HP SmartStart Scripting Toolkit Linux Edition User Guide

April 2006 (First Edition)
Part Number 415597-001

April 2006 (First Edition)

Part Number 415597-001

Audience assumptions

The Toolkit is designed for IT experts with experience in scripting operating system installations and configuring ProLiant server hardware.

Contents

Introduction

In this section

SmartStart Scripting Toolkit

The SmartStart Scripting Toolkit is a server deployment product that delivers an unattended automated installation for high-volume server deployments. This document describes how to best utilize the Toolkit to configure HP ProLiant servers. It also contains information about the Toolkit utilities and how to use them in an unattended environment. This document does **not** include information about installing the operating system.

The Toolkit is designed for IT experts with experience in scripting operating system installations and configuring ProLiant server hardware.

⚠ **CAUTION:** Improper use of the Toolkit utilities can result in loss of critical data. Because of the potential data-loss risk, only experienced individuals should use the Toolkit utilities. Before using the Toolkit, all necessary precautions must be taken to ensure that mission-critical systems remain online if a failure occurs.

Minimum requirements

Before beginning the deployment process, be sure to have the following items available:

- SmartStart Scripting Toolkit Linux Edition
- *HP SmartStart Scripting Toolkit Linux Edition User Guide*
- A Linux workstation (any Linux distribution)

Using the Linux Toolkit environment

In this section

Basic Toolkit strategy

The Linux Toolkit environment is a small, multipurpose Linux environment that can be booted from various media, such as CD, network or PXE, or bootable USB device. After the Linux Toolkit environment is booted, it has virtually the same capabilities as a regular Linux distribution.

The general tasks for setting up a target server for operating system installation include:

1. Setting up the Linux Toolkit boot environment
2. Modifying the install scripts:
   - Configuring system hardware
   - Performing operating system preinstallation tasks
3. Setting up a network share

Steps 1 and 2 are described in more detail in the "Setting up the Linux Toolkit boot environment (on page 5)" and "Modifying the installation scripts (on page 11)" sections. To perform step 3, refer to your Linux system administrator's guide.

Because the Toolkit is Linux-based, this document assumes that the operations described will be performed under Linux. Unless otherwise stated, a full Linux installation of any distribution of Linux released in the last two years enables you to perform all the operations described in this document.

Setting up the Linux Toolkit boot environment

The three main components needed to boot any Linux environment are the bootloader, the Linux kernel, and the Linux file system. For the Linux edition of the SmartStart Scripting Toolkit, these components are:

- **Bootloader:** SYSLINUX

  In general, you must modify only the bootloader configuration to boot the Toolkit in your environment.
- **Kernel:** vmlinuz, a kernel based on SLES 9

NOTE: The kernel is generally static and cannot be modified easily. HP recommends that you use the kernel that is shipped with the Toolkit because it has been tested on all servers supported by the Toolkit.

- **File system:** initrd.img, a reduced SLES 9 environment

  The bootstrap script in the initrd.img file locates the customized install script in the execution path and executes it, beginning the Toolkit process. Because this script does not reside inside initrd.img, you can modify it as often as needed without needing to rebuild the initrd.img file.

The following general steps occur during the boot process:

1. The system boots using the available media (CD, PXE, or USB drive key).
2. The bootloader (SYSLINUX) loads the Toolkit environment.
3. A startup script is executed to finish the environment setup.

SYSLINUX

SYSLINUX is a free third-party bootloader available at the SYSLINUX webpage (http://syslinux.zytor.com/index.php).

SYSLINUX consists of a suite of programs that perform various boot functions. The Toolkit uses the following bootloader programs:

- isolinux.bin—This program enables you to boot from ISO media.
- pxelinux.0—This program enables you to boot using PXE protocol.
- ldlinux.sys—This program enables you to boot from a USB drive key.

The bootloaders each require a configuration file to run:

- isolinux.cfg—This configuration file is used for booting from ISO media.
- default—This configuration file is used for booting using PXE.
- syslinux.cfg—This configuration file is used for booting from a USB drive key.

The following is a sample syslinux.cfg file:

```
say ===============================================================
say HP SmartStart Toolkit Linux Edition 1.30
say Copyright 2001, 2005 Hewlett-Packard Development Company, L.P.
say ---------------------------------------------------------------
say Instructions for boot:
say press <enter> key for normal toolkit boot
say type 'bash' and press <enter> key for bash shell
default toolkit
prompt 1
timeout 300

#
# For Networking, add "network=1" to the append line in the toolkit
section.
#

label toolkit
   kernel vmlinuz

   append initrd=initrd.img root=/dev/ram0 rw ramdisk_size=89000 quiet=1
   ide=nodma ide=noraid pnpbios=off usb=1 sstk_mount=/dev/sda
   sstk_mount_type=vfat sstk_script=custom.sh

label bash
   kernel vmlinuz

   append initrd=initrd.img root=/dev/ram0 rw ramdisk_size=89000 single
   debug console=ttyS0,115200n8 console=tty0 ide=nodma ide=noraid
   pnpbios=off
```

In this example, there are two distinct boot directives: `toolkit` and `bash`. The configuration file instructs the bootloader to prompt the user to pick a boot directive, wait 10 seconds, and then boot the default directive, `toolkit`, if no user input is recorded.

The `toolkit` directive instructs the bootloader to use vmlinuz as the kernel. The `append` line specifies which parameters the bootloader passes to the kernel.

In the boot files (isolinux.cfg, default, and syslinux.cfg), the following options are supported in the `append` statements.

Option	Description
`ssstk_mount=<device>`	This command specifies the device node or name to mount; for example, `/dev/hdc` or `10.0.0.1:/nfs_bootstrap`.
`ssstk_mount_type=<mount type>`	This command specifies the file system type of the device; for example, `nfs`, `vfat`, or `iso9660`.
`ssstk_mount_options=<mount options>`	This command specifies the options for mounting the device; for example, `ro`, or `ro,nolock` for NFS.
`ssstk_script=<script filename>`	This command specifies the administrator-created script that will be executed to continue the process. Typically, the script uses Toolkit tools to configure and update the system and then begins an operating system installation.

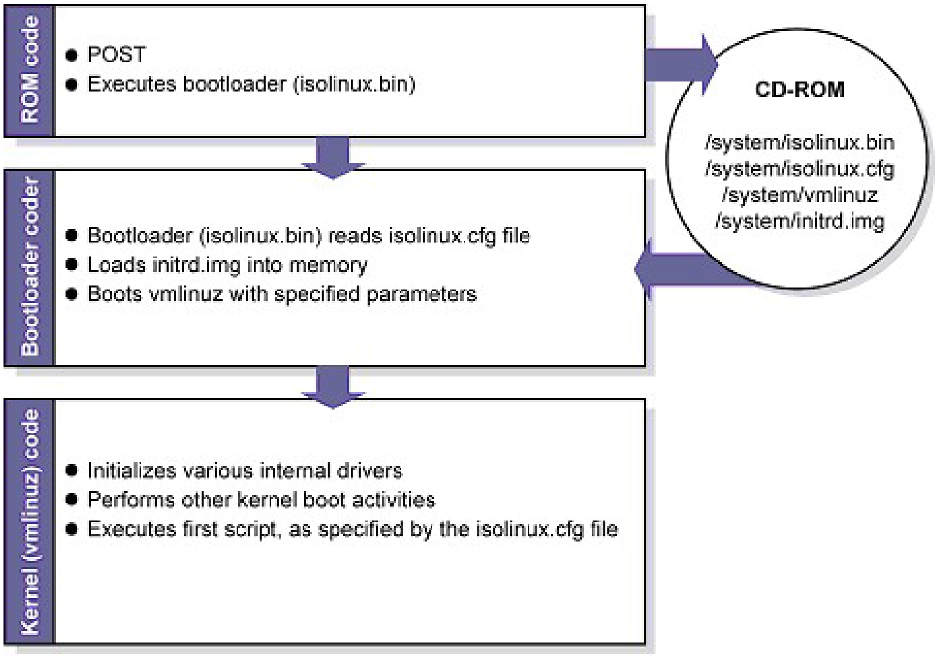
The bootstrap script included with the Toolkit performs the following commands:

```
mount -t $ssstk_mount_type $ssstk_mount /mnt/main -o
$ssstk_mount_options
exec /mnt/main/$ssstk_script
```

For more information about SYSLINUX usage, refer to the SYSLINUX documentation.

Booting from the CD

The following figure illustrates a Toolkit CD boot.


ROM code
POST
Executes bootloader (isolinux.bin)
CD-ROM
/system/isolinux.bin
/system/isolinux.cfg
/system/vmlinuz
/system/initrd.img
Bootloader coder
Bootloader (isolinux.bin) reads isolinux.cfg file
Loads initrd.img into memory
Boots vmlinuz with specified parameters
Kernel (vmlinuz) code
Initializes various internal drivers
Performs other kernel boot activities
Executes first script, as specified by the isolinux.cfg file

Creating a Toolkit bootable CD requires the following general steps:

1. Create a CD build directory.
2. Create an ISO image to be written to CD.

Creating a CD build directory

1. Create a directory on the Linux workstation; for example, ./linuxbootCD.
2. Create a subdirectory on which to store boot files; for example, ./linuxbootCD/system.
3. Copy the necessary boot files to the ./linuxbootCD/system directory:
    - isolinux.bin (the SYSLINUX binary used for ISO media)
    - isolinux.cfg (the boot configuration used by isolinux.bin)
    - initrd.img (the Linux file system)
    - vmlinuz (the Linux kernel)

If needed, copy additional files to ./linuxbootCD. These files might include Toolkit files, configuration files, or third-party tools. Alternately, refer to "Using the Linux Toolkit environment (on page 5)" for information about downloading the files from a network share.

Creating an ISO image

The `mkisofs` command is used to create an ISO image. The following table describes the arguments used with this command.

Argument	Description
`-o linuxbootCD.iso`	This argument is the output of the mkisofs command, the ISO file.
`-b system/isolinux.bin`	This argument sets isolinux.bin as the bootloader.
`-V LinuxBootCD`	This argument sets the volume label of the CD.
`./linuxbootCD`	This argument specifies the target directory that will be the root of the CD.

To create the ISO image, execute the following command at the shell prompt:

```
mkisofs -J -iso-level 3 -R -L -o linuxbootCD.iso \
   -b system/isolinux.bin -c system/boot.cat \
   -V LinuxBootCD \
   -no-emul-boot -boot-load-size 4 \
   -boot-info-table \
   ./linuxbootCD
```

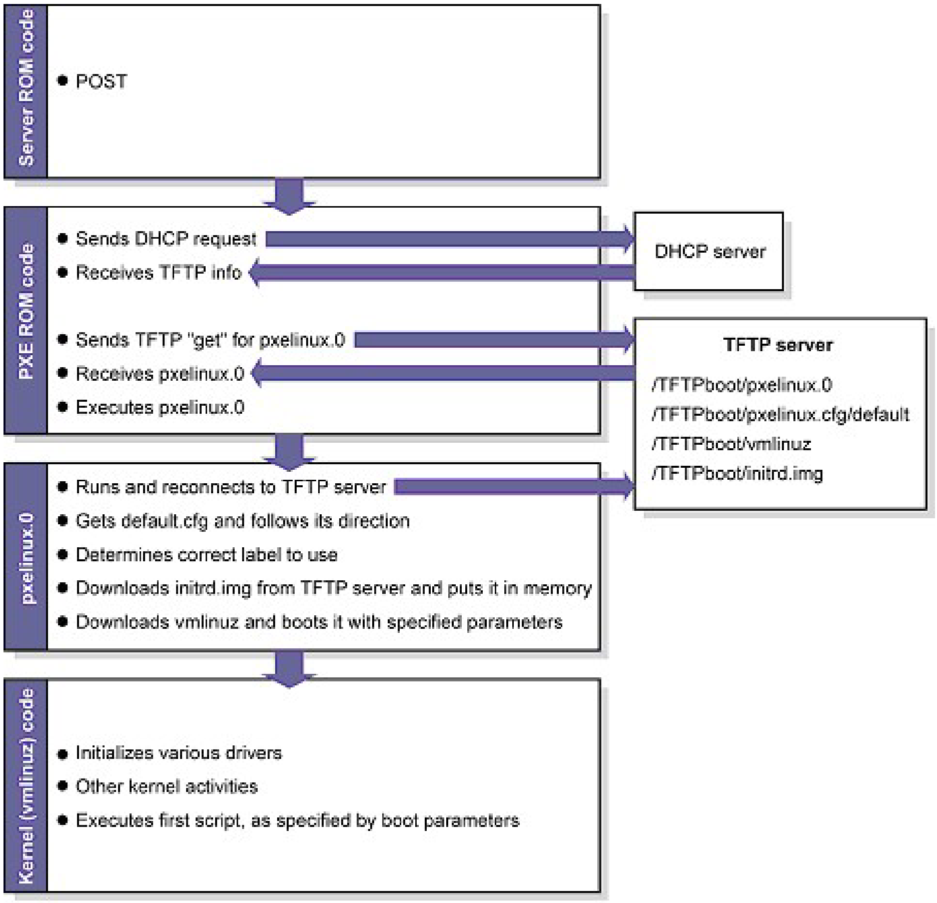
The ISO file can now be written to a CD.

Booting using PXE

IMPORTANT: A basic understanding of DHCP, PXE, and TFTP is required to perform the procedure described in this section.

NOTE: Examples in this section might not be specific to your operating system environment. Refer to the Linux system administrator's guide for more information about your particular environment.

The following figure illustrates a simplified Toolkit PXE boot.


Server ROM code
POST
PXE ROM code
Sends DHCP request
Receives TFTP info
Sends TFTP "get" for pxelinux.0
Receives pxelinux.0
Executes pxelinux.0
DHCP server
TFTP server
/TFTPboot/pxelinux.0
/TFTPboot/pxelinux.cfg/default
/TFTPboot/vmlinuz
/TFTPboot/initrd.img
pxelinux.0
Runs and reconnects to TFTP server
Gets default.cfg and follows its direction
Determines correct label to use
Downloads initrd.img from TFTP server and puts it in memory
Downloads vmlinuz and boots it with specified parameters
Kernel (vmlinuz) code
Initializes various drivers
Other kernel activities
Executes first script, as specified by boot parameters

Setting up a PXE boot environment requires the following general steps:

1. Set up a DHCP server with the appropriate options.
2. Set up a TFTP server with the appropriate options.
3. Populate the TFTP directory share with the Linux Toolkit boot components.

These steps assume that a Linux workstation is used as the DHCP/TFTP server. You might need to download additional components and adapt the following instructions to suit your environment.

Setting up a DHCP server

To set up a DHCP server, create and edit a dhcpd.conf file in the /etc directory of your server. The following is a sample dhcpd.conf file:

```
allow booting;
allow bootp;
ddns-update-style ad-hoc;
```

[You must customize the file by inserting the appropriate DHCP directives, IP ranges, subnet masks, and so on here.]

```
next-server IP_ADDRESS_OF_TFTP_SERVER;
filename "pxelinux.0";
```

The `next-server` command tells the DHCP client where to send the TFTP get request.

The `filename` command tells the DHCP client which file to get. In this example, the file is pxelinux.0.

When you have finished creating the dhcpd.conf file, restart the DHCP server:

```
/etc/init.d/dhcpd restart
```

Setting up a TFTP server

Most Linux installations include a TFTP server and an automated method of launching the server upon receiving a TFTP request. The parent process for detecting a TFTP request and launching the TFTP server is called xinetd. However, you might have to enable the TFTP service. The TFTP file is located in the /etc/xinetd.d/ directory. The following is a sample TFTP file:

```
# default: off
# description: The tftp server serves files using the \
# trivial file transfer protocol. The tftp protocol is \
# often used to boot diskless workstations, download \
# configuration files to network-aware printers and to \
# start the installation process for some operating systems.
service tftp
{
   socket_type = dgram
   protocol = udp
   wait = yes
   user = root
   server = /usr/sbin/in.tftpd
   server_args = -s /tftpboot
   disable = yes
   per_source = 11
   cps = 100 2
}
```

In this example, "disabled" is the default setting, and /tftpboot is the root directory for all client access.

To enable the TFTP service on your server, edit the disable line to read:

```
disable = no
```

Populating the TFTP directory share

Use the /tftpboot directory from the "Setting up a TFTP server (on page 10)" section as the directory from which TFTP clients will get their files. To populate the TFTP directory share:

1. Create a /tftpboot directory, if needed.
2. Copy the necessary boot files to the /tftpboot directory:
   - pxelinux.0 (the SYSLINUX binary used for PXE boot)
   - initrd.img (the Linux file system)
   - vmlinuz (the Linux kernel)
3. Create a pxelinux configuration file subdirectory called /tftpboot/pxelinux.cfg.
4. Copy the default file (the boot configuration used by pxelinux.0) into the /tftpboot/pxelinux.cfg/ directory.

The tftpboot directory should now contain the following items:

```
/tftpboot/pxelinux.0
/tftpboot/initrd.img
/tftpboot/vmlinuz
/tftpboot/pxelinux.cfg/default
```

Booting from a USB drive key

Some applications require the use of a writable medium. While booting from CD is not suitable for this purpose, a USB drive key provides the ideal medium for this type of activity.

NOTE: Booting from a USB drive key is supported only on the HP ProLiant BL20p G3 Server and ProLiant G4 and later servers.

To set up a USB drive key to boot the Toolkit environment:

1. Create a FAT file system. In this example, the USB drive key is /dev/sda:

   ```
   cd toolkit/
   mkdosfs -I /dev/sda
   ```

2. Use SYSLINUX to install the ldlinux.sys bootloader:

   ```
   ./boot_files/syslinux /dev/sda
   ```

3. Copy the boot and Toolkit files to the USB drive key:

   ```
   mount /dev/sda /mnt/usbkey/
   cp boot_files/syslinux.cfg boot_files/vmlinuz boot_files/initrd.img
   /mnt/usbkey/
   cp -a scripts/ /mnt/usbkey/
   cp -a utilities/ /mnt/usbkey/
   cp -a linux_unattend/ /mnt/usbkey/
   ```

4. Customize the scripts for your environment:

   ```
   vi /mnt/usbkey/linux_unattend/rhel4/syslinux-rh.cfg
   ```

   In the syslinux-rh.cfg file, be sure to modify the `sstk_script` boot option parameter to refer to your customized script.

   ```
   vi /mnt/usbkey/syslinux.cfg
   vi /mnt/usbkey/scripts/install_rhel4.sh
   ```

   Be sure to modify the syslinux.cfg and install_rhel4.sh files to refer to your network server.

5. Copy the bootdisk image from the Red Hat CD:

   ```
   cp RHEL4-i386-AS-disc1.iso/images/diskboot.img
   /mnt/usbkey/linux_unattend/rhel4/
   umount /mnt/usbkey/
   ```

6. Test the USB boot process:
   a. Insert the USB drive key in the server. If the server already has a C drive, change the IPL order to ensure the USB drive key boots before the C drive of the primary controller. After the drive key boots, the syslinux information and a prompt appear.
   b. Press the **Enter** key at the "Boot:" prompt. A progress bar and the Toolkit boot messages appear. After processing is complete, the /custom.sh script on the USB drive key is executed.

The drive key has been successfully prepared.

Modifying the installation scripts

Sample scripts are provided with the Toolkit to simplify the installation process. These scripts are used for:

- System hardware configuration
- Operating system preinstallation configuration

However, these scripts **must** be modified for your particular environment.

System hardware configuration

The sample install_rhel4.sh script performs many hardware configuration tasks, including:

- Copying all toolkit utilities from the network share to the target server
- Running hardware discovery to determine server type
- Obtaining server ID information from the hardware discovery file
- Copying server-specific configuration script and data files from the network share
- Loading drivers for storage controllers and any other devices that must be configured
- Running the CONREP utility
- Running the CPQACUXE utility, if needed (this action is server-specific)
- Running any other configuration utilities
- Running hardware discovery to determine the boot controller
- Obtaining the device node of the boot controller (this information is required for the disk-carving portion of the operating system setup)
- Running the operating system-specific setup script

The scripting for these steps **must** be adapted to your server deployment process. In particular, be sure to change the IP address and path of the NFS server to match your environment. You might also need to make other modifications, like adding extra configuration steps (for instance, running HPONCFG to configure iLO) or additional servers.

The sample install_rhel4.sh script is similar to the following:

```
#!/bin/bash

## SAMPLE. Change the NFS mount points to match your environment
export NFS_TOOLKIT_DIR=10.0.0.1:/TOOLKIT

## Internal Variables, do not modify
export TOOLKIT=/TOOLKIT
export NFS_MAIN=/mnt/nfs
export HWDISC_FILE=/TOOLKIT/hwdisc.dat
export SERVERNAME=
export BOOTDEVNODE=

clear
echo "*** Performing RHEL4 installation ***"

echo "Mounting NFS share"
mkdir ${NFS_MAIN}
mount -t nfs ${NFS_TOOLKIT_DIR} ${NFS_MAIN} -o ro,nolock
if [ $? != 0 ]; then
   echo "Unable to mount NFS share, make sure you updated the $0 script
   with the location of your NFS server."
   exec /bin/bash
fi

echo "Copying over toolkit scripts and utilities from NFS share"
cd ${TOOLKIT}
cp -a ${NFS_MAIN}/scripts/* ${TOOLKIT}
cp -a ${NFS_MAIN}/utilities/* ${TOOLKIT}
cp -a ${NFS_MAIN}/data_files ${TOOLKIT}
```

```
echo ""
echo "Loading storage drivers for hardware"
./load_modules.sh

echo ""
echo "Pausing to allow drivers to finish loading"
sleep 5

echo ""
echo "Configure server"

## run hardware discovery
./hwdisc3 -f${HWDISC_FILE}

## use hwquery to fetch the SystemName from hardware discovery file.
## ( extra " " are required )
export "`./hwquery ${HWDISC_FILE} allboards.xml SERVERNAME=SystemName`";

echo "Server Type: ${SERVERNAME}"

case "${SERVERNAME}" in
   "ProLiant DL380 G4" )
     # Apply System Configuration
     ./conrep -l -fdata_files/dl380g4_conrep.dat

     # Apply Array Configuration for Smart Array 6i Controller
     # if present
     ./ifhw ${HWDISC_FILE} allboards.xml "PCI:Smart Array 6i Controller"
     2> /dev/null
     if [ $? = 0 ] ; then
       cd ${TOOLKIT}/cpqacuxe
       ./cpqacuxe -i ../data_files/dl380g4_sa6i_cpqacuxe.dat
     fi

     # Apply Array Configuration for Smart Array P600 Controller if
     # present
     ./ifhw ${HWDISC_FILE} allboards.xml "PCI:Smart Array P600
     Controller" 2> /dev/null
     if [ $? = 0 ] ; then
       cd ${TOOLKIT}/cpqacuxe
       ./cpqacuxe -i ../data_files/dl380g4_p600_cpqacuxe.dat
     fi

     ## ADD EXTRA DL380 G4 Configuration Steps HERE
   ;;

   "ProLiant BL45p G1" )
```

```
      ./conrep -l -fdata_files/bl45pg1_conrep.dat

      # Apply Array Configuration for Smart Array 6i Controller if
      # present
      ./ifhw ${HWDISC_FILE} allboards.xml "PCI:Smart Array 6i Controller"
      2> /dev/null
      if [ $? = 0 ] ; then
        cd ${TOOLKIT}/cpqacuxe
        ./cpqacuxe -i ../data_files/bl45pg1_sa6i_cpqacuxe.dat
      fi

      ## ADD EXTRA BL45p G1 Configuration Steps HERE

   ;;

   "ProLiant ML310 G2" )
      ./conrep -l -fdata_files/ml310g2_conrep.dat

      ./ifhw ${HWDISC_FILE} allboards.xml "PCI:Intel(R) 6300ESB Ultra ATA
      Storage/SATA Controller"
      if [ $? = 0 ] ; then
        echo "Plain SATA found"
        # Plain SATA, set BOOTDEVNODE manually since hardware discovery
        # won't find IDE devices
        export BOOTDEVNODE=/dev/hda
      fi

      ## ADD EXTRA ProLiant ML310 G2 Configuration Steps HERE
   ;;

   ## ADD MORE SERVERS HERE

   ProLiant* )
      echo "No configuration process defined for this ProLiant server"
      echo "Update $0 with steps for this ProLiant server"
      exec /bin/bash
   ;;

   * )
      echo "Unrecognized Server"
      exec /bin/bash
   ;;
esac
```

```
## CONTINUE COMMON INSTALL PROCESS

cd ${TOOLKIT}

echo "Pausing to allow drivers to catch up"
sleep 5

echo ""
echo "Rerun hardware discovery to find boot device"
./hwdisc3 -f${HWDISC_FILE}

## use hwquery to fetch the boot dev node from hardware discovery file.
if [ -z ${BOOTDEVNODE} ]; then
   export `./hwquery ${HWDISC_FILE} allboards.xml BOOTDEVNODE=DevNode`
fi

echo "Boot Device=${BOOTDEVNODE}"

if [ -z ${BOOTDEVNODE} ]; then
   echo "MISSING boot device dev node. Check that the drivers are
   loaded."
   exec /bin/bash
fi

ls -al ${BOOTDEVNODE}*
ln -s ${BOOTDEVNODE} /dev/sssd

echo "### Linux Unattended Install using Kickstart ###"

echo "clearing mbr and a few more sectors"
dd if=/dev/zero of=/dev/sssd bs=512 count=32

echo "forcing kernel to re-read partition table"
sfdisk --re-read /dev/sssd
sleep 5

echo "landing mbr"
dd if=${NFS_MAIN}/linux_unattend/generic.mbr of=/dev/sssd bs=512 count=1

echo "create new 256M FAT16 partition using sfdisk"
echo "0,256,6,*" | sfdisk -uM -D /dev/sssd

echo "forcing kernel to re-read partition table"
sfdisk --re-read /dev/sssd
sleep 5

## make symlink for first partition, usually sda1 or c0d0p1
if test -e ${BOOTDEVNODE}1 ; then
   ln -s ${BOOTDEVNODE}1 /dev/sssd1
elif test -e ${BOOTDEVNODE}p1 ; then
   ln -s ${BOOTDEVNODE}p1 /dev/sssd1
else
   echo "Partition 1 missing, check that partition creation succeeded"
   exec /bin/bash
fi

ls -al /dev/sssd1

cd ${NFS_MAIN}/linux_unattend/rhel4/

echo "landing diskboot.img from RHEL4-disc1/images/"
dd if=diskboot.img of=/dev/sssd1

## mount disk
echo "mounting to /mnt/dos"
mount -t vfat /dev/sssd1 /mnt/dos

##### MAKE SURE YOU MODIFY syslinux-rh.cfg FOR YOUR ENVIRONMENT #######
cp -a syslinux-rh.cfg  /mnt/dos/syslinux.cfg

cd ${TOOLKIT}

## unmount disk
umount /mnt/dos
umount ${NFS_MAIN}

## unmount everything else
#umount -a

#echo "Rebooting"
#sleep 5

#/bin/reboot c:
```

Operating system preinstallation tasks

The process for modifying the install scripts varies depending on the operating system environment. The following example is specific to Red Hat Linux.

To set up a Red Hat kickstart install using the Toolkit Linux Edition:

1. Partition the drive:
    a. Copy the MBR.

       ```
       dd if=generic.mbr of=/dev/sssd
       ```

    b. Create a FAT16 partition.

       ```
       echo "0,256,6,*" | sfdisk -uM -D /dev/sssd
       ```

2. Prepare the partition using one of the following methods:
    - Copy the Red Hat boot disk installation image (diskboot.img) to the partition:

      ```
      dd if=diskboot.img of=/dev/sssd1
      ```

    - Use a bootnet.img diskette file:

    a. Format the partition:

       ```
       mkdosfs /dev/sssd1
       ```

    b. Install the bootloader:

       ```
       syslinux /dev/sssd1
       ```

c. Obtain a Red Hat bootnet.img diskette image from the Red Hat media or the HP website (http://www.hp.com), where there is a downloadable bootnet.img file for each ProLiant server that supports Red Hat Linux.

d. Mount the image on your Linux system.

e. Copy the vmlinuz and initrd.img files from the diskette image to the partition.

f. Copy a syslinux-rh.cfg file that specifies a kickstart install to the partition.

NOTE: The initrd.img from the bootnet.img file is not equivalent to the initrd.img delivered with this Toolkit.

3. Reboot to the FAT16 partition.

The following is a sample syslinux-rh.cfg file:

```
say
say Performing RedHat Kickstart Installation
say
default ks
prompt 1
timeout 6
label linux
  kernel vmlinuz
  append initrd=initrd.img lang= devfs=nomount ramdisk_size=9216
label ks
  kernel vmlinuz
  append ksdevice=eth0
  ks=nfs:10.0.0.1:/STORAGE/linux_unattend/rhel4/anaconda-ks.cfg
  initrd=initrd.img lang= devfs=nomount ramdisk_size=9216 network
```

The following line points the Red Hat install to an NFS share to obtain the kickstart file anaconda-ks.cfg:

```
append ksdevice=eth0
ks=nfs:10.0.0.1:/STORAGE/linux_unattend/rhel4/anaconda-ks.cfg
initrd=initrd.img lang= devfs=nomount ramdisk_size=9216 network
```

You must modify this line to match your deployment environment.

Most recent Red Hat installations provide a kickstart file, called anaconda-ks.cfg, in the /root directory. Modify this file, and put it on your network share for subsequent installations.

Red Hat Linux anaconda-ks.cfg sample file

The operating system-dependent unattended installation file is not created by the Toolkit utilities. The user must create the file separately. In the following example, bold lines indicate modifications made to fully automate the installation of the operating system.

Refer to the operating system documentation or the *Red Hat Linux 9: Red Hat Linux Customization Guide* (http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/custom-guide/part-install-info.html) for a complete description of the options that can be modified in the anaconda-ks.cfg unattended installation file to customize the installation of Red Hat Linux.

```
lang en_US

REM *** Modify the network settings to reflect required
REM *** network settings.

network --bootproto dhcp

REM *** The IP address should be the address of the
REM *** Linux repository server. The /SHAREVOL/RedHatCD
REM *** must be shared as an NFS volume.

nfs --server 192.1.1.3 --dir /SHAREVOL/RedHatCD

device ethernet eepro100

keyboard "us"
zerombr yes
clearpart --Linux
part /boot --size 30
part swap --size 128
part / --size 100 --grow
install

mouse genericps/2
timezone Etc/GMT-6

#xconfig --server "Mach64" --monitor "generic monitor"
skipx

rootpw iscrypted $1$ltK6jzho$7pPbE8WPNAeg44UlXqG27

auth --useshadow --enablemd5

lilo --location partition

reboot

%packages
ElectricFence
setup
filesystem
basesystem
ldconfig
glibc
shadow-utils
mkkickstart
mktemp
termcap
libtermcap
bash
MAKEDEV
SysVinit
XFree86-Mach64
ncurses
info
grep
XFree86-libs
chkconfig
XFree86-xfs
anacron
anonftp
fileutils
mailcap
textutils
```

```
apache
apmd
arpwatch
ash
at
authconfig
autoconf
automake
yp-tools
ypbind
ypserv
zlib
zlib-devel
%post
```

The previous example contains a limited list of packages to be installed. Add to this section any other packages to be installed.

The server deployment configuration and operating system installation process is complete.

Toolkit utilities

In this section

Syntax conventions

Syntax refers to the way a command and parameters must be entered. Unless specified otherwise, enter commands, parameters, and switches in all uppercase or all lowercase letters.

Sample syntax line:

```
SAMPLE [/R|-R] [PATH] FILENAME [...]
```

Command element	Meaning
`SAMPLE`	Specifies the name of the command.
`[ ]`	Indicates a component of the command line. Enter only the information within the brackets, not the brackets themselves.
`/ or -`	Indicates a command line switch for executable files.
`PATH`	Specifies the route the operating system must follow through the directory structure to locate a directory or file. A path and file name must be specified only if the file is not in the current directory.
`FILENAME`	Specifies a file name. This document uses uppercase file names. A device name or a drive letter cannot be specified for a file name.
`...`	Indicates that the previous parameter or switch can be repeated several times in a command. Enter only the information, not the ellipsis (...) itself.

In this document, the length of an example command or syntax might require it to continue on another line. When this happens, the second line (and any additional lines) is indented under the first line.

Placeholder items used in the syntax lines in this chapter include:

- Source—Specifies the location of the data to be transferred to a specified destination or used as input to a command. The source can consist of a drive letter and colon, a directory name, a file name, or a combination of these items.
- Destination—Specifies the location to which the data specified by the source is to be transferred. The destination can consist of a drive letter and colon, a directory name, a file name, or a combination of these items.
- String—Specifies a group of characters to be treated as a unit. A string can include letters, numbers, spaces, or any other characters and is usually enclosed in double quotation marks.

Utility online help

Most Toolkit utilities include usage instructions. To obtain help with the syntax, parameters, and switches of a particular Toolkit utility, enter the file name followed by `/?` in the command line. For example, for usage instructions on the CONREP utility, enter the following command:

```
CONREP /?
```

The utility displays information about its command line syntax, argument, and switches.

Using system utilities

The system utilities control steps in the installation process. System utilities in the Toolkit include:

- REBOOT
- STATEMGR
- HWDISCOVERY
- IFHW
- HWQUERY

Using REBOOT

REBOOT is used from a batch file, in conjunction with other utilities, to control server reboots. This utility enables the user to reboot the server with control over which device is the boot device. If no boot drive argument is passed on to REBOOT, the tool reboots the server using whichever drive is specified as the default drive.

REBOOT command line syntax

```
REBOOT [DRIVE:] [/?]
```

REBOOT command line arguments

Command line argument	Description
`[DRIVE:]`	Valid arguments that can be passed on to REBOOT are A:, C:, CD, PXE, or no argument. By specifying an argument, the drive indicated is set to boot on the next reboot, and the system is restarted. If no argument is provided, the system is set to boot using the defined boot order, and the system is restarted.
`/?`	This argument displays help information.

REBOOT return codes

There are no return codes for the REBOOT utility.

REBOOT command line examples

Command line input	Result
`REBOOT A:`	This command reboots the system to the A: drive.
`REBOOT`	This command reboots the system to the drive that is defined as the default boot drive in the system configuration.
`REBOOT PXE`	This command reboots the system to the PXE NIC.

Using STATEMGR

STATEMGR enables the user to keep track of the execution state during system reboots. This utility saves persistent state information across reboots of the system.

STATEMGR command line syntax

```
STATEMGR [/R | -R] [EVNAME] [/?]
```

- or -

```
STATEMGR [/W | -W] [EVNAME] [VALUE] [/?]
```

STATEMGR command line arguments

Command line input	**Result**
`/R or -R`	This argument reads the state of the environment variable defined by [EVNAME]. The value of the environment variable is returned as a return code.
`/W or -W`	This argument writes the state defined by [VALUE] to an environment variable defined by [EVNAME].
`EVNAME`	This argument creates an environment variable used to represent the state to manage. The variable can be any word that is eight characters or less.
`VALUE`	This argument is used only with the /W or -W arguments to indicate the value of the environment variable to maintain. [VALUE] is limited to integers between 0 and 254. If no value is provided when using /W or -W, the state environment variable is cleared.
`/?`	This argument displays help information.

STATEMGR return codes

Value	**Meaning**
`0`	The command was completed successfully.
n	*N* arguments were ignored because they were not in the *variable=<string>* format.

STATEMGR command line examples

Command line input	**Result**
`STATEMGR /W PHASE 3`	STATEMGR writes the state value 3 to the PHASE environment variable.
`STATEMGR /R PHASE`	STATEMGR reads the PHASE environment variable and returns its value as a return code. If the environment variable has been reset or no value has been stored, the return code is 0.

Using HWDISC

HWDISC provides an inventory of the server being configured. It must be run on each server deployed. It is executed by the server configuration script and captures the following information:

- System ID
- System name
- ROM information

- Processor information
- PCI devices present in the system
- Storage configuration information
- NIC information

User process decisions can be made based on data that is in the file created by this utility.

HWDISC command line syntax

```
hwdisc3 [-ffilename] [/?]
```

HWDISC command line arguments

Command line input	Result
`-ffilename`	This argument specifies the location and name of the Hardware Discovery data file. If no file name is specified, the utility generates a file in the current directory using the default name hwdisc-3.dat.
`/?`	This argument displays help information.

HWDISC return codes

Value	Meaning
`0`	The command was completed successfully. A usage message might be displayed.
`1`	The command contained an invalid parameter.

HWDISC command line examples

Command line input	Result
`hwdisc3 -f/toolkit/hwdisc.dat`	This command generates the file hwdisc.dat in the /toolkit directory.
`hwdisc3`	This command generates the file hwdisc-3.dat in the current directory.

Using IFHW

IFHW is used from a script file, in conjunction with other utilities, to control the deployment. The IFHW utility enables you to make intelligent queries against the hardware discovery file. Queries take the form of a logical expression, and the result of the expression is returned as the return code of the tool, which the hosting script can use to conditionally perform actions.

IFHW command line syntax

```
ifhw [path]hwdiscfilename [path]allboards.xml <expression>
```

IFHW command line arguments

Command line argument	Description
`[path]hwdiscfilename`	This argument specifies the hardware discovery file used to run the query.

Command line argument	Description
`[path]allboards.xml`	This argument specifies the allboards.xml PCI device list file, which is used to convert PCI IDs found in hardware discovery into device names, such as "Smart Array 5i Controller."
`<expression>`	This argument specifies the query expression. Refer to "Expression operators and terms (on page 24)."

IFHW return codes

Value	Meaning
`0`	The expression is true.
`1`	The expression is false.
`2`	The expression was not understood or an argument was invalid.

IFHW command line examples

Command line input	Result
`ifhw hwdisc.xml allboards.xml "PCI:Smart Array 5i"`	This command returns the following error levels: • ERRORLEVEL 0 (True) if the Smart Array 5i is present • ERRORLEVEL 1 (False) if the device is not present • ERRORLEVEL 2 (Error) if the expression could not be understood

Expression operators and terms

Operator or term	Result
`and`	True if both operands are true
`or`	True if either operand is true
`gt`	True if the first operand is greater than the second
`lt`	True if the first operand is less than the second
`gte`	True if the first operand is greater than or equal to the second
`lte`	True if the first operand is less than or equal to the second
`eq`	True if the two operands are equal
`neq`	True if the two operands are not equal
`not`	True if the operand is false
`PCI:`<*string*>	True if a PCI device whose name includes <*string*> is found in the hardware discovery file. <*string*> is case-sensitive.
`HWQ:`<*string*>	The hardware discovery file is searched for <*string*>, and the corresponding value is the value of this term. <*string*> is case-sensitive.
<*string*>	A literal string, used for comparison
<*number*>	A literal number, used for comparison

Expression examples

Expression input	**Result**
`"PCI:Smart Array 5i"`	True if the Smart Array 5i Controller is found in the system
`HWQ:RAM gte 512`	True if the amount of RAM in the hardware discovery file is at least 512
`HWQ:ROMdate neq "11/12/2004"`	True if the ROM date in the hardware discovery file is not 11/12/2004
`HWQ:SystemName eq "ProLiant DL380 G2"`	True if the system name in the hardware discovery file exactly matches "ProLiant DL380 G2"
`HWQ:SystemName eq "ProLiant DL380 G2" and "PCI:Smart Array 5i" and HWQ:ROMdate eq "11/12/2004"`	True if the system is a ProLiant DL380 G2 with a Smart Array 5i Controller present and a ROM date of 11/12/2004
`"PCI:Smart Array 5i" or "PCI:Smart Array 6i"`	True if the system contains a Smart Array 5i Controller or a Smart Array 6i Controller

Using HWQUERY

HWQUERY is used from a script, in conjunction with other utilities, to control the deployment. The HWQUERY utility enables you to use data from the hardware discovery file in your own scripts. HWQUERY cannot alter environment variables directly. To set the variable, the output of HWQUERY must be used by the hosting script. The most common way to use it is to write the output to an intermediate script that is subsequently called by the hosting script.

HWQUERY command line syntax

```
hwquery [path]hwdiscfilename [path]allboards.xml variable=<string> ...
```

HWQUERY command line arguments

Command line argument	**Description**
`[path]hwdiscfilename`	This argument specifies the hardware discovery file used to run the query.
`[path]allboards.xml`	This argument specifies the allboards.xml PCI device list file, which is used to convert PCI IDs found in hardware discovery into device names, such as "Smart Array 5i Controller."
`variable=<string>`	In this argument, *variable* is the name of an environment variable and *<string>* is a PCI device name or the name of an element from the hardware discovery file. Arguments must be in quotes if *<string>* contains spaces. *<string>* is case-sensitive.
`...`	You can specify multiple *variable=<string>* arguments.

HWQUERY return codes

Value	**Meaning**
0	The command was completed successfully
n	*N* arguments were ignored because they were not in the *variable=<string>* format.

HWQUERY command line examples

Command line input	Result
`hwquery hwdisc.xml allboards.xml MY_SYS_RAM=RAM`	For a hwdisc.xml file that contains <RAM>768</RAM>, HWQUERY produces the following: `MY_SYS_RAM=768`
`hwquery hwdisc.xml allboards.xml "TEST=Smart Array"`	For a hwdisc.xml file that indicates a Smart Array 5i Controller is present, HWQUERY produces the following: `TEST=Smart Array 5i Controller`

`hwquery hwdisc.xml allboards.xml MYRAM=RAM MYROMDATE=ROMdate`	For a hwdisc.xml file that contains <RAM>768</RAM> and <ROMdate>11/15/2002</ROMdate>, HWQUERY produces the following:
	`MYRAM=768`
	`MYROMDATE=11/15/2002`

`hwquery hwdisc.xml allboards.xml "TEST=smart array 5i"`	Although the controller is present, HWQUERY produces the following: `TEST=` This behavior is correct. The string is case-sensitive, and the argument uses lowercase lettering instead of the title case found in the allboards.xml file.

Using configuration utilities

The configuration utilities read the source server configuration and duplicate that configuration on a target server through a generated script file. The configuration utilities include:

- CONREP
- CPQACU
- HPONCFG

Using CONREP

CONREP generates a hardware configuration script file used to duplicate the hardware configuration of one ProLiant server onto another.

△ **CAUTION:** Improper modification of the CONREP data files can result in the loss of critical data. Only experienced users of the Toolkit should attempt to modify the data files. Because of the potential risk of data loss, take all necessary precautions to ensure that mission-critical systems remain online if a failure occurs.

CONREP reads the state of the system environment settings to determine the configuration of the server and writes the results to a text file that can be edited by the user. The utility then uses the data in the generated script file to configure the hardware of the target server.

CONREP uses an XML definition file to determine what information to retrieve from and restore to the server. This file can be easily modified to update new features or restrict features when capturing configurations.

IMPORTANT: The file format for the DOS version of CONREP and the current version of CONREP are not compatible.

CONREP command line syntax

```
conrep [-s | -l] [-xfilename] [-ffilename] [-?]
```

CONREP command line arguments

Command line argument	**Description**
`-s`	This argument saves the hardware configuration to a file.
`-l`	This argument loads the hardware configuration from a file and writes it to the target server.
`-xfilename`	This argument defines the name and location of the XML definition file. The default is conrep.xml.
`-ffilename`	This argument defines the name and location of the data file. The default is conrep.dat.
`-?`	This argument displays help information.

CONREP return codes

Value	**Meaning**
0	The command was completed successfully.
1	The data file (systemdata.dat) is bad.
2	The XML definition file (conrep.xml) is bad.

CONREP command file contents

A typical data file generated by CONREP is similar to the following:

```
<Conrep_data>
   <Section name="OS">06 05</Section>
   <Section name="Custom_Post_Message">
      <Line0>Added by Conrep</Line0>
   </Section>
   <Section name="IMD_ServerName">
      <Line0>SSTK test</Line0>
   </Section>
   <Section name="IPL_Order">
      <Index0>00 </Index0>
      <Index1>01 </Index1>
      <Index2>02 </Index2>
      <Index3>03 </Index3>
      <Index4>ff </Index4>
      <Index5>ff </Index5>
      <Index6>ff </Index6>
      <Index7>ff </Index7>
   </Section>
   <Section name="PCI_Devices">
      <Index0>05 </Index0>
      <INT0>01 </INT0>
      <IRQ0>03 </IRQ0>
      <Reserved0>00 </Reserved0>
      <Id0>11 0e 78 b1 </Id0>
      <Index1>04 </Index1>
      <INT1>01 </INT1>
      <IRQ1>07 </IRQ1>
      <Reserved1>00 </Reserved1>
      <Id1>86 80 29 12 </Id1>
      <Index2>06 </Index2>
      <INT2>01 </INT2>
      <IRQ2>0a </IRQ2>
      <Reserved2>00 </Reserved2>
      <Id2>11 0e f0 a0 </Id2>
      <Index3>08 </Index3>
      <INT3>01 </INT3>
      <IRQ3>0b </IRQ3>
      <Reserved3>00 </Reserved3>
      <Id3>66 11 20 02 </Id3>
      <Index4>07 </Index4>
      <INT4>01 </INT4>
      <IRQ4>0f </IRQ4>
      <Reserved4>00 </Reserved4>
      <Id4>11 0e f7 a0 </Id4>
   </Section>
   <Section name="Controller_Order">
      <Id0>0e 11 40 80 </Id0>
      <Slot0>00 </Slot0>
      <BusDev0>00 08 </BusDev0>
      <Rest0>01 </Rest0>
      <Id1>0e 11 ff ff </Id1>
      <Slot1>00 </Slot1>
      <BusDev1>00 78 </BusDev1>
      <Rest1>c1 </Rest1>
   /Section>
   <Section name="Language">ENGUSAus  </Section>
   <Section name="System_WOL">Disabled</Section>
   <Section name="System_APIC">Auto Set</Section>
   <Section name="System_COMA">COM1</Section>
   <Section name="System_COMA_IRQ">IRQ4</Section>
   <Section name="System_COMB">Disabled</Section>
   <Section name="System_COMB_IRQ">Undefined</Section>
```

```
    <Section name="System_LPT">LPT1</Section>
    <Section name="System_LPT_IRQ">IRQ7</Section>
    <Section name="Diskette_Write_Control">Writes_Enabled</Section>
    <Section name="NMI_Debug_Button">Disabled</Section>
    <Section name="ACPI_Power_Button">Disabled</Section>
    <Section name="ASR">Disabled</Section>
    <Section name="ASR_Timeout">10 Minutes</Section>
    <Section name="Thermal_Shutdown">Enabled</Section>
    <Section name="RBSU_Language">01</Section>
    <Section name="PXE_NIC1">Disabled</Section>
    <Section name="PXE_NIC2">Disabled</Section>
    <Section name="BIOS_Console">Disabled</Section>
    <Section name="EMS_Console">Disabled</Section>
    <Section name="Diskette_Boot">Enabled</Section>
    <Section name="NumLock">On</Section>
    <Section name="POST_Speed_Up">Enabled</Section>
    <Section name="Integrated_Diskette_Controller">Enabled</Section>
    <Section name="PCI_Bus_Reset">Enabled</Section>
    <Section name="Hot_Plug_Reservation">Auto Set</Section>
    <Section name="Memory_Protection">Standard ECC Protection</Section>
</Conrep_data>
```

CONREP operating system definition values

Operating system	Major	Minor
NetWare 3.10	0	1
NetWare 3.11	0	2
NetWare 4.1x and 4.2	0	3
NetWare 5.0, 5.1, 5.2, 6.0, and 6.5	0	5
NetWare SMP 4.1x and 4.2	0	4
SCO UNIX®	1	1
SCO 5	1	2
Solaris	1	3
OS/2 2.x, Warp, Warp Connect, Warp Server	2	6
OS/2 2.11 SMP, Warp Server Advanced 4.0 with SMP	2	7
Banyan Vines	3	0
Banyan 8.0	3	2
Windows NT® 3.51	5	4
Windows NT® 4.0	5	5
Windows® 2000	5	6
Windows Server™ 2003	5	6
UnixWare 2.0	6	1

Operating system	Major	Minor
UnixWare 2.1	6	2
UnixWare 7.0	6	3
Linux	6	4
Other	FF	0

Using CPQACUXE

CPQACUXE enables you to configure array controllers on a target server. CPQACUXE reads the configuration information from a data file and applies the configuration to the controllers in the target server. CPQACUXE enables the array configuration existing on one ProLiant ML, DL, or BL server to be replicated on other servers with similar storage configurations.

CPQACUXE has two modes of operation:

- In **Capture** mode, the configurations of all internal and external array controllers connected to a server are saved to a data file. You can then use CPQACUXE and the data file to replicate the array configuration on other servers that have similar storage resources.
- In **Input** mode, the array configuration that is specified in a data file is applied to a target system. The data file can be an unmodified or modified capture file, or you can write the data file from scratch.

CPQACUXE command line syntax

- Capture mode:

```
cpqacuxe -c [path]filename [-?]
```

- Input mode:

```
cpqacuxe -i [path]filename [-?]
```

CPQACUXE command line arguments

Command line argument	Description
`-c [path]filename`	This argument is used on source servers to capture the existing array controller configurations and to write the configurations to the file specified by `[path]filename`. If no file name is specified, the utility generates a file in the current directory using the default name acucapt.ini.
`-i [path]filename`	This argument is used on the target server to specify the input file name. The file name is the data file used by the utility to configure the array controllers. If no file name is specified, the utility generates a file in the current directory using the default name acuinput.ini.
-?	This argument displays help information.

CPQACUXE return codes

If CPQACUXE encounters an error, the error is logged to the error.ini file.

Value	Meaning
0	The command was completed successfully.
1	The command failed. The user is not authenticated to use ACU, or ACU is already running.

CPQACUXE command file contents

A typical array configuration script file generated by CPQACUXE displays a script similar to the following:

NOTE: An asterisk next to a line indicates that the line is not required in Automatic mode.

```
  ; Control Options
  Action = Configure
  Method = Custom

  ; Controller Options
  ; Controller Compaq Smart Array 5300
  Controller = Slot 5
  ClearConfigurationWithDataLoss = No
  LicenseKey = XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
  DeleteLicenseKey = XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
  RAIDArrayID = "XXXXXXXXXXXXXXXXXXXX"
  ReadCache = 50
  WriteCache = 50
  RebuildPriority = Low
  ExpandPriority = Low
  SurfaceScanDelay = N
* SSPState = Disable

  ; Array Options
* Array = A
  OnlineSpare = None
* Drive = 2:0,2:1

  ; Logical Drive Options
* LogicalDrive = 1
  RAID = 1
* Size = 17359
* Sectors = 32
* StripeSize = 256
* ArrayAccelerator = Enable
* ResourceVolumeOwner = N
* LogicalDriveSSPState = Disable
* SSPAdaptersWithAccess = None

  ; HBA SSP Specifications
* HBA_WW_ID = XXXXXXXXXXXXXXXX
* ConnectionName = TestConn
* HostMode = Windows
```

The data file used by CPQACUXE is a text file that contains options and parameters required to configure HP array controllers. The CPQACUXE utility parses the data file in a case-insensitive manner.

Lines of the data file can be blank lines or lines in the form `option = value`. Semicolons are used for comments within data files, and CPQACUXE ignores everything after a semicolon up to the next line.

The following options are valid in data files generated and read by CPQACUXE:

- **Control options** define the overall behavior of CPQACUXE when it processes the scripts and creates configurations. Control options can occur only once in a data file and must be the first options listed.
- **Controller options** define the controller that is to be configured (or the controller that has had its configuration captured). The Controller option must be placed at the beginning of this section in the

data file, but other options in this category can be scripted in any order. One data file can be used to configure several controllers if all controllers are to be configured identically or if each controller is defined separately. When defining each controller configuration separately, all other category options for a defined controller must be entered before starting a new controller listing.

- **Array options** define an array that is to be configured on the controller that is identified previously in the data file. If no controller is previously identified, CPQACUXE sends an error message. The Array option must be at the beginning of this section in the data file, but other options in this category can be scripted in any order.
- **Logical drive options** define a logical drive that is to be configured on an array that is defined previously in the data file. If no array is previously defined, CPQACUXE sends an error message. The LogicalDrive option must be placed at the beginning of this section in the data file, but other options in this category can be scripted in any order.
- **HBA options** define an HBA SSP configuration for a logical drive that is previously defined in the data file. If no logical drive is previously defined, CPQACUXE sends an error message. The HBA_WW_ID option must be at the beginning of this section in the data file, but other options in this category can be scripted in any order.

Control options

The following table describes the control options used to define the overall behavior of CPQACUXE when it processes the scripts and creates the configuration. Each option can have only one of the listed values.

Option	**Value**
ACTION (required) This option defines the configuration action performed.	• CONFIGURE In Configure mode, you can only create new arrays; you cannot modify any existing arrays. The controller must have unassigned physical drives for this mode to be available. • RECONFIGURE In Reconfigure mode, you can use CPQACUXE to modify existing arrays. This procedure does not destroy data unless you specifically want the data to be deleted. In this mode, CPQACUXE does not change an existing option setting unless you specifically script a different value for that option.
METHOD This option defines the configuration method by which the action is performed.	• AUTO (default) CPQACUXE can perform an expansion, extension, or migration without user intervention, depending on the settings that you use for other options. • CUSTOM CPQACUXE uses only the criteria in the input file for the configuration. Default values are used where required.

Controller options

The following table describes the controller options used to define a controller or set of controllers used in the configuration. Each option can have only one of the listed values.

Option	**Value**
CONTROLLER (required) This option identifies the controller that is to be configured.	• ALL Configure all detected controllers in the system identically. • SLOT [N] Configure the internal controller with slot number N. • WWN [N] Configure the external controller with WWN N. • SERIAL NUMBER [N] Configure the shared storage controller with serial number N. • IOCABINET [N],IOBAY [N],IOCHASSIS [N],SLOT [N],CABINET [N],CELL [N] Configure the controller identified by the slot path information.
ClearConfigurationWithDataLoss This option specifies whether to clear the configuration.	• NO (default) The configuration will not be cleared. • YES The configuration will be cleared. Clearing the configuration causes data loss because it deletes all logical drives on the controller. If you clear a configuration, you can write commands later in the script file to create a new configuration from the liberated drive capacity.
LicenseKey This option enables you to enter a license key that is required to activate some controller features.	XXXXX-XXXXX-XXXXX-XXXXX-XXXXX Hyphens can be entered but are not required.
DeleteLicenseKey This option enables you to uninstall an existing controller feature by entering the license key for the feature.	XXXXX-XXXXX-XXXXX-XXXXX-XXXXX Hyphens can be entered but are not required.
RAIDArrayID This option is the user-defined character string that identifies the controller.	"XXXXXXXXXXXXXXXXXXXX" Any of the following characters can be used in the string: a–z A–Z 0–9 ! @ # * ( ) , - _ + : . / [space]
READCACHE This option specifies the percentage of the controller cache reserved for the read-ahead cache.	0, 10, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100
WRITECACHE This option specifies the percentage of the controller cache reserved for the posted-write cache.	0, 10, 25, 30, 40, 50, 60, 70, 75, 80, 90, 100
REBUILDPRIORITY This option specifies the priority to be assigned for logical drive rebuilding.	• LOW • MEDIUM • HIGH
EXPANDPRIORITY This option specifies the priority to be assigned for logical drive expansion.	• LOW • MEDIUM • HIGH

Option	Value
SurfaceScanDelay This option specifies the duration of the surface scan delay in seconds.	1, 2, …, 30
SSPState * This option specifies the SSP state for controllers that support SSP.	• DISABLE Disable SSP for the controller. • ENABLE Enable SSP for the controller. If you enable SSP, you must also specify an adapter for one or more logical drives by using the SSPAdaptersWithAccess command. Otherwise, SSP is automatically disabled.

* Currently, this option applies only to shared-storage controllers, such as the HP StorageWorks Modular Smart Array 1000 (MSA1000) and Smart Array Cluster Storage. The SSPState option is valid only for controllers that enable SSP on a controller basis. RA4x00 controllers enable SSP on a logical drive basis and use the LogicalDriveSSPState command instead.

Array options

The following table describes the array options used to specify a particular array in the configuration. Each option, except the DRIVE option, can have only one of the listed values.

Option	Value
ARRAY (required) This option specifies the array that is being created or reconfigured.	ARRAYLETTER This is a single letter (A–Z or a–f) used to specify the array ID. • In Configure mode, a new array is created. The array letter specified must be the next available array letter in the existing configuration. • In Reconfigure mode, the array letter can identify an existing array, or it can identify the next available array letter in the existing configuration to create a new array.
DRIVE This option specifies the physical drives used for the array. This option is required in Custom mode.	• [X]:[Y],… These values specify Port:Id for controllers that use Port/ID drive numbering schemes or Box:Bay for controllers that use Box/Bay numbering schemes. • [X]:[Y]:[Z],… These values specify Port:Box:Bay for SAS controllers. In Configure mode, the physical drives listed are used to create the new array. In Reconfigure mode, any extra physical drives that you add to the list are used to expand the array, as long as the capacity of the added drives is at least as great as that of existing drives in the array. You cannot remove drives from the array unless the ClearConfigurationWithDataLoss option is set to Yes. In Automatic mode, all available drives are used.

Option	Value
ONLINESPARE This option specifies the online spare used with the array.	In Automatic mode, the following values are valid: • YES The utility will attempt to add spares to each array. • NO The utility will not add spares to each array. In Configure mode, the default value is YES. In Reconfigure mode, CPQACUXE ignores this option and keeps any spares that the existing configuration already has. In Custom mode, you can specify which drives are to be used as spares. • [X]:[Y],... These values specify Port:Id for controllers that use Port/ID drive numbering schemes or Box:Bay for controllers that use Box/Bay numbering schemes. • [X]:[Y]:[Z],... These values specify Port:Box:Bay for SAS controllers. • NONE No spares are added to the array, and any existing spares are removed from the array. In Configure mode, the default value is None. In Reconfigure mode, any existing spares in the array are kept if you do not specify a value for the OnlineSpare option.

Logical drive options

Option	Value
LOGICALDRIVE (required) This option specifies the logical drive number to be configured or reconfigured.	[N] This is a numeric value from 1 to 32. • In Configure mode, you can enter only the ID number of the next possible logical drive in the sequence for the existing configuration. • In Reconfigure mode, you can also enter the ID number of an existing logical drive.
RAID This option specifies the RAID level for this logical drive.	0, 1, 4, 5, ADG • In Configure mode, the default setting is the highest RAID level that the configuration can support. • In Reconfigure mode, the default setting is the existing RAID level for that logical drive. If you specify a different RAID setting, then CPQACUXE either ignores the new setting (in Automatic mode) or attempts to migrate the logical drive to the specified RAID level (in Custom mode).

Option	**Value**
SIZE This option specifies the size of the logical volume in megabytes.	• [N] Specifies the size of the logical drive in megabytes. • MAX (default) Specifies that all the remaining space on the array must be allocated to this logical drive. In Reconfigure mode, the default setting is the existing size of the logical drive. If you enter a larger value, CPQACUXE extends the logical drive to the new size if there is unused drive capacity on the same array, as long as the operating system supports logical drive extension. You cannot reduce the size of the logical drive.
SECTORS (required) This option specifies the Max Boot setting (the number of sectors per track) to be used for this logical volume.	32, 63 Enter `32` to disable Max Boot. Enter `63` to enable Max Boot. • For new logical drives, the default setting is 32. • For an existing logical drive, the default setting is the existing setting. Logical drive performance is likely to decrease with Max Boot enabled.
STRIPESIZE This option specifies the stripe size of the logical drive in kilobytes. If the stripe size is not specified, the default based on the RAID level is chosen automatically.	8, 16, 32, 64, 128, 256 • RAID 0 and RAID 1 drives can use any of the listed stripe sizes. • RAID 4, RAID 5, and RAID ADG drives are limited to 8, 16, 32, or 64.
ARRAYACCELERATOR This option enables the array accelerator for this logical drive.	• ENABLE (default) Enables the array accelerator for this logical drive. • DISABLE Disables the array accelerator for this logical drive.
RESOURCEVOLUMEOWNER This option specifies the logical drive as the owner of a resource volume.	N This value is the logical drive ID of an existing logical drive that owns the resource volume.
LOGICALDRIVESSPSTATE This option is valid only for controllers that enable SSP on a logical drive basis. For other controllers that support SSP, see the `SSPState` command.	• ENABLE Enables SSP for the logical drive. • DISABLE Disables SSP for the logical drive. For existing logical drives, the default setting is the current logical drive setting. For new logical drives, the default setting is Disable.

Option	Value
SSPADAPTERSWITHACCESS This option identifies the SSP adapters that have access to a logical drive.	• [N],[N]... These values specify a list of SSP adapter IDs that are to be given access to the logical drive • NONE No SSP adapters will have access to the logical drive. This command is processed only if either SSPState or LogicalDriveSSPState is set to Enable. Otherwise, this command is ignored.

HBA options

Option	Value
HBA_WW_ID This option specifies which HBA to configure.	WWN [N] Configure the controller with WWN [N].
ConnectionName This option specifies a user-defined string as the name for the specified controller.	"XXXXXXXXXXXXXXX" Any of the following characters can be used in the string: a–z A–Z 0–9 ! @ # * ( ) , - _ + : . / [space]
HostMode This option specifies the host mode for the selected HBA.	Setting the host mode optimizes the storage array for the selected operating system. Valid values include: • Default • Windows • Windows (degrade) • OpenVMS • Tru64 • Linux • Solaris • Netware • HP • Windows_SP2 Host modes are device specific. Not all modes are available on all devices and not all HBAs support a host mode.

CPQACUXE overview input file

The following text displays an overview input file describing all options for configuring one or more array controllers. The overview provides valid options and their values. Required and default values are in **bold** type. Options with no default value will not be changed if they are not specified.

```
;Control Options
Action = Configure|Reconfigure
Method = Custom|Auto

; Controller Options
; There can be multiple controller specifications in the file.
Controller = All|Slot [N]|WWN [N]|SerialNumber [N]|IOCabinet [N],
  IOBay [N],IOChassis [N],Slot [N],Cabinet [N],Cell [N]
```

```
ClearConfigurationWithDataLoss = Yes|No
LicenseKey = XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
DeleteLicenseKey = XXXXX-XXXXX-XXXXX-XXXXX-XXXXX
RAIDArrayID = "XXXXXXXXXXXXXXXXXXXX"
ReadCache = 0|10|20|25|30|40|50|60|70|75|80|90|100
WriteCache = 0|10|20|25|30|40|50|60|70|75|80|90|100
RebuildPriority = Low|Medium|High
ExpandPriority = Low|Medium|High
SurfaceScanDelay = N
SSPState = Enable|Disable

; Array Options
; There can be multiple array specifications in the file.
Array = A|B|C|D|E|F|G|…Z|a|b|c|d|e|f
OnlineSpare = Port:ID,... | Box:Bay,... | Port:Box:Bay |None
Drive = Port:ID,... | Box:Bay,... | Port:Box:Bay,...

; Logical Drive Options
; There can be multiple logical drive specifications in the file.
LogicalDrive = 1|2|3|...32
RAID = 0|1|4|5|ADG
Size = [N]|Max
Sectors = 32|63
StripeSize = 8|16|32|64|128|256
ArrayAccelerator = Enable|Disable
ResourceVolumeOwner = N
LogicalDriveSSPState = Enable|Disable
SSPAdaptersWithAccess = [N],[N]...|None

; HBA Options
; There can be multiple HBA specifications in the file.
HBA_WW_ID = WWN
ConnectionName = UserDefinedName
HostMode = Default|Windows|Windows(degrade)|OpenVMS|Tru64|Linux|
  Solaris|Netware|HP|Windows_SP2
```

Using HPONCFG

HP offers support for the RILOE II, iLO, and iLO 2 features available on ProLiant servers with the HPONCFG utility.

HPONCFG is an online configuration tool used to set up and reconfigure RILOE II, iLO, and iLO 2 without requiring a reboot of the server operating system. The utility runs in a command line mode and must be executed from an operating system command line.

HPONCFG enables you to initially configure features exposed through the RBSU or the RILOE II, iLO, or iLO 2 GUI. This utility is not intended for continued administration. CPQLOCFG should be used for ongoing administration of user rights and network functionality on the server.

Observe the following requirements before using HPONCFG:

- The RILOE II, iLO, or iLO 2 Management Interface Driver must be loaded on the server. HPONCFG displays a warning if the driver is not installed.
- HPONCFG requires minimum RILOE II, iLO, and iLO 2 firmware versions. To determine the minimum firmware version required, see the *HP SmartStart Scripting Toolkit Linux and Windows Editions Support Matrix.*

For more information, see the Remote Management website (http://www.hp.com/servers/lights-out).

HPONCFG command line syntax

```
hponcfg [-help] [-?] [-reset] [-f filename] [-l filename]
   [-w filename] [-get_hostinfo] [-m firmwarelevel]
```

IMPORTANT: Because the -w argument does not capture certain types of information, such as the administrator password, data files created with HPONCFG using the -w argument cannot then be used as input files for HPONCFG, unless they are modified first.

HPONCFG command line arguments

Command line argument	Description
`-help` or `-?`	These arguments display simple help messages.
`-reset`	This argument resets the RILOE II, iLO, or iLO 2 to factory defaults.
`-f` *filename*	This argument sets the RILOE II, iLO, or iLO 2 configuration based on the information in the XML input file named *filename*.
`-l` *filename*	This argument logs replies to the text log file named *filename*.
`-w` *filename*	This argument writes the RILOE II, iLO, or iLO 2 configuration obtained from the device to the XML output file named *filename*.
`-get_hostinfo`	This argument returns the host server name and serial number.
`-m`	This argument indicates to HPONCFG the minimum firmware level that must be present in the management device to execute the RIBCL script. If the minimum level is not met, HPONCFG returns an error without performing any additional actions.

HPONCFG return codes

Value	Meaning
0	Either the script was sent successfully to the device, or there is no RILOE II, iLO, or iLO 2 present in the system.
1	The script could not be sent to the device.

If the script itself fails, errors are reported in the log file created by HPONCFG.

HPONCFG command file contents

HPONCFG can be used to perform the following tasks:

- Obtain an entire configuration
- Obtain a specific configuration
- Set a configuration

Obtaining an entire configuration

HPONCFG can be used to obtain an entire configuration from an iLO, iLO2, or a RILOE II. In this case, the utility executes from the command line without specification of an input file. The name of the output file is given on the command line. For example:

```
HPONCFG /w config.xml
```

In this example, the utility indicated that it obtained the data successfully and wrote it to the output file as requested. The following is a typical example of the contents of the output file:

```
<HPONCFG VERSION = "1.1">
<!--- Generated 04/15/04 15:20:36 --->
```

```
<MOD_DIR_CONFIG>
<DIR_AUTHENTICATION_ENABLED VALUE = "N"/>
<DIR_LOCAL_USER_ACCT VALUE = "Y"/>
<DIR_SERVER_ADDRESS VALUE = ""/>
<DIR_SERVER_PORT VALUE = "25"/>
<DIR_OBJECT_DN VALUE = " "/>
<DIR_OBJECT_PASSWORD VALUE = ""/>
<DIR_USER_CONTEXT_1 VALUE = ""/>
<DIR_USER_CONTEXT_2 VALUE = "_"/>
<DIR_USER_CONTEXT_3 VALUE = ""/>
</MOD_DIR_CONFIG>
<MOD_NETWORK_SETTINGS>
<SPEED_AUTOSELECT VALUE = "Y"/>
<NIC_SPEED VALUE = "100"/>
<FULL_DUPLEX VALUE = "Y"/>
<IP_ADDRESS VALUE = "16.100.241.229"/>
<SUBNET_MASK VALUE = "255.255.252.0"/>
<GATEWAY_IP_ADDRESS VALUE = "16.100.240.1"/>
<DNS_NAME VALUE = "ILOD234KJ44D002"/>
<PRIM_DNS_SERVER value = "16.81.3.242"/>
<DHCP_ENABLE VALUE = "Y"/>
<DOMAIN_NAME VALUE = "americas.cpqcorp.net"/>
<DHCP_GATEWAY VALUE = "Y"/>
<DHCP_DNS_SERVER VALUE = "Y"/>
<DHCP_STATIC_ROUTE VALUE = "Y"/>
<DHCP_WINS_SERVER VALUE = "Y"/>
<REG_WINS_SERVER VALUE = "Y"/>
<PRIM_WINS_SERVER value = "16.81.3.247"/>
<STATIC_ROUTE_1 DEST = "0.0.0.0" GATEWAY = "0.0.0.0"/>
<STATIC_ROUTE_2 DEST = "0.0.0.0" GATEWAY = "0.0.0.0"/>
<STATIC_ROUTE_3 DEST = "0.0.0.0" GATEWAY = "0.0.0.0"/>
</MOD_NETWORK_SETTINGS>
<ADD_USER
USER_NAME = "Administrator"
USER_LOGIN = "Administrator"
PASSWORD = "">
</ADD_USER>
<ADD_USER
USER_NAME = "Landy9"
USER_LOGIN = "mandy9"
PASSWORD = "">
</ADD_USER>
<RESET_RIB VALUE = "Y"/>
</HPONCFG>
```

For security reasons, the user passwords are not returned.

Obtaining a specific configuration

A specific configuration can be obtained using the appropriate XML input file. For example, here are the contents of a typical XML input file, `get_global.xml`:

```
<!-- Sample file for Get Global command -->
<RIBCL VERSION="2.0">
<LOGIN USER_LOGIN="x" PASSWORD="x">
<RIB_INFO MODE="read">
<GET_GLOBAL_SETTINGS />
```

```
</RIB_INFO>
</LOGIN>
</RIBCL>
```

The XML commands are read from the input file `get_global.xml` and are processed by the device:

```
HPONCFG /f get_global.xml /l log.txt > output.txt
```

The requested information is returned in the log file, which, in this example, is named `log.txt`. The contents of the log file are shown below.

```
<GET_GLOBAL_SETTINGS>
<SESSION_TIMEOUT VALUE="30"/>
<ILO_FUNCT_ENABLED VALUE="Y"/>
<F8_PROMPT_ENABLED VALUE="Y"/>
<REMOTE_CONSOLE_PORT_STATUS VALUE="3"/>
<REMOTE_CONSOLE_ENCRYPTION VALUE="N"/>
<PREFER_TERMINAL_SERVICES VALUE="N"/>
<HTTPS_PORT VALUE="443"/>
<HTTP_PORT VALUE="80"/>
<REMOTE_CONSOLE_PORT VALUE="23"/>
<TERMINAL_SERVICES_PORT VALUE="3389"/>
<VIRTUAL_MEDIA_PORT VALUE="17988"/>
<MIN_PASSWORD VALUE="4"/>
</GET_GLOBAL_SETTINGS>
```

Setting a configuration

A specific configuration can be sent to the iLO, iLO2, or RILOE II by using the command format:

```
HPONCFG /f add_user.xml /l log.txt
```

In this example, the input file has contents:

```
<!-- Add user with minimal privileges to test default setting of
assigned privileges to 'N' -->
<RIBCL version="1.2">
<LOGIN USER_LOGIN="x" PASSWORD="x">
<USER_INFO MODE="write">
<ADD_USER USER_NAME="Landy9" USER_LOGIN="mandy9"
PASSWORD="floppyshoes">
<RESET_SERVER_PRIV value="Y" />
<ADMIN_PRIV value="Y" />
</ADD_USER>
</USER_INFO>
</LOGIN>
</RIBCL>
```

The specified user will be added to the device.

HPONCFG command line examples

For HPONCFG command line examples, see the appropriate user guide at the Remote Management website (http://www.hp.com/servers/lights-out).

Technical support

In this section

Reference documentation

For issues or problems not addressed by this guide, refer to the following resources for more information:

- The SmartStart Scripting Toolkit website (http://www.hp.com/servers/sstoolkit)
- The Red Hat Linux website (http://www.redhat.com)

Toolkit support

E-mail support for the SmartStart Scripting Toolkit is available from the HP support website (http://atwnt947.external.hp.com/fd2/email_form.cfm?countrycode=US&langcode=en&sni=437764&lang=en&cc=us).

HP contact information

For the name of the nearest HP authorized reseller:

- In the United States, see the HP US service locator webpage (http://www.hp.com/service_locator).
- In other locations, see the Contact HP worldwide (in English) webpage (http://welcome.hp.com/country/us/en/wwcontact.html).

For HP technical support:

- In the United States, for contact options see the Contact HP United States webpage (http://welcome.hp.com/country/us/en/contact_us.html). To contact HP by phone:
    - Call 1-800-HP-INVENT (1-800-474-6836). This service is available 24 hours a day, 7 days a week. For continuous quality improvement, calls may be recorded or monitored.
    - If you have purchased a Care Pack (service upgrade), call 1-800-633-3600. For more information about Care Packs, refer to the HP website (http://www.hp.com).
- In other locations, see the Contact HP worldwide (in English) webpage (http://welcome.hp.com/country/us/en/wwcontact.html).

Acronyms and abbreviations

ACU

Array Configuration Utility

ConRep

Configuration Replication utility

CPQACUXE

Array Configuration Utility XE

CPQLOCFG

Lights-Out Configuration Utility

DHCP

Dynamic Host Configuration Protocol

GUI

graphical user interface

HBA

host bus adapter

HPONCFG

HP Lights-Out Online Configuration utility

HWDISCOVERY

Hardware Discovery Utility

HWQUERY

Hardware Query Utility

IFHW

IF Hardware Utility

iLO

Integrated Lights-Out

IP

Internet Protocol

MBR

master boot record

NFS

network file system

NIC

network interface controller

PXE

Preboot Execution Environment

RAID

redundant array of inexpensive (or independent) disks

RAM

random access memory

RBSU

ROM-Based Setup Utility

RIBCL

Remote Insight Board Command Language

RILOE II

Remote Insight Lights-Out Edition II

SAS

serial attached SCSI

SLES

SUSE LINUX Enterprise Server

SSP

Selective Storage Presentation

STATEMGR

State Manager utility

TFTP

Trivial File Transfer Protocol

USB

universal serial bus

WWN

World Wide Name

XML

extensible markup language

Index