HP SmartStart Scripting Toolkit Linux Edition Best Practices

October 2005 (Fourth Edition)
Part Number 365443-004

October 2005 (Fourth Edition)

Part Number 365443-004

Contents

Introduction

In this section

Linux Toolkit overview

This document describes how to best utilize the Linux edition of the SmartStart Scripting Toolkit to configure HP ProLiant servers. It also contains information about using the Toolkit utilities in an unattended environment. This document does **not** include information about installing the operating system.

The best practices also include suggestions about how to organize utilities and data files by following a set of operational procedures that standardize configuration procedures and help reduce errors.

△ **CAUTION:** Because of the potential risk of data loss, be sure that all necessary precautions are taken so that mission-critical systems are not disrupted if a failure occurs.

Toolkit changes

Previous versions of the SmartStart Scripting Toolkit utilities were designed for the MS-DOS environment. However, limitations in MS-DOS have become a significant problem as hardware and software have evolved. In response to this issue, HP has migrated the SmartStart Scripting Toolkit to a Linux environment to provide better scripting and better hardware support.

The Linux edition of the Toolkit provides the same functionality as the earlier MS-DOS version. However, tools, arguments, and data files are different in the Linux edition of the Toolkit, so you must update your customized scripts to work in a Linux environment. Data files have been migrated to industry-standard XML format to provide improved extensibility. For more information about the Toolkit utilities, refer to the *HP SmartStart Scripting Toolkit Linux and Win32 Editions User Guide* on the Toolkit website (http://www.hp.com/servers/sstoolkit).

Minimum requirements

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

- *HP SmartStart Scripting Toolkit Linux and Win32 Editions User Guide*
- SmartStart Scripting Toolkit Linux Edition
- A Linux workstation (any Linux distribution)
- SYSLINUX package, downloaded from the Web

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.

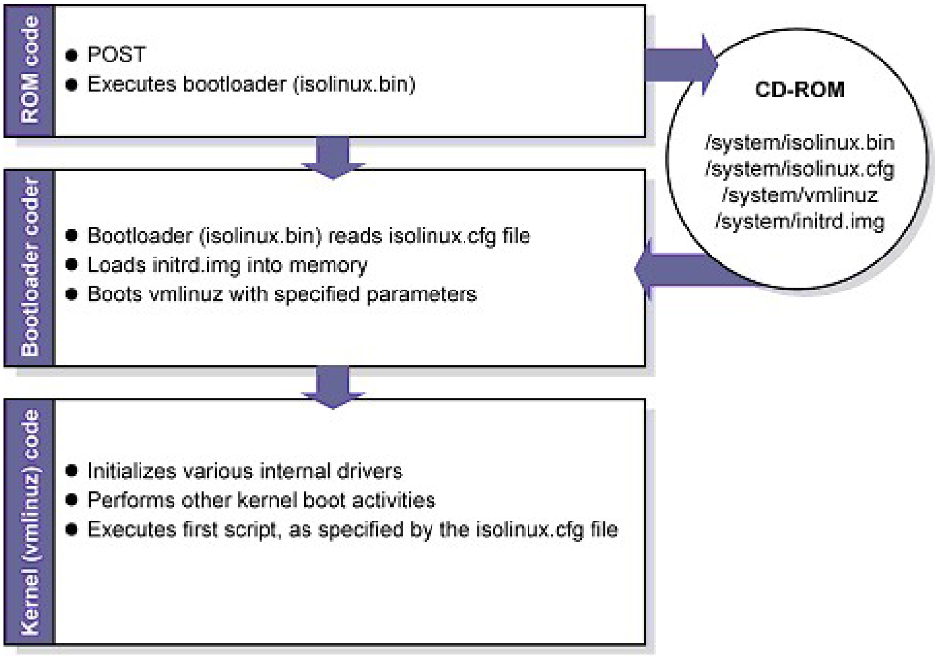
The bootstrap script included with the Toolkit performs the following commands:

```
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
```

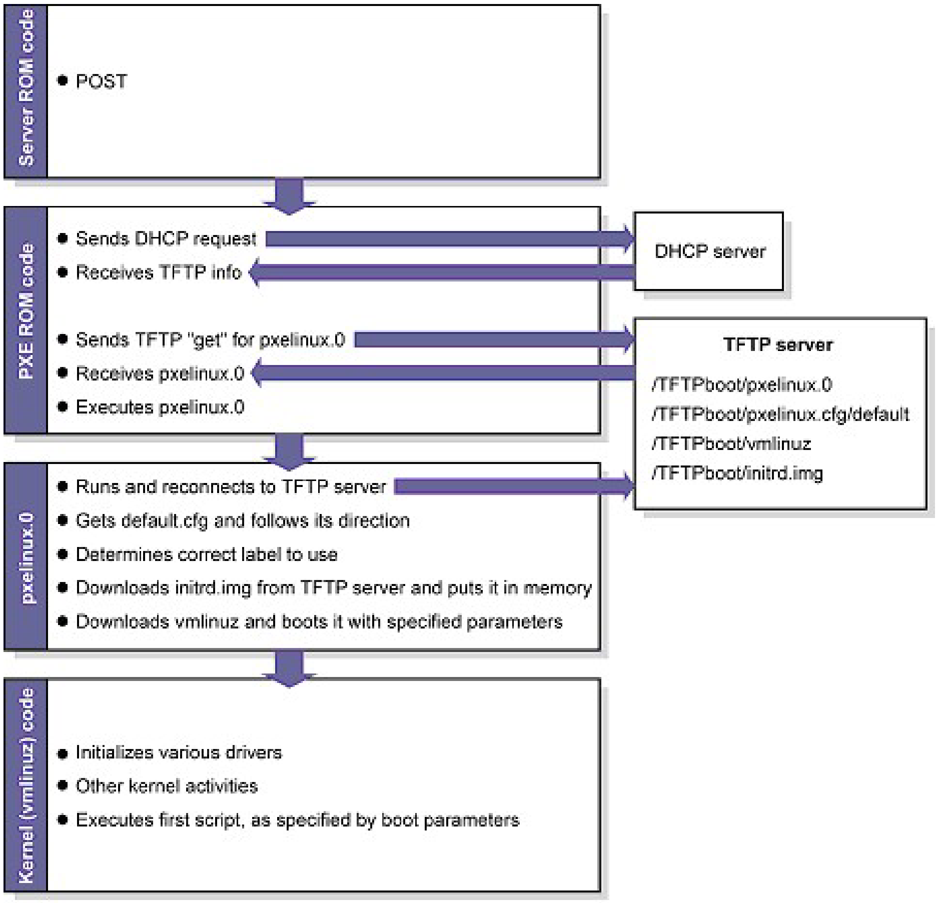
The ISO file can now be written to a CD.

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;
```

[Insert the usual 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-1.3/
   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 `ssstk` 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

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

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

## SAMPLE. Change this to match your NFS server IP/path

echo "Mounting NFS share"
mkdir ${NFS_MAIN}
mount -t nfs 10.0.0.1:/TOOLKIT ${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 utilites 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" )
      ./conrep -l -fdata_files/dl380g4_conrep.dat

      ./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

      ./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 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
         export BOOTDEVNODE=/dev/hda
      fi

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

   ## ADD EXTRA SERVERS HERE

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

## CONTINUE COMMON INSTALL PROCESS

cd ${TOOLKIT}

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 "re-reading partition table"
sfdisk --re-read /dev/sssd

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
sfdisk --re-read /dev/sssd


## 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}

./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
```

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

```
yp-tools
ypbind
```

```
ypserv
zlib
zlib-devel
%post
```

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

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, refer to the HP US service locator webpage (http://www.hp.com/service_locator).
- In other locations, refer to the HP website (http://www.hp.com).

For HP technical support:

- In North America:
  - 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).
- Outside North America, call the nearest HP Technical Support Phone Center. For telephone numbers for worldwide Technical Support Centers, refer to the HP website (http://www.hp.com).

Acronyms and abbreviations

BIOS

Basic Input/Output System

ConRep

Configuration Replication utility

DHCP

Dynamic Host Configuration Protocol

IP

Internet Protocol

MBR

master boot record

NFS

network file system

POST

Power-On Self Test

PXE

Preboot Execution Environment

RAM

random access memory

SLES

SUSE LINUX Enterprise Server

TFTP

Trivial File Transfer Protocol

USB

universal serial bus

XML

extensible markup language

Index