start %path% %host% %port%.
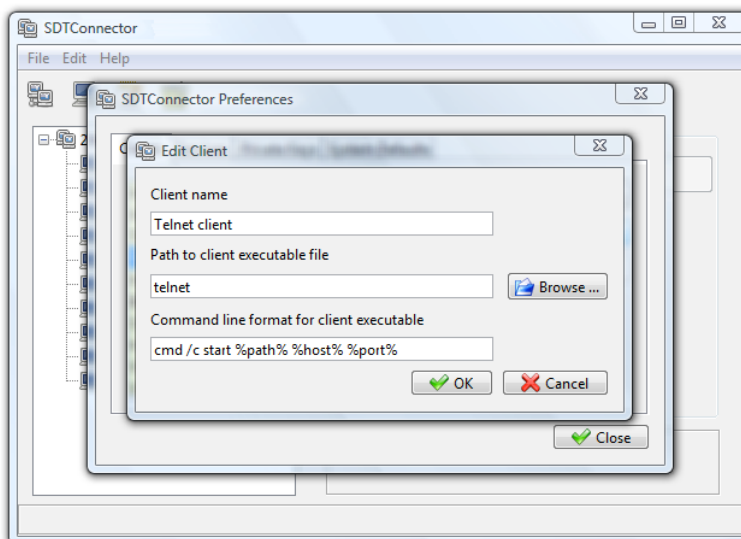
Figure 6-16. Telnet client.

Click OK.

724-746-5500 | blackbox.com

6.3 SDT Connector to Management Console

You can also configure SDT Connector for browser access to the console server's Management Console —and for Telnet or SSH access to the command line. For these connections to the console server itself, you must configure SDT Connector to access the Gateway itself by setting the Gateway (console server) up as a host, and then configuring the appropriate services:

Launch SDT Connector on your PC. Assuming you have already set up the console server as a Gateway in your SDT Connector client (with username/ password etc.), select this newly added Gateway and click the Host icon to create a host. Or, select File -> New Host.

Enter 127.0.0.1 as the Host Address and provide details in Descriptive Name/Notes. Click OK.
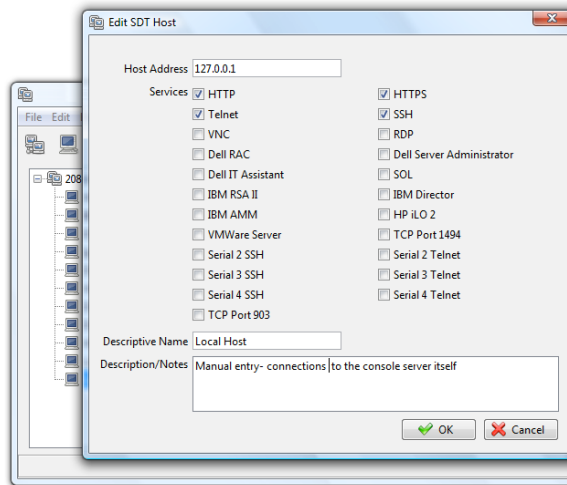
Figure 6-17. Edit SDT host screen.

Click the HTTP or HTTPS Services icon to access the Management Console, and/or click SSH or Telnet to access the command line console.

NOTES: To enable SDT access to the console, you must also configure the console server to allow the port forwarded network access to itself:

Browse to the console server and select Network Hosts from Serial & Network, click Add Host, and in the IP Address/DNS Name field enter 127.0.0.1 (this is the Black Box network loopback address). Then, enter Loopback in Description.

Remove all entries under Permitted Services except for those that you will use to access the Management Console (80/http or 443/https) or the command line (22/ssh or 23/telnet). Scroll to the bottom and click Apply.

Administrators by default have gateway access privileges. For Users to access the console server Management Console, you will need to give those Users the required access privileges. Select Users & Groups from Serial & Network. Click Add User. Enter a Username, Description and Password/Confirm. Select 127.0.0.1 from Accessible Host(s) and click Apply.

6.4 SDT Connector—Telnet or SSH Connect to Serially Attached Devices

You can also use SDT Connector to access text consoles on devices that are attached to the console server serial ports. For these connections, you must configure the SDT Connector client software with a Service that will access the target gateway serial port, and then set the gateway up as a host:

Launch SDT Connector on your PC. Select Edit -> Preferences and click the Services tab. Click Add.

Enter "Serial Port 2" in Service Name and click Add.

Select Telnet client as the Client. Enter 2002 in TCP Port. Click OK, then Close and Close again.
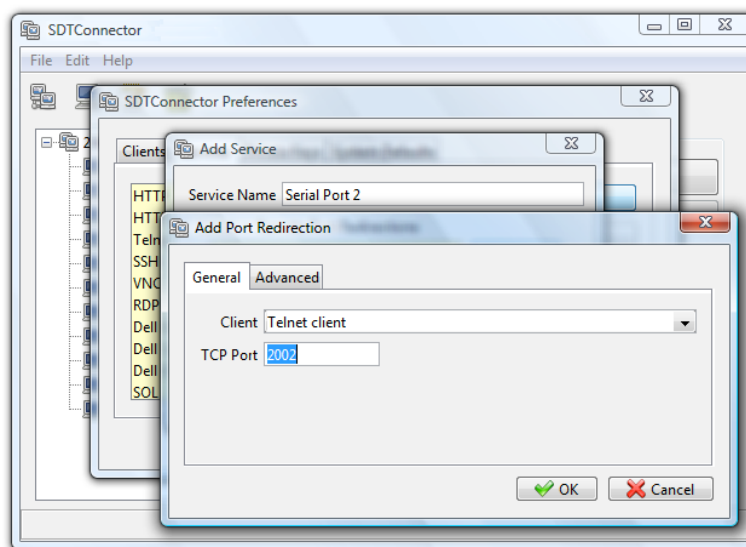
Figure 6-18. Add port redirection.

Assuming you have already set up the target console server as a gateway in your SDT Connector client (with username/password etc), select this gateway and click the Host icon to create a host. Or, select File -> New Host.

Enter 127.0.0.1 as the Host Address and use Serial Port 2 for descriptive name. In Description/notes, enter something such as Loopback ports, or Local serial ports. Click OK.

Click Serial Port 2 icon for Telnet access to the serial console on the device attached to serial port #2 on the gateway.

To enable SDT Connector to access the devices connected to the gateway's serial ports, you must also configure the Console server itself to allow port forwarded network access to itself, and enable access to the nominated serial port:

Browse to the Console server and select Serial Port from Serial & Network.

Click Edit next to selected Port # (for example, Port 2 if the target device is attached to the second serial port). Make sure the port's serial configuration is appropriate for the attached device.

Scroll down to Console server Setting and select Console server Mode. Check Telnet (or SSH) and scroll to the bottom and click Apply.

Select Network Hosts from Serial & Network and click Add Host.

In the IP Address/DNS Name field enter 127.0.0.1 (this is the Black Box network loopback address) and enter Loopback in Description.

Remove all entries under Permitted Services, select TCP, and enter 200n in Port. (This configures the Telnet port enabled in the previous step, so for Port 2 you would enter 2002.)

Click Add, then scroll to the bottom and click Apply.

Administrators by default have gateway and serial port access privileges; however for Users to access the gateway and the serial port, you will need to give those Users the required access privileges. Select Users & Groups from Serial & Network. Click Add User. Enter a Username, Description, and Password/Confirm. Select 127.0.0.1 from Accessible Host(s) and select Port 2 from Accessible Port(s). Click Apply.

6.5 Using SDT Connector for Out-of-Band Connection to the Gateway

You can also set up SDT Connector to connect to the console server (gateway) out-of-band (OoB). OoB access uses an alternate path for connecting to the gateway to that used for regular data traffic. OoB access is useful for when the primary link into the gateway is unavailable or unreliable. Typically, a gateway's primary link is a broadband Internet connection or Internet connection via a LAN or VPN, and the secondary out-of-band connectivity is provided by a dial-up or wireless modem directly attached to the gateway. Out-of-band access enables you to access the hosts and serial devices on the network, diagnose any connectivity issues, and restore the gateway's primary link.

In SDT Connector, to configure OoB access, you provide the secondary IP address of the gateway, and tell SDT Connector how to start and stop the OoB connection. You can start an OoB connection by initiating a dial up connection, or adding an alternate route to the gateway. SDT Connector allows for maximum flexibility. It allows you to provide your own scripts or commands for starting and stopping the OoB connection.
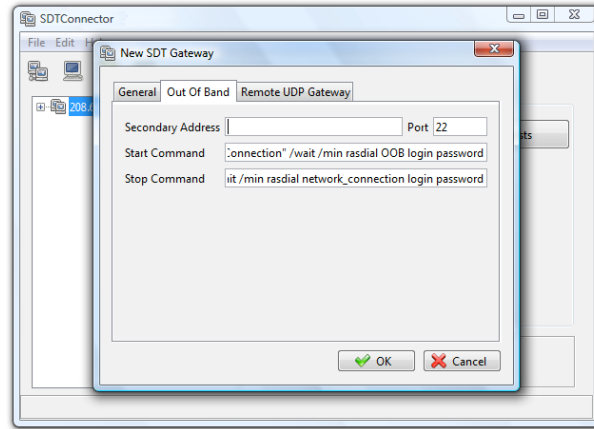
Figure 6-19. Out-of-band access.

To configure SDT Connector for OoB access:

When adding a new Gateway or editing an existing Gateway select the Out Of Band tab.

Enter the secondary, OoB IP address of the gateway (for example, the IP address it is using when dialed in directly). You also may modify the gateway's SSH port if it's not using the default of 22.

Enter the command or path to a script to start the OoB connection in Start Command.

To initiate a pre-configured dial-up connection under Windows, use the following Start Command:

cmd /c start "Starting Out of Band Connection" /wait /min rasdial network_connection login password

where network_connection is the name of the network connection as displayed in Control Panel -> Network Connections, login is the dial-in username, and password is the dial-in password for the connection.

To initiate a pre-configured dial-up connection under Linux, use the following Start Command:

pon network_connection

where network_connection is the name of the connection.

Enter the command or path to a script to stop the OoB connection in Stop Command.

To stop a pre-configured dial-up connection under Windows, use the following Stop Command:

cmd /c start "Stopping Out of Band Connection" /wait /min rasdial network_connection /disconnect

where network connection is the name of the network connection as displayed in Control Panel -> Network Connections.

To stop a pre-configured dial-up connection under Linux, use the following Stop Command:
poff network_connection

To make the OoB connection using SDT Connector:

Select the console server and click Out Of Band. The status bar will change color to indicate that this console server is now accessed using the OoB link rather than the primary link.
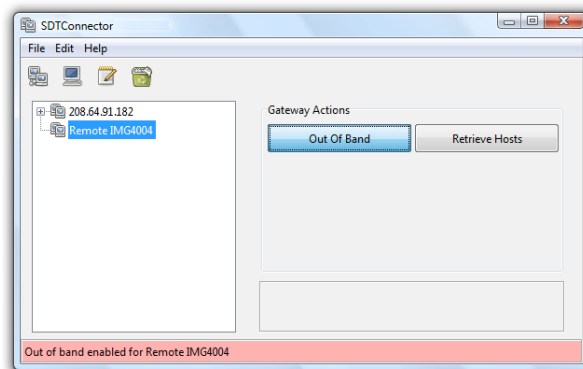
Figure 6-20. OoB connection using SDT connector.

When you connect to a service on a host behind the console server, or to the console server itself, SDT Connector will initiate the OoB connection using the provided Start Command. The OoB connection does not stop (using the provided Stop Command) until you click off Out Of Band under Gateway Actions; then the status bar will return to its normal color.

6.6 Importing (and Exporting) Preferences

To enable the distribution of pre-configured client config files, SDT Connector has an Export/Import facility:

• To save a configuration.xml file (for backup or for importing into other SDT Connector clients), select File -> Export Preferences and select the location where you want to save the configuration file.
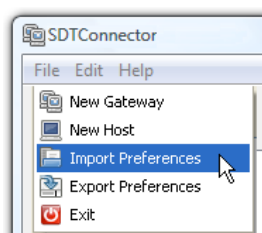
Figure 6-21. Import preferences option.

• To import a configuration, select File -> Import Preferences and select the .xml configuration file to install.

6.7 SDT Connector Public Key Authentication

SDT Connector can authenticate against an SSH gateway using your SSH key pair instead of requiring you to enter your password. This is known as public key authentication.

To use public key authentication with SDT Connector, first you must add the public part of your SSH key pair to your SSH gateway:

Make sure the SSH gateway allows public key authentication, this is typically the default behavior.

If you do not already have a public/private key pair for your client PC (the one running SDT Connector), generate them now using ssh-keygen, PuTTYgen or a similar tool. You may use RSA or DSA; however, leave the passphrase field blank:

PuTTYgen: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

OpenSSH: http://www.openssh.org/
OpenSSH (Windows): http://sshwindows.sourceforge.net/download/

Upload the public part of your SSH key pair (this file is typically named id_rsa.pub or id_dsa.pub) to the SSH gateway, or otherwise add to .ssh/authorized keys in your home directory on the SSH gateway.

Next, add the private part of your SSH key pair (this file is typically named id_rsa or id_dsa) to SDT Connector. Click Edit -> Preferences -> Private Keys -> Add, locate the private key file, and click OK.

You do not have to add the public part of your SSH key pair, the private key calculates it.

724-746-5500 | blackbox.com

SDT Connector will now use public key authentication when connecting through the SSH gateway (console server). You may have to restart SDT Connector to shut down any existing tunnels that were established using password authentication.

If you have a host behind the console server that you connect to by clicking the SSH button in SDT Connector, you may also want to configure access to it for public key authentication as well. This configuration is entirely independent of SDT Connector and the SSH gateway. You must configure the SSH client that SDT Connector launches (for example, Putty, OpenSSH) and the host's SSH server for public key authentication. Essentially what you are using is SSH over SSH, and the two SSH connections are entirely separate.

6.8 Setting up SDT for Remote Desktop Access

The Microsoft Remote Desktop Protocol (RDP) enables the system manager to securely access and manage remote Windows computers—to reconfigure applications and user profiles, upgrade the server's operating system, reboot the machine, etc. Black Box's Secure Tunneling uses SSH tunneling, so this RDP traffic is securely transferred through an authenticated and encrypted tunnel.

SDT with RDP also allows remote Users to connect to Windows XP, Vista, Server2003, and Server 2008 computers and to Windows 2000 Terminal Servers; and to access to all of the applications, files, and network resources (with full graphical interface just as though they were in front of the computer screen at work). To set up a secure Remote Desktop connection, enable Remote Desktop on the target Windows computer that you want to access and configure the RPD client software on the client PC.

6.8.1 Enable Remote Desktop on the Target Windows Computer to be Accessed

To enable Remote Desktop on the Windows computer being accessed:

Open System in the Control Panel and click the Remote tab.
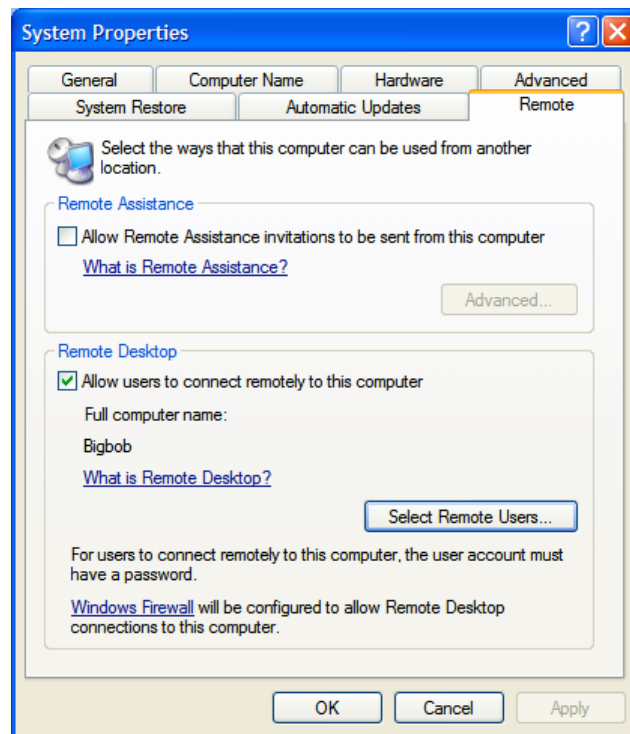
Figure 6-22. System Properties screen.

Check Allow users to connect remotely to this computer.

Click Select Remote Users.
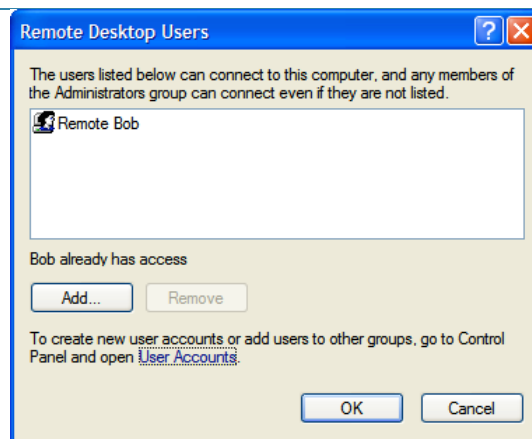
Figure 6-23. Remote Desktop Users dialog box.

To set the user(s) who can remotely access the system with RDP, click Add on the Remote Desktop Users dialog box.

NOTE: If you need to set up new users for Remote Desktop access, open User Accounts in the Control Panel and follow the steps to nominate the new user's name, password, and account type (Administrator or Limited).

NOTE: With Windows XP Professional and Vista, you have only one Remote Desktop session and it connects directly to the Windows root console. With Windows Server 2008, you can have multiple sessions (and with Server 2003 you have three sessions— the console session and two other general sessions). More than one user can have active sessions on a single computer.

When the remote user connects to the accessed computer on the console session, Remote Desktop automatically locks that computer (no other user can access the applications and files). When you come back to your computer at work, you can unlock it by typing CTRL+ALT+DEL.

6.8.2 Configure the Remote Desktop Connection Client

Now that you have the Client PC securely connected to the console server (either locally, or remotely—through the enterprise VPN, or a secure SSH internet tunnel, or a dial-in SSH tunnel), you can establish the Remote Desktop connection from the Client. Simply enable the Remote Desktop Connection on the remote client PC, then point it to the SDT Secure Tunnel port in the console server:

On a Windows client PC:

Click Start. Point to Programs, then to Accessories, then Communications, and click Remote Desktop Connection.

Figure 6-24. Remote Desktop Connection.

In Computer, enter the appropriate IP Address and Port Number:

Where there is a direct local or enterprise VPN connection, enter the IP Address of the console server, and the Port Number of the SDT Secure Tunnel for the console server serial port that you attach to the Windows computer you want to control. For example, if the Windows computer is connected to serial Port 2 on a console server located at 192.168.0.50, then you would enter 192.168.0.50:7302.

Where there is an SSH tunnel (over a dial up PPP connection or over a public internet connection or private network connection), simply enter the localhost as the IP address, 127.0.0.1. For Port Number, enter the source port you created when setting SSH tunneling /port forwarding (in Section 6.1), for example:1234.

Click Option. In the Display section, specify an appropriate color depth (for example, for a modem connection we recommend that you not use over 256 colors). In Local Resources, specify the peripherals on the remote Windows computer that are to be controlled (printer, serial port, etc.).
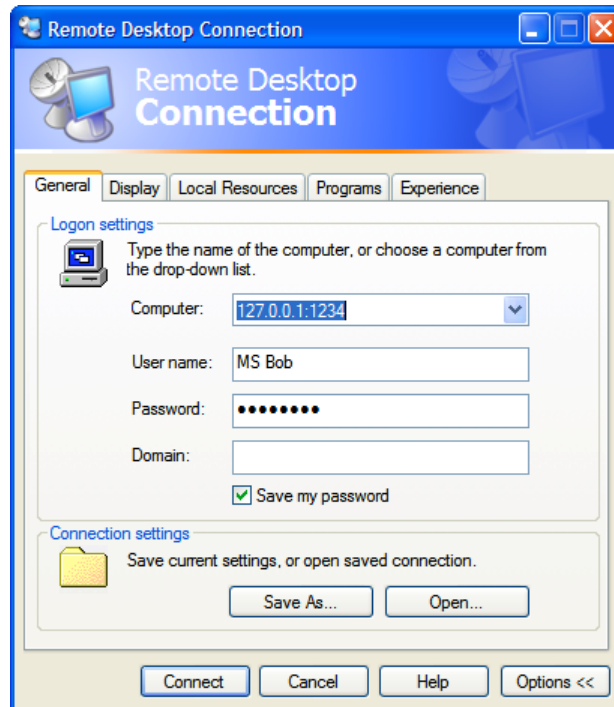
Figure 6-25. Remote Desktop Connection, General tab.

Click Connect.

NOTES: The Remote Desktop Connection software is pre-installed with Windows XP, Vista, and Server 2003/2008. For earlier Windows PCs, you need to download the RDP client:

Go to the Microsoft Download Center site   http://www.microsoft.com/downloads/details.aspx?familyid=80111F21-D48D-426E-96C2-08AA2BD23A49&displaylang=en and click the Download button

This software package will install the client portion of Remote Desktop on Windows 95, Windows 98 and 98 Second Edition, Windows Me, Windows NT 4.0, and Windows 2000. When run, this software allows these older Windows platforms to remotely connect to a computer running current Windows.

On a Linux or UNIX client PC:

Launch the open source rdesktop client:

rdesktop -u windows-user-id -p windows-password -g 1200x950 ms-windows-terminal-server-host-name

Option	Description
-a	Color depth: 8, 16, 24
-r	Device redirection. (Redirect sound on remote machine to local device. -0 –r sound (MS/Windows 2003)
-g	Geometry: width x height or 70% screen percentage
-p	Use –p to receive password prompt

1101 and 1102 Secure Device Servers

You can use GUI front end tools like the GNOME Terminal Services Client tsclient to configure and launch the rdesktop client. (Using tsclient also enables you to store multiple configurations of rdesktop for connection to many servers.)
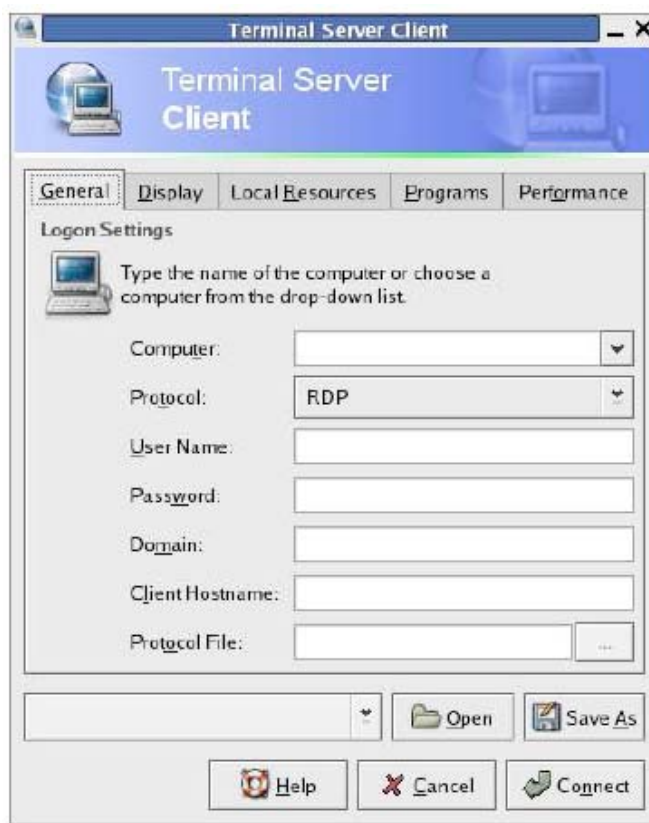
Figure 6-26. RDP protocol.

NOTE: The rdesktop client is supplied with Red Hat 9.0:
      rpm -ivh rdesktop-1.2.0-1.i386.rpm

For Red Hat 8.0 or other distributions of Linux; download source, untar, configure, make, make, then install.

rdesktop currently runs on most UNIX based platforms with the X Window System and can be downloaded from http://www.rdesktop.org/

On a Macintosh client:

Download Microsoft's free Remote Desktop Connection client for Mac OS X

http://www.microsoft.com/mac/otherproducts/otherproducts.aspx?pid=remotedesktopclientSDT SSH Tunnel for VNC

6.9 SDT SSH Tunnel for VNC

With SDT and Virtual Network Computing (VNC), Users, and Administrators can securely access and control Windows 98/NT/2000/XP/2003, Linux, Macintosh, Solaris, and UNIX computers. There's a range of popular free and commercial VNC software available (UltraVNC, RealVNC, TightVNC). To set up a secure VNC connection, install and configure the VNC Server software on the computer the user will access, then install and configure the VNC Viewer software on the Viewer PC.

6.9.1 Install and Configure the VNC Server on the Computer to be Accessed

Virtual Network Computing (VNC) software enables users to remotely access computers running Linux, Macintosh, Solaris, UNIX, all versions of Windows, and most other operating systems.

For Microsoft Windows servers (and clients):

Windows does not include VNC software, so you will need to download, install, and activate a third party VNC Server software package:

RealVNC http://www.realvnc.com is fully cross-platform, so a desktop running on a Linux machine may be displayed on a Windows PC, on a Solaris machine, or on any number of other architectures. There is a Windows server, allowing you to view the desktop of a remote Windows machine on any of these platforms using exactly the same viewer. RealVNC was founded by members of the AT&T team who originally developed VNC.

TightVNC http://www.tightvnc.com is an enhanced version of VNC. It has added features such as file transfer, performance improvements, and read-only password support. They have just recently included a video drive much like UltraVNC. TightVNC is still free, cross-platform (Windows Unix, and Linux), and compatible with the standard (Real) VNC.

UltraVNC http://ultravnc.com is easy to use, fast, and free VNC software that has pioneered and perfected features that the other flavors have consistently refused or been very slow to implement for cross platform and minimalist reasons. UltraVNC runs under Windows operating systems (95, 98, Me, NT4, 2000, XP, 2003). Download UltraVNC from Sourceforge's UltraVNC file list.

For Linux servers (and clients):

Most Linux distributions now include VNC Servers and Viewers and they generally can be launched from the (Gnome/KDE etc) front end; for example, with Red Hat Enterprise Linux 4 there's VNC Server software and a choice of Viewer client software, and to launch:

Select the Remote Desktop entry in the Main Menu -> Preferences menu.

Click the Allow other users… checkbox to allow remote users to view and control your desktop.

Figure 6-27. Remote Desktop Preferences screen.

To set up a persistent VNC server on Red Hat Enterprise Linux 4:

• Set a password using vncpasswd

• Edit /etc/sysconfig/vncservers

• Enable the service with chkconfig vncserver on

• Start the service with service vncserver start

• Edit /home/username/.vnc/xstartup if you want a more advanced session than just twm and an xterm.

For Macintosh servers (and clients):

OSXvnc http://www.redstonesoftware.com/vnc.html is a robust, full-featured VNC server for Mac OS X that allows any VNC client to remotely view and/or control the Mac OS X machine. OSXvnc is supported by Redstone Software.

Most other operating systems (Solaris, HPUX, PalmOS etc) either come with VNC bundled, or have third-party VNC software that you can download.

6.9.2 Install, Configure, and Connect the VNC Viewer

VNC is truly platform-independent so a VNC Viewer on any operating system can connect to a VNC Server on any other operating system. There are Viewers (and Servers) from a wide selection of sources (for example, UltraVNC, TightVNC, or RealVNC) for most operating systems. There are also a wealth of Java viewers available so that any desktop can be viewed with any Java-capable browser (http://en.wikipedia.org/wiki/VNC lists many of the VNC Viewers sources).

Install the VNC Viewer software and set it up for the appropriate speed connection.

NOTE: To make VNC faster, when you set up the Viewer:

      Set encoding to ZRLE (if you have a fast enough CPU).

      Decrease color level (e.g. 64 bit).

      Disable the background transmission on the Server or use a plain wallpaper.

   (Refer to http://doc.uvnc.com for detailed configuration instructions)

To establish the VNC connection, first configure the VNC Viewer, entering the VNC Server IP address.

When the Viewer PC is connected to the console server thru an SSH tunnel (over the public Internet, or a dial-in connection, or private network connection), enter localhost (or 127.0.0.1) as the IP VNC Server IP address; and the source port you entered when setting SSH tunneling /port forwarding (in Section 6.2.6), for example:1234
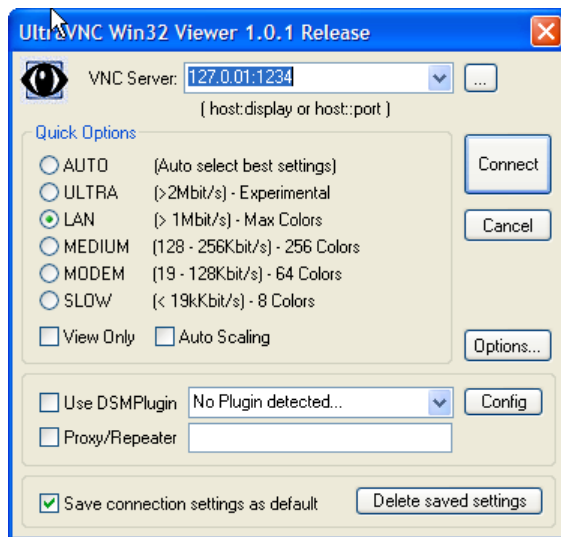
Figure 6-28. VNC server.

When the Viewer PC is connected directly to the console server (that is, locally or remotely through a VPN or dial in connection); and the VNC Host computer is serially connected to the console server; enter the IP address of the console server unit with the TCP port that the SDT tunnel will use. The TCP port will be 7900 plus the physical serial port number (that is, 7901 to 7902, so all traffic directed to port 79xx on the console server is tunneled thru to port 5900 on the PPP connection on serial Port xx). For a Windows Viewer PC using UltraVNC connecting to a VNC Server attached to Port 1 on a console server, it is located at 192.168.0.1.
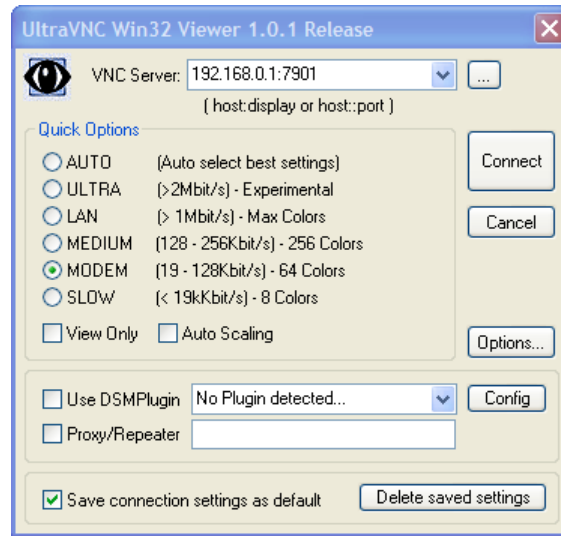
Figure 6-29. IP address of console server unit.

To establish the VNC connection, simply activate the VNC Viewer software on the Viewer PC and enter the password.

Figure 6-30. VNC authentication.

NOTE: For general background reading on Remote Desktop and VNC access we recommend the following:

The Microsoft Remote Desktop How-To.
http://www.microsoft.com/windowsxp/using/mobility/getstarted/remoteintro.mspx
The Illustrated Network Remote Desktop help page.
http://theillustratednetwork.mvps.org/RemoteDesktop/RemoteDesktopSetupandTroubleshooting.html
What is Remote Desktop in Windows XP and Windows Server 2003? by Daniel Petri. http://www.petri.co.il/what's_remote_desktop.htm
Frequently Asked Questions about Remote Desktop. http://www.microsoft.com/windowsxp/using/mobility/rdfaq.mspx
Secure remote access of a home network using SSH, Remote Desktop, and VNC for the home user
http://theillustratednetwork.mvps.org/RemoteDesktop/SSH-RDP-VNC/RemoteDesktopVNCandSSH.html
Taking your desktop virtual with VNC, Red Hat magazine. http://www.redhat.com/magazine/006apr05/features/vnc/ and
http://www.redhat.com/magazine/007may05/features/vnc/
Wikipedia general background on VNC http://en.wikipedia.org/wiki/VNC.

6.10 Using SDT to IP Connect to Hosts that are Serially Attached to the Gateway

Network (IP) protocols like RDP, VNC, and HTTP can also be used for connecting to host devices that are serially connected through their COM port to the console server. To do this you must:

> establish a PPP connection (Section 6.7) between the host and the gateway, then

> set up Secure Tunneling—Ports on the console server (Section 6.7), then

> configure SDT Connector to use the appropriate network protocol to access IP consoles on the host devices that are attached to the Console server serial ports (Section 6.7)

1101 and 1102 Secure Device Servers

6.10.1 Establish a PPP Connection between the Host COM Port and Console Server

(This step is only necessary for serially connected computers.)

First, physically connect the COM port on the host computer you want to access to the serial port on the console server, then:

For non Windows (Linux, UNIX, Solaris, etc.) computers, establish a PPP connection over the serial port. The online tutorial http://www.yolinux.com/TUTORIALS/LinuxTutorialPPP.html presents a selection of methods for establishing a PPP connection for Linux.

For Windows XP and 2003 computers, follow the steps below to set up an advanced network connection between the Windows computer, through its COM port to the console server. Both Windows 2003 and Windows XP Professional allow you to create a simple dial in service which can be used for the Remote Desktop/VNC/HTTP/X connection to the console server:

Open Network Connections in Control Panel and click the New Connection Wizard.
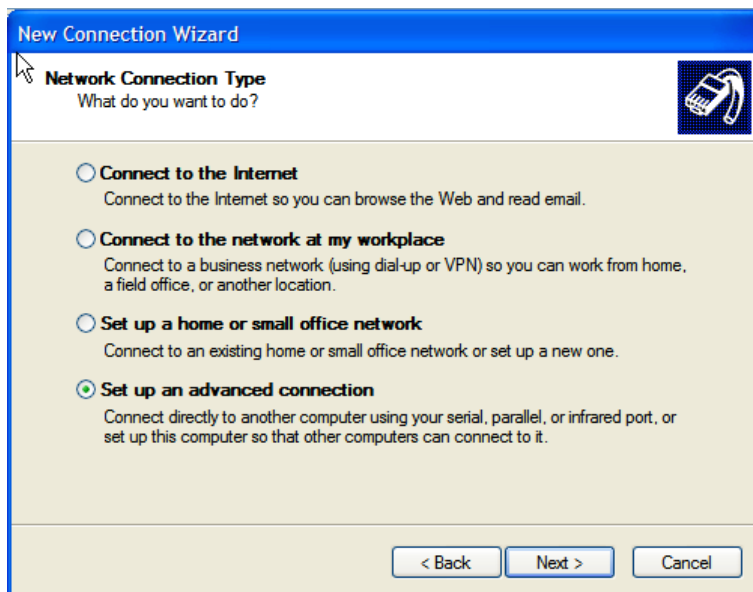
Figure 6-31. New Connection Wizard screen.

Select Set up an advanced connection and click Next.

On the Advanced Connection Options screen, select Accept Incoming Connections and click Next.

Select the Connection Device (i.e. the serial COM port on the Windows computer that you cabled through to the console server). By default, select COM1. The COM port on the Windows computer should be configured to its maximum baud rate. Click Next.

On the Incoming VPN Connection Options screen, select Do not allow virtual private connections and click Next.

724-746-5500 | blackbox.com
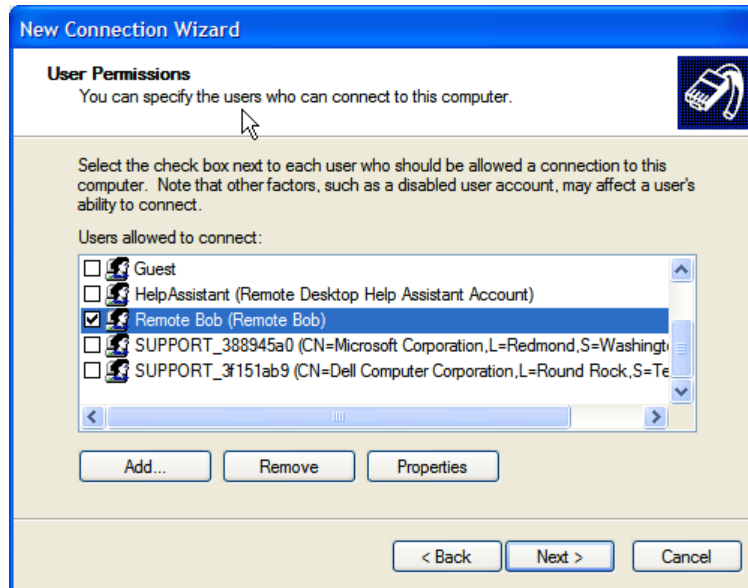
Figure 6-32. User permissions.

Specify which Users will be allowed to use this connection. This should be the same Users who were given Remote Desktop access privileges in the earlier step. Click Next.

On the Network Connection screen, select TCP/IP and click Properties.
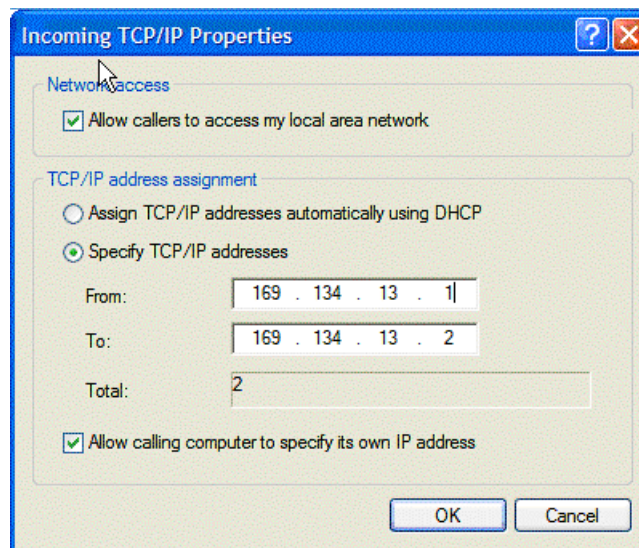
Figure 6-33. Incoming TCP/IP properties.

Select Specify TCP/IP addresses on the Incoming TCP/IP Properties screen, select TCP/IP. Nominate a From: and a To: TCP/IP address, and click Next.

NOTES: You can choose any TCP/IP addresses so long as they are addresses that are not used anywhere else on your network. The From: address
               will be assigned to the Windows XP/2003 computer and the To: address will be used by the console server. For simplicity, use the IP
               address as shown in the illustration above:
                       From: 169.134.13.1
                       To: 169.134.13.2
               Or, you can set the advanced connection and access on the Windows computer to use the console server defaults:
                       Specify 10.233.111.254 as the From: address
                       Select Allow calling computer to specify its own address
               Also, you could use the console server default username and password when you set up the new Remote Desktop User and gave this
               User permission to use the advance connection to access the Windows computer:

1101 and 1102 Secure Device Servers

NOTES (continued): The console server default Username is portXX where XX is the serial port number on the console server.

The default Password is portXX

To use the defaults for a RDP connection to the serial port 2 on the console server, you would have set up a Windows user named port02.

When the PPP connection has been set up, a network icon will appear in the Windows task bar.

NOTE: The above notes describe setting up an incoming connection for Windows XP. The steps are similar for Vista and Windows Server 2003/2008, but the set up screens present slightly differently:

Figure 6-34. Users tab.

You need to put a check in the box for Always allow directly connected devices such as palmtop.

The option to Set up an advanced connection is not available in Windows 2003 if RRAS is configured. If RRAS has been configured, you can enable the null modem connection for the dial-in configuration.

For earlier version Windows computers, follow the steps in Section B, above. To get to the Make New Connection button:

For Windows 2000, click Start, and select Settings. At the Dial-Up Networking Folder, click Network and Dial-up Connections, and click Make New Connection. You may need to first set up a connection over the COM port using Connect directly to another computer before proceeding to Set up an advanced connection.

For Windows 98, double click My Computer on the Desktop, then open Dial-Up Networking and double click.

6.10.2 Set up SDT Serial Ports on Console Server

To set up RDP (and VNC) forwarding on the console server Serial Port that is connected to the Windows computer COM port:

Select the Serial & Network: Serial Port menu option and click Edit (for the particular Serial Port that is connected to the Windows computer COM port).

On the SDT Settings menu, select SDT Mode (this will enable port forwarding and SSH tunneling) and enter a Username and User Password.

SDT Settings

SDT Mode	◎
	Enable access over SSH to a host connected to this serial port.
Username	[_____]
	The login name for PPP. *The default is 'port01'*
User Password	[_____]
	The login secret for PPP. *The default is 'port01'*
Confirm Password	[_____]
	Re-type the password for confirmation.

Figure 6-35. SDT settings screen.

NOTE: When you enable SDT, it will override all other Configuration protocols on that port.

NOTE: If you leave the Username and User Password fields blank, they default to portXX and portXX where XX is the serial port number. The default username and password for Secure RDP over Port 2 is port02.

Make sure the console server Common Settings (Baud Rate, Flow Control) are the same as those set up on the Windows computer COM port and click Apply.

RDP and VNC forwarding over serial ports is enabled on a Port basis. You can add Users who can have access to these ports (or reconfigure User profiles) by selecting Serial & Network: User & Groups menu tag—as described earlier in Chapter 5, Configuring Serial Ports.

6.10.3 Setup SDT Connector to SSH Port Forward over the Console Server Serial Port

In the SDT Connector software running on your remote computer, specify the gateway IP address of your console server and a username/password for a user you set up on the console server that has access to the desired port.

Next, add a New SDT Host. In the Host address, put portxx, where xx = the port you are connecting to. Example: for port 1 you would have a Host Address of: port01. Then select the RDP Service check box.

6.11 SSH Tunneling Using other SSH Clients (for example, PuTTY)

As covered in the previous sections of this chapter, we recommend that you use the SDT Connector client software that is supplied with the console server. There's also a wide selection of commercial and free SSH client programs that can provide the secure SSH connections to the console servers and secure tunnels to connected devices:

PuTTY is a complete (though not very user friendly) freeware implementation of SSH for Win32 and UNIX platforms.

SSHTerm is a useful open source SSH communications package.

SSH Tectia is leading end-to-end commercial communications security solution for the enterprise.

Reflection for Secure IT (formerly F-Secure SSH) is another good commercial SSH-based security solution.

For example, the steps on the next page show how to establish an SSH tunneled connection to a network connected device using the PuTTY client software.
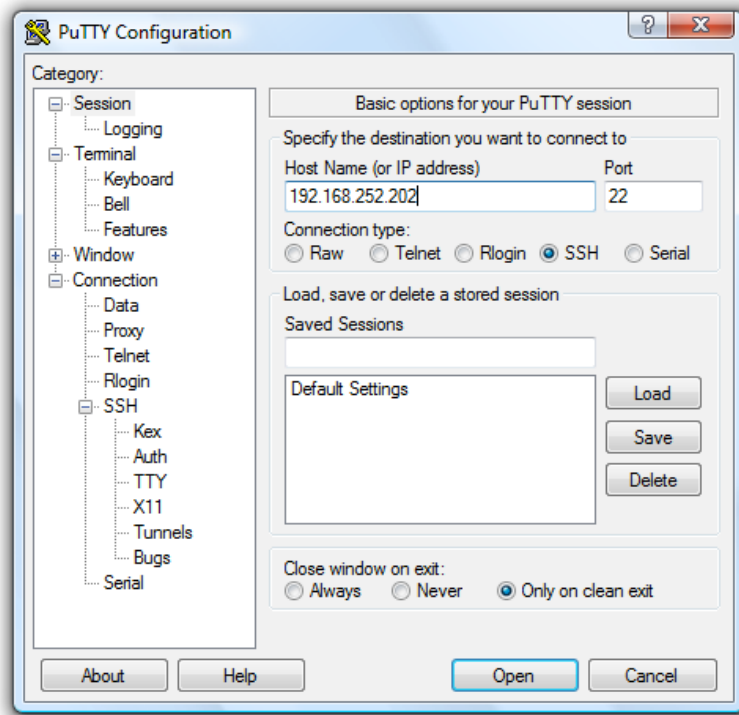
Figure 6-36. PuTTY Configuration screen.

In the Session menu, enter the IP address of the console server in the Host Name or IP address field.

For dial-in connections, this IP address will be the Local Address that you assigned to the console server when you set it up as the Dial-In PPP Server.

For Internet (or local/VPN connections) connections, this will be the console server's public IP address.

Select the SSH Protocol, and the Port will be set as 22.

Go to the SSH -> Tunnels menu and in Add new forwarded port enter any high unused port number for the Source port, for example, 54321. Set the Destination: IP details.

If your destination device is network-connected to the console server and you are connecting using RDP, set the Destination as <Managed Device IP address/DNS Name>:3389. For example, if when setting up the Managed Device as Network Host on the console server, you specified its IP address to be 192.168.253.1 (or its DNS Name was accounts.myco.intranet.com), then specify the Destination as 192.168.523.1:3389 (or accounts.myco.intranet.com:3389). Only devices that are configured as networked Hosts can be accessed using SSH tunneling (except by the "root" user who can tunnel to any IP address the console server can route to).
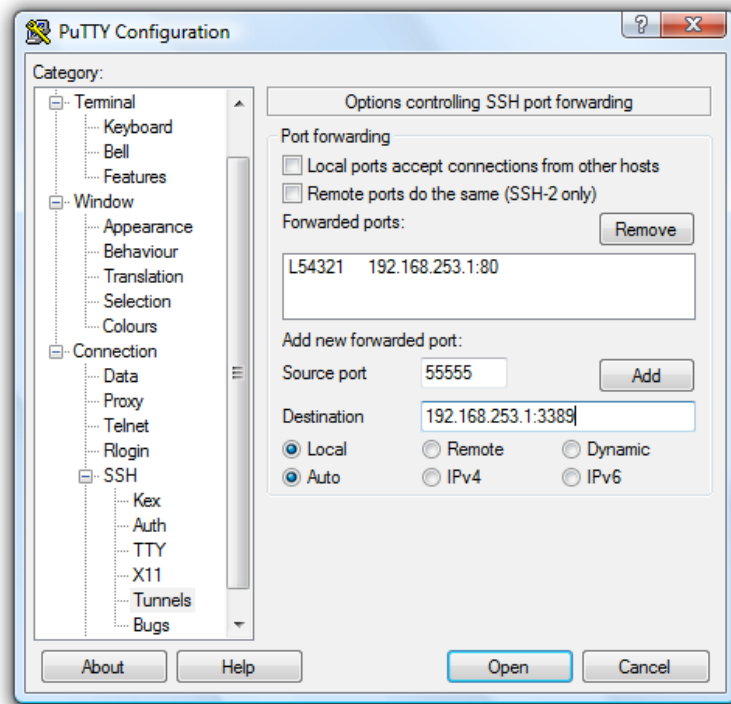
Figure 6-37. Set destination.

If your destination computer is serially connected to the console server, set the Destination as <port label>:3389. For example, if the Label you specified on the serial port on the console server is win2k3, then specify the remote host as win2k3:3389. Or, you can set the Destination as portXX:3389 (where XX is the SDT enabled serial port number). For example, if port 2 is on the console server is to carry the RDP traffic, then specify port02:3389.

NOTE: http://www.jfitz.com/tips/putty_config.html has useful examples on configuring PuTTY for SSH tunneling.

Select Local and click the Add button.

Click Open to SSH connect the Client PC to the console server. You will now be prompted for the Username/Password for the console server user.
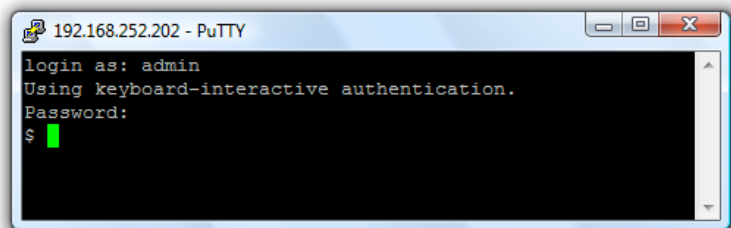
Figure 6-38. Enter username and password.

If you are connecting as a User in the "users" group, then you can only SSH tunnel to Hosts and Serial Ports where you have specific access permission.

If you are connecting as an Administrator (in the "admin" group), then you can connect to any configured Host or Serial Ports (that has SDT enabled).

To set up the secure SSH tunnel for a HTTP browser connection to the Managed Device, specify port 80 (instead of port 3389 that was used for RDP) in the Destination IP address.

To set up the secure SSH tunnel from the Client (Viewer) PC to the console server for VNC, follow the steps above, but when you configure the VNC port redirection, specify port 5900 in the Destination IP address.

1101 and 1102 Secure Device Servers

NOTE: How secure is VNC? VNC access generally allows access to your whole computer, so security is very important. VNC uses a random challenge-response system to provide the basic authentication that allows you to connect to a VNC server. This is reasonably secure and the password is not sent over the network.

Once connected, all subsequent VNC traffic is unencrypted. A malicious user could snoop your VNC session. There are also VNC scanning programs available, which will scan a subnet looking for PCs that are listening on one of the ports that VNC uses.

Tunneling VNC over a SSH connection ensures all traffic is strongly encrypted. No VNC port is ever open to the internet, so anyone scanning for open VNC ports will not be able to find your computers. When tunneling VNC over a SSH connection, the only port that you're opening on your console server is the SDT port 22.

Sometimes it may be prudent to tunnel VNC through SSH even when the Viewer PC and the console server are both on the same local network.

7. Alerts and Logging

This chapter describes the alert generation and logging features of the *console server*. The Alert facility monitors the serial ports, all logins, and the power status, and sends emails, SMS, Nagios, or SNMP alerts when specified trigger events occur.

First, enable and configure the service that will be used to carry the alert (*Section 7.1*).

Then, specify the alert trigger condition and the actual destination to which that particular alert will be sent (*Section 7.2*).

All *console server* models can maintain log records of all access and communications with the *console server* and with the attached serial devices. A log of all system activity is also maintained, as is a history of the status of any attached environmental monitors.

Some models also log access and communications with network attached hosts and maintain a history of the UPS and PDU power status.

If port logs are to be maintained on a remote server, then configure the access path to this location *(Section 7.3)*.

Then you need to activate and set the desired levels of logging for each serial *(Section 7.4)* and/or network port *(Section 7.5)* and/or power UPS (refer to *Chapter 8*).

7.1 Configure SMTP/SMS/SNMP/Nagios Alert Service

The Alerts facility monitors nominated ports/hosts/UPSs/PDUs/EMDs, etc. for trigger conditions. When triggered, the facility sends an alert notification over the nominated alert service. Before setting up the alert trigger, configure these alert services:

7.1.1 Email Alerts

The *console server* uses SMTP (Simple Mail Transfer Protocol) for sending the email alert notifications. To use SMTP, the *Administrator* must configure a valid SMTP server for sending the email:

Select **Alerts & Logging: SMTP &SMS**

Figure 7-1. SMTP and SMS screen.

In the **SMTP Server** field, enter the outgoing mail **Server**'s IP address**.**

If this mail server uses a **Secure Connection**, specify its type.

You may enter a **Sender** email address which will appear as the "*from*" address in all email notifications sent from this *console server*. Many SMTP servers check the sender's email address with the host domain name to verify the address as authentic. So it may be useful to assign an email address for the console server such as *consoleserver2@mydomain.com*

You may also enter a **Username** and **Password** if the SMTP server requires authentication.

You can specify the specific **Subject Line** that will be sent with the email.

Click **Apply** to activate SMTP.

7.1.2 SMS Alerts

The *console server* uses email-to-SMS services to send SMS alert notifications to mobile devices. Sending SMS via email using SMTP (Simple Mail Transfer Protocol) is much faster than sending text pages via a modem using the TAP Protocol. Almost all mobile phone carriers provide an SMS gateway service that forwards email to mobile phones on their networks. There's also a wide selection of SMS gateway aggregators that provide email to SMS forwarding to phones on any carriers. To use SMTP SMS, the *Administrator* must configure a valid SMTP server for sending the email:
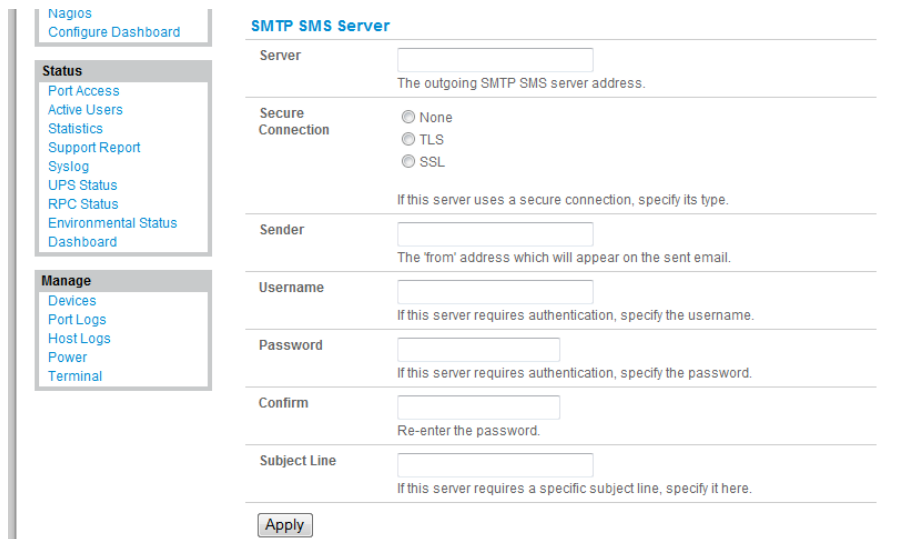
Figure 7-2. SMTP SMS Server screen.

In the **SMTP SMS Server** field in the **Alerts & Logging: SMTP &SMS** menu, enter the IP address of the outgoing mail **Server** (and **Secure Connection** if applicable).

You may enter a **Sender** email address, which will appear as the "*from*" address in all email notifications sent from this *console server*. Some SMS gateway service providers only forward email to SMS when the email has been received from authorized senders. You might need to assign a specific authorized email address for the console server.

You may also enter a **Username** and **Password**, because some SMS gateway service providers use SMTP servers which require authentication.

You can specify the specific **Subject Line** that will be sent with the email. Generally, the email subject will contain a truncated version of the alert notification message (which is contained in full in the body of the email). However some SMS gateway service providers require blank subjects or require specific authentication headers to be included in the subject line.

Click **Apply** to activate SMTP.

7.1.3 SNMP Alerts

The *Administrator* can configure the Simple Network Management Protocol (SNMP) agent that resides on the *console server* to send SNMP trap alerts to an NMS management application:

Select **Alerts & Logging: SNMP**

Enter the SNMP transport protocol. SNMP is generally a **UDP**-based protocol, though infrequently, it uses **TCP** instead.

Enter the IP address of the **SNMP Manager** and the Port to use for connecting (default = 162)

Select the version being used. The *console server* SNMP agent supports SNMP v1, v2, and v3.

Enter the **Community** name for SNMP v1 or 2c. An SNMP community is the group that devices and management stations running SNMP belong to. It helps define where information is sent. SNMP default communities are *private* for Write (and public for Read).

To configure for SNMP v3, you will need to enter an ID and authentication password and contact information for the local *Administrator* (in the **Security Name**).

Click **Apply** to activate SNMP.
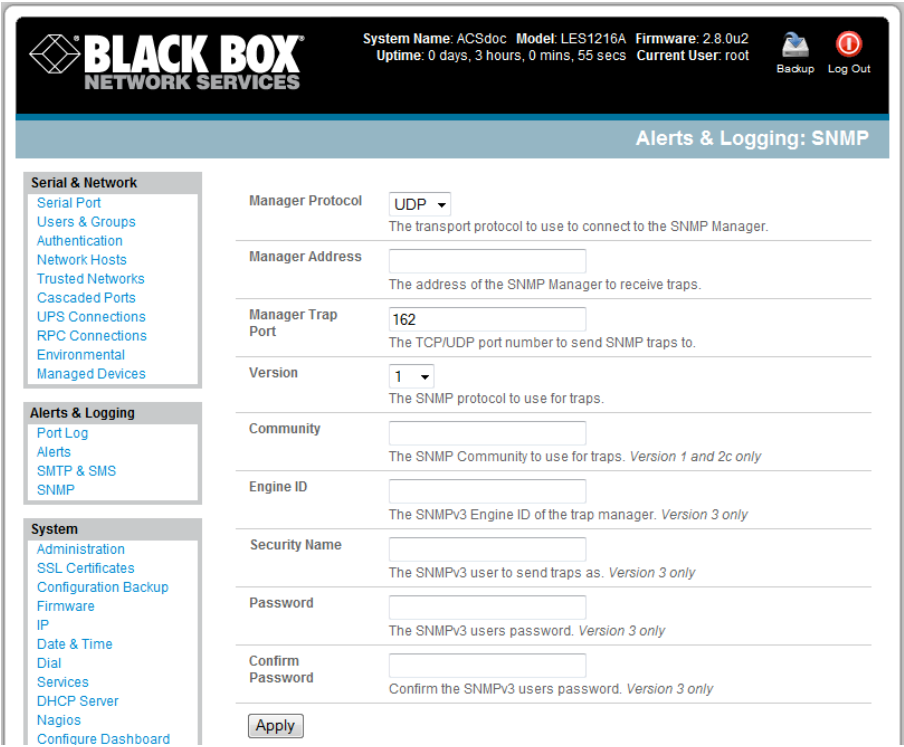
Figure 7-3. SNMP alerts.

NOTE: All *console server*s have the *snmptrap* daemon to send traps/notifications to remote SNMP servers on defined trigger events as detailed above. They also accept SNMP requests from remote SNMP management servers and provides information on network interface, running processes, etc. (refer to *Chapter 15.5—Modifying SNMP Configuration* for more details).

7.1.4 Nagios Alerts

To notify the central Nagios server of Alerts, NSCA must be enabled under **System: Nagios** and Nagios must be enabled for each applicable host or port under **Serial & Network: Network Hosts** or **Serial & Network: Serial Ports** *(refer to Chapter 10).*

7.2 Activate Alert Events and Notifications

The Alert facility monitors the status of the *console server* and connected devices. When an alert event is triggered, the Alert facility notifies a nominated email address or SMS gateway, or the configured SNMP or Nagios server. The data stream from nominated serial ports can be monitored for matched patterns or flow control status changes can be configured to trigger alerts, as can user connections to serial ports and Hosts, or power events.
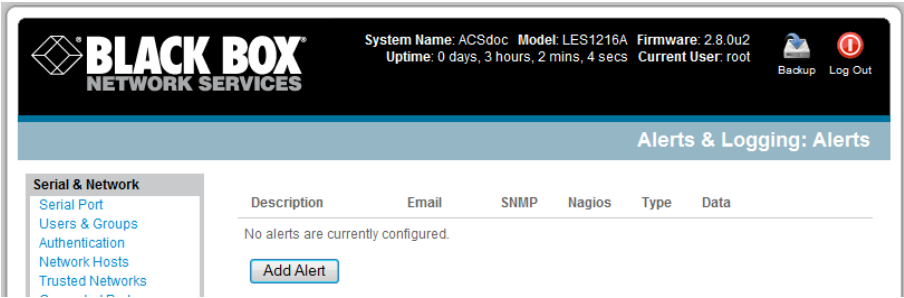
Figure 7-4. Add alert button.

1101 and 1102 Secure Device Servers

Select **Alerts & Logging: Alerts,** which will display all the alerts currently configured. Click **Add Alert.**

7.2.1 Add a New Alert

The first step is to specify the alert service that this event will use for sending notification, who to notify there, and what port/host/device is to be monitored:

Figure 7-5. Add a new alert screen.

At **Add a New Alert,** enter a **Description** for this new alert.

Nominate the email address for the **Email Recipient(s)** and/or the **SMS Recipient(s)** to be notified of the alert. For multiple recipients, enter comma separated addresses.

Activate **SNMP** notification if an SNMP trap is to be sent for this event.

Activate **Nagios** notification to use it for this event. In a SDT Nagios centrally managed environment, you can check the Nagios alert option. On the trigger condition (for matched patterns, logins, power events, and signal changes), an NSCA check "warning" result will be sent to the central Nagios server. This condition is displayed on the Nagios status screen and triggers a notification, which can cause the Nagios central server itself to send out an email or an SMS, page, etc.

7.2.2 Configuring General Alert Types

Next, you must select the Alert Type (**Connection**, **Signal**, **Pattern Match**, **UPS Power Status**, and **Power Sensor**) to monitor. You can configure a selection of different Alert types and any number of specific triggers.

Connection Alert—This alert will be triggered when a user connects or disconnects from the applicable Host or Serial Port, or when a Slave connects or disconnects from the applicable UPS (and you must specify the applicable connections to **Apply Alert To**).
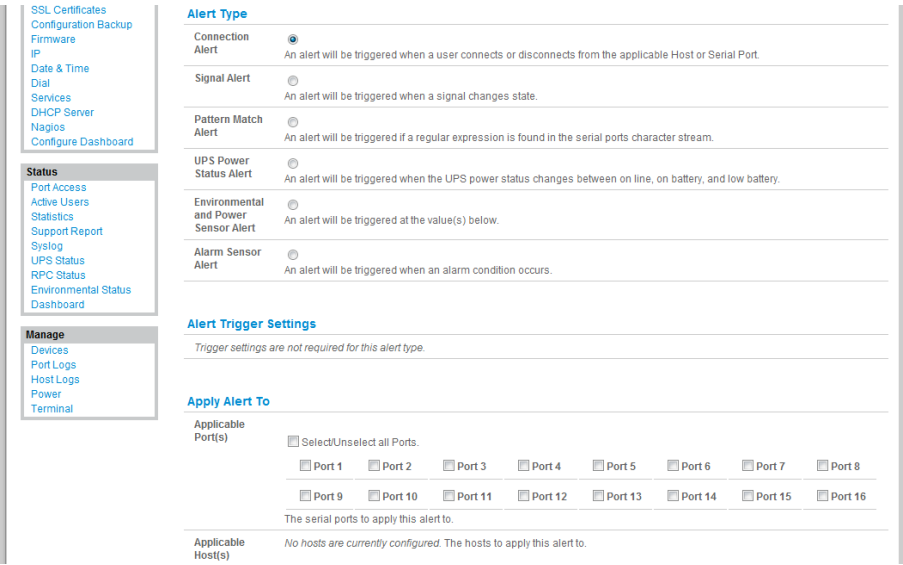
Figure 7-6. General alert types.

Serial Port Signal Alert—This alert will be triggered when the specified signal changes state and applies to serial ports only. You must specify the particular **Signal Type** (DSR, DCD or CTS) trigger condition and the **Applicable Ports(s).**
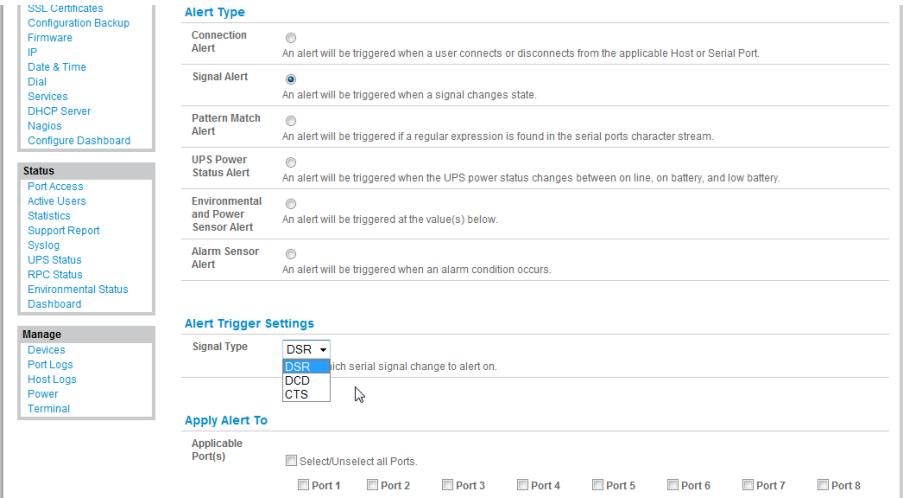
Figure 7-7. Serial port signal alert.

Serial Port Pattern Match Alert—This alert will be triggered if a regular expression is found in the serial ports character stream that matches the regular expression you enter in the **Pattern** field. This alert type will only be applied to serial ports selected as **Applicable Ports(s).**

Figure 7-8. Serial port pattern match alert.

UPS Power Status Alert— This alert will be triggered when the UPS power status changes between on line, on battery, and low battery. This status will only be monitored on the **Applicable UPS(es)** you select.

Power Alert—(next section).

7.2.3 Configuring Power Alert Type

This alert type monitors UPSes, RPCs, and power devices.

Figure 7-9. Power alert.

Select **Power Alert** to activate.

Specify which **Sensor Type** to alert on (Power Load and Battery Charge).

Set the levels at which **Critical** and/or **Warning** alerts are to be sent. You can also specify **High** and/or **Low Set Points** for sending alerts and the **Hysteresis** to be applied before resetting the alerts.

NOTE: Specify the Set Point values in amps (current) for power load and % (percentage) for battery charge.

Specify the applicable UPSes, RPCs (and RPC outlets) to **Apply Alert To**.

7.3 Remote Log Storage

Before activating Serial or Network Port Logging on any port or UPS logging, you must specify where those logs are to be saved:
Select the **Alerts & Logging: Port Log** menu option and specify the **Server Type** to use, and the details to enable log server access.

Figure 7-11. Remote log storage.

7.4 Serial Port Logging

In *Console Server* mode, activity logs of all serial port activity can be maintained. These records are stored on an off-server, or in the Advanced Console Server flash memory. To specify which serial ports have activities recorded and to what level data is to be logged:

Select **Serial & Network: Serial Port** and **Edit** the port to be logged.

Specify the **Logging Level** of for each port as:

Level 0   Turns off logging for the selected port.

Level 1   Logs all connection events to the port.

Level 2   Logs all data transferred to and from the port, all changes in hardware flow control status, and all *User* connection events.

Click **Apply.**

NOTE: A cache of the most recent 8K of logged data per serial port is maintained locally (in addition to the Logs that are transmitted for remote/USB flash storage). To view the local cache of logged serial port data, select **Manage: Port Logs.**

7.5 Network TCP or UDP Port Logging

The *console server* can be configured to log access to and communications with network attached Hosts.
For each Host, when you set up the Permitted Services that you authorize to use, you also must set up the level of logging to maintain for each service.

Specify the logging level to maintain for that particular TDC/UDP port/service, on that particular Host:

Level 0   Turns off logging for the selected TDC/UDP port to the selected Host.

Level 1   Logs all connection events to the port.

Level 2   Logs all data transferred to and from the port.

Click **Add** then click **Apply.**

8. Power Management

Black Box *console server*s manage embedded software that you can use to manage connected Power Distribution Systems (PDUs), IPMI devices, and Uninterruptible Power Supplies (UPSs) supplied by a number of vendors.

8.1 Remote Power Control (RPC)

The *console server* Management Console monitors and controls Remote Power Control (RPC) devices using the embedded PowerMan and Network UPS Tools open source management tools and the Black Box power management software. RPCs include power distribution units (PDUs) and IPMI power devices.

You can control serial PDUs invariably using their command line console, so you could manage the PDU through the *console server* using a remote Telnet client. Also, you could use proprietary software tools supplied by the vendor. This generally runs on a remote Windows PC, and you could configure the *console server* serial port to operate with a serial COM port redirector in the PC (as detailed in *Chapter 5)*.

Similarly, you can control network-attached PDUs with a browser (for example, with SDT as detailed in *Chapter 6.3)*, an SNMP management package, or using the vendor-supplied control software. Servers and network-attached appliances with embedded IPMI service processors or BMCs invariably have their own management tools (like SoL) that provide secure management when connected with SDT Connector.

For simplicity, you can now control all these devices through one window using the Management Console's RPC remote power control tools.

8.1.1 RPC Connection

Serial and network connected RPCs must first be connected to, and configured to communicate with, the *console server*:

For serial RPCs, connect the PDU to the selected serial port on the *console server*. From the **Serial and Network: Serial Port** menu, configure the **Common Settings** of that port with the RS-232 properties, etc. required by the PDU (refer to *Chapter 5.1.1 Common Settings*). Then select **RPC** as the **Device Type**.

For each network-connected RPC, go to **Serial & Network: Network Hosts** menu and configure the RPC as a connected Host by specifying it as **Device Type: RPC** and clicking **Apply** (refer to *Section 5.4, Network Hosts*).

Figure 8-1. Select RPC connections menu.

Select the **Serial & Network: RPC Connections** menu. This will display all the RPC connections that have already been configured.

Click **Add RPC**.

Connected Via presents a list of serial ports and network Host connections that you have set up with device type RPC (but have yet to connect to a specific RPC device):

When you select **Connect Via** for a Network RPC connection, then the corresponding Host Name/Description that you set up for that connection will be entered as the **Name** and **Description** for the power device.

Or, if you select to **Connect Via** a Serial connection, enter a **Name** and **Description** for the power device.

Figure 8-2. Add RPC screen.

Select the appropriate **RPC Type** for the PDU (or IPMI) being connected:

If you are connecting to the RPC via the network, you will be presented with the IPMI protocol options and the SNMP RPC Types currently supported by the embedded Network UPS Tools.

If you are connecting to the RPC by a serial port, you will be presented with all the serial RPC types currently supported by the embedded PowerMan and the Black Box power manager:
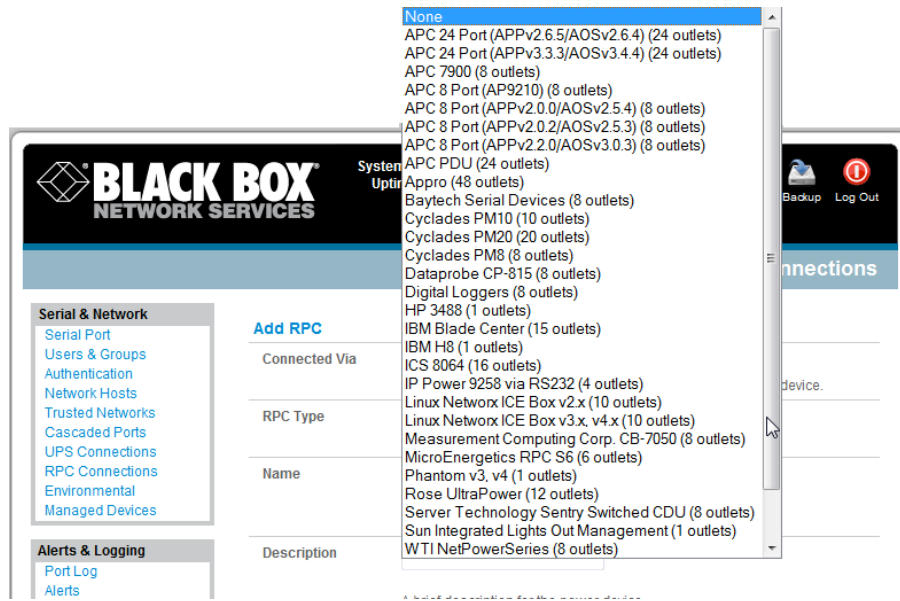
Figure 8-3. RPC descriptions.

Enter the **Username** and **Password** used to login into the RPC (Note that these login credentials are not related to the *Users* and access privileges you configured in *Serial & Networks: Users & Groups*).

If you selected SNMP protocol, enter the SNMP v1 or v2c Community for Read/Write access (by default this would be "private").

Check **Log Status** and specify the **Log Rate** (minutes between samples) if you want the status from this RPC to be logged. View these logs from the **Status: RPC Status** screen.

Click **Apply.**

For SNMP PDUs, the *console server* probes the configured RPC to confirm the RPC Type matches and reports the number of outlets it finds that can be controlled. If unsuccessful, it will report **Unable to probe outlets** and you'll need to check the RPC settings or network/serial connection.

For serially connected RPC devices, a new Managed Device (with the same name as given to the RPC) will be created. The *console server* will then configure the RPC with the number of outlets specified in the selected RPC Type or will query the RPC itself for this information.

NOTE: The Black Box console servers support most popular network and serial PDUs. If your PDU is not on the default list, then you can add support directly (as covered in Chapter 15—Advanced Configurations) or add the PDU support to either the Network UPS Tools or PowerMan open source projects.

Configure IPMI service processors and BMCs so that all authorized users can use the Management Console to remotely cycle power and reboot computers, even when their operating system is unresponsive. To set up IPMI power control, the Administrator first enters the IP address/domain name of the BMC or service processor (for example, a Dell DRAC) in Serial & Network: Network Hosts, then in Serial & Network: RPC Connections specifies the RPC Type to be IPMI1.5 or 2.0.

8.1.2 RPC Access Privileges and Alerts

You can now set PDU and IPMI alerts using **Alerts & Logging: Alerts** (refer to *Chapter 7*). You can also assign which user can access and control which particular outlet on each RPC using **Serial & Network: User & Groups** (refer to *Chapter 5*).

8.1.3 User Power Management

The Power Manager enables both *Users* and *Administrator*s to access and control the configured serial and network attached PDU power strips, and servers with embedded IPMI service processors or BMCs.

Select the **Manage: Power** and the particular **Target** power device to be controlled (and the Outlet to be controlled if the RPC supports outlet level control).

The outlet status is displayed and you can initiate the **Action** you want to take by selecting the appropriate icon:

Turn ON

Turn OFF

Cycle

Status

You will only be presented with icons for those operations that are supported by the **Target** you have selected.
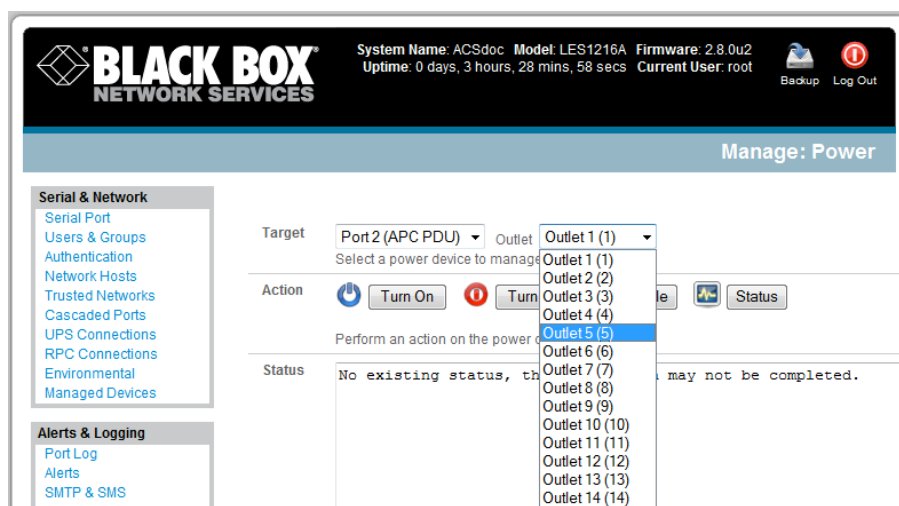
Figure 8-4. Selected operations supported by target.

8.1.4 RPC Status

You can monitor the current status of your network and serially connected PDUs and IPMI RPCs.

Select the **Status: RPC Status** menu and a table with the summary status of all connected RPC hardware will be displayed.

Click on **View Log** or select the **RPCLogs** menu and you will be presented with a table of the history and detailed graphical information on the selected RPC.

Click **Manage** to query or control the individual power outlet. This will take you to the **Manage: Power** screen.

8.2 Uninterruptible Power Supply Control (UPS)

You can configure all Black Box *console server*s to manage locally and remotely connected UPS hardware using Network UPS Tools.
Network UPS Tools (NUT) is a group of open source programs that provide a common interface for monitoring and administering UPS hardware. These programs ensure safe shutdowns of the systems that are connected. NUT is built on a networked model with a layered scheme of drivers, server, and clients (covered in some detail in *Chapter 8.2.6*).
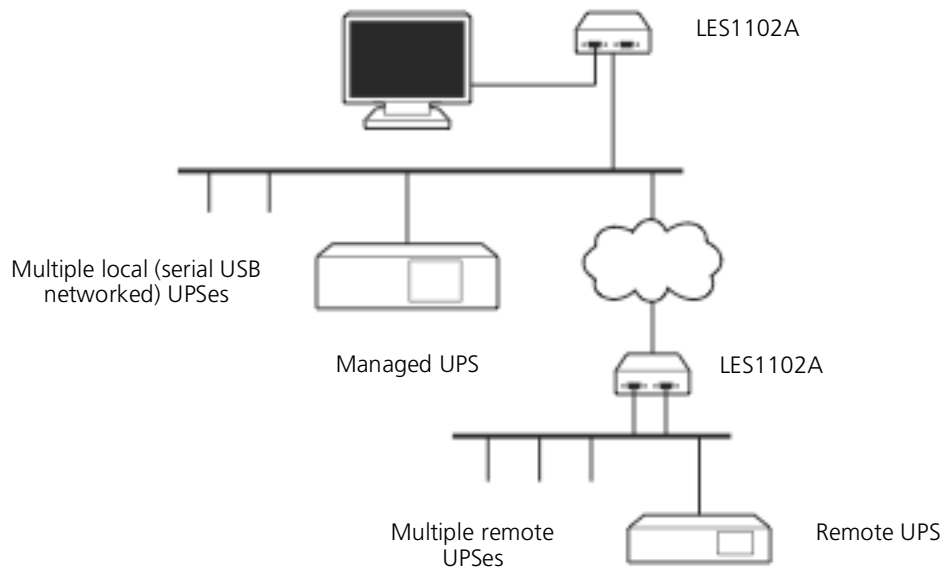
Figure 8-5. Connecting to remote UPS.

8.2.1 Managed UPS Connections

A **Managed UPS** is a UPS that is directly connected as a Managed Device to the *console server*. You can connect it via serial or USB cable or by the network. The *console server* becomes the *master* of this UPS, and runs a *upsd* server to allow other computers that are drawing power through the UPS (*slaves*) to monitor the UPS status and take appropriate action, such as shutdown when the UPS battery is low.
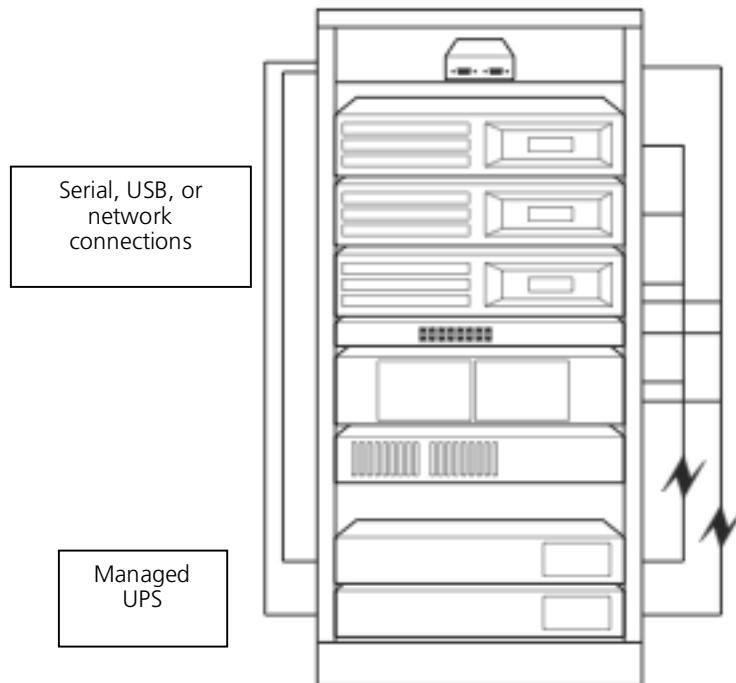
Figure 8-6. Managed UPS connections.

The *console server* may or may not be drawing power itself through the Managed UPS. When the UPS's battery power reaches critical, the *console server* signals and waits for *slaves* to shut down, then powers off the UPS.

1101 and 1102 Secure Device Servers

Serial and network connected UPSes must first be connected to, and configured to communicate with the *console server:*

For serial UPSes attach the UPS to the selected serial port on the *console server*. From the **Serial and Network: Serial Port** menu, configure the **Common Settings** of that port with the RS-232 properties, etc. required by the UPS (refer to *Chapter 5.1.1—Common Settings*). Then select **UPS** as the **Device Type**.

For each network connected UPS, go to the **Serial & Network: Network Hosts** menu and configure the UPS as a connected Host by specifying it as **Device Type: UPS** and clicking **Apply.**

No such configuration is required for USB connected UPS hardware.
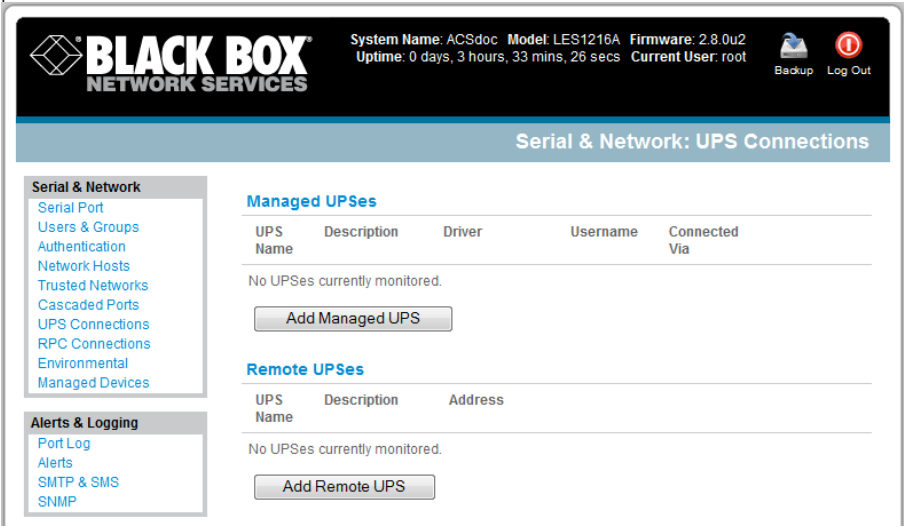
Figure 8-7. UPS connections.

Select the **Serial & Network: UPS Connections** menu. The **Managed UPSes** section will display all the UPS connections that have already been configured.

Click **Add Managed UPS**.

Figure 8-8. Add managed UPS screen.

Select if the UPS will be **Connected Via** USB, over a pre-configured serial port, or via SNMP/HTTP/HTTPS over the preconfigured network Host connection.

When you select a network UPS connection, then the corresponding Host Name/Description that you set up for that connection will be entered as the **Name** and **Description** for the power device. Or, if you selected to **Connect Via** a USB or serial connection then you will need to enter a **Name** and **Description** for the power device (and these details will also be used to create a new Managed Device entry for the serial/USB connected UPS devices).

Enter the login details. This **Username** and **Password** is used by *slaves* of this UPS (that is, other computers that are drawing power through this UPS) to connect to the *console server* to monitor the UPS status so they can shut themselves down when battery power is low. Monitoring will typically be performed using the *upsmon* client running on the slave server (refer to *Section 8.2.3*)

NOTE: These login credentials are not related to the Users and access privileges you configured in Serial & Networks: Users & Groups.

If you have multiple UPSes and require them to be shut down in a specific order, specify the **Shutdown Order** for this UPS. This is a whole positive number, or *-1*. *0s* shut down first, then *1s, 2s*, etc. *-1s* are not shut down at all. Defaults to *0*.

Select the **Driver** that you will use to communicate with the UPS. Most *console servers* are preconfigured so the drop down menu presents a full selection of drivers from the latest Network UPS Tools (NUT version 2.4).

Click **New Options** in **Driver Options** if you need to set driver-specific options for your selected NUT driver and hardware combination (more details at http://www.networkupstools.org/doc).
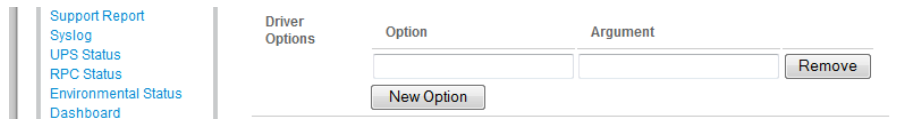
Figure 8-9. New option screen.

Check **Log Status** and specify the **Log Rate** (minutes between samples) if you want the status from this UPS to be logged. You can view these logs from the **Status: UPS Status** screen.

If you have enabled Nagios services, then you will presented with an option for Nagios monitoring. Check **Enable Nagios** to enable this UPS to be monitored using Nagios central management.

Check **Enable Shutdown Script** if this is the UPS providing power to the *console server* itself and if a critical power failure occurs, you can perform any "*last gasp*" actions on the *console server* before power is lost. Place a custom script in */etc/config/scripts/ups-shutdown* (you may use the provided */etc/scripts/ups-shutdown* as a template). This script only runs when then UPS reaches critical battery status.

Click **Apply.**

NOTE: You can also customize the upsmon, upsd, and upsc settings for this UPS hardware directly from the command line.

8.2.2 Remote UPS Management

A **Remote UPS** is a UPS that is connected as a Managed Device to a remote *console server* that is monitored (but not managed) by your *console server*.

You can configure the *upsc* and *upslog* clients in the Black Box *console server* to monitor remote servers that are running Network UPS Tools managing their locally connected UPSes. These remote servers might be other Black Box *console server*s or generic Linux servers running NUT. You can centrally monitor all these distributed UPSes (which may be spread in a row in a data center, around a campus property, or across the country) through the one central *console server* window. To add a Remote UPS:

Select the **Serial & Network: UPS Connections** menu. The **Remote UPSes** section will display all the remote UPS devices being monitored.

Click **Add Remote UPS**.
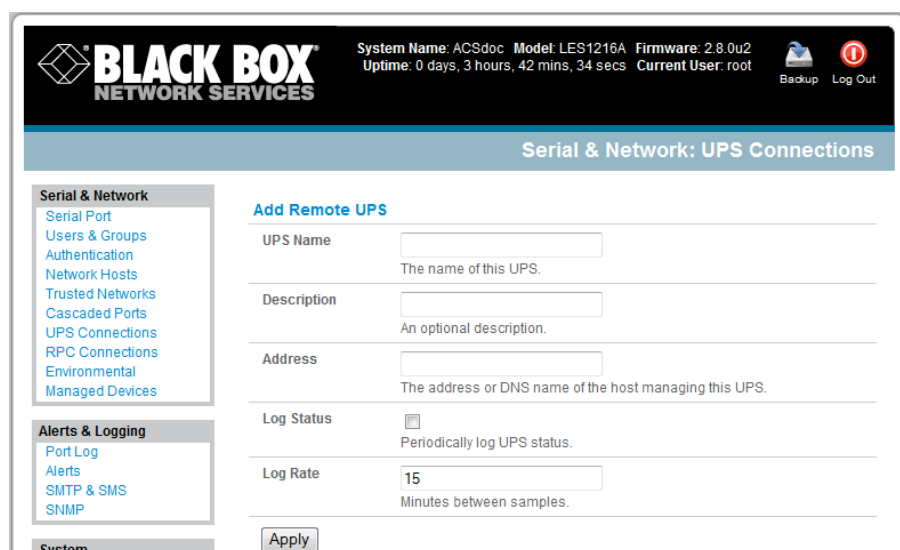
Figure 8-10. Add remote UPS screen.

Enter the **Name** of the particular remote UPS that you want to remotely monitor. This name must be the name that the remote UPS was configured with on the remote *console server* (because the remote *console server* may itself have multiple UPSes attached that it manages locally with NUT). Optionally, enter a **Description.**

Enter the IP **Address** or DNS name of the remote *console server** that is managing the remote UPS. (*This may be another Black Box *console server* or it may be a generic Linux server running Network UPS Tools.)

NOTE: An example where centrally monitor remotely distributed UPSes is useful is a campus or large business site where there's a multitude of computer and other equipment sites spread afar, each with their own UPS supply … and many of these (particularly the smaller sites) will be USB or serially connected.

Having a console server at these remote sites would enable the system manager to centrally monitor the status of the power supplies at all sites, and centralize alarms. So he/she can be warned to initiate a call-out or shut-down.

Check **Log Status** and specify the **Log Rate** (minutes between samples) if you want the status from this UPS to be logged. You can view these logs from the **Status: UPS Status** screen.

Check **Enable Shutdown Script** if this remote UPS is the UPS providing power to the *console server* itself. If the UPS reaches critical battery status, the custom script in */etc/config/scripts/ups-shutdown* runs, enabling you to perform any "*last gasp*" actions.

Click **Apply.**

8.2.3 Controlling UPS Powered Computers

One of the advantages of having a Managed UPS is that you can configure computers that draw power through that UPS to shut down gracefully if you have UPS problems.

For Linux computers, set up *upsmon* on each computer and direct them to monitor the *console server* that is managing their UPS. This will set the specific conditions that will be used to initiate a power down of the computer. Non-critical servers may be powered down some seconds after the UPS starts running on battery. In contrast, more critical servers may not be shut down until a low battery warning is received). Refer to the online NUT documentation for details on how to do this:

http://eu1.networkupstools.org/doc/2.2.0/INSTALL.html
http://linux.die.net/man/5/upsmon.conf
http://linux.die.net/man/8/upsmon

An example upsmon.conf entry might look like:

MONITOR managedups@192.168.0.1 1 username password slave

- *managedups* is the UPS Name of the Managed UPS

- *192.168.0.1* is the IP address of the Black Box *console server*

- *1* indicates the server has a single power supply attached to this UPS

- *username* is the Username of the Managed UPS

- password is the Password of the Manager UPS

There are NUT monitoring clients available for Windows computers (WinNUT).

If you have an RPC (PDU), you can shut down UPS powered computers and other equipment if if the they don't have a client running (for example, communications, and surveillance gear). Set up a UPS alert and using this to trigger a script that controls a PDU to shut off the power (refer to *Chapter 15*).

8.2.4 UPS Alerts

You can set UPS alerts using **Alerts & Logging: Alerts** (refer to *Chapter 7— Alerts & Logging*).

8.2.5 UPS Status

You can monitor the current status of your network, serially or USB connected Managed UPSes, and any configured Remote UPSes.

Select the **Status: UPS Status** menu and a table with the summary status of all connected UPS hardware displays.

Click on any particular UPS **System** name in the table and more detailed graphical information on the selected UPS System appears.

1101 and 1102 Secure Device Servers
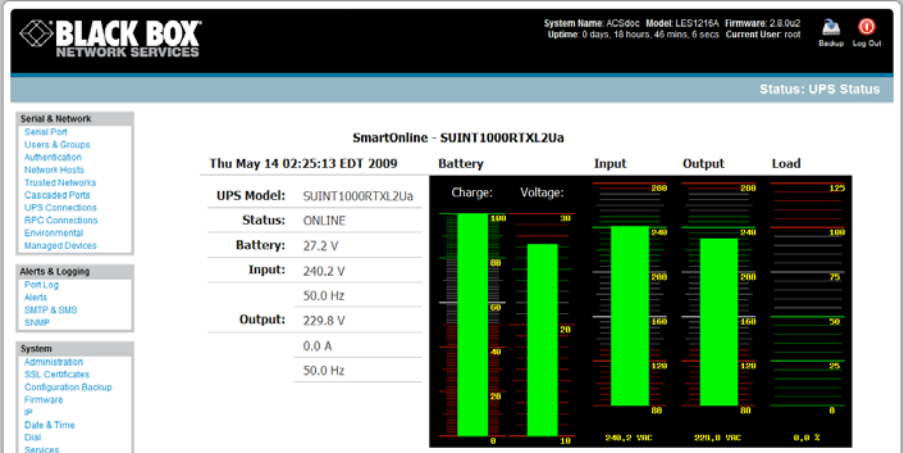
Figure 8-11. UPS graph.

Click on any particular **All Data** for any UPS System in the table for more status and configuration information about the selected UPS System.

Select **UPS Logs** and you will be presented with the log table of the load, battery charge level, temperature, and other status information from all the Managed and Monitored UPS systems. This information will be logged for all UPSes that were configured with **Log Status** checked. The information is also presented graphically.
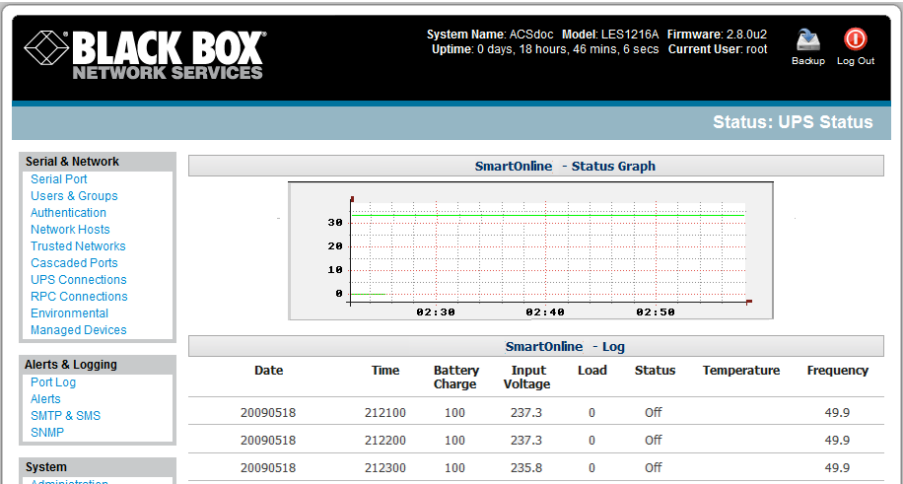
Figure 8-12. Log table.

8.2.6 Overview of Network UPS Tools (NUT)

NUT is built on a networked model with a layered scheme of drivers, server and clients. Configure NUT using the Management Console as described above, or configure the tools and manage the UPSes directly from the command line. This section provides an overview of NUT. You can find full documentation at *http://www.networkupstools.org/doc*.
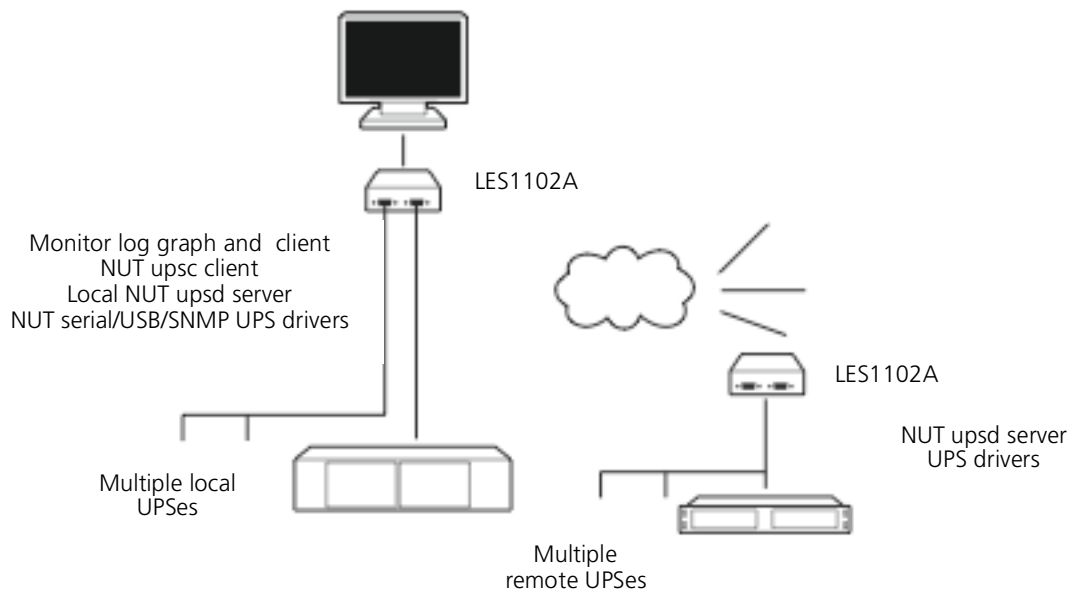
Figure 8-13. NUT.

NUT is built on a networked model with a layered scheme of drivers, server and clients:

The **driver** programs talk directly to the UPS equipment and run on the same host as the NUT network server (*upsd*). Drivers are provided for a wide assortment of equipment from most of the popular UPS vendors and understand the specific language of each UPS. They communicate with serial, USB, and SNMP network connected UPS hardware and map the communications back to a compatibility layer. This means both an expensive "smart" protocol UPS and a simple "power strip" model can be handled transparently.

The NUT network **server** program *upsd* is responsible for passing status data from the drivers to the client programs via the network. *upsd* can cache the status from multiple UPSes and then serve this status data to many clients. *upsd* also contains access control features to limit the abilities of the clients (only authorized hosts may monitor or control the UPS hardware).

There are a number of NUT **clients** that connect to *upsd* to check on the status of the UPS hardware and do things based on the status. These clients can run on the same host as the NUT server or they can communicate with the NUT server over the network (enabling them to monitor any UPS anywhere):

> The *upsc* client provides a quick way to poll the status of a UPS server. Use it inside shell scripts and other programs that need UPS data but don't want to include the full interface.

> The *upsmon* client enables servers that draw power through the UPS to shutdown gracefully when the battery power reaches critical.

> There are also logging clients (*upslog*) and third party interface clients (Big Sister, Cacti, Nagios, Windows, and more). Refer to *www.networkupstools.org/client-projects*.)

The latest release of NUT (2.4) also controls PDU systems. It can do this either natively using SNMP or through a *binding* to Powerman (open source software from Livermore Labs that also is embedded in Black Box *console servers*).

These NUT clients and servers all are embedded in each Black Box *console server* (with a Management Console presentation layer added) —and they also are run remotely on distributed *console servers* and other remote NUT monitoring systems. This layered distributed NUT architecture enables:

Multiple manufacturer support: NUT can monitor UPS models from 79 different manufacturers—and PDUs from a growing number of vendors—with a unified interface.

Multiple architecture support: NUT can manage serial and USB connected UPS models with the same common interface. Network-connected USB and PDU equipment can also be monitored using SNMP.

Multiple clients monitoring one UPS: Multiple systems may monitor a single UPS using only their network connections. There is a wide selection of client programs that support monitoring UPS hardware via NUT (Big Sister, Cacti, Nagios, and more).

1101 and 1102 Secure Device Servers

Central management of multiple NUT servers: A central NUT client can monitor multiple NUT servers that may be distributed throughout the data center, across a campus, or around the world.

NUT supports the more complex power architectures found in data centers, communications centers, and distributed office environments where many UPSes from many vendors power many systems with many clients. Each of the larger UPSes power multiple devices, and many of these devices are in turn dual powered.

9. Authentication

The *console server* is a dedicated Linux computer with a myriad of popular and proven Linux software modules for networking, secure access (OpenSSH), and communications (OpenSSL), and sophisticated user authentication (PAM, RADIUS, TACACS+, and LDAP).

This chapter details how the *Administrator* can use the Management Console to establish remote AAA authentication for all connections to the *console server* and attached serial and network host devices.

This chapter also covers how to establish a secure link to the Management Console using HTTPS and using OpenSSL and OpenSSH to establish a secure Administration connection to the *console server*.

9.1 Authentication Configuration

Authentication can be performed locally, or remotely using an LDAP, Radius, or TACACS+ authentication server. The default authentication method for the *console server* is Local.
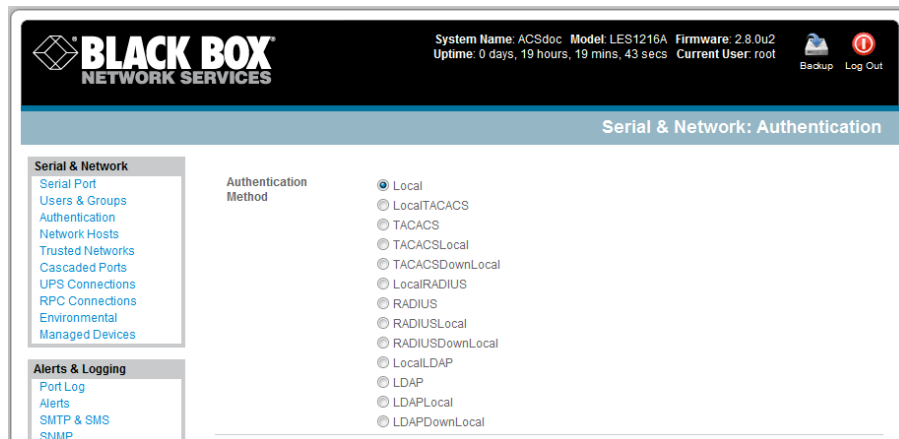
Figure 9-1. Authentication screen.

Any authentication method that is configured will be used for authentication of any user who attempts to log in through Telnet, SSH, or the Web Manager to the *console server* and any connected serial port or network host devices.

You can configure the *console server* to the default (**Local**) or using an alternate authentication method (**TACACS**, **RADIUS,** or **LDAP**). Optionally, you can select the order in which local and remote authentication is used:

Local *TACACS* *⁄RADIUS⁄LDAP*: Tries local authentication first, falling back to remote if local fails.

*TACACS ⁄RADIUS⁄***LDAP** **Local**: Tries remote authentication first, falling back to local if remote fails.

*TACACS ⁄RADIUS⁄***LDAP** **Down Local**: Tries remote authentication first, falling back to local if the remote authentication returns an error condition (for example, if the remote authentication server is down or inaccessible).

9.1.1 Local Authentication

Select **Serial and Network: Authentication** and check **Local**.

Click **Apply.**

9.1.2 TACACS Authentication

Perform the following procedure to configure the TACACS+ authentication method to use whenever the *console server* or any of its serial ports or hosts is accessed:

Select **Serial and Network: Authentication** and check **TACAS** or **LocalTACACS** or **TACACSLocal** or **TACACSDownLocal**

Figure 9-2. TACACS screen.

Enter the **Server Address** (IP or host name) of the remote Authentication/Authorization server. Multiple remote servers may be specified in a comma-separated list. Each server is tried in succession.

In addition to multiple remote servers, you can also enter separate lists of Authentication/Authorization servers and Accounting servers. If no Accounting servers are specified, the Authentication/Authorization servers are used instead.

Enter the **Server Password.**

Click **Apply.** TACAS+ remote authentication will now be used for all user access to *console server* and serially or network attached devices.

TACACS+

The Terminal Access Controller Access Control System (TACACS+) security protocol is a recent protocol developed by Cisco. It provides detailed accounting information and flexible administrative control over the authentication and authorization processes. TACACS+ allows for a single access control server (the TACACS+ daemon) to provide authentication, authorization, and accounting services independently. Each service can be tied into its own database to take advantage of other services available on that server or on the network, depending on the capabilities of the daemon. There is a draft RFC detailing this protocol. You can find further information on configuring remote TACACS+ servers at the following sites:

http://www.cisco.com/en/US/tech/tk59/technologies_tech_note09186a0080094e99.shtml

http://www.cisco.com/en/US/products/sw/secursw/ps4911/products_user_guide_chapter09186a00800eb6d6.html

http://cio.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scprt2/sctplus.htm

9.1.3 RADIUS Authentication

Perform the following procedure to configure the RADIUS authentication method to use whenever the *console server* or any of its serial ports or hosts is accessed:

Select **Serial and Network: Authentication** and check **RADIUS** or **LocalRADIUS** or **RADIUSLocal** or **RADIUSDownLocal**.

Figure 9-3. RADIUS screen.

Enter the **Server Address** (IP or host name) of the remote Authentication/ Authorization server. Multiple remote servers may be specified in a comma-separated list. Each server is tried in succession.

In addition to multiple remote servers, you can also enter separate lists of Authentication/Authorization servers and Accounting servers. If no Accounting servers are specified, the Authentication/Authorization servers are used instead.

Enter the **Server Password.**

Click **Apply.** RADIUS remote authentication will now be used for all user access to *console server* and serially or network-attached devices.

RADIUS: The Remote Authentication Dial-In User Service (RADIUS) protocol was developed by Livingston Enterprises as an access server authentication and accounting protocol. The RADIUS server can support a variety of methods to authenticate a user. When it is provided with the username and original password given by the user, it can support PPP, PAP, or CHAP, UNIX login, and other authentication mechanisms. You can find further information on configuring remote RADIUS servers at the following sites:

http://www.microsoft.com/technet/prodtechnol/windowsserver2003/library/DepKit/d4fe8248-eecd-49e4-88f6-9e304f97fefc.mspx

http://www.cisco.com/en/US/tech/tk59/technologies_tech_note09186a00800945cc.shtml

http://www.freeradius.org/

9.1.4 LDAP Authentication

Perform the following procedure to configure the LDAP authentication method to use whenever the *console server* or any of its serial ports or hosts is accessed:

Select **Serial and Network: Authentication** and check **LDAP** or **LocalLDAP** or **LDAPLocal** or **LDAPDownLocal**
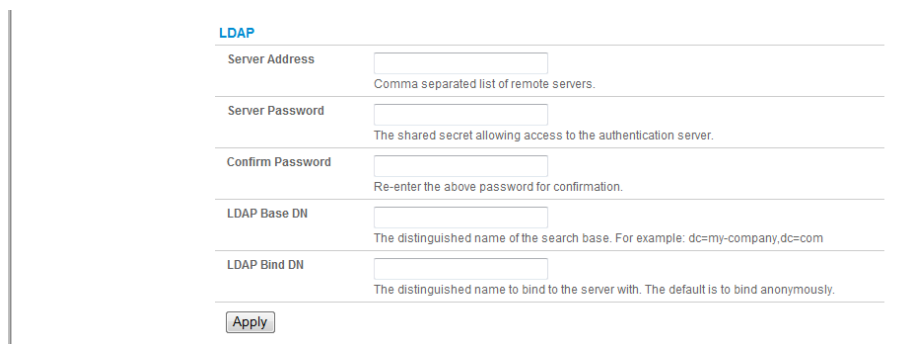
Figure 9-4. LDAP screen.

Enter the **Server Address** (IP or host name) of the remote Authentication server. Multiple remote servers may be specified in a comma-separated list. Each server is tried in succession.

Enter the **Server Password.**

NOTE: To interact with LDAP requires that the user account exist on our console server to work with the remote server. (You can't just create the user on your LDAP server and not tell the console server about it.) You need to add the user account.

Click **Apply.** LDAP remote authentication will now be used for all user access to *console server* and serially or network attached devices.

LDAP: The Lightweight Directory Access Protocol (LDAP) is based on the X.500 standard, but is significantly simpler and more readily adapted to meet custom needs. The core LDAP specifications are all defined in RFCs. LDAP is a protocol used to access information stored in an LDAP server.

You can find further information on configuring remote RADIUS servers at the following sites:

http://www.ldapman.org/articles/intro_to_ldap.html
http://www.ldapman.org/servers.html
http://www.linuxplanet.com/linuxplanet/tutorials/5050/1/
http://www.linuxplanet.com/linuxplanet/tutorials/5074/4/

9.1.5 RADIUS/TACACS User Configuration

Users may be added to the local *console server* appliance. If they are not added and they log in via remote AAA, a user will be added for them. This user will not show up in the Black Box configurators unless they are specifically added, at which point they are transformed into a completely local user. The newly added user must authenticate from the remote AAA server, and will have no access if it is down.

If a local user logs in, they may be authenticated/authorized from the remote AAA server, depending on the chosen priority of the remote AAA. A local user's authorization is the union of local and remote privileges.

Example 1: User Tim is locally added, and has access to ports 1 and 2. He is also defined on a remote TACACS server, which says he has access to ports 3 and 4. Tim may log in with either his local or TACACS password, and will have access to ports 1 through 4. If TACACS is down, he will need to use his local password, and will only be able to access ports 1 and 2.

1101 and 1102 Secure Device Servers

Example 2: User Ben is only defined on the TACACS server, which says he has access to ports 5 and 6. When he attempts to log in, a new user will be created for him, and he will be able to access ports 5 and 6. If the TACACS server is down he will have no access.

Example 3: User Paul is defined on a RADIUS server only. He has access to all serial ports and network hosts.

Example 4: User Don is locally defined on an appliance using RADIUS for AAA. Even if Don is also defined on the RADIUS server, he will only have access to those serial ports and network hosts he has been authorized to use on the appliance.

If a "no local AAA" option is selected, then root will still be authenticated locally.

You can add remote users to the admin group via either RADIUS or TACACS. Users may have a set of authorizations set on the remote TACACS server. Users automatically added by RADIUS will have authorization for all resources, whereas those added locally will still need their authorizations specified.

LDAP has not been modified, and will still need locally defined users.

9.2 PAM (Pluggable Authentication Modules)

The *console server* supports RADIUS, TACACS+, and LDAP for two-factor authentication *via* PAM (Pluggable Authentication Modules). PAM is a flexible mechanism for authenticating users. A number of new ways of authenticating users have become popular. The challenge is that each time a new authentication scheme is developed, you need to rewrite all the necessary programs (login, ftpd, *etc*.) to support it.
PAM provides a way to develop programs that are independent of authentication scheme. These programs need "authentication modules" to be attached to them at run-time in order to work. Which authentication module is attached depends on the local system setup and is at the discretion of the local *Administrator*.

The *console server* family supports PAM with the following modules added for remote authentication:

RADIUS	- pam_radius_auth	(http://www.freeradius.org/pam_radius_auth/)
TACACS+	- pam_tacplus	(http://echelon.pl/pubs/pam_tacplus.html)
LDAP	- pam_ldap	(http://www.padl.com/OSS/pam_ldap.html)

Further modules can be added as required.

Changes may be made to files in /etc/config/pam.d/ that will persist, even if the authentication configurator runs.

Users added on demand: When a user attempts to log in, but does not already have an account on the *console server*, a new user account will be created. This account will have no rights, and no password set. It will not appear in the Black Box configuration tools. Automatically added accounts will not be able to log in if the remote servers are unavailable. RADIUS users are currently assumed to have access to all resources, so they will only be authorized to log in to the *console server*. RADIUS users will be authorized each time they access a new resource.

Admin rights granted over AAA: Users may be granted *Administrator* rights via networked AAA. For TACACS a priv-lvl of 12 or above indicates an *Administrator*. For RADIUS, *Administrator*s are indicated via the Framed Filter ID. (See the example configuration files below for example.)

Authorization via TACACS for both serial ports and host access: Permission to access resources may be granted via TACACS by indicating a Black Box Appliance and a port or networked host the user may access. (See the example configuration files below.)

TACACS Example:
user = tim {
  service = raccess {
     priv-lvl = 11
     port1 = les1102/port02
     port2 = 192.168.254.145/port05
  }
  global = cleartext mit
}
RADIUS Example:
paul   Cleartext-Password := "luap"
     Service-Type = Framed-User,
     Fall-Through = No,
     Framed-Filter-Id=":group_name=admin"
]
The list of groups may include any number of entries separated by a comma. If the admin group is included, the user will be made an *Administrator*. If there is already a Framed-Filter-Id, simply add the list of *group_names* after the existing entries, including the separating colon ":".

9.3 SSL Certificate

The *console server* uses the Secure Socket Layer (SSL) protocol for encrypted network traffic between itself and a connected user. When establishing the connection, the *console server* has to expose its identity to the user's browser using a cryptographic certificate. The default certificate that comes with the *console server* device upon delivery is for testing purposes only.

⚠ *The System Administrator should not rely on the default certificate as the secured global access mechanism for use through Internet.*

Activate your preferred browser and enter https://*IP address*. Your browser may respond with a message that verifies the security certificate is valid but notes that it is not necessarily verified by a certifying authority. To proceed, you need to click *yes* if you are using Internet Explorer or select *accept this certificate permanently* (or *temporarily*) if you are using Mozilla Firefox.

You will then be prompted for the *Administrator* account and password as normal.

We recommend that you generate and install a new base64 X.509 certificate that is unique for a particular *console server*.

To do this, the *console server* must be enabled to generate a new cryptographic key and the associated Certificate Signing Request (CSR) that needs to be certified by a Certification Authority (CA). A certification authority verifies that you are the person who you claim you are, and signs and issues a SSL certificate to you. To create and install a SSL certificate for the *console server*:

Figure 9-5. SSL certificates screen.

1101 and 1102 Secure Device Servers

Select **System: SSL Certificate** and fill out the fields as explained below:

Common name
This is the network name of the *console server* once it is installed in the network (usually the fully qualified domain name). It is identical to the name that is used to access the *console server* with a web browser (without the "http://" prefix). In case the name given here and the actual network name differ, the browser will pop up a security warning when the *console server* is accessed using HTTPS.

Organizational Unit
Use this field to specify which department within an organization the *console server* belongs to.

Organization
The name of the organization that the *console server* belongs to.

Locality/City
The city where the organization is located.

State/Province
The state or province where the organization is located.

Country
The country where the organization is located. This is the two-letter ISO code, for example, DE for Germany, or US for the USA. (Note: Enter the country code in CAPITAL LETTERS.)

Email
The email address of a contact person that is responsible for the *console server* and its security.

Challenge Password
Some certification authorities require a challenge password to authorize later changes on the certificate (for example, revocation of the certificate). The password must be at least 4 characters long.

Confirm Challenge Password
Confirmation of the Challenge Password.

Key length
This is the length of the generated key in bits. 1024 Bits are supposed to be sufficient for most cases. Longer keys may result in slower response time of the *console server* when establishing a connection.

Once this is done, click on the button **Generate CSR** which will initiate the Certificate Signing Request generation. The CSR can be downloaded to your administration machine with the **Download** button.

Send the saved CSR string to a Certification Authority (CA) for certification. You will get the new certificate from the CA after a more or less complicated traditional authentication process (depending on the CA).

Upload the certificate to the *console server* using the **Upload** button as shown on the next page.
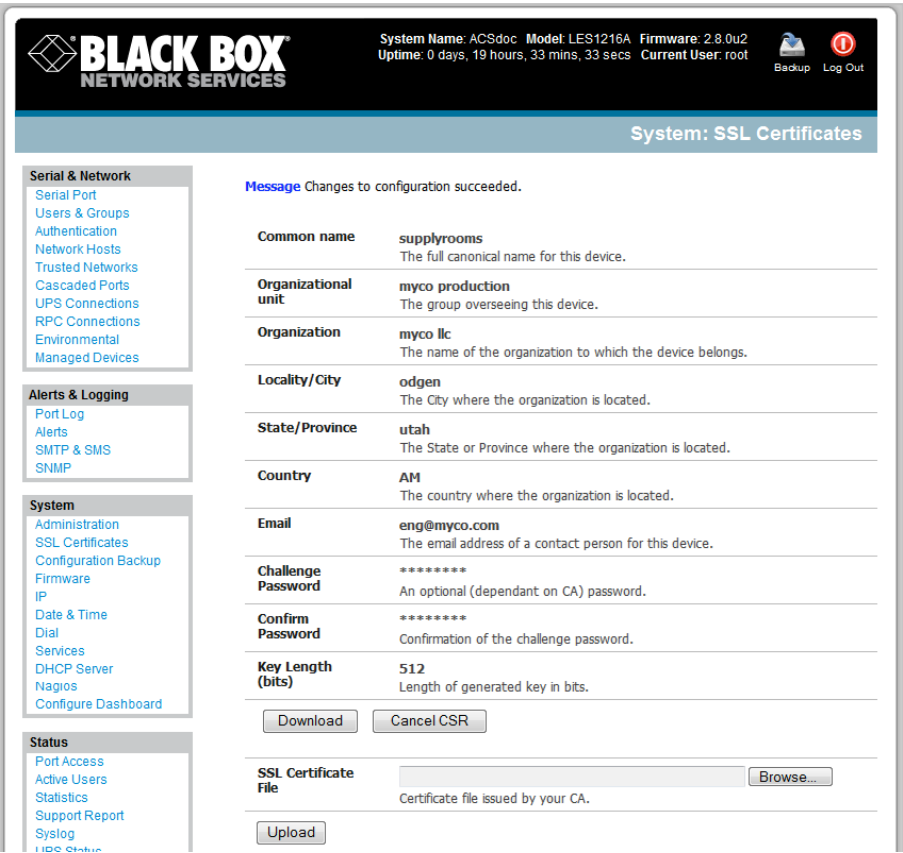
Figure 9-6. Upload button.

After completing these steps, the *console server* has its own certificate that is used for identifying the *console server* to its users.

NOTE: You can find information on issuing certificates and configuring HTTPS from the command line in Chapter 14.

10. Nagios Integration

Nagios is a powerful, highly extensible open source tool for monitoring network hosts and services. The core Nagios software package will typically be installed on a server or virtual server, the central Nagios server.

*Console server*s operate in conjunction with a central/upstream Nagios server to distribute and monitor attached network hosts and serial devices. They embed the NSCA (Nagios Service Checks Acceptor) and NRPE (Nagios Remote Plug-in Executor) add-ons—this allows them to communicate with the central Nagios server, so you won't need a dedicated slave Nagios server at remote sites.

Even if distributed monitoring is not required, the *console server*s can be deployed locally alongside the Nagios monitoring host server, to provide additional diagnostics and points of access to managed devices.
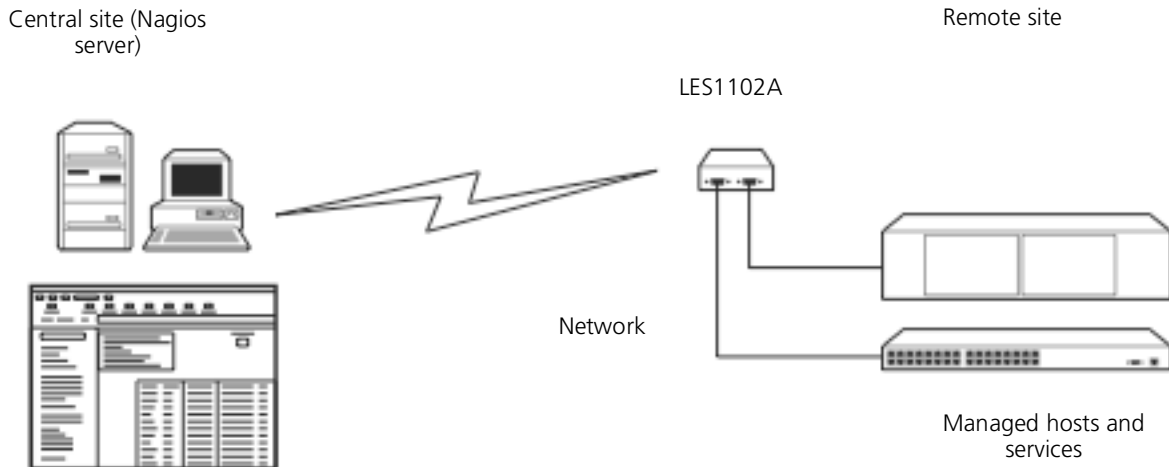
Figure 10-1. Nagios application.

SDT for Nagios extends the capabilities of the central Nagios server beyond monitoring, enabling it to be used for central management tasks. It incorporates the SDT Connector client, enabling point-and-click access and control of distributed networks of *console server*s and their attached network and serial hosts, from a central location.

NOTE: If you have an existing Nagios deployment, you may want to use the *console server* gateways in a distributed monitoring server capacity only. If this case and you are already familiar with Nagios, skip ahead to Section 10.3.

10.1 Nagios Overview

Nagios provides central monitoring of the hosts and services in your distributed network. Nagios is freely downloadable, open source software. This section offers a quick background of Nagios and its capabilities. A complete overview, FAQ, and comprehensive documentation are available at: http://www.nagios.org

Nagios does take some time to install and configure; however, once Nagios is up and running however, it provides an outstanding network monitoring system.

With Nagios you can:

Display tables showing the status of each monitored server and network service in real time.

Use a wide range of freely available plug-ins to make detailed checks of specific services—for example, don't just check that a database is accepting network connections, check that it can actually validate requests and return real data.

Display warnings and send warning e-mails, pager, or SMS alerts when a service failure or degradation is detected.

Assign contact groups who are responsible for specific services in specific time frames.

10.2 Central Management and Setting Up SDT for Nagios

The Black Box Nagios solution has three parts: the Central Nagios server, Distributed Black Box *console servers*, and the SDT for Nagios software.
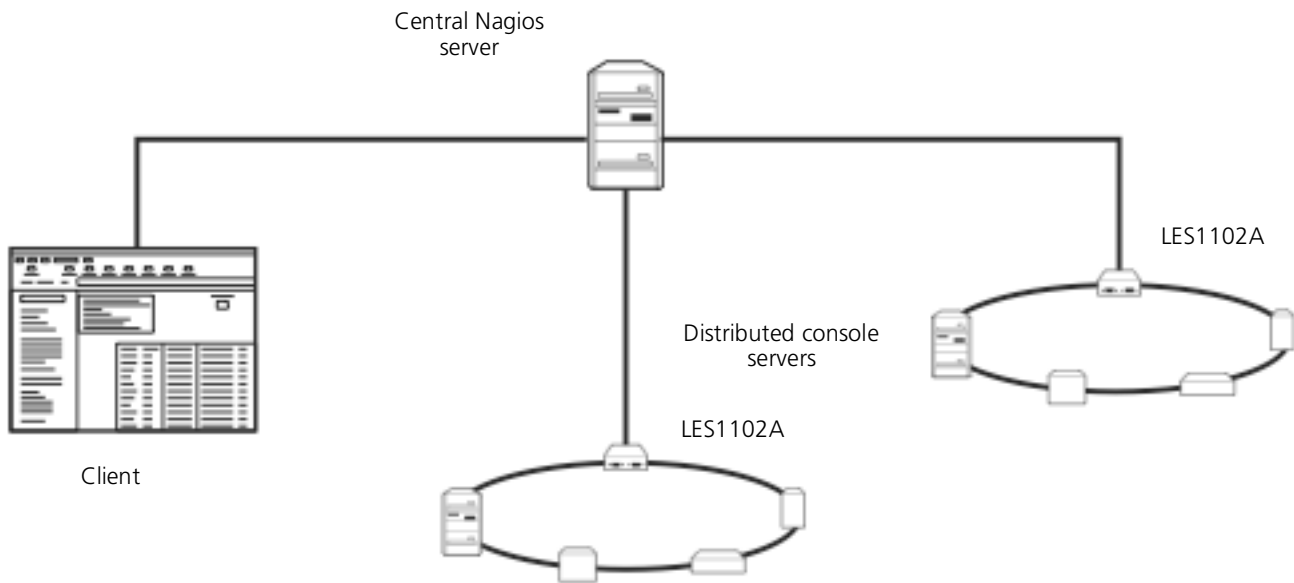
Figure 10-2. Nagios setup.

Central Nagios server

- A vanilla Nagios 2.x or 3.x installation (typically on a Linux server) generally running on a blade, PC, virtual machine, etc. at a central location. Runs a web server that displays the Nagios GUI.

- Imports configuration from distributed *console servers* using the SDT for Nagios Configuration Wizard.

Distributed *console server*s

- Black Box *console servers*.

- Serial and network hosts are attached to each *console server*.

- Each runs Nagios plug-ins, NRPE, and NSCA add-ons, but not a full Nagios server.

Clients

- Typically a client PC, laptop, etc., running Windows, Linux, or Mac OS X.

- Runs SDT Connector client software 1.5.0 or later.

- Possibly remote to the central Nagios server or distributed *console servers* (i.e. a road warrior).

- May receive alert emails from the central Nagios server or distributed *console servers.*

- Connects to the central Nagios server web UI to view status of monitored hosts and serial devices.

- Uses SDT Connector to connect through the *console servers* to manage monitored hosts and serial devices.

SDT Nagios setup involves the following steps:

1. Install Nagios and the NSCA and NRPE add-ons on the central Nagios server *(Section 10.2.1—Set up central Nagios server)*.
   Configure each Black Box distributed *console server* for Nagios monitoring, alerting, and SDT Nagios integration *(Section 10.2.2— Set up distributed* Black Box *servers)*
   .

2. Run the SDT for Nagios Configuration Wizard on the central Nagios server *(Section 10.2.1— Set up SDT Nagios on central Nagios server)* and perform any additional configuration tasks.

3. Install SDT Connector on each client..

10.2.1 Setup Central Nagios Server

SDT for Nagios requires a central Nagios server running Nagios 2.x or 3.x. Nagios 1.x is not supported. The Nagios server software is available for most major distributions of Linux using the standard package management tools. Your distribution will have documentation available on how to install Nagios. This is usually the quickest and simplest way to get up and running.

Note that you will need the core Nagios server package, and at least one of the NRPE or NSCA add-ons. NSCA is required to use the alerting features of the Black Box distributed hosts; installing both NRPE and NSCA is recommended.

You will also require a web server such as Apache to display the Nagios web UI (and this may be installed automatically depending on the Nagios packages).

Or, you may wish to download the Nagios source code directly from the Nagios website, and build and install the software from scratch. The Nagios website (http://www.nagios.org) has several Quick Start Guides that walk through this process.

Once you are able to browse to your Nagios server and see its web UI and the local services it monitors by default, you are ready to continue.

10.2.2 Setup Distributed Console Servers

This section provides a brief walkthrough on configuring a single *console server* to monitor the status of one attached network host (a Windows IIS server running HTTP and HTTPS services) and one serially attached device (the console port of a network router), and to send alerts back to the Nagios server when an *Administrator* connects to the router or server.

This walkthrough provides an example, but details of the configuration options are described in the next section. This walkthrough also assumes the network host and serial devices are already physically connected to the *console server*. The first step is to set up the Nagios features on the *console server*:

Figure 10-3. Nagios screen.

Browse the Black Box *console server* and select System: Nagios on the *console server* Management Console. Check Nagios service Enabled. Enter the Host Name and the Nagios Host Address (for example, IP address) that the central Nagios server will use to contact the distributed Black Box *console server*.

Enter the IP address that the distributed Black Box *console server* will use to contact the central Nagios server in Nagios Server Address.

Enter the IP address that the clients running SDT Connector will use to connect through the distributed Black Box servers in SDT Gateway address. Check Prefer NRPE, NRPE Enabled, and NRPE Command Arguments.

Check NSCA Enabled, choose an NSCA Encryption Method and enter and confirm an NSCA Secret. Remember these details because you will need them later on. For NSCA Interval, enter: *5.*

Click Apply.

Next, you must configure the attached Window network host and specify the services you will be checking with Nagios (HTTP and HTTPS):

Select Network Hosts from the Serial & Network menu and click Add Host.

Enter the IP Address/DNS Name of the network server, for example: *192.168.1.10* and enter a Description, for example: *Windows 2003 IIS Server.* Remove all Permitted Services. This server will be accessible using Terminal Services, so check TCP, Port 3389 and log level 1 and click Add. Remove and re-add the service to enable logging.
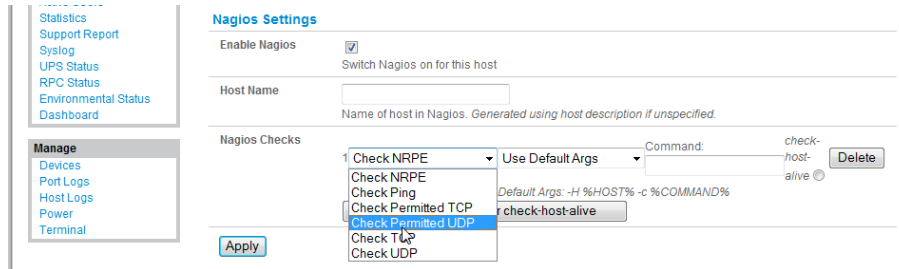
Figure 10-4. Nagios settings screen.

Scroll down to Nagios Settings and check Enable Nagios.

Click New Check and select Check Ping. Click check-host-alive.

Click New Check and select Check Permitted TCP. Select Port 3389

Click New Check and select Check TCP. Select Port 80.

Click New Check and select Check TCP. Select Port 443.

Click Apply.

Similarly, you now must configure the serial port to the router to be monitored by Nagios:

Select Serial Port from the Serial & Network menu.

Locate the serial port that has the router console port attached and click Edit.

Make sure the serial port settings under *Common Settings* are correct and match the attached router's console port.

Click *Console server* Mode, and select Logging Level 1.

Check Telnet (SSH access is not required, as SDT Connector is used to secure the otherwise insecure Telnet connection).

Scroll down to Nagios Settings and check Enable Nagios.

Check Port Log and Serial Status.

Click Apply.

Now you can set the *console server* to send alerts to the Nagios server:

Select Alerts from the Alerts & Logging menu and click Add Alert.

In Description enter: *Administrator connection.*

Check Nagios (NSCA).

In Applicable Ports check the serial port that has the router console port attached. In Applicable Hosts check the IP address/DNS name of the IIS server.

Click Connection Alert.

Click Apply.

Finally, you need to add a *User* for the client running SDT Connector:

Select *Users & Groups* from the *Serial & Network* menu.

Click Add User.

In Username, enter: *sdtnagiosuser*, then enter and confirm a Password.

In Accessible Hosts click the IP address/DNS name of the IIS server, and in Accessible Ports click the serial port that has the router console port attached.

Click Apply.

10.3 Configuring Nagios Distributed Monitoring

To activate the *console server* Nagios distributed monitoring:

Nagios integration must be enabled and a path established to the central/upstream Nagios server.

If the *console server* is to periodically report on Nagios monitored services, then the NSCA client embedded in the *console server* must be configured—the NSCA program enables scheduled check-ins with the remote Nagios server and is used to send passive check results across the network to the remote server.

If the Nagios server is to actively request status updates from the *console server*, then the NRPE server embedded in the *console server* must be configured— the NRPE server is the Nagios daemon for executing plug-ins on remote hosts.

Each of the Serial Ports and each of the Hosts connected to the *console server* that you want to monitor must have Nagios enabled and any specific Nagios checks configured.

Configure the central/upstream Nagios monitoring host.

10.3.1 Enable Nagios on the Console Server

Select System: Nagios on the *console server* Management Console and tick the Nagios service Enabled.

Enter the Nagios Host Name that the *Console server* will be referred to in the Nagios central server—this will be generated from local System Name (entered in System: Administration) if unspecified.

In Nagios Host Address enter the IP address or DNS name that the upstream Nagios server will use to reach the *console server*— if unspecified this will default to the first network port's IP (*Network (1)* as entered in System: IP).

In Nagios Server Address enter the IP address or DNS name that the *console server* will use to reach the upstream Nagios monitoring server.

Check the Disable SDT Nagios Extensions option if you want to disable the SDT Connector integration with your Nagios server at the head end— this would only be checked if you want to run a vanilla Nagios monitoring.

If not, enter the IP address or DNS name that the SDT Nagios clients will use to reach the *console server* in SDT Gateway Address.

When NRPE and NSCA are both enabled, NSCA is preferred method for communicating with the upstream Nagios server— check Prefer NRPE to use NRPE whenever possible (that is, for all communication except for alerts).

10.3.2 Enable NRPE Monitoring
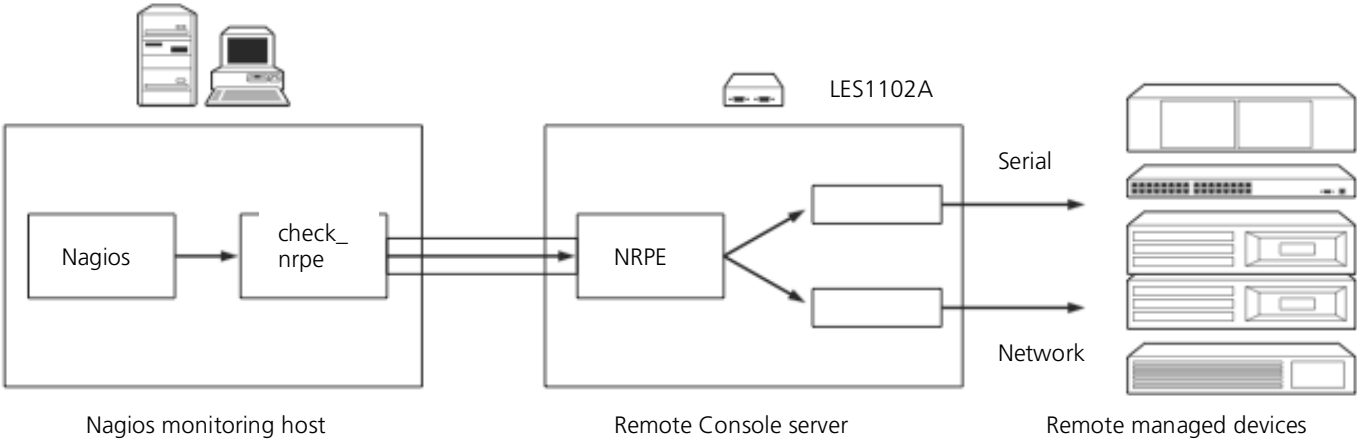
Figure 10-5. NRPE monitoring structure.

Enabling NRPE allows you to execute plug-ins (such as *check_tcp* and *check_ping*) on the remote *Console server* to monitor serial or network attached remote servers. This will offload CPU load from the upstream Nagios monitoring machine. This is especially valuable if you are monitoring hundreds or thousands of hosts. To enable NRPE:

Select System: Nagios and check NRPE Enabled

Enter the details for the user connection to the upstream Nagios monitoring server and again refer to the sample Nagios configuration example below for details about how to configure specific NRPE checks.

By default, the *console server* will accept a connection between the upstream Nagios monitoring server and the NRPE server with SSL encryption, without SSL, or tunneled through SSH. The security for the connection is configured at the Nagios server.

10.3.3 Enable NSCA Monitoring
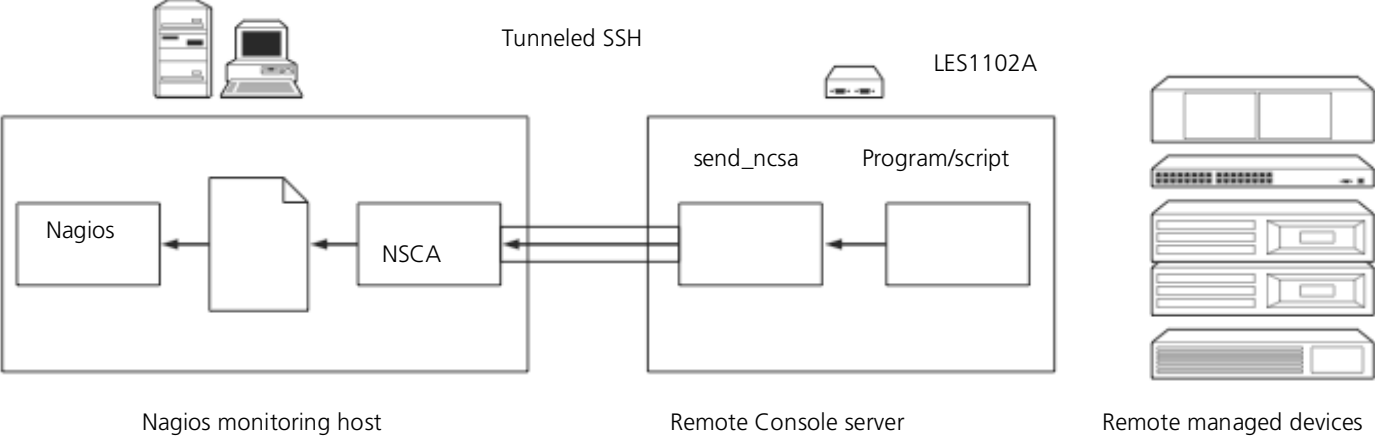
Figure 10-6. NCSA monitoring structure.

NSCA is the mechanism that allows you to send passive check results from the remote *console server* to the Nagios daemon running on the monitoring server. To enable NSCA:

Select System: Nagios and check NSCA Enabled.

Select the Encryption to be used from the drop down menu, then enter a Secret password and specify a check Interval.

Refer to the sample Nagios configuration section below for some examples of configuring specific NSCA checks.

10.3.4 Configure Selected Serial Ports for Nagios Monitoring

The individual Serial Ports connected to the *console server* to be monitored must be configured for Nagios checks. See Chapter 10 for details on enabling Nagios monitoring for Hosts that are network connected to the *console server*. To enable Nagios to monitor a device connected to the *console server* serial port:

Select Serial & Network: Serial Port and click Edit on the serial Port # you want to monitor.

Select Enable Nagios, specify the name of the device on the upstream server and determine the check you want to run on this port. Serial Status monitors the handshaking lines on the serial port and Check Port monitors the data logged for the serial port.

10.3.5 Configure Selected Network Hosts for Nagios Monitoring

The individual Network Hosts connected to the *console server* that you want to monitor must also be configured for Nagios checks:

Select Serial & Network: Network Port and click Edit on the Network Host you want to monitor.

Select Enable Nagios, specify the name of the device as it will appear on the upstream Nagios server.

Click New Check to add a specific check which will be run on this host.

Select Check Permitted TCP/UDP to monitor a service that you have previously added as a Permitted Service.

Select Check TCP/UDP to specify a service port that you want to monitor, without allowing external (SDT Connector) access.

Select Check TCP to monitor.

The Nagios Check nominated as the *check-host-alive* check is the check used to determine whether the network host itself is up or down. Typically this will be *Check Ping*—although in some cases the host will be configured not to respond to pings.

If no *check-host-alive* check is selected, the host will always be assumed to be up.

You may deselect *check-host-alive* by clicking Clear *check-host-alive*.

If required, customize the selected Nagios Checks to use custom arguments.

Click Apply.

10.3.6 Configure the Upstream Nagios Monitoring Host

Refer to the Nagios documentation (http://www.nagios.org/docs/) for configuring the upstream server:

 The section entitled *Distributed Monitoring* steps through what you need to do to configure NSCA on the upstream server (under *Central Server Configuration*).

 NRPE Documentation was recently added that steps through configuring NRPE on the upstream server http://nagios.sourceforge.net/docs/nrpe/NRPE.pdf.

At this stage, Nagios at the upstream monitoring server is configured, and individual serial port and network host connections on the *console server* are configured for Nagios monitoring. If NSCA is enabled, each selected check will be executed once over the period of the check interval. If NRPE is enabled, then the upstream server will be able to request status updates under its own scheduling.

10.4 Advanced Distributed Monitoring Configuration

10.4.1 Sample Nagios Configuration

An example configuration for Nagios is listed below. It shows how to set up a remote *Console server* to monitor a single host, with both network and serial connections. For each check it has two configurations, one each for NRPE and NSCA. In practice, these would be combined into a single check which used NSCA as a primary method, falling back to NRPE if a check was late— for details, see the Nagios documentation (http://www.nagios.org/docs/) on *Service and Host Freshness Checks*

```
.
; Host definitions
;
; Black Box console server
define host{
    use             generic-host
```

```
host_name          Black Box
alias              Console server
address            192.168.254.147
}

; Managed Host
define host{
    use              generic-host
    host_name        server
    alias            server
    address          192.168.254.227
}

; NRPE daemon on gateway
define command {
        command_name    check_nrpe_daemon
        command_line    $USER1$/check_nrpe -H 192.168.254.147 -p 5666
        }

define service {
        service_description  NRPE Daemon
        host_name            Black Box
        use                  generic-service
        check_command        check_nrpe_daemon
        }

; Serial Status
define command {
        command_name    check_serial_status
        command_line    $USER1$/check_nrpe -H 192.168.254.147 -p 5666 -c check_serial_$HOSTNAME$
        }

define service {
        service_description  Serial Status
        host_name            server
        use                  generic-service
        check_command        check_serial_status
        }

define service {
        service_description  serial-signals-server
        host_name            server
        use                  generic-service
        check_command        check_serial_status
        active_checks_enabled   0
        passive_checks_enabled  1
        }

define servicedependency{
        name                          Black Box_nrpe_daemon_dep
        host_name                     Black Box
        dependent_host_name           server
        dependent_service_description         Serial Status
        service_description           NRPE Daemon
        execution_failure_criteria    w,u,c
        }

; Port Log
define command{
        command_name    check_port_log
        command_line    $USER1$/check_nrpe -H 192.168.254.147 -p 5666 -c port_log_$HOSTNAME$
    }

define service {
        service_description  Port Log
        host_name            server
        use                  generic-service
```

```
        check_command          check_port_log
        }


define service {
        service_description  port-log-server
        host_name              server
        use                    generic-service
        check_command          check_port_log
        active_checks_enabled  0
        passive_checks_enabled 1
        }


define servicedependency{
        name                            Black Box_nrpe_daemon_dep
        host_name                       Black Box
        dependent_host_name             server
        dependent_service_description   Port Log
        service_description             NRPE Daemon
        execution_failure_criteria      w,u,c
        }

; Ping
define command{
        command_name    check_ping_via_Black Box
        command_line    $USER1$/check_nrpe -H 192.168.254.147 -p 5666 -c host_ping_$HOSTNAME$
    }


define service {
        service_description  Host Ping
        host_name              server
        use                    generic-service
        check_command          check_ping_via_Black Box
        }


define service {
        service_description  host-ping-server
        host_name              server
        use                    generic-service
        check_command          check_ping_via_Black Box
        active_checks_enabled  0
        passive_checks_enabled 1
        }


define servicedependency{
        name                            Black Box_nrpe_daemon_dep
        host_name                       Black Box
        dependent_host_name             server
        dependent_service_description   Host Ping
        service_description             NRPE Daemon
        execution_failure_criteria      w,u,c
        }

; SSH Port
define command{
    command_name    check_conn_via_Black Box
    command_line    $USER1$/check_nrpe -H 192.168.254.147 -p 5666 -c host_$HOSTNAME$_$ARG1$_$ARG2$
    }


define service {
        service_description  SSH Port
        host_name              server
        use                    generic-service
        check_command          check_conn_via_Black Box!tcp!22
        }


define service {
        service_description  host-port-tcp-22-server
                                        ; host-port-<protocol>-<port>-<host>
        host_name              server
```

724-746-5500 | blackbox.com

```
    use                    generic-service
    check_command          check_conn_via_Black Box!tcp!22
    active_checks_enabled   0
    passive_checks_enabled  1
}

define servicedependency{
    name                          Black Box_nrpe_daemon_dep
    host_name                     Black Box
    dependent_host_name           server
    dependent_service_description SSH Port
    service_description     NRPE Daemon
    execution_failure_criteria  w,u,c
}
```

10.4.2 Basic Nagios Plug-Ins

Plug-ins are compiled executables or scripts that can be scheduled to run on the *console server* to check the status of a connected host or service. This status is then communicated to the upstream Nagios server that uses the results to monitor the current status of the distributed network. Each *console server* is preconfigured with a selection of the checks that are part of the Nagios plug-ins package:

check_tcp and *check_udp* are used to check open ports on network hosts

check_ping is used to check network host availability

check_nrpe is used to execute arbitrary plug-ins in other devices

Each *console server* is preconfigured with two checks that are specific to Black Box:

check_serial_signals is used to monitor the handshaking lines on the serial ports

check_port_log is used to monitor the data logged for a serial port.

10.4.3 Number of Supported Devices

Ultimately the number of devices any particular *console server* can support depends upon the number of checks made, and how often they are performed. Access method will also play a part. The table below shows the performance of three of the *console servers*:

Time	No encryption	3DES	SSH tunnel
NSCA for single check	~ ½ second	~ ½ second	~ ½ second
NSCA for 100 sequential checks	100 seconds	100 seconds	100 seconds
NSCA for 10 sequential checks, batched upload	1 ½ seconds	2 seconds	1 second
NSCA for 100 sequential checks, batched upload	7 seconds	11 seconds	6 seconds

	No encryption	SSL	no encryption - tunneled over existing SSH session
NRPE time to service 1 check	1/10th second	1/3rd second	1/8th second
NRPE time to service 10 simultaneous checks	1 second	3 seconds	1 ¼ seconds
Maximum number of simultaneous checks before timeouts	30	20 (1,2 and 8) or 25 (16 and 48 port)	25 (8 port), 35 (16 and 48 port)

The results were from running tests 5 times in succession with no timeouts on any runs. There are a number of ways to increase the number of checks you can do.

Usually when using NRPE checks, an individual request will need to set up and tear down an SSL connection. This overhead can be avoided by setting up an SSH session to the *console server* and tunneling the NRPE port. This allows the NRPE daemon to run securely without SSL encryption, because SSH will provide the security.

When the *console server* submits NSCA results, it staggers them over a certain time period (for example, 20 checks over 10 minutes will result in two check results every minute). Staggering the results like this means that if the power fails or other incident causes multiple problems, the individual freshness checks will be staggered too.

NSCA checks are also batched. In the previous example, the two checks per minute are sent through in a single transaction.

10.4.4 Distributed Monitoring Usage Scenarios

Below are a number of distributed monitoring Nagios scenarios:

Local office

In this scenario, the *console server* is set up to monitor each managed device's console. Configure it to make a number of checks, either actively at the Nagios server's request, or passively at preset intervals, and submit the results to the Nagios server in a batch.

You can augment the *console server* at the local office site by one or more Intelligent Power Distribution Units (IPDUs) to remotely control the power supply to the managed devices.

Network checks over Ethernet
Serial checks over RS-232
Power monitoring and manipulaton via IPDU

Console server

Hosts

IPDU
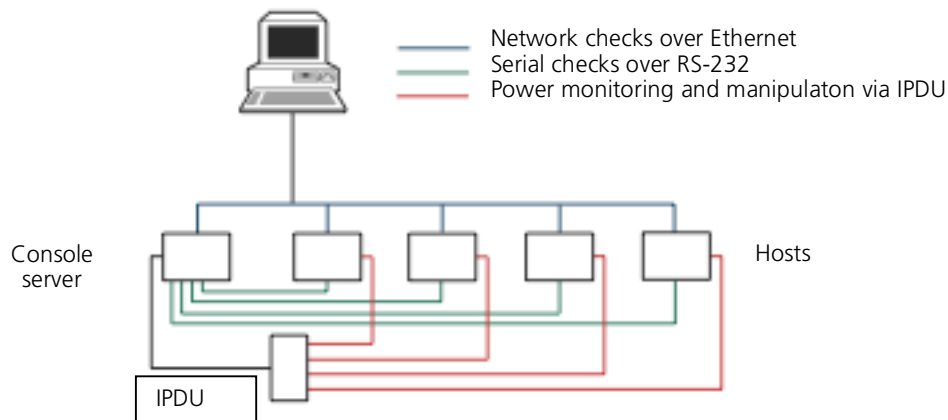
Figure 10-7. Using Nagios in a local office.

Remote site

In this scenario, configure the *console server* NRPE server or NSCA client to actively check configured services and upload the checks to the Nagios server that's waiting passively. You can also configure it to service NRPE commands to perform checks on demand.
In this situation, the *console server* will perform checks based on both serial and network access.

Modem

Nagios

Network checks over Internet
Serial checks over RS-232
Results updated to Nagios via NSCA

Internet

Firewall
Outgoing connections only

Modem
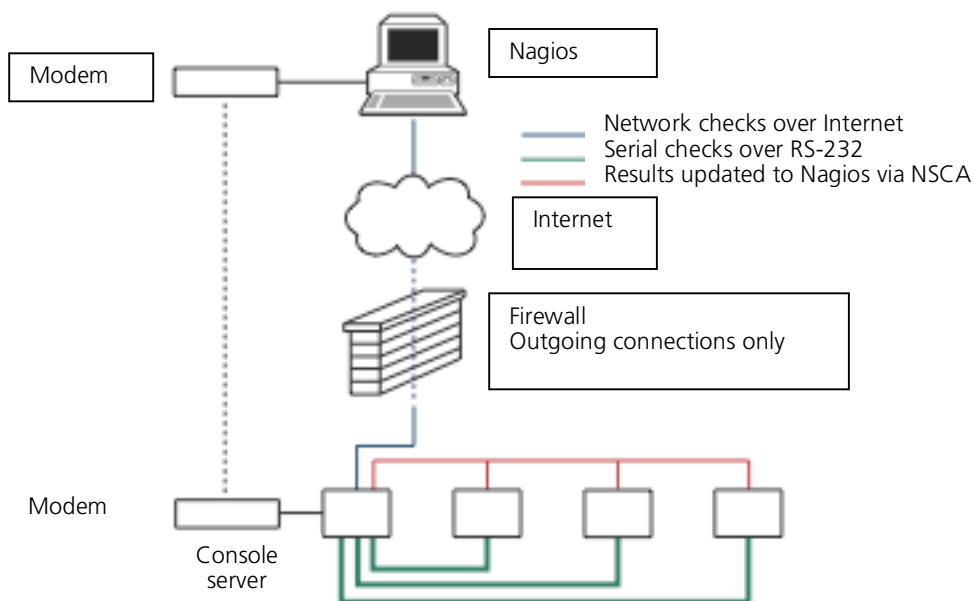
Console server

Figure 10-8. Using Nagios in a remote site.

Remote site with restrictive firewall

In this scenario, the role of the *console server* will vary. One aspect may be to upload check results through NSCA. Another may be to provide an SSH tunnel to allow the Nagios server to run NRPE commands.
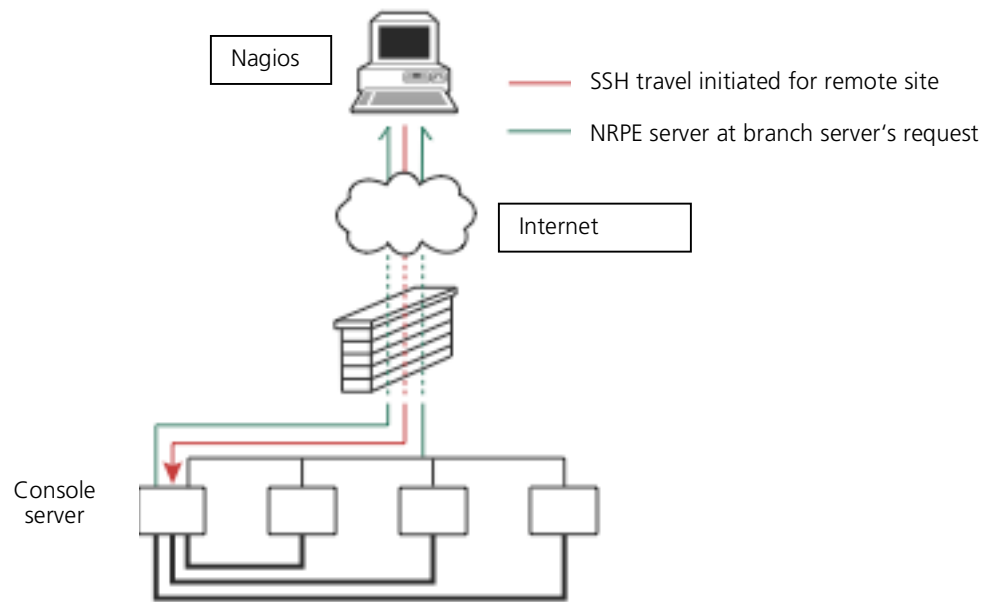
Figure 10-9. Using Nagios in a remote site with a restrictive firewall.

Remote site with no network access

In this scenario the *console server* allows dial-in access for the Nagios server. Periodically, the Nagios server will establish a connection to the *console server* and execute any NRPE commands, before dropping the connection.

11. System Management

This chapter describes how the *Administrator* can perform a range of general *console server* system administration and configuration tasks such as:

• Applying *Soft* and *Hard* Resets to the gateway.

• Re-flashing the Firmware.

• Configuring the Date, Time, and NTP.

• Setting up Backup of the configuration files.

System administration and configuration tasks that are covered elsewhere include:

• Resetting the System Password and entering a new System Name and Description (*Chapter 4.2*).

• Setting the System IP Address (*Chapter 4.3).*

• Setting the permitted Services by which to access the gateway (*Chapter 4.4*).

• Configuring the Dashboard (*Chapter 12*).

11.1 System Administration and Reset

The *Administrator* can reboot or reset the gateway to default settings.

A *soft* reset is affected by:

Selecting Reboot in the System: Administration menu and clicking Apply.

| Syslog
UPS Status
RPC Status
Environmental Status
Dashboard | Reboot | ☐ |
| | | Safely reboot the device. |
| | Apply | |
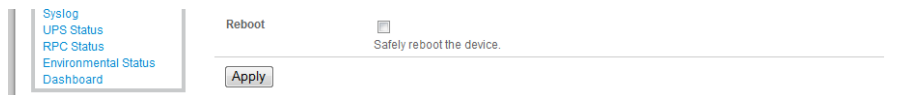
Figure 11-1. Reboot the gateway.

The *console server* reboots with all settings (*for example,* the assigned network IP address) preserved. This *soft* reset disconnects all users and ends any established SSH sessions.

A *soft* reset will also occur when you switch OFF power from the *console server*, and then switch the power back ON. If you cycle the power and the unit is writing to flash, you could corrupt or lose data, so rebooting the software is the safer option.

A *hard* erase (*hard reset*) is performed by:

Pushing the *Erase* button on the rear panel twice. A ball-point pen or bent paper clip is a suitable tool for this procedure. Do not use a graphite pencil. Press the button gently twice (within a couple of seconds) while the unit is powered ON. This will reset the *console server* back to its factory default settings and clear the *console server*'s stored configuration information.

The *hard* erase will clear all custom settings and return the unit back to factory default settings (*i.e.* the IP address will be reset to 192.168.0.1).
You will be prompted to log in and must enter the default administration username and administration password:
Username: root
Password: default
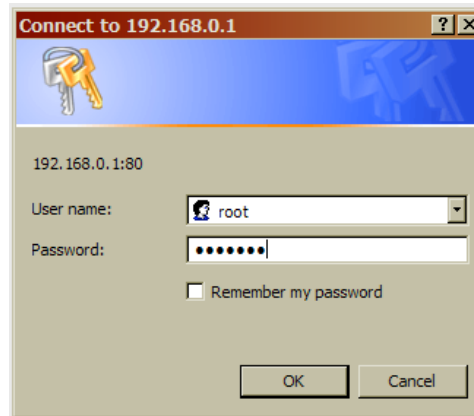
Figure 11-2. Hard erase.

11.2 Upgrade Firmware

Before upgrading, make sure you are already running the most current firmware in your gateway. Your *console server* will not allow you to upgrade to the same or an earlier version.

The Firmware version is displayed in each page's header.

Or select Status: Support Report and note the Firmware Version.
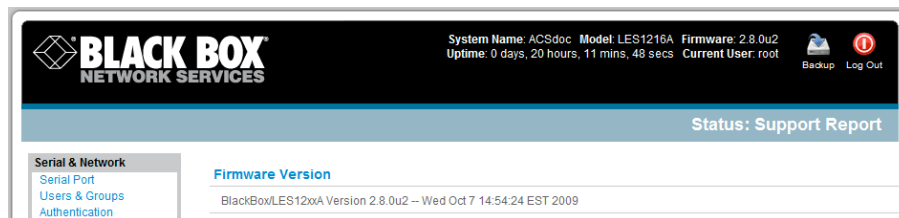
Figure 11-3. Firmware version.

To upgrade, you first must download the latest firmware image from the Black Box.web site.

*S*ave this downloaded firmware image file to a system on the same subnet as the *console server*.

Download and read the *release_notes.txt* for the latest information.

To upload the firmware image file to your *console server*, select System: Firmware.

Specify the address and name of the downloaded Firmware Upgrade File, or Browse the local subnet and locate the downloaded file.
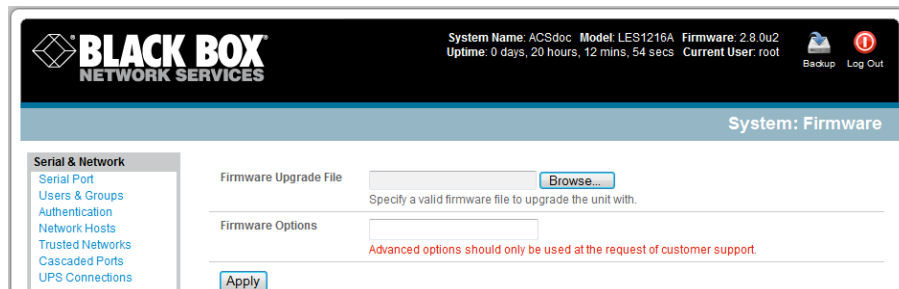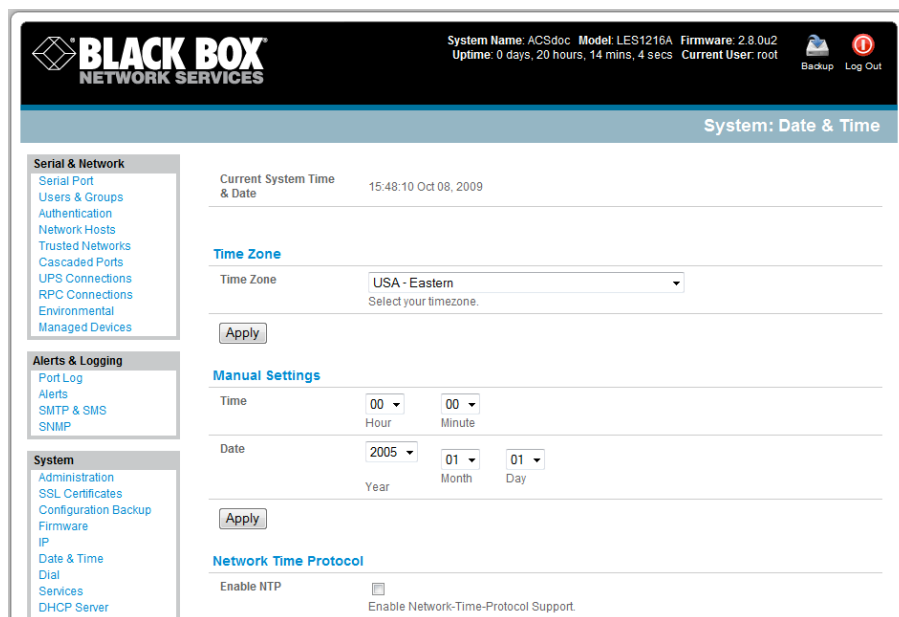
Figure 11-4. Upgrade firmware.

1101 and 1102 Secure Device Servers

Click Apply and the *console server* appliance will perform a soft reboot and start upgrading the firmware. This process will take several minutes. After the firmware upgrade completes, click here to return to the Management Console. Your *console server* will have retained all its pre-upgrade configuration information.

11.3 Configure Date and Time

We recommend that you set the local Date and Time in the *console server* as soon as it is configured. Features like Syslog and NFS logging use the system time for time-stamping log entries, while certificate generation depends on a correct *Timestamp* to check the validity period of the certificate.

Figure 11-5. Set date and time.

Select the System: Date & Time menu option.

Manually set the Year, Month, Day, Hour and Minute using the Date and Time selection boxes, then click Apply.

The gateway can synchronize its system time with a remote time server using the Network Time Protocol (NTP). Configuring the NTP time server ensures that the *console server* clock will be accurate soon after the Internet connection is established. Also if NTP is not used, the system clock will reset randomly every time the *console server* is powered up. To set the system time using NTP:

Select the Enable NTP checkbox on the Network Time Protocol page.

Enter the IP address of the remote NTP Server and click Apply.

You must now also specify your local time zone so the system clock can show local time (and not UTP):

Set your appropriate region/locality in the Time Zone selection box and click Apply.

11.4 Configuration Backup

We recommend that you back up the *console server* configuration whenever you make significant changes (such as adding new Users or Managed Devices) or before performing a firmware upgrade.

Select the System: Configuration Backup menu option or click the [Backup] icon.

NOTE: You can also back up the configuration files from the command line (refer to Chapter 14).
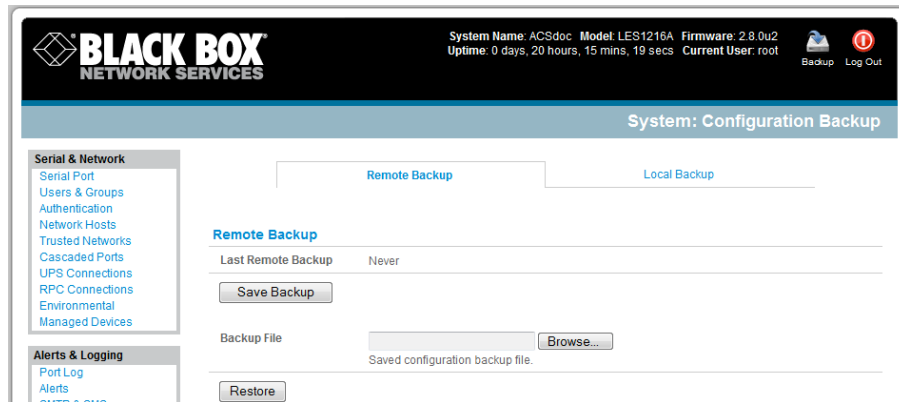
Figure 11-6. Configuration backup screen.

With all *console servers*, you can save the backup file remotely on your PC and you can restore configurations from remote locations: Click Save Backup in the Remote Configuration Backup menu.

The config backup file (*System Name_date_config.opg*) will be downloaded to your PC and saved in the location you nominate. To restore a remote backup:

Click Browse in the Remote Configuration Backup menu and select the Backup File you want to restore.

Click Restore and click OK. This will overwrite all the current configuration settings in your *console server*.

After saving a local configuration backup, you may choose to use it as the alternate default configuration. When the *console server* is reset to factory defaults, it will then load your alternate default configuration instead of its factory settings:

To set an alternate default configuration, check Load On Erase and click Apply.

NOTE: Before selecting *Load On Erase*, make sure that you have tested your alternate default configuration by clicking Restore.

If your alternate default configuration causes the *console server* to not boot, recover your unit to factory settings using the following steps:

If the configuration is stored on an external USB storage device, unplug the storage device and reset to factory defaults (see Section 11.1).

If the configuration is stored on an internal USB storage device, reset it to factory defaults using a specially prepared USB storage device:

The USB storage device must be formatted with a Windows FAT32/VFAT file system on the first partition or the entire disk; most USB thumb drives are already formatted this way.

The file system must have the volume label: OPG_DEFAULT.

Insert this USB storage device into an external USB port on the *console server* and reset to factory defaults as described in Section 12.1.

- After recovering your *console server*, make sure the problem configuration is no longer selected for Load On Erase.

1101 and 1102 Secure Device Servers

12. Status Reports

This chapter describes the dashboard feature and the status reports that are available:

• Port Access and Active Users

• Statistics

• Support Reports

• Syslog

• Dashboard

Other status reports that are covered elsewhere include:

• UPS Status (*Chapter 8.2*)

• RPC Status (*Chapter 8.1*)

12.1 Port Access and Active Users

The *Administrator* can see which *Users* have access privileges with which serial ports:

Select the **Status: Port Access**
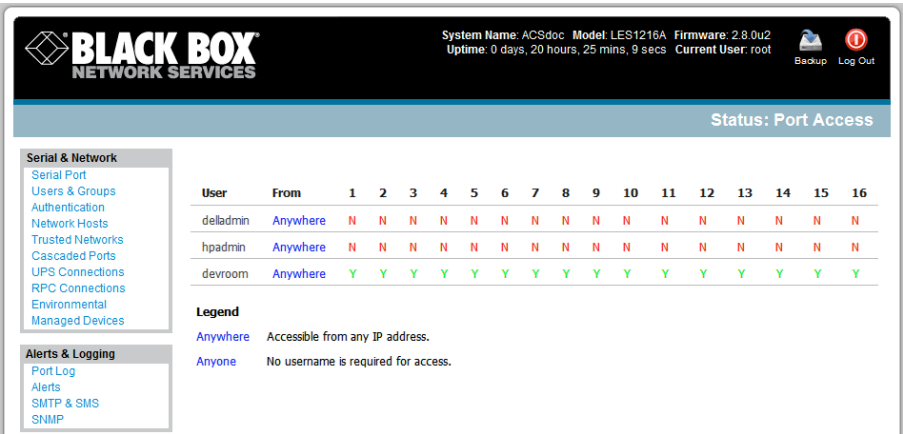
Figure 12-1. Port access status screen.

The *Administrator* can also see the current status as to *Users* who have active sessions on those ports:

Select the **Status: Active Users**

12.2 Statistics

The Statistics report provides a snapshot of the status, current traffic, and other activities and operations of your *console server*:

Select the **Status: Statistics**

Figure 12-2. Statistics status.

You can find detailed statistics reports by selecting the various submenus.

12.3 Support Reports

The Support Report provides useful status information that will assist the Black Box Technical Support team to solve any problems you may experience with your *console server*.

If you do experience a problem and have to contact tech support, make sure you include the Support Report with your email support request. The Support Report is generated when the issue is occurring, and is attached in plain text format.
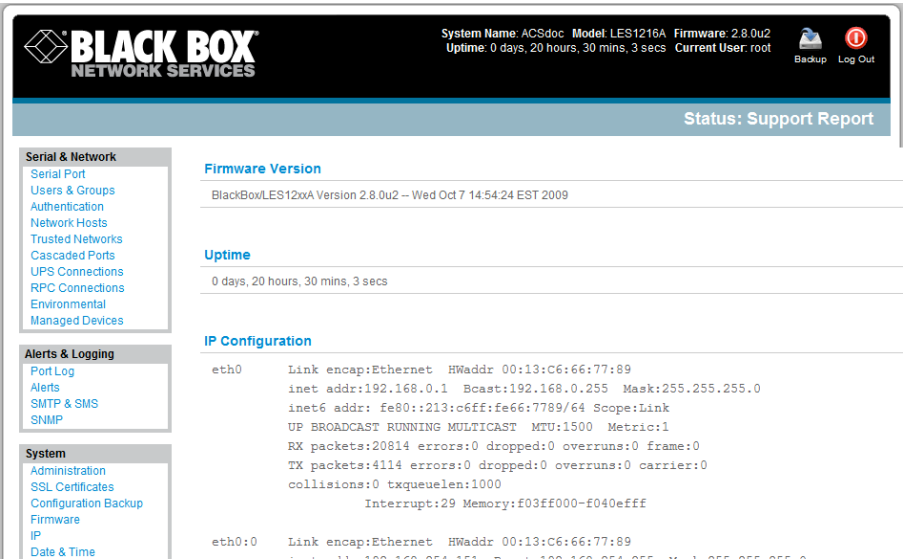
Figure 12-3. Support report.

Select **Status: Support Report** and you will be presented with a status snapshot.

Save the file as a text file and attach it to your support email.

12.4 Syslog

The Linux System Logger in the *console server* maintains a record of all system messages and errors:

Select **Status: Syslog**

You can redirect the syslog record to a remote Syslog Server:

Enter the remote **Syslog Server Address** and **Syslog Server Port** details and click **Apply.**

The console maintains a local Syslog. To view the local Syslog file:

Select **Status: Syslog**

To make it easier to find information in the local Syslog file, use the provided pattern matching filter tool.

Specify the **Match Pattern** that you want to search for (*for example,* the search for *mount* is shown below) and click **Apply.** The Syslog will then be represented with only those entries that actually include the specified pattern.

Figure 12-4. Syslog specified by match pattern.

12.5 Dashboard

The Dashboard provides the *Administrator* with a summary of the status of the *console server* and its Managed Devices. You can configure custom dashboards for each user group.

Figure 12-5. Dashboard summary.

12.5.1 Configuring the Dashboard

Only users who are members of the *admin* group (and the *root* user) can configure and access the dashboard. To configure a custom dashboard:

Select **System: Configure Dashboard** and select the user (or group) you are configuring this custom dashboard layout for.

Click **Next.**
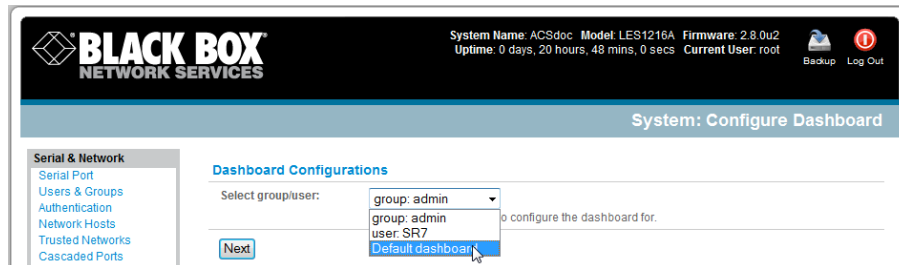
Figure 12-6. Custom dashboard.

NOTE: You can configure a custom dashboard for any admin user or for the admin group or you can reconfigure the default dashboard.

The Status:Dashboard screen is the first screen displayed when admin users (other than root) log into the console manager. If you log in as "John," and John is member of the admin group and there is a dashboard layout configured for John, then you will see the dashboard for John upon log-in and each time you click on the Status:Dashboard menu item.

If there is no dashboard layout configured for John, but there is an admin group dashboard configured, then you will see the admin group dashboard instead. If there is no user dashboard or admin group dashboard configured, then you will see the default dashboard.

The root user does not have its own dashboard.

Use the above configuration options to enable admin users to setup their own custom dashboards.

The Dashboard displays six *widgets*. These widgets include each of the Status screens (alerts, devices, ports ups, rpc, and environmental status) and a custom script screen. The *admin* user can configure which of these widget is to be displayed where:

Go to the **Dashboard layout** panel and select which widget is to be displayed in each of the six display locations (widget1 …6).

Click **Apply.**
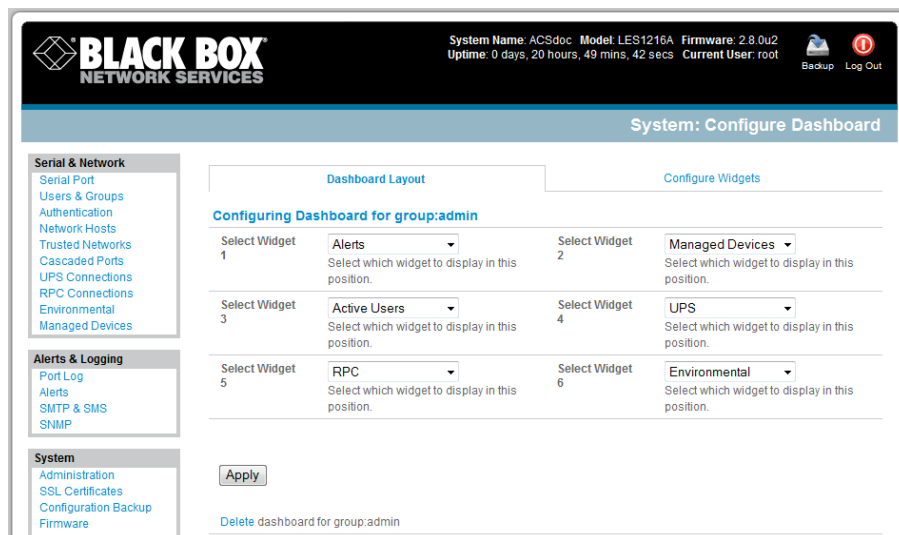
Figure 12-7. Configure dashboard screen.

NOTE: The Alerts widget is a new screen that shows the current alerts status. When an alert gets triggered, a corresponding .XML file is created in /var/run/alerts/. The dashboard scans all these files and displays a summary status in the alerts widget. When an alert is deleted, the corresponding .XML files that belong to that alert are also deleted.

1101 and 1102 Secure Device Servers

To configure what is to be displayed by each widget:

Go to the **Configure widgets** panel and configure each selected widget (for example, specify which UPS status is to be displayed on the *ups widget* or the maximum number of Managed Devices to be displayed in the *devices widget*).

Click **Apply.**
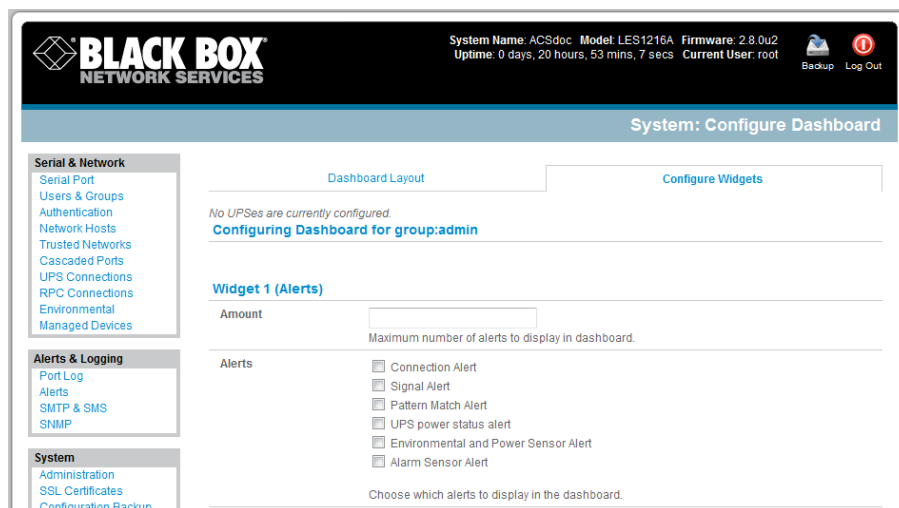
Figure 12-8. Configure widgets.

NOTE: Dashboard configuration is stored in the /etc/config/config.xml file. Each configured dashboard will increase the config file. If this file gets too big, you can run out of memory space on the console manager.

12.5.2 Creating Custom Widgets for the Dashboard

To run a custom script inside a dashboard widget:

Create a file called "*widget-<name>.sh*" in the folder */etc/config/scripts/* where *<name>* can be anything. You can have as many custom dashboard files as you want.

Inside this file you can put any code you want. When configuring the dashboard, choose "*widget-<name>.sh*" in the dropdown list. The dashboard will run the script and display the output of the script commands directly on the screen, inside the specific widget.

The best way to format the output would be to send HTML commands back to the browser by adding echo commands in the script:

echo '<table>'

You can of course run any command and its output will be displayed in the widget window directly.

Below is an example script that writes the current date to a file, and then echos HTML code back to the browser. The HTML code gets an image from a specific URL and displays it in the widget.

#!/bin/sh

date >> /tmp/test
echo '<table>'
echo '<tr><td> This is my custom script running </td></tr>'
echo '<tr><td>'
echo '<img src="http://www.vinras.com/images/linux-online-inc.jpg">'
echo '</td></tr>'
echo '</table>'

exit 0

13. Management

The *console server* has a small number of **Manage** reports and tools that are available to both *Administrator*s and *Users*:

• Access and control authorized devices.

• View serial port logs and host logs for those devices.

• Use SDT Connector or the java terminal to access serially attached consoles.

• Control power devices (where authorized).

All other Management Console menu items are available to *Administrators* only.

13.1 Device Management

To display the Managed Devices and their associated serial, network, and power connections:

Select **Manage: Devices.** The *Administrator* will be presented with a list of all configured Managed Devices, whereas the *User* will only see the Managed Devices they (or their Group) has been given access privileges for.
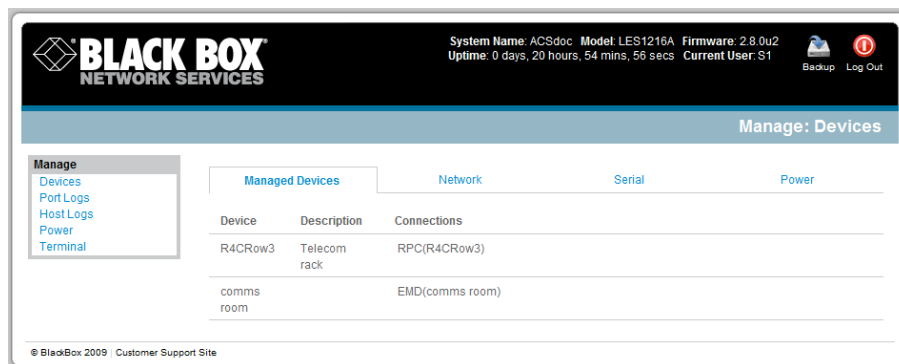
Figure 13-1. Managed devices screen.

Select **Serial Network** or **Power** for a view of the specific connections. The user can then take a range of actions using these serial, network or power connections by selecting the **Action** icon or the related Manage menu item. (For example, selecting the *Manager Power* icon [or **Manage: Power** from the menu] would enable the user to power Off/On/Cycle any power outlet on any PDU the user has been given access privileges to [refer to *Chapter 8* for details]).
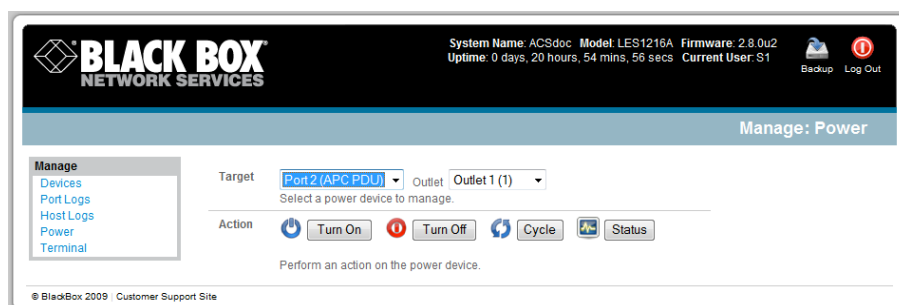
Figure 13-2. Viewing specific connections.

13.2 Port and Host Logs

Administrators and *Users* can view logs of data transfers to connected devices.

Select **Manage: Port Logs** and the serial Port # to be displayed.
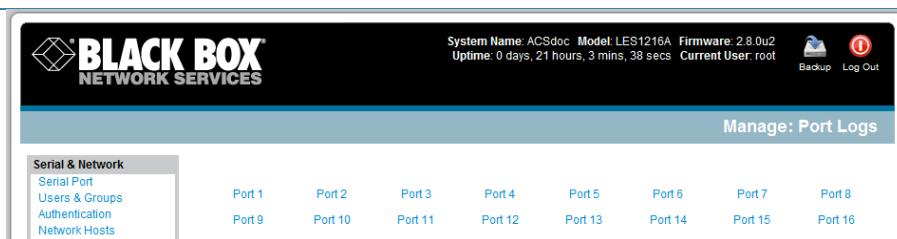
Figure 13-3. Port logs.

To display Host logs, select **Manage: Host Logs** and the Host to be displayed.

13.3 Serial Port Terminal Connection

Administrator and *Users* can communicate directly with the *console server* command line and with devices attached to the *console server* serial ports using SDT Connector and their local tenet client, or use a java terminal in their browser.

Select **Manage: Terminal.**
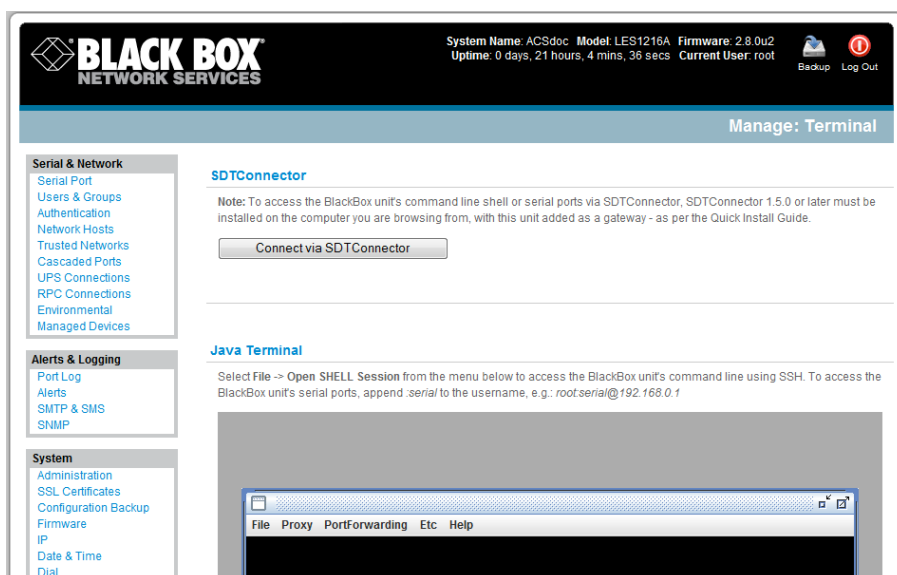
Figure 13-4. Managing terminal.

Click **Connect to SDT Connector** to access the *console server's* command line shell or the serial ports via SDT Connector. This will to activate the SDT Connector client on the computer you are browsing from and load your local telnet client to connect to the command line or serial port using SSH.
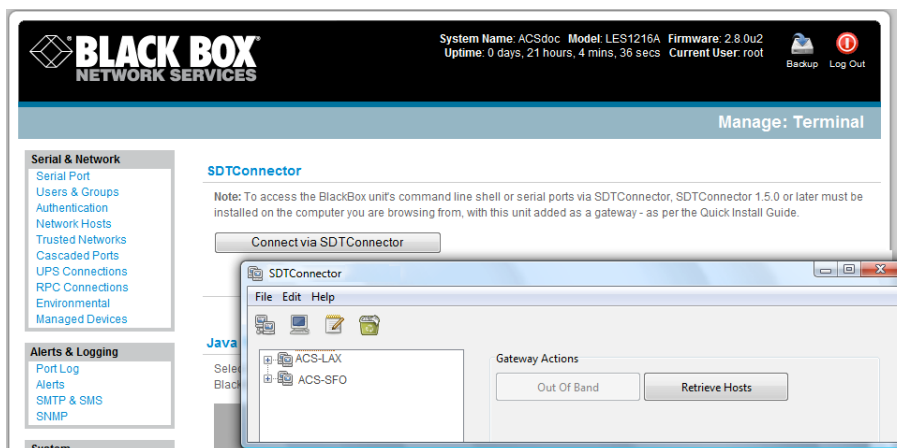
Figure 13-5. Connect to SDT connector.

NOTE: You must install SDT Connector on the computer you are browsing from and add and the *console server* as a gateway as detailed in Chapter 6.

The alternate to using SDT Connector and your local telnet client is to run the open source *jcterm* java terminal applet into your browser to connect to the *console server* and attached serial port devices. *jcterm* does have some JRE compatibility issues that may prevent it from loading.

Select **Manage: Terminal.** The *jcterm* java applet is downloaded from the *console server* to your browser and the virtual terminal will be displayed.

Select **File** -> **Open SHELL Session** from the *jcterm* menu to access the command line using SSH.

To access the *console server's* command line, enter its TCP address (e.g. *192.168.254.198*) as **hostname** and the Username, for example, *root@192.168.254.198*. Then enter the Password.

To access the *console server*'s serial ports, append *:serial* to the username. With the gateway's TCP address (for example, *192.168.254.198*), the Username (for example, *root*), enter *root:serial@192.168.254.198.* Then enter Password and select the TCP Port address for the serial port to be accessed. By default *3001* is selected (that is, Port 1). To access Port 4 for example, change this to *3004* for the Username.

13.4 Power Management

Administrators and *Users* can access and manage the connected power devices.

Select **Manage: Power**

14. Configuration from the Command Line

For those who prefer to configure their *console server* at the Linux command line level (rather than use a browser and the Management Console), this chapter describes how to use command line access and the **config** tool to manage the *console server* and configure the ports, etc.

This *config* documentation in this chapter walks through command line configuration to deliver the functions provided using the Management Console GUI.

For advanced and custom configurations and for details using other tools and commands, refer to the next chapter.

When displaying a command, the convention used in the rest of this chapter is to use single quotes ('') for user-defined values (for example, descriptions and names). Element values without single quotes must be typed exactly as shown.

After the initial section on accessing the *config* command, the menu items in this document follow the same structure as the menu items in the web GUI.

14.1 Accessing *config* from the Command Line

The *console server* runs a standard Linux kernel and embeds a suite of open source applications. If you do not want to use a browser and the Management Console tools, you can configure the *console server* and manage connected devices from the command line using standard Linux commands and applications such as *ifconfig, gettyd, stty, powerman, nut etc*. Without care, these configurations may not withstand a *power-cycle-reset* or *reconfigure*.

Black Box provides a number of custom command line utilities and scripts to make it simple to configure the *console server* and make sure the changes are stored in the *console server*'s flash memory, etc.

In particular, the **config** utility allows you to manipulate the system configuration from the command line. With *config,* you can activate a new configuration by running the relevant configurator, which performs the action needed to make the configuration changes live.

To access *config* from the command line:

Power on the *console server* and connect the "terminal" device:

If you are connecting using the serial line, plug a serial cable between the *console server* local DB9 console port and terminal device. Configure the serial connection of the terminal device you are using to 115200 bps, 8 data bits, no parity, and one stop bit.

If you are connecting over the LAN, then you will need to interconnect the Ethernet ports and direct your terminal emulator program to the IP address of the *console server* (192.168.0.1 by default).

Log on to the *console server* by pressing "return" a few times. The *console server* will request a username and password. Enter the username *root* and the password *default*. You should now see the command line prompt which is a hash (#).

This chapter is not intended to teach you Linux. We assume you already have a certain level of understanding before you execute Linux kernel level commands.

The *config* tool

Syntax

config [-ahv] [-d id] [-g id] [-p path] [-r configurator] [-s id=value] [-P id]

Description

The *config* tool is designed to perform multiple actions from one command if needed, so options can be chained together.

The *config* tool allows you to manipulate and query the system configuration from the command line. Using *config,* you can activate the new configuration by running the relevant *configurator* that performs the action needed to make the configuration changes live.

The custom user configuration is saved in the */etc/config/config.xml* file. This file is transparently accessed and edited when configuring the device using the Management Console browser GUI. Only the user *"root"* can configure from the shell.

By default, the *config* elements are separated by a '.' character. The root of the *config* tree is called *<config>*. To address a specific element place a '.' between each node/branch. For example, to access and display the description of *user1* type:

config -g config.users.user1.description

The root node of the *config* tree is <config>. To display the entire *config* tree, type:

config -g config

To display the help text for the *config* command, type:

config –h

The *config* application resides in the */bin* directory. The environmental variable called *PATH* contains a route to the */bin* directory. This allows a user to simply type *config* at the command prompt instead of the full path */bin/config*.

Options

-a –run-all	Run all registered configurators. This performs every configuration synchronization action pushing all changes to the live system.
-h –help	Display a brief usage message.
-v –verbose	Log extra debug information.
-d –del=id	Remove the given configuration element specified by a '.' separated identifier.
-g –get=id	Display the value of a configuration element.
-p –path=file	Specify an alternate configuration file to use. The default file is located at */etc/config/config.xml.*
-r –run=configurator	Run the specified registered configurator. Registered configurators are listed below.
-s --set=id=value	Change the value of configuration element specified by a '.' separated identifier.
-e --export=file	Save active configuration to file.
-i --import=file	Load configuration from file.
-t --test-import=file	Pretend to load configuration from file.
-S --separator=char	The pattern to separate fields with, default is '.'
-P --password=id	Prompt user for a value. Hash the value, then save it in id.

The registered configurators are:

alerts	*ipconfig*
auth	*nagios*
cascade	*power*
console	*serialconfig*
dhcp	*services*
dialin	*slave*
eventlog	*systemsettings*
hosts	*time*
ipaccess	*ups*
	users

There are three ways to delete a config element value. The simplest way is use the *delete-node* script detailed later in this chapter. You can also assign the config element to " ", or delete the entire config node using *-d*:

> # */bin/config -d 'element name'*

All passwords are saved in plaintext *except* the user passwords and the system passwords, which are encrypted.

NOTE: The config command does not verify whether the nodes edited/added by the user are valid. This means that any node may be added to the tree. If a user runs the following command:

> # */bin/config -s config.fruit.apple=sweet*

The configurator will not complain, but this command is useless. When the configurators are run (to turn the config.xml file into live config) they will simply ignore this <fruit> node. Administrators must make sure of the spelling when typing config commands. Incorrect spelling for a node will not be flagged.

Most configurations made to the XML file will be immediately active. To make sure that *all* configuration changes are active, especially when editing user passwords, run all the configurators:

> # */bin/config -a*

For information on backing up and restoring the configuration file, refer to *Chapter 15, Advanced Configuration*.

14.2 Serial Port Configuration

The first set of configurations you need to make to any serial port are the RS-232 common settings. For example, setup serial port 5 to use the following properties:

Baud Rate	*9600*
Parity	*None*
Data Bits	*8*
Stop Bits	*1*
label	*Myport*
log level	*0*
protocol	*RS232*
flow control	*None*

To do this, use the following commands:

> # *config -s config.ports.port5.speed=9600*
> # *config -s config.ports.port5.parity=None*
> # *config -s config.ports.port5.charsize=8*
> # *config -s config.ports.port5.stop=1*
> # *config -s config.ports.port5.label=myport*
> # *config -s config.ports.port5.loglevel=0*
> # *config -s config.ports.port5.protocol=RS232*
> # *config -s config.ports.port5.flowcontrol=None*

The following command will synchronize the live system with the new configuration:

> # *config -r serialconfig*

NOTE: Supported serial port baud-rates are '50', '75', '110', '134', '150', '200', '300', '600', '1200', '1800', '2400', '4800', '9600', '19200', '38400', '57600', '115200', and '230400'.

Supported parity values are 'None', 'Odd', 'Even', 'Mark' and 'Space'.

Supported data-bits values are '8', '7', '6', and '5'.

Supported stop-bits values are '1', '1.5', and '2'.

Supported flow-control values are 'Hardware', 'Software', and 'None'.

Additionally, before any port can function properly, you need to set the port mode. Set any port to run in one of the five possible modes (refer *Chapter 5* for details): [*Console server* mode|Device mode|SDT mode|Terminal server mode|Serial bridge mode]. All these modes are mutually exclusive.

Console server mode

The command to set the port in *portmanager* mode:

> *# config -s config.ports.port5.mode=portmanager*

To set the following optional config elements for this mode:

Data accumulation period	*100 ms*
Escape character	*% (default is ~)*
log level	*2 (default is 0)*
Shell power command menu	*Enabled*
RFC2217 access	*Enabled*
Limit pot to 1 connection	*Enabled*
SSH access	*Enabled*
TCP access	*Enabled*
telnet access	*Disabled*
Unauthorized telnet access	*Disabled*

> *# config -s config.ports.port5.delay=100*
> *# config -s config.ports.port5.escapechar=%*
> *# config -s config.ports.port5.loglevel=2*
> *# config -s config.ports.port5.powermenu=on*
> *# config -s config.ports.port5.rfc2217=on*
> *# config -s config.ports.port5.singleconn=on*
> *# config -s config.ports.port5.ssh=on*
> *# config -s config.ports.port5.tcp=on*
> *# config -d config.ports.port5.telnet*
> *# config -d config.ports.port5.unauthtel*

Device Mode

For a device mode port, set the port type to *ups, rpc,* or *enviro:*

> *# config -s config.ports.port5.device.type=[ups | rpc | enviro]*

For port 5 as a UPS port:

> *# config -s config.ports.port5.mode=reserved*

For port 5 as an RPC port:

> *# config -s config.ports.port5.mode=powerman*

For port 5 as an Environmental port:

> *# config -s config.ports.port5.mode=reserved*

SDT mode

To enable access over SSH to a host connected to serial port 5:

> *# config -s config.ports.port5.mode=sdt*

> *# config -s config.ports.port5.sdt.ssh=on*

To configure a username and password when accessing this port with Username = user1 and Password = secret:

> *# config -s config.ports.port#.sdt.username=user1*
> *# config -s config.ports.port#.sdt.password=secret*

Terminal server mode

Enable a TTY login for a local terminal attached to serial port 5:

> *# config -s config.ports.port5.mode=terminal*
> *# config -s config.ports.port5.terminal=[vt220 | vt102 | vt100 | linux | ansi]*

The default terminal is vt220.

Serial bridge mode

Create a network connection to a remote serial port via RFC-2217 on port 5:

> *# config -s config.ports.port5.mode=bridge*

Optional configurations for the network address of RFC-2217 server of 192.168.3.3 and TCP port used by the RFC-2217 service = 2500:

> *# config -s config.ports.port5.bridge.address=192.168.3.3*
> *# config -s config.ports.port5.bridge.port=2500*

To enable RFC-2217 access: *# config -s config.ports.port5.bridge.rfc2217=on*

To redirect the serial bridge over an SSH tunnel to the server: *# config -s config.ports.port5.bridge.ssh.enabled=on*

Syslog settings

Additionally, the global system log settings can be set for any specific port, in any mode:

 # config -s config.ports.port#.syslog.facility='facility'
'facility' can be:
> *Default*
> *local 0-7*
> *auth*
> *authpriv*
> *cron*
> *daemon*
> *ftp*
> *kern*
> *lpr*
> *mail*
> *news*
> *user*
> *uucp*

 # config -s config.ports.port#.syslog.priority='priority'
'priority' can be:
> *Default*
> *warning*
> *notice*
> *Info*
> *error*
> *emergency*
> *debug*
> *critical*
> *alert*

14.3 Adding and Removing Users

First, determine the total number of existing Users (if you have no existing Users you can assume this is 0):

> *# config -g config.users.total*

This command should display *config.users.total 1*. Note that if you see *config.users.total*, this means you have 0 Users configured.

724-746-5500 | blackbox.com

Your new User will be the existing total plus 1. If the previous command gave you 0, then you start with user number 1. If you already have 1 user your new user will be number 2, etc.

To add a user (with Username=John, Password=secret and Description=mySecondUser) issue the commands:

> # config -s config.users.total=2 (assuming we already have 1 user configured)
> # config -s config.users.user2.username=John
> # config -s config.users.user2.description=mySecondUser
> # config -P config.users.user2.password

NOTE: The -P parameter will prompt the user for a password, and encrypt it. You can encrypt the value of any config element using the -P parameter, but only encrypted user passwords and system passwords are supported. If any other element value were to be encrypted, the value will become inaccessible and will have to be reset.

To add this user to specific groups (admin/users):

> # config -s config.users.user2.groups.group1='groupname'
> # config -s config.users.user2.groups.group2='groupname2'
> etc...

To give this user access to a specific port:

> # config -s config.users.user2.port1=on
> # config -s config.users.user2.port2=on
> # config -s config.users.user2.port5=on
> etc...

To remove port access:

> # config -s config.users.user2.port1=''   (the value is left blank)
> or simply:
> # config -d config.users.user2.port1

The port number can be anything from 1 to 48, depending on the available ports on the specific *console server*.

For example, assume we have an RPC device connected to port 1 on the *console server* and the RPC is configured. To give this user access to RPC outlet number 3 on the RPC device, run the 2 commands below:

> # config -s config.ports.port1.power.outlet3.users.user2=John
> # config -s config.ports.port1.power.outlet3.users.total=2 (total number of users that have access to this outlet)

If more users are given access to this power outlet, then increment the 'config.ports.port1.power.outlet3.users.total' element accordingly.

To give this user access to network host 5 (assuming the host is configured):

> # config -s config.sdt.hosts.host5.users.user1=John
> # config -s config.sdt.hosts.host5.users.total=1 (total number of users having access to host)

To give another user called "Peter" access to the same host:

> # config -s config.sdt.hosts.host5.users.user2=Peter
> # config -s config.sdt.hosts.host5.users.total=2 (total number of users having access to host)

To edit any of the user element values, use the same approach as when adding user elements; that is, use the "-s" parameter. If any of the config elements do not exist, they will automatically be created.

To delete the user called John, use the delete-node script:

> # ./delete-node config.users.user2

The following command will synchronize the live system with the new configuration:

> # config -r users

14.4 Adding and Removing User Groups

The *console server* is configured with a few default user groups (even though only two of these groups are visible in the Management Console GUI). To find out how many groups are already present:

> *# config -g config.groups.total*

Assume this value is six. Make sure you number any new groups you create from seven and up.

To add a custom group to the configuration with Group name=Group7, Group description=MyGroup and Port access= 1,5 you'd issue the commands:

> *# config -s config.groups.group7.name=Group7*
> *# config -s config.groups.group7.description=MyGroup*
> *# config -s config.groups.total=7*
> *# config -s config.groups.group7.port1=on*
> *# config -s config.groups.group7.port5=on*

Assume we have an RPC device connected to port 1 on the console manager, and the RPC is configured. To give this group access to RPC outlet number 3 on the RPC device, run the two commands below:

> *# config -s config.ports.port1.power.outlet3.groups.group1=Group7*
> *# config -s config.ports.port1.power.outlet3.groups.total=1 (total number of groups that have access to this outlet)*

If more groups are given access to this power outlet, then increment the '*config.ports.port1.power.outlet3.groups.total*' element accordingly.

To give this group access to network host 5:

> *# config -s config.sdt.hosts.host5.groups.group1=Group7*
> *# config -s config.sdt.hosts.host5.groups.total=1 (total number of groups having access to host)*

To give another group called 'Group8' access to the same host:

> *# config -s config.sdt.hosts.host5.groups.group2=Group8*
> *# config -s config.sdt.hosts.host5.groups.total=2 (total number of users having access to host)*

To delete the group called Group7, use the following command:

> *# rmuser Group7*

Attention: The *rmuser* script is a generic script to remove any config element from config.xml correctly. However, any dependencies or references to this group will not be affected. Only the group details are deleted. The *Administrator* is responsible for going through *config.xml* and removing group dependencies and references manually, specifically if the group had access to a host or RPC device.
The following command will synchronize the live system with the new configuration:

> *# config –a*

14.5 Authentication

To change the type of authentication for the *console server*:

config -s config.auth.type='authtype'
'authtype' can be:
> *Local*
> *LocalTACACS*
> *TACACS*
> *TACACSLocal*
> *TACACSDownLocal*
> *LocalRADIUS*
> *RADIUS*
> *RADIUSLocal*
> *RADIUSDownLocal*
> *LocalLDAP*
> *LDAP*
> *LDAPLocal*
> *LDAPDownLocal*

To configure TACACS authentication:

> # config -s config.auth.tacacs.auth_server='comma separated list' (list of remote authentiction and authorization servers.)
> # config -s config.auth.tacacs.acct_server='comma separated list' (list of remote accounting servers. If unset, Authentication and Authorization Server Address will be used.)
> # config -s config.auth.tacacs.password='password'

To configure RADIUS authentication:

> # config -s config.auth.radius.auth_server='comma separated list' (list of remote authentiction and authorization servers.)
> # config -s config.auth.radius.acct_server='comma separated list' (list of remote accounting servers. If unset, Authentication and Authorization Server Address will be used.)
> # config -s config.auth.radius.password='password'

To configure LDAP authentication:

> # config -s config.auth.ldap.server='comma separated list' (list of remote servers.)
> # config -s config.auth.ldap.basedn='name' (The distinguished name of the search base. For example: dc=my-company,dc=com)
> # config -s config.auth.ldap.binddn='name' (The distinguished name to bind to the server with. The default is to bind anonymously.)
> # config -s config.auth.radius.password='password'

The following command will synchronize the live system with the new configuration:

> # config -r auth

14.6 Network Hosts

To determine the total number of currently configured hosts:

> # config -g config.sdt.hosts.total

Assume this value is equal to 3. If you add another host, make sure you increment the total number of hosts from 3 to 4:

> # config -s config.sdt.hosts.total=4

If the output is *config.sdt.hosts.total* then assume 0 hosts are configured.

Add power device host

To add a UPS/RPC network host with the following details:

IP address/ DNS name	192.168.2.5
Host name	remoteUPS
Description	UPSroom3
Type	UPS
Allowed services	ssh port 22 and https port 443
Log level for services	0

Issue the commands below:

> # config -s config.sdt.hosts.host4.address=192.168.2.5
> # config -s config.sdt.hosts.host4.name=remoteUPS
> # config -s config.sdt.hosts.host4.description=UPSroom3
> # config -s config.sdt.hosts.host4.device.type=ups
> # config -s config.sdt.hosts.host4.tcpports.tcpport1=22
> # config -s config.sdt.hosts.host4.tcpports.tcpport1.loglevel=0
> # config -s config.sdt.hosts.host4.udpports.udpport2=443
> # config -s config.sdt.hosts.host4.udpports.udpport2.loglevel=0

The *loglevel* can have a value of 0 or 1.

The default services that you should configure are: *22/tcp (ssh), 23/tcp (telnet), 80/tcp (http), 443/tcp (https), 1494/tcp (ica), 3389/tcp (rdp), 5900/tcp (vnc)*

1101 and 1102 Secure Device Servers

Add other network host

To add any other type of network host with the following details:

IP address/ DNS name	192.168.3.10
Host name	OfficePC
Description	MyPC
Allowed sevices	ssh port 22,https port 443
log level for services	1

Issue the commands below. If the Host is not a PDU or UPS power device or a server with IPMI power control, then leave the device type blank:

> *# config -s config.sdt.hosts.host4.address=192.168.3.10*
> *# config -s config.sdt.hosts.host4.description=MyPC*
> *# config -s config.sdt.hosts.host4.name=OfficePC*
> *# config -s config.sdt.hosts.host4.device.type='' (leave this value blank)*
> *# config -s config.sdt.hosts.host4.tcpports.tcpport1=22*
> *# config -s config.sdt.hosts.host4.tcpports.tcpport1.loglevel=1*
> *# config -s config.sdt.hosts.host4.udpports.tcppport2=443*
> *# config -s config.sdt.hosts.host4.udpports.tcpport2.loglevel=1*

If you want to add the new host as a managed device, make sure you use the current total number of managed devices + 1, for the new device number.

To get the current number of managed devices:

> *# config -g config.devices.total*

Assuming we already have one managed device, our new device will be device 2. Issue the following commands:

> *# config -s config. devices.device2.connections.connection1.name=192.168.3.10*
> *# config -s config. devices.device2.connections.connection1.type=Host*
> *# config -s config. devices.device2.name=OfficePC*
> *# config -s config. devices.device2.description=MyPC*
> *# config -s config.devices.total=2*

The following command will synchronize the live system with the new configuration:

> *# config –hosts*

14.7 Trusted Networks

You can further restrict remote access to serial ports based on the source IP address. To configure this via the command line, you need to do the following:

Determine the total number of existing trusted network rules. If you have no existing rules, you can assume this is 0.

 # config -g config.portaccess.total

This command should display *config.portaccess.total 1*

Note that if you *see config.portaccess.total* this means you have 0 rules configured.

Your new rule will be the existing total plus 1. So if the previous command gave you 0, then you start with rule number 1. If you already have 1 rule, your new rule will be number 2, etc.

If you want to restrict access to serial port 5 to computers from a single class C network (*192.168.5.0* for example), you need to issue the following commands (assuming you have a previous rule in place).

Add a trusted network:

> *# config -s config.portaccess.rule2.address=192.168.5.0*
> *# config -s "config.portaccess.rule2.description=foo bar"*
> *# config -s config.portaccess.rule2.netmask=255.255.255.0*
> *# config -s config.portaccess.rule2.port5=on*
> *# config -s config.portaccess.total=2*

The following command will synchronize the live system with the new configuration:

 # config -r serialconfig

14.8 Cascaded Ports

To add a new slave device with the following settings:

IP address/DNS name	192.168.0.153
Description	Console in office 42
Label	les1102-2
Number of ports	2

The following commands must be issued:

 # config -s config.cascade.slaves.slave1.address=192.168.0.153
 # config -s "config.cascade.slaves.slave1.description=CM in office 42"
 # config -s config.cascade.slaves.slave1.label=les1102-2
 # config -s config.cascade.slaves.slave1.ports=2

The total number of slaves must also be incremented. If this is the first slave you're adding, type:

 # config -s config.cascade.slaves.total=1

Increment this value when adding more slaves.

NOTE: If a slave is added using the CLI, then the master SSH public key will need to be manually copied to every slave device before cascaded ports will work (refer to *Chapter 6*).

The following command will synchronize the live system with the new configuration:

 # config -r cascade

14.9 UPS Connections

Managed UPSes

Before adding a managed UPS, make sure that at least 1 port has been configured to run in 'device mode', and that the device is set to 'ups'.

To add a managed UPS with the following values:

Connected via	Port 1
UPS name	My UPS
Description	UPS in room 5
Username to connect to UPS	User2
Password to connect to UPS	secret
shutdown order	2 (0 shuts down first)
Driver	genericups
Driver option - option	option
Driver option - argument	argument
Logging	Enabled
Log interval	2 minutes
Run script when power is critical	Enabled

 # config -s config.ups.monitors.monitor1.port=/dev/port01

If the port number is higher than 9, for example, port 13, enter:

 # config -s config.ups.monitors.monitor1.port=/dev/port13

 # config -s "config.ups.monitors.monitor1.name=My UPS"
 # config -s "config.ups.monitors.monitor1.description=UPS in room 5"
 # config -s config.ups.monitors.monitor1.username=User2
 # config -s config.ups.monitors.monitor1.password=secret
 # config -s config.ups.monitors.monitor1.sdorder=2
 # config -s config.ups.monitors.monitor1.driver=genericups

 # config -s config.ups.monitors.monitor1.options.option1.opt=option
 # config -s config.ups.monitors.monitor1.options.option1.arg=argument
 # config -s config.ups.monitors.monitor1.options.total=1
 # config -s config.ups.monitors.monitor1.log.enabled=on
 # config -s config.ups.monitors.monitor1.log.interval=2
 # config -s config.ups.monitors.monitor1.script.enabled=on

Make sure to increment the total monitors:

 # config -s config.ups.monitors.total=1

The five commands below will add the UPS to Managed devices. Assuming there are already two managed devices configured:

 # config -s "config.devices.device3.connections.connection1.name=My UPS"
 # config -s "config.devices.device3.connections.connection1.type=UPS Unit"
 # config -s "config.devices.device3.name=My UPS"
 # config -s "config.devices.device3.description=UPS in toom 5"
 # config -s config.devices.total=3

To delete this managed UPS:

 # config -d config.ups.monitors.monitor1

Decrement *monitors.total* when deleting a managed UPS.

Remote UPSes

To add a remote UPS with the following details (assuming this is our first remote UPS):

UPS name	oldUPS
Description	UPS in room 2
Address	192.168.50.50
Log status	Disabled
Log rate	240 seconds
Run shutdown script	Enabled

 # config -s config.ups.remotes.remote1.name=oldUPS
 # config -s "config.ups.remotes.remote1.description=UPS in room 2"
 # config -s config.ups.remotes.remote1.address=192.168.50.50
 # config -d config.ups.remotes.remote1.log.enabled
 # config -s config.ups.remotes.remote1.log.interval=240
 # config -s config.ups.remotes.remote1.script.enabled=on
 # config -s config.ups.remotes.total=1

The following command will synchronize the live system with the new configuration:

 # config -a

14.10 RPC Connections

You can add an RPC connection from the command line. We do not recommend that you do this because of dependency issues.

Before adding an RPC, the Management Console GUI code makes sure that at least one port has been configured to run in 'device mode', and that the device is set to 'rpc'.

To add an RPC with the following values:

RPC type	APC 7900
Connected via	Port 2
UPS name	MyRPC
Description	RPC in room 5
Login name for device	rpclogin
Login password for device	secret
SNMP community	v1 or v2c
Logging	Enabled
Log interval	600 second
Number of power outlets	4 (depends on the type/model of the RPC)

```
# config -s config.ports.port2.power.type=APC 7900
# config -s config.ports.port2.power.name=MyRPC
# config -s "config.ports.port2.power.description=RPC in room 5"
# config -s config.ports.port2.power.username=rpclogin
# config -s config.ports.port2.power.password=secret
# config -s config.ports.port2.power.snmp.community=v1
# config -s config.ports.port2.power.log.enabled=on
# config -s config.ports.port2.power.log.interval=600
# config -s config.ports.port2.power.outlets=4
```

The following five commands are used by the Management Console to add the RPC to "Managed Devices":

```
# config -s config.devices.device3.connections.connection1.name=myRPC
# config -s "config.devices.device3.connections.connection1.type=RPC Unit"
# config -s config.devices.device3.name=myRPC
# config -s "config.devices.device3.description=RPC in room 5"
# config -s config.devices.total=3
```

The following command will synchronize the live system with the new configuration:

 # config –a

14.11 Managed Devices

To add a managed device: (also see UPS, RPC connections and Environmental)

```
# config -s "config.devices.device8.name=my device"
# config -s "config.devices.device8.description=The eighth device"
# config -s "config.devices.device8.connections.connection1.name=my device"
# config -s config.devices.device8.connections.connection1.type=[serial | Host | UPS | RPC]
# config -s config.devices.total=8        (decrement this value when deleting a managed device)
```

To delete the above managed device:

```
# config -d config.devices.device8
```

The following command will synchronize the live system with the new configuration:

```
# config –a
```

14.12 Port Log

To configure serial/network port logging:

```
# config -s config.eventlog.server.address='remote server ip address'
# config -s config.eventlog.server.logfacility='facility'
```

'facility' can be:
> Daemon
> Local 0-7
> Authentication
> Kernel
> User
> Syslog
> Mail
> News
> UUCP

```
# config -s config.eventlog.server.logpriority='priority'
```
'priority' can be:
> Info
> Alert
> Critical
> Debug
> Emergency
> Error

Notice
Warning

Assume the remote log server needs a username 'name1' and password 'secret':

> *# config -s config.eventlog.server.username=name1*
> *# config -s config.eventlog.server.password=secret*

To set the remote path as '/Black Box/logs' to save logged data:

> *# config -s config.eventlog.server.path=/Black Box/logs*
> *# config -s config.eventlog.server.type=[none | syslog | nfs | cifs | usb]*

If the server type is set to usb, none of the other values need to be set. The mount point for storing on a remote USB device is */var/run/portmanager/logdir*

The following command will synchronize the live system with the new configuration:

> *# config –a*

14.13 Alerts

You can add an email, SNMP, or NAGIOS alert by following the steps below.

The general settings for all alerts

Assume this is our second alert, and we want to send alert emails to john@Black Box.com and sms's to peter@Black Box.com:

> # config -s config.alerts.alert2.description=MySecondAlert
> # config -s config.alerts.alert2.email=john@Black Box.com
> # config -s config.alerts.alert2.email2=peter@Black Box.com

To use NAGIOS to notify of this alert:

> *# config -s config.alerts.alert2.nsca.enabled=on*

To use SNMP to notify of this alert:

> *# config -s config.alerts.alert2.snmp.enabled=on*

Increment the total alerts:

> # config -s config.alerts.total=2

Below are the specific settings depending on the type of alert required:

Connection Alert

To trigger an alert when a user connects to serial port 5 or network host 3:

> *# config -s config.alerts.alert2.host3='host name'*
> *# config -s config.alerts.alert2.port5=on*
> *# config -s config.alerts.alert2.sensor=temp*
> *# config -s config.alerts.alert2.signal=DSR*
> *# config -s config.alerts.alert2.type=login*

Signal Alert

To trigger an alert when a signal changes state on port 1:

> *# config -s config.alerts.alert2.port1=on*
> *# config -s config.alerts.alert2.sensor=temp*
> *# config -s config.alerts.alert2.signal=[DSR | DCD | CTS]*
> *# config -s config.alerts.alert2.type=signal*

Pattern Match Alert

To trigger an alert if the regular expression '.*0.0% id' is found in serial port 10's character stream.

*# config -s "config.alerts.alert2.pattern=.*0.0% id"*
 # config -s config.alerts.alert2.port10=on
 # config -s config.alerts.alert2.sensor=temp
 # config -s config.alerts.alert2.signal=DSR
 # config -s config.alerts.alert2.type=pattern

UPS Power Status Alert

To trigger an alert when *myUPS* (on localhost) or *thatUPS* (on remote host *192.168.0.50*) power status changes between on line, on battery and low battery.

 # config -s config.alerts.alert2.sensor=temp
 # config -s config.alerts.alert2.signal=DSR
 # config -s config.alerts.alert2.type=ups
 # config -s config.alerts.alert2.ups1=myUPS@localhost
 # config -s config.alerts.alert2.ups2=thatUPS@192.168.0.50

Power Sensor Alert

 # config -s config.alerts.alert2.enviro.high.critical='critical value'
 # config -s config.alerts.alert2.enviro.high.warning='warning value'
 # config -s config.alerts.alert2.enviro.hysteresis='value'
 # config -s config.alerts.alert2.enviro.low.critical='critical value'
 # config -s config.alerts.alert2.enviro.low.warning='warning value'
 # config -s config.alerts.alert2.enviro1='Enviro sensor name'
 # config -s config.alerts.alert2.outlet#='RPCname'.outlet#
'alert2.outlet#' increments sequentially with each added outlet. The second 'outlet#' refers to the specific RPC power outlets.
 # config -s config.alerts.alert2.rpc#='RPC name'
 # config -s config.alerts.alert2.sensor=[temp | humid | load | charge]
 # config -s config.alerts.alert2.signal=DSR
 # config -s config.alerts.alert2.type=enviro
 # config -s config.alerts.alert2.ups1='UPSname@hostname'

Example: To configure a load sensor alert for outlets 2 and 4 for an RPC called 'RPCInRoom20':

 # config -s config.alerts.alert2.outlet1='RPCname'.outlet2
 # config -s config.alerts.alert2.outlet2='RPCname'.outlet4
 # config -s config.alerts.alert2.enviro.high.critical=300
 # config -s config.alerts.alert2.enviro.high.warning=280
 # config -s config.alerts.alert2.enviro.hysteresis=20
 # config -s config.alerts.alert2.enviro.low.critical=50
 # config -s config.alerts.alert2.enviro.low.warning=70
 # config -s config.alerts.alert2.rpc1=RPCInRoom20
 # config -s config.alerts.alert2.sensor=load
 # config -s config.alerts.alert2.signal=DSR
 # config -s config.alerts.alert2.type=enviro

The following command will synchronize the live system with the new configuration:

 # config -r alerts

14.14 SMTP and SMS

To set-up an SMTP mail or SMS server with the following details:

Outgoing server address   mail.Black Box.com
Secure connection type   SSL
Sender   John@Black Box.com
Server username   john
Server password   secret
Subject line   SMTP alerts

 # config -s config.system.smtp.server=mail.Black Box.com
 # config -s config.system.smtp.encryption=SSL (can also be TLS or None)
 # config -s config.system.smtp.sender=John@Black Box.com
 # config -s config.system.smtp.username=john
 # config -s config.system.smtp.password=secret

 # config -s config.system.smtp.subject=SMTP alerts

To set-up an SMTP SMS server with the same details as above:

 # config -s config.system.smtp.server2=mail.Black Box.com
 # config -s config.system.smtp.encryption2=SSL (can also be TLS or None)
 # config -s config.system.smtp.sender2=John@Black Box.com
 # config -s config.system.smtp.username2=john
 # config -s config.system.smtp.password2=secret
 # config -s config.system.smtp.subject2=SMTP alerts

The following command will synchronize the live system with the new configuration:

config -a

14.15 SNMP

To set-up the SNMP agent on the device:

 # config -s config.system.snmp.protocol=[UDP | TCP]
 # config -s config.system.snmp.trapport='port number'   (default is 162)
 # config -s config.system.snmp.address='NMS IP network address'
 # config -s config.system.snmp.commnity='community name' (v1 and v2c only)
 # config -s config.system.snmp.engineid='ID'   (v3 only)
 # config -s config.system.snmp.username='username'   (v3 only)
 # config -s config.system.snmp.password='password'   (v3 only)
 # config -s config.system.snmp.version=[1 | 2c | 3]

The following command will synchronize the live system with the new configuration:

 # config -a

14.16 Administration

To change the administration settings to:

System Name	og.mydomain.com
System Password (root account)	secret
Description	Device in office 2

 # config -s config.system.name=og.mydomain.com
 # config -P config.system.password   (will prompt user for a password)
 # config -s "config.system.location=Device in office 2 "

NOTE: The -P parameter will prompt the user for a password, and encrypt it. You can encrypt the value of any config element using the -P parameter, but only encrypted user passwords and system passwords are supported. If any other element value were to be encrypted, the value will become inaccessible and will have to be reset.

The following command will synchronize the live system with the new configuration:

 # config –a

14.17 IP Settings

To configure the primary network interface with static settings:

IP address	192.168.0.23
Netmask	255.255.255.0
Default gateway	192.168.0.1
DNS server 1	192.168.0.1
DNS server 2	192.168.0.2

 # config -s config.interfaces.wan.address=192.168.0.23
 # config -s config.interfaces.wan.netmask=255.255.255.0
 # config -s config.interfaces.wan.gateway=192.168.0.1
 # config -s config.interfaces.wan.dns1=192.168.0.1
 # config -s config.interfaces.wan.dns2=192.168.0.2

 # config -s config.interfaces.wan.mode=static
 # config -s config.interfaces.wan.media=[Auto | 100baseTx-FD | 100baseTx-HD | 10baseT-HD] 10baseT-FD

To enable bridging between all interfaces:

 # config -s config.system.bridge.enabled=on

To enable IPv6 for all interfaces

 # config -s config.system.ipv6.enabled=on

To configure the management LAN interface, use the same commands as above but replace:

config.interfaces.wan, with *config.interfaces.lan*

NOTE: Not all devices have a management LAN interface.

To configure a failover device in case of an outage:

 # config -s config.interfaces.wan.failover.address1='ip address'
 # config -s config.interfaces.wan.failover.address2='ip address'
 # config -s config.interfaces.wan.failover.interface=[eth1 | console | modem]

The network interfaces can also be configured automatically:

 # config -s config.interfaces.wan.mode=dhcp
 # config -s config.interfaces.lan.mode=dhcp

The following command will synchronize the live system with the new configuration:

 # /bin/config —run=ipconfig

The following command will synchronize the live system with the new configuration:

 # config -r ipconfig

14.18 Date and Time Settings

To enable NTP using a server at pool.ntp.org, issue the following commands:

 # config -s config.ntp.enabled=on
 # config -s config.ntp.server=pool.ntp.org

Alternatively, you can manually change the clock settings:

To change running system time:

 # date 092216452005.05   Format is MMDDhhmm[[CC]YY][.ss]

Then the following command will save this new system time to the hardware clock:

 # /bin/hwclock -systohc

Alternatively, to change the hardware clock:

 # /bin/hwclock -- set --date=092216452005.05   Format is MMDDhhmm[[CC]YY][.ss]

Then the following command will save this new hardware clock time as the system time:

 # /bin/hwclock -hctosys

To change the timezone:

 # config -s config.system.timezone=US/Eastern

1101 and 1102 Secure Device Servers

The following command will synchronize the live system with the new configuration:

> *# config -r time*

14.19 DHCP Server

To enable the DHCP server on the console management LAN, with settings:

Default lease time	200000 seconds
Maximum lease time	300000 seconds
DNS server1	192.168.2.3
DNS server2	192.168.2.4
Domain name	company.com
Default gateway	192.168.0.1
IP pool 1 start address	192.168.0.20
IP pool 1 end address	192.168.0.100
Reserved IP address	192.168.0.50
MAC to reserve IP for	00:1e:67:82:72:d9
Name to identify this host	John-PC

Issue the commands:

> *# config -s config.interfaces.lan.dhcpd.enabled=on*
> *# config -s config.interfaces.lan.dhcpd.defaultlease=200000*
> *# config -s config.interfaces.lan.dhcpd.maxlease=300000*
> *# config -s config.interfaces.lan.dhcpd.dns1=192.168.2.3*
> *# config -s config.interfaces.lan.dhcpd.dns2=192.168.2.4*
> *# config -s config.interfaces.lan.dhcpd.domain=company.com*
> *# config -s config.interfaces.lan.dhcpd.gateway=192.168.0.1*
> *# config -s config.interfaces.lan.dhcpd.pools.pool1.start=192.168.0.20*
> *# config -s config.interfaces.lan.dhcpd.pools.pool1.end=192.168.0.100*
> *# config -s config.interfaces.lan.dhcpd.pools.total=1*
> *# config -s config.interfaces.lan.dhcpd.staticips.staticip1.ip=192.168.0.50*
> *# config -s config.interfaces.lan.dhcpd.staticips.staticip1.mac=00:1e:67:82:72:d9*
> *# config -s config.interfaces.lan.dhcpd.staticips.staticip1.host=John-PC*
> *# config -s config.interfaces.lan.dhcpd.staticips.total=1*

The following command will synchronize the live system with the new configuration:

> *# config –a*

14.20 Services

You can manually enable or disable network servers from the command line. For example, if you wanted to guarantee the following server configuration:

HTTP Server	Enabled
HTTPS Server	Disabled
Telnet Server	Disabled
SSH Server	Enabled
SNMP Server	Disabled
Ping Replies (Respond to ICMP echo requests)	Disabled

> *# config -s config.services.http.enabled=on*
> *# config -d config.services.https.enabled*
> *# config -d config.services.telnet.enabled*
> *# config -s config.services.ssh.enabled=on*
> *# config -d config.services.snmp.enabled*
> *# config -d config.services.pingreply.enabled*

To set secondary port ranges for any service:

> *# config -s config.services.telnet.portbase='port base number'*   *Default: 2000*
> *# config -s config.services.ssh.portbase='port base number'*   *Default: 3000*
> *# config -s config.services.tcp.portbase='port base number'*   *Default: 4000*
> *# config -s config.services.rfc2217.portbase='port base number'*   *Default: 5000*
> *# config -s config.services.unauthtel.portbase='port base number*   *Default: 6000*

The following command will synchronize the live system with the new configuration:

config –a

14.21 NAGIOS

To configure NAGIOS with the following settings:

NAGIOS host name	console at R3 (Name of this system)
NAGIOS host address	192.168.0.1 (IP to find this device at)
NAGIOS server address	192.168.0.10 (upstream NAGIOS server)
Enable SDT for NAGIOS ext.	Enabled
SDT gateway address	192.168.0.1 (defaults to host address)
Prefer NRPE over NSCA	Disabled (defaults to Disabled)

 # config -s config.system.nagios.enabled=on
 # config -s config.system.nagios.name=les1116
 # config -s config.system.nagios.address=192.168.0.1
 # config -s config.system.nagios.server.address=192.168.0.10
 # config -s config.system.nagios.sdt.disabled=on   (diables SDT for nagios extensions)
 # config -s config.system.nagios.sdt.address=192.168.0.1
 # config -s config.system.nagios.nrpe.prefer=''

To configure NRPE with following settings:

NRPE port	5600 (port to listen on for nrpe. Defualts to 5666)
NRPE user	user1 (User to run as. Defaults to nrpe)
NRPE group	group1 (Group to run as. Defaults to nobody)
Allow command arguments	Enabled

 # config -s config.system.nagios.nrpe.enabled=on
 # config -s config.system.nagios.nrpe.port=5600
 # config -s config.system.nagios.user=user1
 # config -s config.system.nagios.nrpe.group=group1
 # config -s config.system.nagios.nrpe.cmdargs=on

To configure NSCA with the following settings:

NSCA encryption	BLOWFISH (can be: [None | XOR | DES | TRPLEDES | CAST-256 | BLOWFISH | TWOFISH | RIJNDAEL-256 | SERPENT | GOST]
NSCA password	secret
NSCA check-in interval	5 minutes
NSCA port	5650 (defaults to 5667)
user to run as	User1 (defaults to nsca)
group to run as	Group1 (defaults to nobody)

 # config -s config.system.nagios.nsca.enabled=on
 # config -s config.system.nagios.nsca.encryption=BLOWFISH
 # config -s config.system.nagios.nsca.secret=secret
 # config -s config.system.nagios.nsca.interval=2
 # config -s config.system.nagios.nsca.port=5650
 # config -s config.system.nagios.nsca.user=User1
 # config -s config.system.nagios.nsca.group=Group1

Then synchronize the live system with the new configuration using *# config -a*

15. Advanced Configuration

Black Box *console server*s run the embedded Linux operating system. So *Administrator* class users can configure the *console server* and monitor and manage attached serial console and host devices from the command line using Linux commands and the *config* utility as described in *Chapter 14*.

The Linux kernel in the *console server* also supports GNU *bash* shell script enabling the *Administrator* to run custom scripts. This chapter presents a number of useful scripts and scripting tools including:

delete-node which is a general script for deleting users, groups, hosts, UPSes etc.

ping-detect which will run specified commands when a specific host stops responding to ping requests.

This chapter then details how to perform advanced and custom management tasks using Black Box commands, Linux commands, and the open source tools embedded in the *console server*:

portmanager serial port management

raw data access to the ports and modems

iptables modifications and updating IP filtering rules

modifying SNMP with *net-snmpd*

public key authenticated SSH communications

SSL, configuring HTTPS and issuing certificates

using **pmpower** for *NUT* and *PowerMan* power device management

using *IPMItools*

CDK custom development kit

15.1 Custom Scripting

The *console server* supports GNU *bash* shell commands (refer to the *Appendix*) enabling the *Administrator* to run custom scripts.

15.1.1 Custom Script to Run when Booting

The */etc/config/rc.local* script runs whenever the system boots. By default, this script file is empty. You can add any commands to this file if you want them to run at boot time (for example, if you wanted to display *hello world*:)

```
#!/bin/sh
echo "Hello World!"
```

If this script has been copied from a Windows machine, you may need to run the following command on the script before *bash* can run it successfully:

```
# dos2unix /etc/config/rc.local
```

Another scenario would be to call another custom script from the */etc/config/rc.local* file, making sure that your custom script will run whenever the system is booted.

15.1.2 Running Custom Scripts when Alerts are Triggered

Whenever an alert gets triggered, specific scripts get called. These scripts all reside in */etc/scripts/*. Below is a list of the default scripts that run for each applicable alert:

For a connection alert (when a user connects or disconnects from a port or network host): */etc/scripts/portmanager-user-alert* (for port connections) or */etc/scripts/sdt-user-alert* (for host connections)

For a signal alert (when a signal on a port changes state): */etc/scripts/portmanager-signal-alert*

For a pattern match alert (when a specific regular expression is found in the serial ports character stream): */etc/scripts/portmanager-pattern-alert*

For a UPS status alert (when the UPS power status changes between on line, on battery, and low battery): */etc/scripts/ups-status-alert*

For power and alarm sensor alerts (power load, and battery charge alerts): */etc/scripts/environmental-alert*

For an interface failover alert: */etc/scripts/interface-failover-alert*

All of these scripts do a check to see whether you have created a custom script to run instead. The code that does this check is shown below (an extract from the file /etc/scripts/portmanager-pattern-alert):

```
# If there's a user-configured script, run it instead
scripts[0]="/etc/config/scripts/pattern-alert.${ALERT_PORTNAME}"
scripts[1]="/etc/config/scripts/portmanager-pattern-alert"
for (( i=0 ; i < ${#scripts[@]} ; i++ )); do
        if [ -f "${scripts[$i]}" ]; then
                exec /bin/sh "${scripts[$i]}"
        fi
done
```

This code shows that there are two alternative scripts that can be run instead of the default one. This code first checks whether a file "*/etc/config/scripts/pattern-alert.${ALERT_PORTNAME}*" exists. The variable ${ALERT_PORTNAME} must be replaced with "port01" or "port13" or whichever port the alert should run for. If this file cannot be found, the script checks whether the file "*/etc/config/scripts/portmanager-pattern-alert*" exists. If either of these files exists, the script calls the *exec* command on the first file that it finds and runs that custom file/script instead.

As an example, you can copy the */etc/scripts/portmanager-pattern-alert* script file to */etc/config/scripts/portmanager-pattern-alert*:

```
# cd /
# mkdir /etc/config/scripts   (if the directory does not already exist)
# cp /etc/scripts/portmanager-pattern-alert /etc/config/scripts/portmanager-pattern-alert
```

The next step will be to edit the new script file. First, open the file */etc/config/scripts/portmanager-pattern-alert* using vi (or any other editor), and remove the lines that check for a custom script (the code from above). This will prevent the new custom script from repeatedly calling itself. After these lines have been removed, edit the file, or add any additional scripting to the file.

15.1.3 Example Script—Power Cycling on Pattern Match

For example, we have an RPC (PDU) connected to port 1 on a *console server* and also have some telecommunications device connected to port 2 (which is powered by the RPC outlet 3). Now assume the telecom device transmits a character stream "EMERGENCY" out on its serial console port every time that it encounters some specific error, and the only way to fix this error is to power cycle the telecom device.

The first step is to setup a pattern-match alert on port 2 to check for the pattern "EMERGENCY".

Next we need to create a custom script to deal with this alert:

```
# cd /
# mkdir /etc/config/scripts   (if the directory does not already exist)
# cp /etc/scripts/portmanager-pattern-alert /etc/config/scripts/portmanager-pattern-alert
```

NOTE: Make sure to remove the *if* statement (which checks for a custom script) from the new script, in order to prevent an infinite loop.

The *pmpower* utility is used to send power commands to an RPC device in order to power cycle our telecom device:

```
# pmpower -l port01 -o 3 cycle  (The RPC is on serial port 1. The telecom device is powered by RPC outlet 3)
```

We can now append this command to our custom script. This will guarantee that our telecom device will be power cycled every time the console reads the "EMERGENCY" character stream on port 2.

15.1.4 Example Script— Multiple Email Notifications on Each Alert

If you want to send more than one email when an alert triggers, you have to create a replacement script using the method described above and add the appropriate lines to your new script.

Currently, there is a script */etc/scripts/alert-email* that runs from within all the alert scripts (for example, *portmanager-user-alert or environmental-alert*). The alert-email script sends the email. The line that invokes the email script is as follows:

```
/bin/sh /etc/scripts/alert-email $suffix &
```

If you want to send another email to a single address or the same email to many recipients, edit the custom script appropriately. You can follow the examples in any of the seven alert scripts listed above. In particular, consider the *portmanager-user-alert* script. If you need to send the same alert

1101 and 1102 Secure Device Servers

email to more than one email address, find the lines in the script responsible for invoking the alert-email script, then add the following lines below the existing lines:

export TOADDR="emailaddress@domain.com"
/bin/sh /etc/scripts/alert-email $suffix &

These two lines assign a new email address to TOADDR and invoke the alert-email script in the background.

15.1.5 Deleting Configuration Values from the CLI

The *delete-node* script is provided to help with deleting nodes from the command line. The "*delete-node*" script takes one argument, the node name you want to delete (for example, "*config.users.user1*" or "*config.sdt.hosts.host1*").

delete-node is a general script for deleting any node you desire (users, groups, hosts, UPSes, etc.) from the command line. The script deletes the specified node and shuffles the remainder of the node values.

For example, if we have five users configured and we use the script to delete user 3, then user 4 will become user 3, and user 5 will become user 4.

This creates an obvious complication because this script does NOT check for any other dependencies that the node being deleted may have. You are responsible for making sure that any references and dependencies connected to the deleted node are removed or corrected in the config.xml file.

The script treats all nodes the same. The syntax to run the script is # *./delete-node {node name}.* To remove user 3:

>   # *./delete-node config.users.user3*

The *delete-node* script

```
#!/bin/bash
#User must provide the node to be removed. For example, "config.users.user1"
# Usage: delete-node {full node path}

if [ $# != 1 ]
then
        echo "Wrong number of arguments"
        echo "Usage: delnode {full '.' delimited node path}"
        exit 2
fi

# test for spaces
TEMP=`echo "$1" | sed 's/.* .*/N/'`
if [ "$TEMP" = "N" ]
then
        echo "Wrong input format"
        echo "Usage: delnode {full '.' delimited node path}"
        exit 2
fi

# testing if node exists
TEMP=`config -g config | grep "$1"`
if [ -z "$TEMP" ]
then
        echo "Node $1 not found"
        exit 0
fi

# LASTFIELD is the last field in the node path e.g. "user1"
# ROOTNODE is the upper level of the node e.g. "config.users"
# NUMBER is the integer value extracted from LASTFIELD e.g. "1"
# TOTALNODE is the node name for the total e.g. "config.users.total"
# TOTAL is the value of the total number of items before deleting e.g. "3"
# NEWTOTAL is the modified total i.e. TOTAL-1
# CHECKTOTAL checks if TOTAL is the actual total items in .xml

LASTFIELD=${1##*.}
ROOTNODE=${1%.*}
NUMBER=`echo $LASTFIELD | sed 's/^[a-zA-Z]*//g'`
TOTALNODE=`echo ${1%.*} | sed 's/\(.*\)/\1.total/'`
TOTAL=`config -g $TOTALNODE | sed 's/.* //'`
```

```
NEWTOTAL=$[ $TOTAL -1 ]

# Make backup copy of config file
cp /etc/config/config.xml /etc/config/config.bak
echo "backup of /etc/config/config.xml saved in /etc/config/config.bak"

if [ -z $NUMBER ] # test whether a singular node is being \
#deleted e.g. config.sdt.hosts
then

        echo "deleting $1 "
        config -d "$1 "

        echo Done
        exit 0

elif [ $NUMBER = $TOTAL ] # Test if only one item exists
then
        echo "only one item exists"
        # Deleting node
        echo "Deleting $1 "
        config -d "$1 "

        # Modifying item total.
        config -s "$TOTALNODE=0"

        echo Done
        exit 0

elif [ $NUMBER -lt $TOTAL ] # more than one item exists
then

        # Modify the users list so user numbers are sequential
        # by shifting the users into the gap one at a time...

        echo "Deleting $1 "

        LASTFIELDTEXT=`echo $LASTFIELD | sed 's/[0-9]//g'`
        CHECKTOTAL=`config -g $ROOTNODE.$LASTFIELDTEXT$TOTAL`

        if [ -z "$CHECKTOTAL" ]
        then
                echo "WARNING: "$TOTALNODE" greater than number of items"
        fi

        COUNTER=1
        while [ $COUNTER != $((TOTAL-NUMBER+1)) ]
        do

                config -g $ROOTNODE.$LASTFIELDTEXT$((NUMBER+COUNTER)) \
                | while read LINE
                do
                        config -s \
                        "`echo "$LINE" | sed -e "s/$LASTFIELDTEXT$((NUMBER+ \
                        COUNTER))/$LASTFIELDTEXT$((NUMBER+COUNTER-1))/" \
                        -e 's/ /=/'`"

                done

                let COUNTER++
        done

        # deleting last user
        config -d $ROOTNODE.$LASTFIELDTEXT$TOTAL

        # Modifying item total.
        config -s "$TOTALNODE=$NEWTOTAL"
```

```
        echo Done
        exit 0
else

        echo "error: item being deleted has an index greater than total items. Increase the total count variable."
        exit 0
fi
```

15.1.6 Power Cycle Any Device when a Ping Request Fails

The *ping-detect* script is designed to run specified commands when a monitored host stops responding to ping requests.

The first parameter taken by the *ping-detect* script is the hostname/IP address of the device to ping. Any other parameters are then regarded as a command to run whenever the ping to the host fails. *ping-detect* can run any number of commands.

Below is an example using *ping-detect* to power cycle an RPC (PDU) outlet whenever a specific host fails to respond to a ping request. The *ping-detect* runs from */etc/config/rc.local* to make sure that the monitoring starts whenever the system boots.

Suppose we have a serially controlled RPC connected to port01 on a *console server* and have a router powered by outlet 3 on the RPC (and the router has an internal IP address of 192.168.22.2). The following instructions will show you how to continuously ping the router. When the router fails to respond to a series of pings, the *console server* will send a command to RPC outlet 3 to power cycle the router, and write the current date/time to a file:

Copy the *ping-detect* script to */etc/config/scripts/* on the *console server*

Open */etc/config/rc.local* using vi

Add the following line to *rc.local*:

/etc/config/scripts/ping-detect 192.168.22.2 /bin/bash -c "pmpower -l port01 -o 3 cycle && date" > /tmp/output.log &

The above command will cause the *ping-detect* script to continuously ping the host at 192.168.22.2 which is the router. If the router crashes, it will no longer respond to ping requests. If this happens, the two commands *pmpower* and *date* will run. The output from these commands is sent to the file */tmp/output.log* so that we have a record. The *ping-detect* is also run in the background using the "*&*".

Remember the *rc.local* script only runs by default when the system boots. You can manually run the *rc.local* script or the *ping-detect* script if desired.

The *ping-detect* script

The above is just one example of using the *ping-detect* script. The idea of the script is to run any number of commands when a specific host stops responding to ping requests. Here are details of the *ping-detect* script itself:

```
#!/bin/sh
# Usage: ping-detect HOST [COMMANDS...]
# This script takes 2 types of arguments: hostname/IPaddress to ping, and the commands to
#  run if the ping fails 5 times in a row. This script can only take one host/IPaddress per
# instance. Multiple independent commands can be sent to the script. The commands will be
# run one after the other.
#
# PINGREP is the entire reply from the ping command
# LOSS is the percentage loss from the ping command
# $1 must be the hostname/IPaddress of device to ping
# $2... must be the commands to run when the pings fail.
COUNTER=0
TARGET="$1"
shift
# loop indefinitely:
while true
do
        # ping the device 10 times
        PINGREP=`ping -c 10 -i 1 "$TARGET" `
        #get the packet loss percentage
        LOSS=`echo "$PINGREP" |grep "%" | sed -e 's/.* \([0-9]*\)% .*/\1/'`
        if [ "$LOSS" -eq "100" ]
        then
                COUNTER=`expr $COUNTER + 1`
        else
                COUNTER=0
```

```
        sleep 30s
fi
if [ "$COUNTER" -eq 5 ]
then
        COUNTER=0
        "$@"
        sleep 2s
fi
done
```

15.1.7   Running Custom Scripts When a Configurator is Invoked

A configurator is responsible for reading the values in */etc/config/config.xml* and making the appropriate changes live. Some changes made by the configurators are part of the Linux configuration itself, such as user passwords or *ipconfig*.

Currently there are nineteen configurators. Each one is responsible for a specific group of config (for example, the "*users*" configurator makes the user configurations in the *config.xml file* live). To see all the available configurators, type the following from a command line prompt:

> # config

When a change is made using the Management Console web GUI, the appropriate configurator automatically runs. This can be a problem if another *Administrator* makes a change using the Management Console. The configurator could possibly overwrite any custom CLI/linux configurations you may have set.

The solution is to create a custom script that runs after each configurator runs. After each configurator runs, it will check whether that appropriate custom script exists. You can then add any commands to the custom script and they will be invoked after the configurator runs.

The custom scripts must be in the correct location:

> */etc/config/scripts/config-post-*

To create an alerts custom script:

> # cd /etc/config/scripts
> # touch config-post-alerts
> # vi config-post-alerts

You could use this script to recover a specific backup config or overwrite a config or make copies of config files, etc.

15.1.8 Backing Up the Configuration and Restoring Using a Local USB Stick

The ***/etc/scripts/backup-usb*** script is written to save and load custom configuration using a USB flash disk. Before saving configuration locally, you must prepare the USB storage device for use. To do this, disconnect all USB storage devices except for the storage device you want to use.

Usage: */etc/scripts/backup-usb* COMMAND [FILE]

COMMAND:

> check-magic -- check volume label
> set-magic -- set volume label
> save [FILE] -- save configuration to USB
> delete [FILE] -- delete a configuration tarbal from USB
> list -- list available config backups on USB
> load [FILE] -- load a specific config from USB
> load-default -- load the default configuration
> set-default [FILE] -- set which file becomes the default

The first thing to do is to check if the USB disk has a label:

> # /etc/scripts/backup-usb check-magic

If this command returns "Magic volume not found", then run the following command:

> # /etc/scripts/backup-usb set-magic

To save the configuration:

> # /etc/scripts/backup-usb save config-20May

To check if the backup was saved correctly:

> # /etc/scripts/backup-usb list

If this command does not display "* config-20May" then there was an error saving the configuration.

The set-default command takes an input file as an argument and renames it to "default.opg". This default configuration remains stored on the USB disk. The next time you want to load the default config, it will be sourced from the new default.opg file. To set a config file as the default:

> # /etc/scripts/backup-usb set-default config-20May

To load this default:

> # /etc/scripts/backup-usb load-default

To load any other config file:

> # /etc/scripts/backup-usb load {filename}

The /etc/scripts/backup-usb script can be executed directly with various COMMANDS or called from other custom scripts you may create. We recommend that you do not customize the /etc/scripts/backup-usb script itself at all.

15.1.9 Backing Up the Configuration Off-Box

If you do not have a USB port on your console server, you can back up the configuration to an off-box file. Before backing up you need to arrange a way to transfer the backup off-box. This could be via an NFS share, a Samba (Windows) share to USB storage, or copied off-box via the network. If backing up directly to off-box storage, make sure it is mounted.

/tmp is not a good location for the backup except as a temporary location before transferring it off-box. The /tmp directory will not survive a reboot. The /etc/config directory is not a good place either, because it will not survive a restore.

Backup and restore should be done by the root user to make sure correct file permissions are set. The config command is used to create a backup tarball:

> config -e <Output File>

The tarball will be saved to the indicated location. It will contain the contents of the /etc/config/ directory in an uncompressed and unencrypted form.

Example nfs storage:

> # mount -t nfs 192.168.0.2:/backups /mnt # config -e /mnt/les4108.config # umount/mnt/

Example transfer off-box via scp:

> # config -e /tmp/les4108.config
> # scp /tmp/les4108.config 192.168.0.2:/backups

The config command is also used to restore a backup:

> config -i <Input File>

This will extract the contents of the previously created backup to /tmp, and then synchronize the /etc/config directory with the copy in /tmp. One problem that can crop up here is that there is not enough room in /tmp to extract files to. The following command will temporarily increase the size of /tmp:

> mount -t tmpfs -o remount,size=2048k tmpfs /var

If restoring to either a new unit or one that has been factory defaulted, make sure that the process generating SSH keys either stops or completes before restoring configuration. If this is not done, then a mix of old and new keys may be put in place. SSH uses these keys to avoid man-in-the-middle attacks. Logging in may be disrupted.

15.2 Advanced Portmanager

Black Box's *portmanger* program manages the *console server* serial ports. It routes network connection to serial ports, checks permissions, and monitors and logs all the data flowing to/from the ports.

15.2.1 Portmanager commands

pmshell

The *pmshell* command acts similar to the standard *tip* or *cu* commands, but all serial port access is directed *via* the portmanager.
Example: To connect to port 8 *via* the portmanager:

> # *pmshell -l port08*

pmshell Commands:

> Once connected, the pmshell command supports a subset of the '~' escape commands that tip/cu support. For SSH you must prefix the escape with an additional '~' command (that is, use the '~~' escape)

> Send Break: Typing the character sequence '~b' will generate a BREAK on the serial port.

> History: Typing the character sequence '~h' will generate a history on the serial port.

> Quit pmshell: Typing the character sequence '~.' will exit from pmshell.

> Set RTS to 1 run the command: *pmshell --rts=1*

> Show all signals: # *pmshell –signals*

> > DSR=1 DTR=1 CTS=1 RTS=1 DCD=0

> Read a line of text from the serial port: # *pmshell –getline*

pmchat

The *pmchat* command acts similar to the standard *chat* command, but all serial port access is directed *via* the portmanager.

Example: To run a chat script *via* the portmanager:

pmchat -v -f /etc/config/scripts/port08.chat < /dev/port08

For more information on using *chat* (and *pmchat*), consult the UNIX man pages:

http://techpubs.sgi.com/library/tpl/cgibin/getdoc.cgi?coll=linux&db=man&fname=/usr/share/catman/man8/chat.8.html

pmusers

The *pmusers* command is used to query the portmanager for active user sessions.

Example: To detect which users are currently active on which serial ports:

> # *pmusers*

This command will output nothing if there are no active users currently connected to any ports. Otherwise, it will respond with a sorted list of usernames per active port:

> *Port 1:*
> > *user1*
> > *user2*
> *Port 2:*
> > *user1*
> *Port 8:*
> > *user2*

 The above output indicates that a user named *"user1"* is actively connected to ports 1 and 2, while *"user2"* is connected to both ports 1 and 8.

1101 and 1102 Secure Device Servers

portmanager daemon

There is normally no need to stop and restart the daemon. To restart the daemon normally, just run the command:

> *# portmanager*

Supported command line options are:

> Force portmanager to run in the foreground: *--nodaemon*
>
> Set the level of debug logging: *--loglevel={debug,info,warn,error,alert}*
>
> Change which configuration file it uses: *-c /etc/config/portmanager.conf*

Signals

Sending a SIGHUP signal to the portmanager will cause it to re-read its configuration file.

15.2.2 External Scripts and Alerts

The portmanager can execute external scripts on certain events.

When the portmanager opens a port:

It attempts to execute */etc/config/scripts/portXX.init* (where XX is the number of the port, *e.g.* 08). The script is run with STDIN and STDOUT both connected to the serial port.

If the script cannot be executed, then portmanager will execute */etc/config/scripts/portXX.chat via* the chat command on the serial port.

When an alert occurs on a port:

The portmanager will attempt to execute */etc/config/scripts/portXX.alert* (where XX is the port number, for example, 08)

The script is run with STDIN containing the data which triggered the alert, and STDOUT redirected to /dev/null, NOT to the serial port. If you want to communicate with the port, use *pmshell* or *pmchat* from within the script.

If the script cannot be executed, then the alert will be mailed to the address configured in the system administration section.

When a user connects to any port:

If a file called */etc/config/pmshell-start.sh* exists it is run when a user connects to a port. It is provided 2 arguments, the "Port number" and the "Username". Here is a simple example:

> *</etc/config/pmshell-start.sh >*
> *#!/bin/sh*
> *PORT="$1"*
> *USER="$2"*
> *echo "Welcome to port $PORT $USER"*
> *< /etc/config/pmshell-start.sh>*

The return value from the script controls whether the user is accepted or not, if 0 is returned (or nothing is done on exit as in the above script) the user is permitted, otherwise the user is denied access.

Here is a more complex script which reads from configuration to display the port label if available and denies access to the root user:

> *</etc/config/pmshell-start.sh>*
> *#!/bin/sh*
> *PORT="$1"*
> *USER="$2"*
> *LABEL=$(config -g config.ports.port$PORT.label | cut -f2- -d' ')*
> *if ["$USER" == "root"]; then*
> *   echo "Permission denied for Super User"*
> *   exit 1*
> *fi*
> *if [-z "$LABEL"]; then*
> *echo "Welcome $USER, you are connected to Port $PORT"*
> *else*
> *echo "Welcome $USER, you are connected to Port $PORT ($LABEL)"*

fi
</etc/config/pmshell-start.sh>

15.3 Raw Access to Serial Ports

15.3.1 Access to Serial Ports

You can use *tip* and *stty* to completely bypass the *portmanager* and have raw access to the serial ports.

When you run *tip* on a *portmanager* controlled port, *portmanager* closes that port, and stops monitoring it until *tip* releases control of it.

With *stty*, the changes made to the port only "stick" until that port is closed and opened again. People probably will not want to use *stty* for more than initial debugging of the serial connection.

If you want to use *stty* to configure the port, you can put *stty* commands in */etc/config/scripts/portXX.init,* which gets run whenever portmanager opens the port.

Otherwise, any setup you do with *stty* will get lost when the portmanager opens the port. (The reason that portmanager sets things back to its *config* rather than using whatever is on the port, is so the port is in a known good state, and will work, no matter what things are done to the serial port outside of portmanager.)

15.3.2 Accessing the Console/Modem Port

The console dial-in is handled by *mgetty*, with automatic PPP login extensions. *mgetty* is a smart *getty* replacement, designed to be used with Hayes compatible data and data/fax modems. *mgetty* knows about modem initialization, manual modem answering (your modem doesn't answer if the machine isn't ready), UUCP locking (you can use the same device for dial-in and dial-out). *mgetty* provides very extensive logging facilities. All standard *mgetty* options are supported.

Enabling Boot Messages on the Console:

If you are not using a modem on the DB9 console port and instead want to connect to it directly via a Null Modem cable, enable verbose mode, which allows you to see the standard linux start-up messages. Follow these commands:

 # /bin/config --set=config.console.debug=on # /bin/config --run=console # reboot

If at some point in the future you chose to connect a modem for dial-in out-of-band access, you can reverse the procedure with the following commands.

 # /bin/config --del=config.console.debug # /bin/config --run=console # reboot

15.4 IP Filtering

The *console server* uses the *iptables* utility to provide a stateful firewall of LAN traffic. By default, rules are automatically inserted to allow access to enabled services and serial port access *via* enabled protocols. The commands that add these rules are contained in configuration files:

/etc/config/ipfilter

This is an executable shell script that runs whenever the LAN interface is brought up and whenever modifications are made to the *iptables* configuration as a result of CGI actions or the *config* command line tool.

The basic steps performed are as follows:

The current *iptables* configuration is erased.

If a customized IP-Filter script exists it is executed and no other actions are performed.

Standard policies are inserted that will drop all traffic not explicitly allowed to and through the system.

Rules are added which explicitly allow network traffic to access enabled services, for example, TTP, SNMP, *etc*.

Rules are added that explicitly allow traffic network traffic access to serial ports over enabled protocols, for example, Telnet, SSH, and raw TCP.

1101 and 1102 Secure Device Servers

If the standard system firewall configuration is not adequate for your needs you can bypass it safely by creating a file at **/etc/config/filter-custom** containing commands to build a specialized firewall. This firewall script will run whenever the LAN interface is brought up (including initially) and will override any automated system firewall settings.

Below is a simple example of a custom script that creates a firewall using the *iptables* command. Only incoming connections from computers on a C-class network 192.168.10.0 will be accepted when this script is installed at */etc/config/filter-custom.* Note that when this script is called, any pre-existing chains and rules have been flushed from *iptables*:

```
#/bin/sh
# Set default policies to drop any incoming or routable traffic
# and blindly accept anything from the 192.168.10.0 network.
iptables —policy FORWARD DROP
iptables —policy INPUT DROP
iptables —policy OUTPUT ACCEPT
# Allow responses to outbound connections back in.
iptables —append INPUT \
          —match state —state ESTABLISHED,RELATED —jump ACCEPT
# Explicitly accept any connections from computers on
# 192.168.10.0/24
iptables —append INPUT —source 192.168.10.0/24 —jump ACCEPT
```

There's good documentation about using the *iptables* command at the Linux *netfilter* website *http://netfilter.org/documentation/index.html.* There are also many high-quality tutorials and HOWTOs available *via* the *netfilter* website, in particular peruse the tutorials listed on the *netfilter* HOWTO page.

15.5 Modifying SNMP Configuration

Adding More than One SNMP Server

To add more than one SNMP server for alert traps, add the first SNMP server using the Management Console or the command line *config* tool. Secondary and any further SNMP servers are added manually using *config.*

Log in to the *console server*'s command line shell as root or an admin user. Refer back to the Management Console UI or user documentation for descriptions of each field.

To set the Manager Protocol field:

```
config --set config.system.snmp.protocol2=UDP or
config --set config.system.snmp.protocol2=TCP
```

To set the Manager Address field:

```
config --set config.system.snmp.address2=w.x.y.z
```
.. replacing *w.x.y.z* with the IP address or DNS name.

To set the Manager Trap Port field:

```
config --set config.system.snmp.trapport2=162
```
.. replacing 162 with the TCP/UDP port number

To set the Version field:

```
config --set config.system.snmp.version2=1 or
config --set config.system.snmp.version2=2c or
config --set config.system.snmp.version2=3
```

To set the Community field (SNMP version 1 and 2c only):

```
config --set config.system.snmp.community2=yourcommunityname
```
.. replacing *yourcommunityname* with the community name

To set the Engine ID field (SNMP version 3 only):

```
config --set config.system.snmp.engineid2=800000020109840301
```
.. replacing *800000020109840301* with the engine ID

724-746-5500 | blackbox.com

To set the Username field (SNMP version 3 only):

  config --set config.system.snmp.username2=yourusername
.. replacing *yourusername* with the username
  config.system.snmp.username2 (3 only)

To set the Engine ID field (SNMP version 3 only)
  config --set config.system.snmp.password2=yourpassword
.. replacing *yourpassword* with the password
Once the fields are set, apply the configuration with the following command:

config --run snmp

You can add a third or more SNMP servers by incrementing the "2" in the above commands, for example, config.system.snmp.protocol3, config.system.snmp.address3, etc.

15.6 Secure Shell (SSH) Public Key Authentication

This section covers how to generate public and private keys in a Linux and Windows environment and configure SSH for public key authentication. The steps to use in a Clustering environment are:

• Generate a new public and private key pair.

• Upload the keys to the Master and to each Slave *console server*.

• Fingerprint each connection to validate.

15.6.1 SSH Overview

Popular TCP/IP applications such as telnet, rlogin, ftp, and others transmit their passwords unencrypted. Doing this across pubic networks like the Internet can have catastrophic consequences. It leaves the door open for eavesdropping, connection hijacking, and other network-level attacks.

Secure Shell (SSH) is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over insecure channels.

OpenSSH, the de facto open source SSH application, encrypts all traffic (including passwords) to effectively eliminate these risks. Additionally,

OpenSSH provides a myriad of secure tunneling capabilities, as well as a variety of authentication methods.

OpenSSH is the port of OpenBSD's excellent OpenSSH[0] to Linux and other versions of Unix. OpenSSH is based on the last free sample implementation with all patent-encumbered algorithms removed (to external libraries), all known security bugs fixed, new features reintroduced, and many other clean-ups. <u>http://www.openssh.com/</u> The only changes in the Black Box SSH implementation are:

PAM support

EGD[1]/PRNGD[2] support and replacements for OpenBSD library functions that are absent from other versions of UNIX

The config files are now in */etc/config. for example*
        /etc/config/sshd_config instead of */etc/sshd_config*
        /etc/config/ssh_config instead of */etc/ssh_config*
        /etc/config/users/<username>/.ssh/ instead of */home/<username>/.ssh/*

15.6.2 Generating Public Keys (Linux)

To generate new SSH key pairs use the Linux *ssh-keygen* command. This will produce an RSA or DSA public/private key pair and you will be prompted for a path to store the two key files, for example, *id_dsa.pub* (the public key) and *id_dsa* (the private key). For example:

        $ ssh-keygen -t [rsa|dsa]
        Generating public/private [rsa|dsa] key pair.
        Enter file in which to save the key *(/home/user/.ssh/id_[rsa|dsa]):*
        Enter *passphrase* (empty for no passphrase):
        Enter same passphrase again:
        Your identification has been saved in */home/user/.ssh/id_[rsa|dsa].*
        Your public key has been saved in */home/user/.ssh/id_[rsa|dsa].pub.*

The key fingerprint is:
28:aa:29:38:ba:40:f4:11:5e:3f:d4:fa:e5:36:14:d6 user@server
$

Create a new directory to store your generated keys. You can also name the files after the device they will be used for. For example:

$ mkdir keys
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa): */home/user/keys/control_room*
Enter *passphrase* (empty for no passphrase):
Enter same *passphrase* again:
Your identification has been saved in */home/user/keys/control_room*
Your public key has been saved in */home/user/keys/control_room.pub*.
The key fingerprint is:
28:aa:29:38:ba:40:f4:11:5e:3f:d4:fa:e5:36:14:d6 user@server
$

Make sure that there is no password associated with the keys. If there is a password, then the Black Box devices will have no way to supply it as runtime.

Full documentation for the *ssh-keygen* command can be found at http://www.openbsd.org/cgi-bin/man.cgi?query=ssh-keygen

15.6.3 Installing the SSH Public/Private Keys (Clustering)

For Black Box *console servers*, the keys can be simply uploaded through the web interface, on the **System: Administration** page. This enables you to upload stored RSA or DSA Public Key pairs to the Master and apply the Authorized key to the slave and is described in Chapter 4. Once complete, you then proceed to Fingerprinting as described below.

15.6.4 Installing SSH Public Key Authentication (Linux)

Alternately, the public key can be installed on the unit remotely from the linux host with the *scp* utility as follows.

Assuming the user on the Management Console is called "fred"; the IP address of the *console server* is 192.168.0.1 (default); and the public key is on the *linux/unix* computer in ~/.ssh/id_dsa.pub. Execute the following command on the *linux/unix* computer:

*scp ~/.ssh/id_dsa.pub *
[*root@192.168.0.1:/etc/config/users/fred/.ssh/authorized_keys*](mailto:root@192.168.0.1)

The authorized_keys file on the *console server* needs to be owned by "fred", so login to the Management Console as **root** and type:

chown fred /etc/config/users/fred/.ssh/authorized_keys

Master LES1102A

Slave
LES1101A

authorized_key
ssh-rsa
AAAAB3NzaC1yC2Efg4+t
GHIAAA==name@client1

id_rsa
---BEGIN RSA
PRIVATE KEY----
MIBogIDAAKCAQEA
yIPGsNf5+aOLnPUMc
nujXXPGIQGyD3b79
KZg3UZMjZI525sCy
Opv4TjTvTK6a7QIYt
GYTByUdl

Slave
LES1102A

authorized_key
ssh-rsa
AAAAB3NzaC1yC2Efg4+t
GHIAAA==name@client1

id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2Efg4+tGHIAAA==name@client1
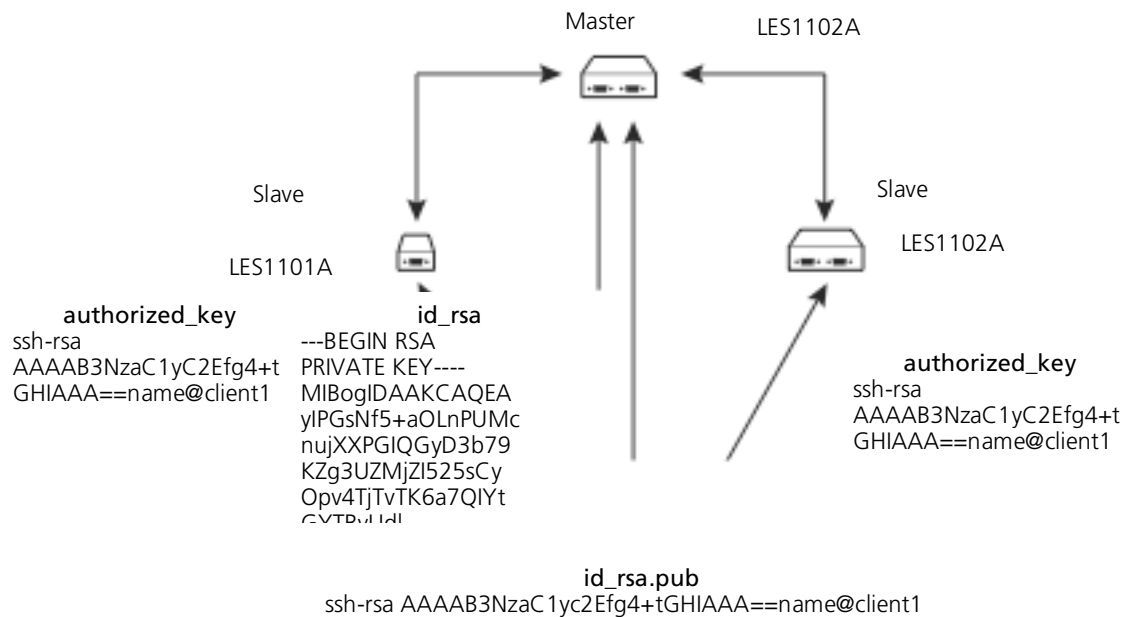
Figure 15-1.

If the Black Box device selected to be the server will only have one client device, then the *authorized_keys* file is simply a copy of the public key for that device. If one or more devices will be clients of the server, then the *authorized_keys* file will contain a copy of all of the public keys. RSA and DSA keys may be freely mixed in the authorized_keys file. For example, assume we already have one server, called bridge_server, and two sets of keys, for the control_room and the plant_entrance:

$ ls /home/user/keys control_room control_room.pub plant_entrance plant_entrance.pub $ cat /home/user/keys/control_room.pub /home/user/keys/plant_entrance.pub > /home/user/keys/authorized_keys_bridge_server

Master
LES1102A

id_dsa
-----BEGIN DSA
PRIVATE KEY----
MIIBugBAAKBgQCR
kixjJ0SKuiREXTM
xDPFp9HqBvEg7Ww9
oynY4QNiXj1YU7T
871TLQiAhn3yp72Wy
7Z5C3sLF8o46Go

Master
LES1102A

id_rsa
-----BEGIN DSA
PRIVATE KEY----
MIIEogIBAAKCAQEA
yIPGsNf5+a0LnPUMc
nujXXPGiQGyD3b79
KZf3UZ4MjZI525sCy
Opv4TjTvTK6e8QIYt
GYTByUdl

LES1102A

ssh-dsa
AAAAB3NzaZr+OV01C8gdgz
XDg==name@client2

ssh-rsa
AAAAB3NzaC1yc2Efg4+tG
HIAAA==name@client1

authorized_keys
ssh-rsa AAAAB3NzaC1yc2Efg4+IGHI
ssh-dss AAAAB3NzaZr+OV01C8gdgz
XDg==name@client2

Figure 15-2.

More documentation on OpenSSH can be found at:

http://openssh.org/portable.html

1101 and 1102 Secure Device Servers

http://www.openbsd.org/cgi-bin/man.cgi?query=ssh&sektion=1
http://www openbsd.org/cgi-bin/man.cgi?query=sshd.

15.6.5 Generating Public/Private Keys for SSH (Windows)

This section describes how to generate and configure SSH keys using Windows.

First create a new user from the Black Box Management (the following example uses a user called "testuser"), making sure it is a member of the "users" group.

If you do not already have a public/private key pair you can generate them now using *ssh-keygen*, *PuTTYgen* or a similar tool:

PuTTYgen: *http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html*

OpenSSH: *http://www.openssh.org/*

OpenSSH (Windows): *http://sshwindows.sourceforge.net/download/*

For example, using PuTTYgen, make sure you have a recent version of the *puttygen.exe* (available from *http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html*) Make sure you have a recent version of WinSCP (available from *http://winscp.net/eng/download.php*)
To generate a SSH key using PuTTY *http://sourceforge.net/docs/F02/#clients*:

Execute the PUTTYGEN.EXE program.

Select the desired key type *SSH2 DSA* (you may use RSA or DSA) within the *Parameters* section.

It is important that you leave the passphrase field blank.

Click on the *Generate* button.

Follow the instruction to move the mouse over the blank area of the program in order to create random data used by PUTTYGEN to generate secure keys. Key generation will occur once PUTTYGEN has collected sufficient random data.
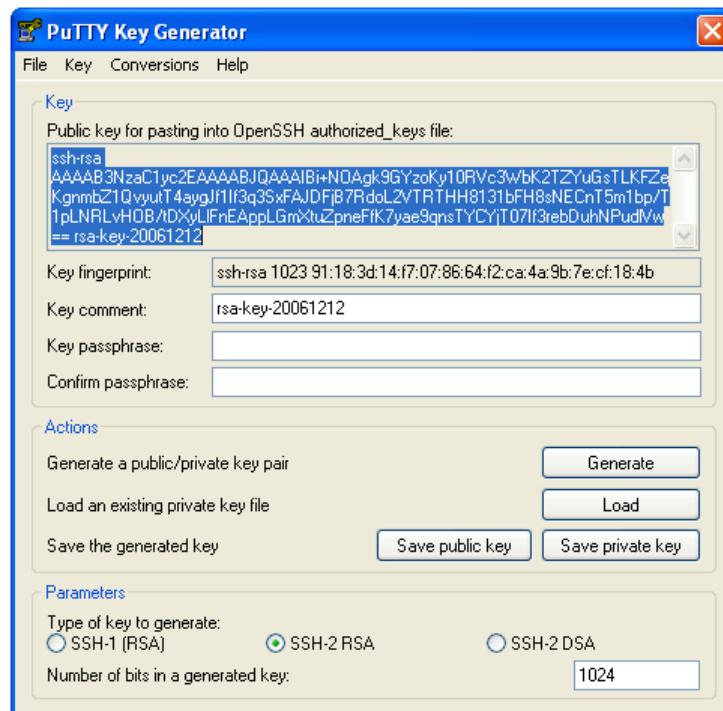
Figure 15-3.

Create a new file " *authorized_keys* " (with notepad) and copy your public key data from the "Public key for pasting into OpenSSH authorized_keys file" section of the PuTTY Key Generator, and paste the key data to the "authorized_keys" file. Make sure there is only one line of text in this file.

724-746-5500 | blackbox.com

Use WinSCP to copy this "authorized_keys" file into the users home directory: e.g. */etc/config/users/testuser/.ssh/authorized_keys* of the Black Box gateway which will be the SSH server. You will need to make sure this file is in the correct format with the correct permissions with the following commands:

> *# dos2unix *
> */etc/config/users/testuser/.ssh/authorized_keys && chown testuser *
> */etc/config/users/testuser/.ssh/authorized_keys*

Using WinSCP copy the attached sshd_config over */etc/config/sshd_config* on the server (Makes sure public key authentication is enabled).

Test the Public Key by logging in as "testuser" to the client Black Box device and typing (you should not need to enter anything): # ssh -o StrictHostKeyChecking=no <server-ip>

To automate connection of the SSH tunnel from the client on every power-up you need to make the *clients /etc/config/rc.local* look like the following:

> *#!/bin/sh*
> *ssh -L9001:127.0.0.1:4001 -N -o StrictHostKeyChecking=no testuser@<server-ip> &*

This will run the tunnel redirecting local port 9001 to the server port 4001.

15.6.6 Fingerprinting

Fingerprints are used to ensure you are establishing an SSH session to who you think you are. On the first connection to a remote server you will receive a fingerprint that you can use on future connections.

This fingerprint is related to the host key of the remote server. Fingerprints are stored in *~/.ssh/known_hosts*.

To receive the fingerprint from the remote server, log in to the client as the required user (usually root) and establish a connection to the remote host:

 # ssh remhost
> *The authenticity of host 'remhost (192.168.0.1)' can't be established.*
> *RSA key fingerprint is 8d:11:e0:7e:8a:6f:ad:f1:94:0f:93:fc:7c:e6:ef:56.*
> *Are you sure you want to continue connecting (yes/no)?*

At this stage, answer yes to accept the key. You should get the following message:

> *Warning: Permanently added 'remhost,192.168.0.1' (RSA) to the list of known hosts.*

You may be prompted for a password, but there is no need to log in— you have received the fingerprint and can Ctrl-C to cancel the connection. If the host key changes you will receive the following warning, and not be allowed to connect to the remote host:

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@   WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!@
@   IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Someone could be eavesdropping on you right now (man-in-the-middle attack)!

It is also possible that the RSA host key has just been changed.

The fingerprint for the RSA key sent by the remote host is

ab:7e:33:bd:85:50:5a:43:0b:e0:bd:43:3f:1c:a5:f8.

Please contact your system *Administrator*.

Add correct host key in */.ssh/known_hosts* to get rid of this message.

Offending key in */.ssh/known_hosts:1*

RSA host key for *remhost* has changed and you have requested strict checking.

Host key verification failed.

1101 and 1102 Secure Device Servers

If the host key has been legitimately changed, it can be removed from the ~/.ssh/known_hosts file and the new fingerprint added. If it has not changed, this indicates a serious problem that should be investigated immediately.

15.6.7   SSH Tunneled Serial Bridging

You have the option to apply SSH tunneling when two Black Box console servers are configured for serial bridging.
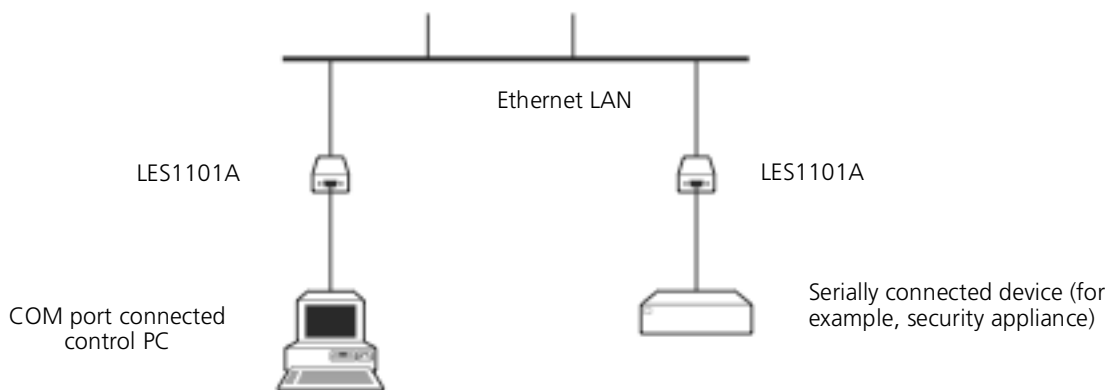
Figure 15-4. SSH tunneling and bridging.

As detailed in *Chapter 5*, the *Server* console server is setup in *Console server* mode with either RAW or RFC2217 enabled and the *Client* console server is set up in Serial Bridging Mode with the Server Address, and Server TCP Port (4000 + port for RAW or 5000 + port # for RFC2217) specified:

Select **SSH Tunnel** when configuring the **Serial Bridging Setting.**

Figure 15-5. Serial bridge settings.

Next, you will need to set up SSH keys for each end of the tunnel and upload these keys to the *Server* and *Client* console servers.

Client Keys:

The first step in setting up ssh tunnels is to generate keys. Ideally, you will use a separate, secure machine to generate and store all keys to be used on the *console servers*. If this is not ideal for your situation, keys may be generated on the *console servers* themselves.

It is possible to generate only one set of keys, and reuse them for every SSH session. While we do not recommend this, each organization will need to balance the security of separate keys against the additional administration they bring.

Generated keys may be one of two types—RSA or DSA (and it is beyond the scope of this document to recommend one over the other). RSA keys will go into the files *id_rsa* and *id_rsa.pub*. DSA keys will be stored in the files *id_dsa* and *id_dsa.pub*.

For simplicity going forward, the term *private key* will be used to refer to either *id_rsa* or *id_dsa* and *public key* to refer to either *id_rsa.pub* or *id_dsa.pub*.

724-746-5500 | blackbox.com

Client #1   Server   Client #2

Id_dsa   id_dsa.pub   authorized keys   id_rsa.pub   id.rsa
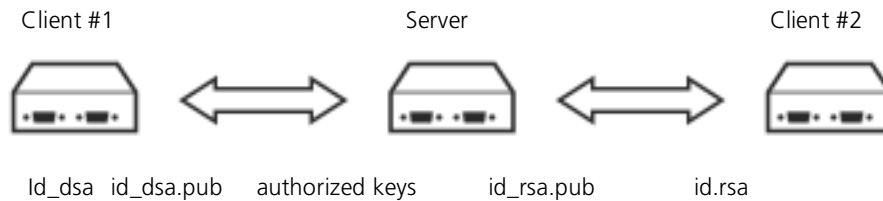
Figure 16-6. Keys.

To generate the keys using OpenBSD's OpenSSH suite, we use the *ssh-keygen* program:

```
$ ssh-keygen -t [rsa|dsa]
Generating public/private [rsa|dsa] key pair.
Enter file in which to save the key (/home/user/.ssh/id_[rsa|dsa]):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_[rsa|dsa].
Your public key has been saved in /home/user/.ssh/id_[rsa|dsa].pub.
The key fingerprint is:
28:aa:29:38:ba:40:f4:11:5e:3f:d4:fa:e5:36:14:d6 user@server
$
```

It is advisable to create a new directory to store your generated keys. It is also possible to name the files after the device they will be used for. For example:

```
$ mkdir keys
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):  /home/user/keys/control_room
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/keys/control_room
Your public key has been saved in /home/user/keys/control_room.pub.
The key fingerprint is:
28:aa:29:38:ba:40:f4:11:5e:3f:d4:fa:e5:36:14:d6 user@server
$
```

You should ensure there is no password associated with the keys. If there is a password, then the *console servers* will have no way to supply it as runtime.

Authorized Keys:

If the *console server* selected to be the server will only have one client device, then the *authorized_keys* file is simply a copy of the public key for that device. If one or more devices will be clients of the server, then the *authorized_keys* file will contain a copy of all of the public keys. RSA and DSA keys may be freely mixed in the *authorized_keys* file.

For example, assume we already have one server, called *bridge_server*, and two sets of keys, for the *control_room* and the *plant_entrance*:

```
$ ls /home/user/keys

control_room control_room.pub plant_entrance plant_entrance.pub

$ cat /home/user/keys/control_room.pub

/home/user/keys/plant_entrance.pub >

/home/user/keys/authorized_keys_bridge_server
```

Uploading Keys:

The keys for the server can be uploaded through the web interface, on the **System: Administration** page as detailed earlier. If only one client will be connecting, then simply upload the appropriate public key as the authorized keys file. Otherwise, upload the authorized keys file constructed in the previous step.

Each client will then need its own set of keys uploaded through the same page. Take care to ensure that the correct type of keys (DSA or RSA) go in the correct spots, and that the public and private keys are in the correct spot.

15.6.8   SDT Connector Public Key Authentication

SDT Connector can authenticate against a *console servers* using your SSH key pair, rather than requiring you to enter your password (i.e. public key authentication).

To use public key authentication with SDT Connector, you must first create an RSA or DSA key pair (using *ssh-keygen, PuTTYgen* or a similar tool) and add the public part of your SSH key pair to the Black Box gateway—as described in the earlier section.

Next, add the private part of your SSH key pair (this file is typically named *id_rsa* or *id_dsa*) to SDT Connector client. Click **Edit -> Preferences -> Private Keys -> Add**, locate the private key file, and click **OK**. You do not have to add the public part of your SSH key pair, it is calculated using the private key.

SDT Connector will now use public key authentication when SSH connecting through the *console server*. You may have to restart SDT Connector to shut down any existing tunnels that were established using password authentication.

If you have a host behind the *console server* that you connect to by clicking the SSH button in SDT Connector, you can also configure it for public key authentication. Essentially what you are using is SSH over SSH, and the two SSH connections are entirely separate, and the host configuration is entirely independent of SDT Connector and the *console server*. You must configure the SSH client that SDT Connector launches (e.g. Putty, OpenSSH) and the host's SSH server for public key authentication.

15.7   Secure Sockets Layer (SSL) Support

Secure Sockets Layer (SSL) is a protocol developed by Netscape for transmitting private documents *via* the Internet. SSL works by using a private key to encrypt data that's transferred over the SSL connection.

The *console server* includes OpenSSL. The OpenSSL Project is a collaborative effort to develop a robust, commercial-grade, full-featured, and Open Source toolkit implementing the Secure Sockets Layer (SSL v2/v3) and Transport Layer Security (TLS v1) protocols as well as a full-strength general purpose cryptography library. The project is managed by a worldwide community of volunteers that use the Internet to communicate, plan, and develop the OpenSSL toolkit and its related documentation.

OpenSSL is based on the excellent SSLeay library developed by Eric A. Young and Tim J. Hudson. The OpenSSL toolkit is licensed under an Apache-style license, which basically means that you are free to get and use it for commercial and non-commercial purposes subject to some simple license conditions. In the *console server,* OpenSSL is used primarily in conjunction with 'http' to have secure browser access to the GUI management console across insecure networks.

More documentation on OpenSSL is available from:

http://www.openssl.org/docs/apps/openssl.html
http://www.openssl.org/docs/HOWTO/certificates.txt

15.8 HTTPS

The Management Console can be served using HTTPS by running the webserver *via sslwrap*. The server can be launched on request using *inetd*.

The HTTP server provided is a slightly modified version of the *fnord-httpd* from *http://www.fefe.de/fnord/*

The SSL implementation is provided by the *sslwrap* application compiled with OpenSSL support. You can find more detailed documentation at *http://www.rickk.com/sslwrap/*

If your default network address is changed or the unit is to be accessed *via* a known Domain Name, you can use the following steps to replace the default SSL Certificate and Private Key with ones tailored for your new address.

15.8.1 Generating an Encryption Key

To create a 1024 bit RSA key with a password, issue the following command on the command line of a linux host with the *openssl* utility installed:

openssl genrsa -des3 -out ssl_key.pem 1024

15.8.2 Generating a Self-Signed Certificate with OpenSSL

This example shows how to use OpenSSL to create a self-signed certificate. OpenSSL is available for most Linux distributions *via* the default package management mechanism. (Windows users can check *http://www.openssl.org/related/binaries.html*)

To create a 1024 bit RSA key and a self-signed certificate, issue the following *openssl* command from the host you have *openssl* installed on:

> *openssl req -x509 -nodes -days 1000 *
> *-newkey rsa:1024 -keyout ssl_key.pem -out ssl_cert.pem*

You will be prompted to enter a lot of information. Most of it doesn't matter, but the "Common Name" should be the domain name of your computer (*for example,* test.Black Box.com). When you have entered everything, the certificate will be created in a file called *ssl_cert.pem*.

15.8.3 Installing the Key and Certificate

We recommend that you use an SCP (Secure Copying Protocol) client to copy files securely to the *console server* unit. The *scp* utility is distributed with OpenSSH for most Unix distributions, while Windows users can use something like the PSCP command line utility available with PuTTY.

You can install remotely the files created in the steps above with the *scp* utility as follows:

> *scp ssl_key.pem root@<address of unit>:/etc/config/*
> *scp ssl_cert.pem root@<address of unit>:/etc/config/*

or using PSCP:

> *pscp -scp ssl_key.pem root@<address of unit>:/etc/config/*
> *pscp -scp ssl_cert.pem root@<address of unit>:/etc/config/*

PuTTY and the PSCP utility can be downloaded from: *http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html*

More detailed documentation on the PSCP can be found: *http://the.earth.li/~sgtatham/putty/0.58/htmldoc/Chapter5.html#pscp*

15.8.4   Launching the HTTPS Server

Note that the easiest way to enable the HTTPS server is from the web Management Console. Simply click the appropriate checkbox in **Network -> Services -> HTTPS Server** and the HTTPS server will be activated (assuming the *ssl_key.pem* & *ssl_cert.pem* files exist in the */etc/config* directory).

Alternatively *inetd* can be configured to launch the secure *fnord* server from the command line of the unit as follows.

Edit the *inetd* configuration file. From the unit command line:

> *vi /etc/config/inetd.conf*

Append a line:

> *443 stream tcp nowait root sslwrap -cert /etc/config/ssl_cert.pem -key /etc/config/ssl_key.pem -exec /bin/httpd /home/httpd"*

Save the file and signal *inetd* of the configuration change.

> *kill -HUP `cat /var/run/inetd.pid`*

The HTTPS server should be accessible from a web client at a URL similar to this: *https://<common name of unit>*

More detailed documentation about the *openssl* utility can be found at the website: *http://www.openssl.org/*

15.9 Power Strip Control

The *console server* supports a growing list of remote power-control devices (RPCs) that you can configure using the Management Console. These RPCs are controlled using the open source *PowerMan* and *Network UPS Tools* and with Black Box's *pmpower* utility.

15.9.1 The PowerMan Tool

PowerMan provides power management in a data center or compute cluster environment. It performs operations such as power on, power off, and power cycle via remote power controller (RPC) devices.

Synopsis
powerman [-option] [targets]
pm [-option] [targets]

1101 and 1102 Secure Device Servers

Options
-1, --on   Power ON targets.
-0, --off   Power OFF targets.
-c, --cycle   Power cycle targets.
-r, --reset Assert hardware reset for targets (if implemented by RPC).
-f, --flash Turn beacon ON for targets (if implemented by RPC).
-u, --unflash   Turn beacon OFF for targets (if implemented by RPC).
-l, --list   List available targets. If possible, output will be compressed into a host range (see TARGET SPECIFICATION below).
-q, --query   Query plug status of targets. If none specified, query all targets. Status is not cached; each time this option is used, powermand queries the appropriate RPC's. Targets connected to RPC's that could not be contacted (e.g. due to network failure) are reported as status "unknown". If possible, output will be compressed into host ranges.
-n, --node   Query node power status of targets (if implemented by RPC). If no targets specified, query all targets. In this context, a node in the OFF state could be ON at the plug but operating in standby power mode.
-b, --beacon   Query beacon status (if implemented by RPC). If no targets are specified, query all targets.
-t, --temp   Query node temperature (if implemented by RPC). If no targets are specified, query all targets. Temperature information is not interpreted   by powerman and is reported as received from the RPC on one line per target, prefixed by target name.
-h, --help Display option summary.
-L, --license   Show powerman license information.
-d, --destination host[:port] Connect to a powerman daemon on non-default host and optionally port.
-V, --version   Display the powerman version number and exit.
-D, --device   Displays RPC status information. If targets are specified, only RPC's matching the target list are displayed.
-T, --telemetry   Causes RPC telemetry information to be displayed as commands are processed. Useful for debugging device scripts.
-x, --exprange   Expand host ranges in query responses.

For more details refer *http://linux.die.net/man/1/powerman*

Also refer *powermand* (*http://linux.die.net/man/1/powermand*) documentation and *powerman.conf* (*http://linux.die.net/man/5/powerman.conf*)

Target Specification

powerman target hostnames may be specified as comma separated or space separated hostnames or host ranges. Host ranges are of the general form: prefix[n-m,l-k,...], where n < m and l < k, etc., This form should not be confused with regular expression character classes (also denoted by ''[]''). For example, foo[19] does not represent foo1 or foo9, but rather represents a degenerate range: foo19.

This range syntax is meant only as a convenience on clusters with a prefix NN naming convention and specification of ranges should not be considered necessary—the list foo1,foo9 could be specified as such, or by the range foo[1,9].

Some examples of powerman targets follows.

Power on hosts bar,baz,foo01,foo02,...,foo05: *powerman --on bar baz foo[01-05]*

Power on hosts bar,foo7,foo9,foo10: *powerman --on bar,foo[7,9-10]*

Power on foo0,foo4,foo5: *powerman --on foo[0,4-5]*

As a reminder to the reader, some shells will interpret brackets ([and]) for pattern matching. Depending on your shell, you might need to enclose ranged lists within quotes. For example, in tcsh, the last example above should be executed as:

powerman --on "foo[0,4-5]"

15.9.2 The pmpower Tool

The *pmpower* utility is a high level tool for manipulating remote preconfigured power devices connected to the *console server* via a serial or network connection. The PDU UPS and IPMI power devices are variously controlled using the open source *PowerMan, IPMItool,* or *Network UPS Tools*, and Black Box's *pmpower* utility arches over these tools so the devices can be controlled through one command line:

pmpower [-?h] [-l device | -r host] [-o outlet] [-u username] [-p password] action

   -?/-h This help message.
   -l   The serial port to use.
   -o   The outlet on the power target to apply to
   -r   The remote host address for the power target
   -u   Override the configured username
   -p   Override the configured password
   on   This *action* switches the specified device or outlet(s) on
   off   This *action* switches the specified device or outlet(s) off
   cycle This *action* switches the specified device or outlet(s) off and on again

724-746-5500 | blackbox.com

status This *action* retrieves the current status of the device or outlet

Examples:
To turn outlet 4 of the power device connected to serial port 2 on: # *pmpower -l port02 -o 4 on*

To turn an IPMI device off located at IP address 192.168.1.100 (where username is 'root' and password is 'calvin': # *pmpower -r 192.168.1.100 -u root -p calvin off*

Default system Power Device actions are specified in */etc/powerstrips.xml*. Custom Power Devices can be added in */etc/config/powerstrips.xml*. If an action is attempted that has not been configured for a specific Power Device, *pmpower* will exit with an error.

15.9.3 Adding New RPC Devices

There are a number of simple paths to adding support for new RPC devices.

The first is to have scripts to support the particular RPC included in either the open source *PowerMan* project (*http://sourceforge.net/projects/powerman)* or the open source *NUT UPS Tools* project. The *PowerMan* device specifications are rather weird and it is suggested that you leave the actual writing of these scripts to the PowerMan authors. Documentation on how they work can be found at *http://linux.die.net/man/5/powerman.dev*. The *Network UPS Tools (NUT)* project has recently moved on from its UPS management origins to also cover SNMP PDUs (and embrace PowerMan). Black Box progressively includes the updated *PowerMan* and *NUT* build into the *console server* firmware releases.

The second path is to directly add support for the new RPC devices (or to customize the existing RPC device support) on your particular *console server*. The **Manage: Power** page uses information contained in */etc/powerstrips.xml* to configure and control devices attached to a serial port. The configuration also looks for (and loads) */etc/config/powerstrips.xml* if it exists.

The user can add his own support for more devices by putting definitions for them into */etc/config/powerstrips.xml*. This file can be created on a host system and copied to the Management Console device using *scp*. Alternatively, login to the Management Console and use *ftp* or *wget* to transfer files.

Here is a brief description of the elements of the XML entries in */etc/config/powerstrips.xml*.

```
<powerstrip>
        <id>Name or ID of the device support</id>
        <outlet port="port-id-1">Display Port 1 in menu</outlet>
        <outlet port="port-id-2">Display Port 2 in menu</outlet>
        ...
        <on>script to turn power on</on>
        <off>script to power off</off>
        <cycle>script to cycle power</cycle>
        <status>script to write power status to /var/run/power-status</status>
        <speed>baud rate</speed>
        <charsize>character size</charsize>
        <stop>stop bits</stop>
        <parity>parity setting</parity>
</powerstrip>
```

The *id* appears on the web page in the list of available device types to configure.

The outlets describe targets that the scripts can control. For example, a power control board may control several different outlets. The port-id is the native name for identifying the outlet. This value will be passed to the scripts in the environment variable *outlet*, allowing the script to address the correct outlet.

There are four possible scripts: *on, off, cycle*, and *status*.

When a script is run, its standard input and output is redirected to the appropriate serial port. The script receives the outlet and port in the *outlet* and *port* environment variables respectively.

The script can be anything that can be executed within the shell.

All of the existing scripts in */etc/powerstrips.xml* use the *pmchat* utility.

pmchat works just like the standard unix "chat" program, only it ensures interoperation with the port manager.

The final options, *speed, charsize, stop,* and *parity* define the recommended or default settings for the attached device.

15.10 IPMItool

The *console server* includes the *ipmitool* utility for managing and configuring devices that support the Intelligent Platform Management Interface (IPMI) version 1.5 and version 2.0 specifications.

IPMI is an open standard for monitoring, logging, recovery, inventory, and control of hardware that is implemented independent of the main CPU, BIOS, and OS. The service processor (or Baseboard Management Controller, BMC) is the brain behind platform management and its primary purpose is to handle the autonomous sensor monitoring and event logging features.

The *ipmitool* program provides a simple command-line interface to this BMC. It features the ability to read the sensor data repository (SDR) and print sensor values, display the contents of the System Event Log (SEL), print Field Replaceable Unit (FRU) inventory information, read and set LAN configuration parameters, and perform remote chassis power control.

SYNOPSIS
ipmitool [**-c**|**-h**|**-v**|**-V**] **-I** *open <command>*
ipmitool [**-c**|**-h**|**-v**|**-V**] **-I** *lan* **-H** *<hostname>*
　　　　[**-p** *<port>*]
　　　　[**-U** *<username>*]
　　　　[**-A** *<authtype>*]
　　　　[**-L** *<privlvl>*]
　　　　[**-a**|**-E**|**-P**|**-f** *<password>*]
　　　　[**-o** *<oemtype>*]
　　　　<command>
ipmitool [**-c**|**-h**|**-v**|**-V**] **-I** *lanplus* **-H** *<hostname>*
　　　　[**-p** *<port>*]
　　　　[**-U** *<username>*]
　　　　[**-L** *<privlvl>*]
　　　　[**-a**|**-E**|**-P**|**-f** *<password>*]
　　　　[**-o** *<oemtype>*]
　　　　[**-C** *<ciphersuite>*]
　　　　<command>

DESCRIPTION
This program lets you manage Intelligent Platform Management Interface (IPMI) functions of either the local system, via a kernel device driver, or a remote system, using IPMI V1.5 and IPMI v2.0. These functions include printing FRU information, LAN configuration, sensor readings, and remote chassis power control.

IPMI management of a local system interface requires a compatible IPMI kernel driver to be installed and configured. On Linux, this driver is called *OpenIPMI* and it is included in standard distributions. On Solaris, this driver is called *BMC* and is included in Solaris 10. Management of a remote station requires the IPMI-over-LAN interface to be enabled and configured. Depending on the particular requirements of each system, it may be possible to enable the LAN interface using *ipmitool* over the system interface.

OPTIONS
-a　　　　Prompt for the remote server password.
-A *<authtype>*
　　　　Specify an authentication type to use during IPMIv1.5 *lan* session activation. Supported types are NONE, PASSWORD, MD5, or OEM.
-c　　　　Present output in CSV (comma separated variable) format. This is not available with all commands.
-C *<ciphersuite>*
　　　　The remote server authentication, integrity, and encryption algorithms to use for IPMIv2 *lanplus* connections. See table 22-19 in the IPMIv2 specification. The default is 3 which specifies RAKP-HMAC-SHA1 authentication, HMAC-SHA1-96 integrity, and AES-CBC-128 encryption algorightms.
-E　　　　The remote server password is specified by the environment variable *IPMI_PASSWORD*.
-f *<password_file>*
　　　　Specifies a file containing the remote server password. If this option is absent, or if password_file is empty, the password will default to NULL.
-h　　　　Get basic usage help from the command line.
-H *<address>*
　　　　Remote server address, can be IP address or hostname. This option is required for *lan* and *lanplus* interfaces.
-I *<interface>*
　　　　Selects IPMI interface to use. Supported interfaces that are compiled in are visible in the usage help output.
-L *<privlvl>*
　　　　Force session privilege level. Can be CALLBACK, USER, OPERATOR, ADMIN. Default is ADMIN.
-m *<local_address>*
　　　　Set the local IPMB address. The default is 0x20 and there should be no need to change it for normal operation.
-o *<oemtype>*
　　　　Select OEM type to support. This usually involves minor hacks in place in the code to work around quirks in various BMCs from various manufacturers. Use *-o list* to see a list of current supported OEM types.

-p *<port>*
> Remote server UDP port to connect to. Default is 623.

-P *<password>*
> Remote server password is specified on the command line. If supported, it will be obscured in the process list.
> > **Note!** Specifying the password as a command line option is not recommended.

-t *<target_address>*
> Bridge IPMI requests to the remote target address.

-U *<username>*
> Remote server username, default is NULL user.

-v
> Increase verbose output level. This option may be specified multiple times to increase the level of debug output. If given three times you will get hexdumps of all incoming and outgoing packets.

-V
> Display version information.

If no password method is specified, then *ipmitool* will prompt the user for a password. If no password is entered at the prompt, the remote server password will default to NULL.

SECURITY

The *ipmitool* documentation highlights that there are several security issues to be considered before enabling the IPMI LAN interface. A remote station has the ability to control a system's power state as well as being able to gather certain platform information. To reduce vulnerability, we strongly advise that the IPMI LAN interface only be enabled in 'trusted' environments where system security is not an issue or where there is a dedicated secure 'management network' or access has been provided through an *console server*.

Further, we strongly advise that you do not enable IPMI for remote access without setting a password, and that password should not be the same as any other password on that system.

When an IPMI password is changed on a remote machine with the IPMIv1.5 *lan* interface, the new password is sent across the network as clear text. This could be observed and then used to attack the remote system. We recommend that IPMI password management only be done over IPMIv2.0 *lanplus* interface or the system interface on the local station.

For IPMI v1.5, the maximum password length is 16 characters. Passwords longer than 16 characters will be truncated.

For IPMI v2.0, the maximum password length is 20 characters; longer passwords are truncated.

COMMANDS

help

This can be used to get command-line help on *ipmitool* commands. It may also be placed at the end of commands to get option usage help.

ipmitool help

Commands:
> *raw*   Send a RAW IPMI request and print
> > response
> *lan*   Configure LAN Channels
> *chassis* Get chassis status and set power
> > state
> *event*   Send pre-defined events to MC
> *mc*   Management Controller status and
> > global enables
> *sdr*   Print Sensor Data Repository
> > entries and readings
> *sensor* Print detailed sensor information
> *fru*   Print built-in FRU and scan SDR
> > for FRU locators
> *sel*   Print System Event Log (SEL)
> *pef*   Configure Platform Event Filtering
> > (PEF)
> *sol*   Configure IPMIv2.0 Serial-over-LAN
> *isol*   Configure IPMIv1.5 Serial-over-LAN
> *user*   Configure Management Controller
> > users
> *channel*   Configure Management Controller
> > channels
> *session*   Print session information

exec   Run list of commands from file
set   Set runtime variable for shell and
        exec

ipmitool chassis help

Chassis Commands: status, power, identify, policy, restart_cause, poh, bootdev

ipmitool chassis power help

chassis power Commands: status, on, off, cycle, reset, diag, soft

You will find more details on *ipmitools* at http://*ipmitool*.sourceforge.net/manpage.html

15.11 Custom Development Kit (CDK)

As detailed in this manual, customers can copy scripts, binaries, and configuration files directly to the *console server*.

Black Box also freely provides a development kit that allows changes to be made to the software in *console server* firmware image. The customer can use the CDK to:

generate a firmware image without certain programs, such as telnet, which may be banned by company policy.

generate an image with new programs, such as custom Nagios plug-in binaries or company specific binary utilities.

generate an image with custom defaults, for example, it may be required that the *console server* be configured to have a specific default serial port profile which is reverted to even in event of a factory reset.
place configuration files into the firmware image, which cannot then be modified e.g. # /bin/config –-set= tools update the configuration files in /etc/config which are read/write, whereas the files in /etc are read only and cannot be modified.

The CDK essentially provides a snapshot of the Black Box build process (taken after the programs have been compiled and copied to a temporary directory *romfs*) just before the compressed file systems are generated. You can obtain a copy of the Black Box CDK for the particular appliance you are working with from Black Box

NOTE: The CDK is free.

15.12 Scripts for Managing Slaves

When the *console server*s are cascaded the Master is in control of the serial ports on the Slaves, and the Master's Management Console provides a consolidated view of the settings for its own and all the Slave's serial ports. The Master does not provide a fully consolidated view, for example, *Status: Active Users* only displays those users active on the Master's ports and you will need to write a custom bash script that parses the port logs if you want to find out who's logged in to cascaded serial ports from the master.

You will probably also want to enable remote or USB logging, because local logs only buffer 8K of data and don't persist between reboots.

This script would, for example, parse each port log file line by linee. Each time it sees '*LOGIN: username*', it adds username to the list of connected users for that port. Each time it sees '*LOGOUT: username,*' it removes it from the list. The list can then be nicely formatted and displayed. You can run the script on the remote log server. To enable log storage and connection logging:

- Select *Alerts & Logging: Port Log*
- *Configure* log storage
- Select *Serial & Network: Serial Port*, *Edit* the serial port(s)
- Under *Console server*, select *Logging Level 1* and click Apply
There's a useful tutorial on creating a bash script CGI at http://www.yolinux.com/TUTORIALS/LinuxTutorialCgiShellScript.html

Similarly, the Master does maintain a view of the status of the slaves:
- Select *Status: Support Report*
- Scroll down to *Processes*
- Look for: */bin/ssh -MN -o ControlPath=/var/run/cascade/%h slavename*
- These are the slaves that are connected
- Note the end of the Slaves' names will be truncated, so the first 5 characters must be unique

Alternatively, you can write a custom CGI script as described above. The currently connected Slaves can be determined by running: *ls /var/run/cascade* and the configured slaves can be displayed by running: *config -g config.cascade.slaves*

Appendix A. Linux Commands and Source Code

The *console server* platform is a dedicated Linux computer, optimized to provide monitoring and secure access to serial and network consoles of critical server systems and their supporting power and networking infrastructure.

Black Box *console servers* are built on the 2.4 uCLinux kernel as developed by the uCLinux project. This is GPL code and source can be found at *http://cvs.uclinux.org*. Some uCLinux commands have config files that can be altered (e.g. *portmanager, inetd, init, ssh/sshd/scp/sshkeygen, ucd-snmpd, samba, fnord, sslwrap*). Other commands you can run and do neat stuff with (e.g. *loopback, bash (shell), ftp, hwclock, iproute, iptables, netcat, ifconfig, mii-tool, netstat, route, ping, portmap, pppd, routed, setserial, smtpclient, stty, stunel, tcpdump, tftp, tip, traceroute*) Below are most of the standard uCLinux and BusyBox commands (and some custom Black Box commands) that are in the default build tree. The *Administrator* can use these to configure the *console server*, and monitor and manage attached serial console and host devices:

addgroup *	Add a group or add an user to a group
adduser *	Add an user
agetty	alternative Linux getty
arp	Manipulate the system ARP cache
arping	Send ARP requests/replies
bash	GNU Bourne-Again Shell
busybox	Swiss army knife of embedded Linux commands
cat *	Concatenate FILE(s) and print them to stdout
chat	Useful for interacting with a modem connected to stdin/stdout
chgrp *	Change file access permissions
chmod *	Change file access permissions
chown *	Change file owner and group
config	Black Box tool to manipulate and query the system configuration from the command line
cp *	Copy files and directories
date *	Print or set the system date and time
dd *	Convert and copy a file
deluser *	Delete USER from the system
df *	Report filesystem disk space usage
dhcpd	Dynamic Host Configuration Protocol server
discard	Network utility that listens on the discard port
dmesg *	Print or control the kernel ring buffer
echo *	Print the specified ARGs to stdout
erase	Tool for erasing MTD partitions
eraseall	Tool for erasing entire MTD partitions
false *	Do nothing, unsuccessful
find	Search for files
flashw	Write data to individual flash devices
flatfsd	Daemon to save RAM file systems back to FLASH
ftp	Internet file transfer program
gen-keys	SSH key generation program
getopt *	Parses command options
gettyd	Getty daemon
grep *	Print lines matching a pattern
gunzip *	Compress or expand files
gzip *	Compress or expand files
hd	ASCII, decimal, hexadecimal, octal dump
hostname *	Get or set hostname or DNS domain name
httpd	Listen for incoming HTTP requests
hwclock	Query and set hardware clock (RTC)
inetd	Network super-server daemon
inetd-echo	Network echo utility
init	Process control initialization
ip	Show or manipulate routing, devices, policy routing and tunnels
ipmitool	Linux IPMI manager
iptables	Administration tool for IPv4 packet filtering and NAT

ip6tables	Administration tool for IPv6 packet filtering
iptables-restore	Restore IP Tables
iptables-save	Save IP Tables
kill *	Send a signal to a process to end gracefully
ln *	Make links between files
login	Begin session on the system
loopback	Black Box loopback diagnostic command
loopback1	Black Box loopback diagnostic command
loopback2	Black Box loopback diagnostic command
loopback8	Black Box loopback diagnostic command
loopback16	Black Box loopback diagnostic command
loopback48	Black Box loopback diagnostic command
ls *	List directory contents
mail	Send and receive mail
mkdir *	Make directories
mkfs.jffs2	Create an MS-DOS file system under Linux
mknod *	Make block or character special files
more *	File perusal filter for crt viewing
mount *	Mount a file system
msmtp	SMTP mail client
mv *	Move (rename) files
nc	TCP/IP Swiss army knife
netflash	Upgrade firmware on ucLinux platforms using the blkmem interface
netstat	Print network connections, routing tables, interface statistics etc
ntpd	Network Time Protocol (NTP) daemon
pgrep	Display process(es) selected by regex pattern
pidof	Find the process ID of a running program
ping	Send ICMP ECHO_REQUEST packets to network hosts
ping6	IPv6 ping
pkill	Sends a signal to process(es) selected by regex pattern
pmchat	Black Box command similar to the standard chat command (via portmanager)
pmdeny	
pminetd	
pmloggerd	
pmshell	Black Box command similar to the standard *tip* or *cu* but all serial port access is directed via the portmanager.
pmusers	Black Box command to query portmanager for active user sessions
portmanager	Black Box command that handles all serial port access
portmap	DARPA port to RPC program number mapper
pppd	Point-to-Point protocol daemon
ps *	Report a snapshot of the current processes
pwd *	Print name of current/working directory
reboot *	*Soft* reboot
rm *	Remove files or directories
rmdir *	Remove empty directories
routed	Show or manipulate the IP routing table
routed	Show or manipulate the IP routing table
routef	IP Route tool to flush IPv4 routes
routel	IP Route tool to list routes
rtacct	Applet printing /proc/net/rt_acct
rtmon	RTnetlink listener
scp	Secure copy (remote file copy program)
sed *	Text stream editor
setmac	Sets the MAC address
setserial	Sets and reports serial port configuration
sh	Shell
showmac	Shows MAC address

sleep *	Delay for a specified amount of time
smbmnt	Helper utility for mounting SMB file systems
smbmount	Mount an SMBFS file system
smbumount	SMBFS umount for normal users
snmpd	SNMP daemon
snmptrap	Sends an SNMP notification to a manager
sredird	RFC 2217 compliant serial port redirector
ssh	OpenSSH SSH client (remote login program)
ssh-keygen	Authentication key generation, management, and conversion
sshd	OpenSSH SSH daemon
sslwrap	Program that allows plain services to be accessed via SSL
stty	Change and print terminal line settings
stunnel	Universal SSL tunnel
sync *	Flush file system buffers
sysctl	Configure kernel parameters at runtime
syslogd	System logging utility
tar *	The tar archiving utility
tc	Show traffic control settings
tcpdump	Dump traffic on a network
telnetd	Telnet protocol server
tftp	Client to transfer a file from/to tftp server
tftpd	Trivial file Transfer Protocol (tftp) server
tip	Simple terminal emulator/cu program for connecting to modems and serial devices
top	Provide a view of process activity in real time
touch *	Change file timestamps
traceroute	Print the route packets take to network host
traceroute6	Traceroute for IPv6
true *	Returns an exit code of TRUE (0)
umount *	Unmount file systems
uname *	Print system information
usleep *	Delay for a specified amount of time
vconfig *	Create and remove virtual ethernet devices
vi *	Busybox clone of the VI text editor
w	Show who is logged on and what they are doing
zcat *	Identical to gunzip -c

Commands above which are appended with '*' come from BusyBox (the Swiss Army Knife of embedded Linux)
http://www.busybox.net/downloads/BusyBox.html. Others are generic Linux commands and most commands the **-h** or **--help** argument to provide a terse runtime description of their behavior. More details on the generic Linux commands can found online at
http://en.tldp.org/HOWTO/HOWTO-INDEX/howtos.html and http://www.faqs.org/docs/Linux-HOWTO/Remote-Serial-Console-HOWTO.html

An updated list of the commands may found using **ls** command to view all the commands actually available in the /bin directory in your *console server*.

There were a number of Black Box tools listed above that make it simple to configure the *console server* and make sure the changes are stored in the *console server*'s flash memory, etc. These commands are covered in the previous chapters and include:

config which allows manipulation and querying of the system configuration from the command line. With *config* a new configuration can be activated by running the relevant configurator, which performs the action necessary to make the configuration changes live.

portmanager which provides a buffered interface to each serial port. It is supported by the pmchat and pmshell commands which ensure all serial port access is directed via the *portmanager*.

pmpower is a configurable tool for manipulating remote power devices that are serially or network connected to the *console server*.

SDT Connector is a java client applet that provides point-and-click SSH tunneled connections to the *console server* and Managed Devices.

1101 and 1102 Secure Device Servers

There are also a number of other CLI commands related to other open source tools embedded in the *console server* including:

PowerMan provides power management for many preconfigured remote power controller (RPC) devices. For CLI details refer
http://linux.die.net/man/1/powerman

Network UPS Tools (NUT) provides reliable monitoring of UPS and PDU hardware and ensure safe shutdowns of the systems which are connected - with a goal to monitor every kind of UPS and PDU. For CLI details refer *http://www.networkupstools.org*

Nagios is a popular enterprise-class management tool that provides central monitoring of the hosts and services in distributed networks. For CLI details refer http://www.nagios.org

Many components of the *console server* software are licensed under the GNU General Public License (version 2), which Black Box supports. You may obtain a copy of the GNU General Public License at **http://www.fsf.org/copyleft/gpl.html**. Black Box will provide source code for any of the components of the software licensed under the GNU General Public License upon request.

The *console server* also embodies the *okvm* console management software. This is GPL code and the full source is available from *http://okvm.sourceforge.net*.

The *console server* BIOS (boot loader code) is a port of **uboot**, which is also a GPL package with source openly available.

The *console server* CGIs (the html code, xml code and web config tools for the Management Console) are proprietary to Black Box, however the code will be provided to customers, under NDA.

Also inbuilt in the *console server* is a Port Manager application and Configuration tools as described in *Chapters 15* and *16*. These both are proprietary to Black Box, but open to customers (as above).

The *console server* also supports GNU *bash* shell script enabling the *Administrator* to run custom scripts. GNU *bash*, version 2.05.0(1)-release (arm-Black Box-linux-gnu) offers the following shell commands:

```
alias [-p] [name[=value] ... ]          local name[=value] ...
bg [job_spec]                           logout
bind [-lpvsPVS] [-m keymap] [-f fi      popd [+N | -N] [-n]
    break [n]                           printf format [arguments]
builtin [shell-builtin [arg ...]]       pushd [dir | +N | -N] [-n]
case WORD in [PATTERN [| PATTERN]       pwd [-PL]
cd [-PL] [dir]                          read [-ers] [-t timeout] [-p promp]
command [-pVv]                          readonly [-anf] [name ...] or read return [n]
command [arg ...]                       select NAME [in WORDS ... ;] do
compgen [-abcdefjkvu] [-o option]       COMMANDS
complete [-abcdefjkvu] [-pr] [-o o]     set [--abefhkmnptuvxBCHP] [-o opti]
continue [n]                            shift [n]
declare [-afFrxi] [-p] name[=value]     shopt [-pqsu] [-o long-option] opt
dirs [-clpv] [+N] [-N]                  source filename
disown [-h] [-ar] [jobspec ...]         suspend [-f]
echo [-neE] [arg ...]                   test [expr]
enable [-pnds] [-a] [-f filename]       time [-p] PIPELINE
eval [arg ...]                          times
exec [-cl] [-a name] file [redirec]     trap [arg] [signal_spec ...]
exit [n]                                true
export [-nf] [name ...] or export       type [-apt] name [name ...]
false                                   typeset [-afFrxi] [-p] name[=value ulimit [-
fc [-e ename] [-nlr] [first] [last]     SHacdflmnpstuv] [limit]
fg [job_spec]                           umask [-p] [-S] [mode]
for NAME [in WORDS ... ;] do COMMA      unalias [-a] [name ...]
function NAME { COMMANDS ; } or NA      unset [-f] [-v] [name ...]
getopts optstring name [arg]            until COMMANDS; do COMMANDS; done
hash [-r] [-p pathname] [name ...]      variables - Some variable names an wait [n]
help [-s] [pattern ...]                 while COMMANDS; do COMMANDS; done {
history [-c] [-d offset] [n] or hi      COMMANDS ; }
if COMMANDS; then COMMANDS; [ elif
jobs [-lnprs] [jobspec ...] or job kill [-s sigspec
| -n signum | -si let arg [arg ...]
```

724-746-5500 | blackbox.com