SystemImager® v3.4.1 Manual

Brian Elliott Finley
dann frazier
Austin Gonyou
Ari Jort
Jason R. Mastaler
Greg Pratt
Ben Spade
Denise Walters
Curtis Zinzilieta

SystemImager® v3.4.1 Manual

by
Brian Elliott Finley
dann frazier
Austin Gonyou
Ari Jort
Jason R. Mastaler
Greg Pratt
Ben Spade
Denise Walters
Curtis Zinzilieta

Published $Date: 2005-08-18 14:11:22 -0600 (Thu, 18 Aug 2005) $

SystemImager was created by Brian Elliott Finley, who continues to maintain it as the team lead.

SystemImager is software that automates Linux installs, software distribution, and production deployment. SystemImager makes it easy to do automated installs (clones), software distribution, content or data distribution, configuration changes, and operating system updates to your network of Linux machines. You can even update from one Linux release version to another!

It can also be used to ensure safe production deployments. By saving your current production image before updating to your new production image, you have a highly reliable contingency mechanism. If the new production enviroment is found to be flawed, simply roll-back to the last production image with a simple update command!

Some typical environments include: high performance clusters, Internet server farms, database server farms, computer labs, and corporate desktop environments.

Be sure to view the CREDITS file for a listing of other people who have contributed code or documentation that has been incorporated into SystemImager. Many thanks go to these people, as their relentless pursuit in the discovery bugs and the occasional code contribution are invaluable.

Trademark Notices

Table of Contents

List of Examples

Chapter 1. Introduction to SystemImager®

1.1. SystemImager Overview

SystemImager, a component of the System Installation Suite, is software that allows you to install Linux to large numbers of similar machines relatively easily. You can also use it for software distribution, configuration, and operating system updates, in addition to using it to update from one Linux release or version to another. SystemImager can also be used for content management on web servers.

SystemImager is most useful in environments where you have large numbers of identical machines, including Internet server farms, high performance clusters, computer labs, or corporate desktop environments where all workstations have the same basic hardware configuration. For more background information on design goals and product development, see "SystemImager Background."

1.2. Who Should Use This Guide

This guide is for system administrators who install and configure systems in a network environment. Those who would benefit from using SystemImager include:

- Organizations that have Internet server farms
- Organizations that manage many workstations or servers
- Organizations doing super-computing/clustering with Linux
- Anyone who needs to maintain identical configurations on a large number of machines
- Manufacturing organizations that must automate the software preload process for Linux based machines.

1.3. How SystemImager Works

SystemImager allows you to retrieve an entire system image from a golden client, which is a manually installed, customized machine, to an image server, which is the machine that will hold and distribute system images. You can deploy the images to any number of client systems from the image server.

After initial image deployment, you can update the client systems by syncing them to an updated image on the image server. Updates are fast and efficient because only the modified parts of files are pulled to the client.

1.3.1. Supported Distributions

SystemImager uses System Configurator (http://sourceforge.net/projects/systemconfig) to custom configure autoinstall clients for specific distributions. Through System Configurator, SystemImager supports all major Linux distributions and most others, including custom or in-house distributions. Using

"footprints," System Configurator works with distributions based on their system configuration style rather than needing to know the name of the distribution. To determine a system's footprint, System Configurator identifies the configuration files in use and associates that footprint with a configuration style. It then correctly makes settings, such as hostname or IP address, without needing to know the name of the distribution.

Therefore, SystemImager is distribution agnostic in nearly all areas. With few exceptions, all distribution specific knowledge exists within the SystemConfigurator tool, which supports a very large range of distributions. If you find a distribution that does not work with SystemImager, please file a bug report.

NOTE: Linux kernel 2.0.x-based distributions are not supported at this time, nor are there plans to add support for Linux kernel 2.0.x-based distributions.

1.3.2. System Requirements

- Your image server must have enough disk space to hold the images to be installed on your client systems.
- All clients that will use the same image should have hardware that is as similar as possible. Most importantly they should use the same chipset on the network device(s) and the same number and kind of hard drive(s) (e.g. IDE, SCSI, Mylex Hardware RAID, etc.) The hard drives may be of different capacities, and disks may be larger with no problem and smaller within reason.

  Advanced users can modify the `/etc/systemimager/autoinstallscript.conf` file within an image to make adjustments, then run the **mkautoinstallscript** command to install an image on clients with varying disk configurations.
- For PXE installations, you need a compatible TFTP server running on the boot server, which is usually the same machine as the image/DHCP pserver. Debian provides such servers in the tftpd-hpa and atftpd packages, while Red Hat 7.0 and later include such a server in the tftp-server package. H. Peter Anvin maintains the tftp-hpa package that provides the required functionality.
- In addition to a compatible TFTP server, PXE network-based installations may also require a PXE daemon to run on your image server. This requirement depends on the firmware used on the client side and the capabilities of your DHCP server.

1.4. Glossary of Terms

addclients

a command that tells your image server which image to install on the autoinstall clients by creating soft links to the master autoinstall script with the name of each host that will receive that image. **addclients** also allows you to populate the `etc/hosts` file with sequential host names and IP addresses. The information in this file is necessary for some SystemImager operations.

autoinstall media

media that is used to boot an autoinstall client to begin the autoinstall process. Autoinstall media can be a floppy, a CDROM, the network, or the local hard drive of the autoinstall client.

autoinstall script

One or more scripts associated with an image, each unique to a specific hardware/partitioning/filesystem/network configuration. The **getimage** command creates an initial autoinstallscript, which can be regenerated later, possibly with different options, using the **mkautoinstallscript** command. The autoinstall script (also called the ".master script") is downloaded and executed by the autoinstall client, and performs most of the autoinstall process. Names of autoinstall scripts begin with the image name and end in .master. For example: `my_webserver_image_v1.master`

daemon (http://www.tuxedo.org/~esr/jargon/html/entry/daemon.html)

a program that lies in wait for something to trigger it to action, for example, a web server daemon waiting for a Web page request to begin its retrieval function.

getimage

a command run from the image server to pull a system image from a golden client.

golden client

a manually-installed, customized machine from which an image is taken for deployment to client systems.

image server

the machine that holds and distributes the system images.

local.cfg

a configuration file that can be used for autoinstall clients in lieu of DHCP, DNS, and/or the `/var/lib/systemimager/scripts/hosts` file on the image server.

mkdhcpserver

a command that creates a SystemImager -appropriate `/etc/dhcpd.conf` file. DHCP can be used to assign IP addresses to autoinstall clients.

mkdhcpstatic

a command to modify the `/etc/dhcpd.conf` file, adding static entries for autoinstall clients based on the IP addresses handed out to these clients by the DHCP server.

mkbootserver

A utility for configuring a network boot server. Currently just supports configuring PXE boot servers for i386 clients.

prepareclient

a command you must run on the golden client immediately prior to running **getimage** on the image server. **prepareclient** prepares the golden client to have its image retrieved and creates an `etc/systemimager` directory with information about the golden client, such as the disk partitioning scheme(s).

updateclient

a command that updates or synchronizes client systems to a new or updated image after the initial autoinstall, enabling software and content distribution.

Chapter 2. Installing SystemImager

2.1. How Does it Work?

The SystemImager concept is that an image server retrieves a golden client's entire system image and deploys it to any number of client systems. A golden client is a system you have customized to work exactly the way you want. You can re-compile the kernel, install custom software, and do any configuration file tweaking you like. The **getimage** command pulls the golden image to the image server for deployment to other systems.

Once you have deployed the initial image to your client systems, you can update/upgrade the client systems by syncing them to an updated image on the image server. Only the modified parts of files are pulled to the client for a fast, efficient, and accurate mass update/upgrade.


Tools other than SystemImager are available for doing automatic installations, such as Red Hat's Kickstart, which installs systems based on a list of pre-defined packages. However, such package-based installs can be very limiting because they don't have an automated way to deal with non-packaged files. If you re-compile your kernel, add a piece of non-packaged software, or modify certain configuration files, package-based installation methods usually require you to do some sort of scripting or programming to deal with these "special cases."

2.2. Obtaining SystemImager

SystemImager is currently packaged in .deb, and RPM formats. SystemImager is a part of the Debian GNU/Linux 3.0 distribution, starting with 3.0 (woody). For earlier releases and other distributions, download packages from http://systemimager.org/ (http://systemimager.org).

2.3. Selecting A Machine To Use As An Image Server

Because SystemImager uses other network services such as DHCP, an existing server that provides these services often makes a good choice for an image server. In addition, the image server you choose needs to have enough disk space to hold the images you want to deploy. SystemImager stores images as an uncompressed directory structure, so a quick analysis of the disk usage on your golden client will give you a good estimate of the space required on the image server. If you plan to do multiple simultaneous image updates, poor processor performance on your image server can cause a bottleneck. Keep this in mind when selecting the number and speed of processors for your image server.

2.4. Installing an Image Server

To create a SystemImager image server, you must install a systemimager-boot package(s)and then a systemimager-server package that supports your clients.

The boot packages support multiple configurations. Because different client configurations require different drivers, kernel versions, etc., SystemImager allows you to install different boot packages, which are known as "boot flavors."

Each SystemImager release provides the "standard" flavors for each supported architecture. For example, the 3.0.0 release should have these packages: systemimager-boot-i386-standard_3.0.0-1_all.deb, systemimager-boot-ia64-standard_3.0.0-1_all.deb, systemimager-i386boot-standard.noarch.rpm, and systemimager-ia64boot-standard.noarch.rpm

SystemImager and its standard boot flavors support most common hardware configurations. The `.config` files list the options for this kernel.

You can use other flavors at any time to support alternate client configurations, and mulitple boot flavors can be installed simultaneously. However, boot packages are strongly coupled to the version of SystemImager for which they were created. The SystemImager boot package version strings should match the version string of the installed version of SystemImager.

To install an image server for Debian GNU/Linux 3.0 (woody) and later, run:

1. **# apt-get update**
2. **# apt-get install systemimager-server**

To obtain the latest systemimager release on a Debian system running stable or testing, you must add the following line to the `/etc/apt/sources.list` file:

```
deb http://download.systemimager.org/debian stable main
```

These packages are synchronized with the packages in the unstable distribution periodically. Because systemimager's dependencies are available in both the latest stable and unstable releases, we just copy over the binary packages - we do not rebuild them.

For RPM based distributions:

1. Download and install the systemimager-server RPM and required dependencies from http://systemimager.org/download/ (http://systemimager.org/download/).
2. Install each package with the **rpm** command.

   # rpm -Uvh *.rpm
3. The rpms leave the systemimager service disabled by default, so you will need to start the systemimager service in order to make images available to your clients via rsync.

   To make systemimager automatically start each time you boot your system, run:

   chkconfig --add systemimager

You can list the set of runlevels where systemimager will be automatically started by running:

chkconfig --list systemimager

To start systemimager by hand, run:

service systemimager start

For other distributions: If your distribution does not support RPMs or .debs, you must build and install from the source tarball; binary tarball installations are no longer supported. Instructions for building SystemImager can be found in the `README` file within the source tarball.

Installing Flamethrower for multicast installations. See Chapter 4, HOWTO Use Flamethrower for Multicast Installs

2.5. Selecting A Machine To Use As A Golden Client

A golden client must have similar hardware to the machines that will later use its image (i.e. the same network adapter chipset and the same storage adapters). You cannot currently deploy a SCSI-based image to an IDE based-system (or vice-versa) without manually editing the master script and various files within the image. The disk size(s) of the golden client can be smaller than the other machines that will use its image and larger within reason.

2.6. Installing SystemImager Client Software on a Golden Client

To create a golden client, you must install the systemimager-client package.

For Debian GNU/Linux 3.0 (woody) and later, run:

1. **# apt-get update**
2. **# apt-get install systemimager-client**

For Debian 2.2 (potato), you must add the following line to the file `/etc/apt/sources.list`:

```
deb http://systemimager.org/debian /
```

For RPM based distributions:

1. Download and install the systemimager-client RPM and required dependencies from http://systemimager.org/download/ (http://systemimager.org/download/).
2. Install each package with the **rpm** command.

 # rpm -Uvh *.rpm

For other distributions: If your distribution does not support RPMs or .debs, you must use build and install from the source tarball; binary tarball installations are no longer supported. Instructions for building SystemImager can be found in the `README` file within the source tarball.

2.7. Creating an Image on the Golden Client

To create an image for deployment with the SystemImager tool, install and configure a Linux distribution and any additional software that you want the image to contain on a system you will use as your golden client. You will deploy the image of that system (or golden client) onto other machines.

System Installer, a component of the System Installation Suite to which SystemImager belongs, allows you to install Linux directly to an image, bypassing the golden client step. System Installer packages and documentation can be found at http://systeminstaller.sourceforge.net (http://systeminstaller.sourceforge.net) Images created with SystemInstaller are interchangeable with those created using the SystemImager tools.

2.8. Upgrading SystemImager

Although SystemImager upgrades are automated in most ways, you must perform some upgrade processes manually to prevent losing user customizations.

2.8.1. Regenerating autoinstallscripts

With each release of SystemImager, use the **mkautoinstallscript** command to update the autoinstall scripts stored in `/var/lib/systemimager/scripts`. Installations using older scripts may fail.

mkautoinstallscript overwrites the pre-existing script for an image. If you make any changes to your autoinstall scripts (also known as .master scripts), backup those scripts to forward port your changes to the new release.

Example 2-1. Generating new autoinstallscripts

/usr/sbin/mkautoinstallscript -image myimage -post-install reboot -ip-assignment dhcp

If you require customizations to your autoinstallscript, edit the appropriate .master file in `/var/lib/systemimager/scripts/`.

2.8.2. Migrating Customizations of the /etc/systemimager/rsyncd.conf file to SystemImager 3.0.0

Prior to the release of SystemImager 3.0.0, your changes to `/etc/systemimager/rsyncd.conf` could be made within the file, but were susceptible to upgrade issues. With version 3.0.0, you can make these changes in a separate file that is maintained across upgrades. See the mkrsyncd_conf(8) man page for details.

2.8.3. Regenerating boot media

Each time you upgrade Systemimager, you must also upgrade the boot media you use to boot the autoinstall system. Use **mkautoinstallcd** or **mkautoinstalldiskette** to regenerate removable media. If you network boot, update the files in your /tftpboot directory and pass all necessary options to the kernel. These options occasionally change - check /etc/systemimager/pxelinux.cfg/syslinux.cfg for examples.

2.8.4. What happened to the binary tarballs?

The SystemImager 3.0.0 release deprecates the binary tarball releases and introduces the boot packages feature, which requires that various components of the SystemImager system have some sort of version control. Without relying on a package management system, version control requires significant effort, so SystemImager 3.0.0 does not support unpackaged bits.

However, with the build system based on make, you can run commands like **make install_server_all**, etc., which is now the preferred method for installing without using a package manager, although you need to track down all build dependencies and runtime dependencies by hand.

Chapter 3. Using SystemImager

3.1. Installation Procedures Overview

1. Using the instructions in Chapter 2, install the SystemImager server package on the machine you have chosen as your image server.
2. Install Linux on your golden client and customize as desired.
3. Using the instructions in Chapter 2, install the SystemImager client software on the golden client.
4. Run the **prepareclient** command on your golden client.
5. Choose and configure the method for assigning IP addresses to your autoinstall clients. This information is required for the **getimage** command in the next step; however, you can change these settings later by running the **mkautoinstallscript** command.
6. Run **getimage** on the image server to pull the golden client to the image server.
7. Run **addclients** on the image server to tell it which clients will receive what image and to populate the image server's `/etc/hosts` and `/var/lib/systemimager/scripts/hosts` file.
8. Autoinstall the golden image on other machines.

 See the SystemImager Tools section in this chapter for detailed tool descriptions and functions.

3.1.1. Detailed Installation Instructions

1. Install the SystemImager server package on the machine you have chosen as your image server, using the instructions in Chapter 2.
2. Install Linux on your "golden client" and customize as desired. Remember that the software installed will eventually constitute the golden image for all other nodes installed with SystemImager. Don't worry too much about getting it exactly right the first time, as you can easily use SystemImager to make incremental changes to your image and deploy those changes without doing a complete re-install.
3. Install the SystemImager client software on the golden client using the instructions in Chapter 2.
4. On the golden client, run the command **prepareclient** as root. This will create various files in your /etc/systemimager directory that contain information on your disk partition scheme, filesystem types, etc. **prepareclient** will also start an rsync daemon to allow its files to be transferred to a server. Your golden client is now ready to have its image pulled by an image server.

 If you are not in ssh mode, all files on your golden client are openly accessible to anyone on your network. Once you have pulled the image from your golden client, you can deactivate the

rsync daemon by killing the **rsync** process or by simply rebooting the golden client. This **rsync** server will not start automatically on future reboots.

5. Choose and configure the method for assigning IP addresses to your autoinstall clients.

   The most common way to assign IP addresses to autoinstall clients is DHCP. To simplify the configuration of the DHCP configuration file (`/etc/dhcpd.conf`), SystemImager includes a utility called **mkdhcpserver**. This utility asks you for all the information it needs to create a DHCP configuration file that is appropriate for your installation of SystemImager. After installation, you can use DHCP to assign static IP addresses to your clients on an ongoing basis by running the **mkdhcpstatic** command after your clients have booted and received an IP address. **mkdhcpstatic** will modify your `/etc/dhcpd.conf` file on the imageserver to include static entries for each of your hosts.

   Alternately, you can put hostname, imageserver, and networking information in a configuration file on a floppy diskette. The format of the configuration file is simply `VARIABLE=value` for all the appropriate settings. The name of this file must be `local.cfg` and it must exist on the root of the floppy or hard drive. The floppy can be formatted with either ext2 or fat. An example local.cfg file, such as the one below, can be found with the documentation files that are usually installed in `/usr/share/doc`.

   Example 3-1. Contents of an example `local.cfg` provided with SystemImager:

```
HOSTNAME=www1
DOMAINNAME=systemimager.org
DEVICE=eth0
IPADDR=10.0.0.99
NETMASK=255.255.255.0
NETWORK=10.0.0.0
BROADCAST=10.0.0.255
GATEWAY=10.0.0.1
    GATEWAYDEV=eth0
IMAGESERVER=10.0.0.3
IMAGENAME=oracle_db_server-1.0
```

   Alternatively, if you are using a running system's hard drive as the boot media, you can run **updateclient -autoinstall -server <imageserver> -configure-from eth0** , which will create a `local.cfg` file at the root of the client's hard drive containing the existing live network settings. When the autoinstall client boots, it will look for this file and use the provided values instead of getting them from DHCP and the `/var/lib/systemimager/scripts/hosts` file on the image server.

   A `local.cfg` file on a floppy will work with any of the autoinstall media. You can even put the configuration file on the autoinstall floppy itself. If you use a `local.cfg` file on a hard drive and on a floppy, the settings on the floppy will override the settings on the hard drive.

   Example 3-2. Running mkdhcpserver

```
[root@imageserver]# mkdhcpserver
```

Example 3-3. Running updateclient with the "-autoinstall" and "-config" options

Note that the options *-autoinstall*, *-server*, and *-configure-from* are abbreviated below as *-a*, *-s*, and *-c*. You can abbreviate options to minimum uniqueness with most SystemImager commands.

Minimum uniqueness means that if two options for a single command are similar, such as the *-image* and *-ip-assignment* options to **getimage,** you can abbreviate them to *-im* and *-ip*.

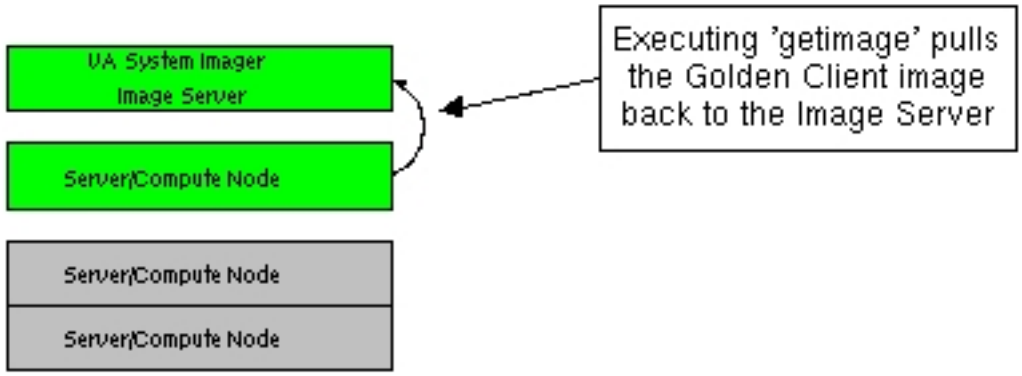
```
[root@server7]# updateclient -a -s imageserver -c eth0
Retrieving SystemImager kernel...
Retrieving SystemImager initial ramdisk...
Adding SystemImager entry in /etc/lilo.conf...
running /sbin/lilo -d 50 -D systemimager ...
Ignoring entry 'delay'
Ignoring entry 'default'
Added linux
Added systemimager *

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
Below are the contents of your /local.cfg file.  Make sure that all
the variables are filled in and that they contain the proper values.
You may edit the file directly if you need to change any of the values.
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

#
# "SystemImager"
#
#  Copyright (C) 1999-2002 Bald Guy Software <brian.finley@baldguysoftware.com>
#
# This file is: /local.cfg
#
HOSTNAME=server7
DOMAINNAME=mydomain.com
DEVICE=eth0
IPADDR=192.168.1.7
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
GATEWAY=192.168.1.254
GATEWAYDEV=eth0
IMAGESERVER=192.168.1.203
```

6. Run **getimage** on the image server to pull the image from the golden client to the image server.

Example 3-4. Pull the golden client's image to the image server



The basic syntax is: "**getimage -golden-client** [client_hostname] **-image** [image_name]"

Where [client_hostname] is the hostname or IP address of the "golden client" and [image_name] is the name you want to give to this image. You can see many other options with "**man getimage**."

Example 3-5. Running getimage

```
[root@imageserver]# getimage -golden-client my-golden-client -image web_server_image
```

getimage contacts the golden client and requests its `/etc/systemimager/mounted_filesystems` file, which contains the list of mounted filesystems and their mount points. **getimage** pulls out the mount points for the filesystems that are unsupported and creates an exclusion list. The filesystems SystemImager currently supports are ext2, ext3 and reiserfs. All other filesytems will be ignored, including proc, nfs, devpts, iso9660, etc.

7. **addclients** creates a symbolic link to the master script for the image to which each specified autoinstall client is assigned. **addclients** also populates the image server's `/etc/hosts` and `/var/lib/systemimager/scripts/hosts` files. The hosts file provides the default mechanism used by autoinstall clients to look up their hostnames.

   When **addclients** is run without arguments, it takes you through three configuration screens interactively.

   a. In the first configuration screen **addclients**, asks you to specify the hostname pattern of your autoinstall clients. Autoinstall client hostnames are comprised of [basename][number]. For example, if you choose "proxy" as your basename, and 3-10 as your range, you will define the following autoinstall clients:
   proxy3
   proxy4
   proxy5
   proxy6
   proxy7
   proxy8
   proxy9
   proxy10

   b. In the second screen, you map the clients defined in Section 1 to an image.

Each invocation of **addclients** allows you to map a single range of clients to an image. If you want to map non-consecutive ranges of clients to an image, or want to map different client ranges to different images, you must execute the **addclients** command multiple times.

c. In the third configuration screen, the **addclients** command asks you for a range of IP addresses from which it populates your `/etc/hosts` and `/var/lib/systemimager/scripts/hosts` files. When an autoinstall clients boots, it will attempt to retrieve the latter file from the image server and use it to look up its hostname. If this step fails, the client will attempt to do a reverse DNS lookup. If you have PTR records configured for each of your autoinstall clients, you can skip the third configuration step; however, it is recommended to complete it because it is more robust.

Example 3-6. Entries in /etc/hosts created by addclients

If you give **addclients** a range of IPs from 192.168.1.1-192.168.1.99, a base hostname server, and a hostname extension range of 1-99, then it would generate the following `/etc/hosts` file:

```
192.168.1.1    server1.mydomain.com  server1
192.168.1.2    server2.mydomain.com  server2
192.168.1.3    server3.mydomain.com  server3
192.168.1.4    server4.mydomain.com  server4
192.168.1.5    server5.mydomain.com  server5
192.168.1.6    server6.mydomain.com  server6
192.168.1.7    server7.mydomain.com  server7
192.168.1.8    server8.mydomain.com  server8
192.168.1.9    server9.mydomain.com  server9
192.168.1.10   server10.mydomain.com  server10
192.168.1.11   server11.mydomain.com  server11

[ ... etc, etc, etc ... ]

192.168.1.97    server97.mydomain.com  server97
192.168.1.98    server98.mydomain.com  server98
192.168.1.99    server99.mydomain.com  server99
```

8. Autoinstall the golden image on other machines.

   You can use one of four methods to autoinstall the clients:

   - Boot the system from a floppy diskette.

     Run **mkautoinstalldiskette** to create an autoinstall diskette that you can use with any machine unless you add a customized `/local.cfg` file to the diskette.

   - Boot the system from a CDROM.

Run **mkautoinstallcd** to create an ISO image that can be burned to CDROM. You can use the CDROM to boot your autoinstall clients and use it with any machine.

- Boot the system from its own hard drive.

  If your client is already running Linux and uses LILO as its boot loader, you can simply copy the **updateclient** command and run it with the `-autoinstall` option.

  Example 3-7. Booting the autoinstall media from a system's hard drive

  ```
  [root@server7]# updateclient -a -s imageserver -c eth0
  ```

- Boot the system from the network. If your systems are network-boot capable, using PXE for example, you can start an autoinstall without using local media.

  PXE is usually enabled through a BIOS setting. Booting can be unstable and client side firmware is not consistent. If you must boot without physically touching your machines, try the `-autoinstall` option to **updateclient** first.

  SystemImager comes with the **mkbootserver** utility to help configure a PXE server. Running **mkbootserver** is an iterative process. It will attempt to generate an appropriate tftproot directory, configure your tftp server, and run various tests to see if things are functioning properly. Once **mkbootserver** detects an error, it will fail out and generate an error message. When you have corrected the error, you can re-execute **mkbootserver**, and repeat until it exits successfully. **mkbootserver** will probably not work with all PXE clients. If it fails to work with your configuration, please submit a bug report to http://sourceforge.net/tracker/?group_id=259&atid=100259.

3.1.2. SystemImager Tools

3.1.2.1. the prepareclient command

- After configuring the golden client, run the **prepareclient** command to create a file with the partition information from your disks and put it in the `/etc/systemimager/partitionschemes` directory.
- **prepareclient** will also create an rsync(1) configuration file (/tmp/rsyncd.conf) and start rsync in server mode (rsync--daemon). This step allows the image server to pull the image from the client but will not cause the rsync daemon to be restarted after the golden client is rebooted, helping avoid security concerns from sharing a golden client's root filesystem via rsync.

3.1.2.2. The getimage command

- After running **prepareclient**, run the the **getimage** command on the image server. For example : **getimage -golden-client 192.168.1.1 -image my_webserver_image_v1**

- **getimage** contacts the golden client and requests its `/etc/systemimager/mounted_filesystems` file, which contains the list of mounted filesystems and the devices on which they are mounted. It pulls out the mount points for the filesystems that are unsupported and creates an exclusion list. Currently supported filesystems are ext2, ext3, and reiserfs. All other filesystems are unsupported, including proc, devpts, iso9660, etc.
- **getimage** then pulls the golden client's entire system image, excluding the filesystems in the exclusion list, by connecting to the rsync daemon running on the golden client. All the files from the client will be copied over, recreating the file and directory hierarchy in the image directory.
- You can also use **getimage** to update an existing image by simply specifying an existing image name, for example, **getimage -golden-client 192.168.1.1. -image <imagename>. getimage** then updates the image to match the files on your golden client. When you do this, only the parts of files that are different will be copied over. Files that exist in the old image but not on the golden client will be deleted, and files that exist in both places but have changed will be updated. **getimage** is one way to update an image when new security patches or other system updates come out. However, this method is revision control on an image-by-image basis, and not true revision control where individual file revisions are tracked on a line-by-line basis. The recommended method is never to overwrite a known working image. Revision control on an image-by-image basis also ties in to the **updateclient** command. By default, all images are stored in the parent directory of `/var/lib/systemimager/images/` in a directory that bears the image name. For example: `/var/lib/systemimager/images/my_webserver_image_v1/`

3.1.2.3. Autoinstall scripts

- After **getimage** has pulled the files to the image directory on the imageserver, it creates a customized autoinstall script. The autoinstall script in this example is named "my_webserver_image_v1.master". All autoinstall scripts are placed in the `/var/lib/systemimager/scripts` directory.
- The disk partitioning information left behind by the **prepareclient** command adds the necessary commands to re-partition the disk(s) on the autoinstall clients.
- File system information is taken from the `/etc/fstab` file in the image (i.e.: `/var/lib/systemimager/images/my_webserver_image_v1/etc/fstab`) and used to determine the appropriate file system creation commands and to determine mount points for the autoinstall process. Networking information is added to the autoinstall script based on command line options passed to **getimage** or information it prompts you for. This information is added in variable form as the autoinstall client will determine the values for things such as its hostname and IP address during the autoinstall process.

3.1.2.4. The addclients Command

- After running **getimage**, run the **addclients** command, which asks you for the series of hostnames you will be installing by combining a base host name and a number range. For example, if your base host name is "www", and your number range is from "1" to "3," then the resultant host names would be "www1, www2, www3". **addclients** then prompts you to choose the image that will be installed to

these hosts and creates soft links for each hostname that point to the master autoinstall script for that image. For example: "www3.sh -> web_server_image_v1.master".

- If the image is updated and you allow **getimage** to update the master autoinstall script also, then each of the associated soft links will point to the updated autoinstall script. If individual host configuration is necessary, the soft link for that host can be removed and replaced with a copy of the master autoinstall script that can then be customized for that host. This customization is a manual process and is up to the system administrator.

3.1.2.5. Additional Installation Information

- The unattended install procedure is flexible and works with almost any available hardware. You can also easily modify it to work with new or special hardware. A miniature Linux distribution called Brian's Own Embedded Linux (boel) is used for autoinstalls. It consists of a customized kernel and an initial ram disk that contains only the specific commands and utilities necessary to perform autoinstalls. The same kernel and initial ram disk (`initrd.gz`) can be used to boot from floppy disks, CDROMs, the network, or any running Linux system's local hard drive.

  The **mkautoinstalldiskette** and **mkautoinstallcd** commands use the **syslinux(2)** utility to create floppies and CDROMs that will boot the SystemImager kernel and initial ram disk. **pxelinux(2)**, which is a sister tool to **syslinux**, allows the same kernel and initial ram disk to boot PXE capable machines from the network. Both **syslinux** and **pxelinux** need a configuraton file, but the two tools can use the same one and SystemImager handles this for you.

- The autoinstall kernel contains all the necessary drivers for the majority of systems. Custom kernels can be compiled to meet special disk and network driver requirements. To use a custom kernel, simply copy it to `/tftpboot/kernel`. All of the autoinstall media is created from `/tftpboot/kernel` and `/tftpboot/initrd.gz` on the image server.

- Once the kernel has booted, it mounts the initial ram disk as its root filesystem. The kernel then executes an initialization script on the ram disk that has been written to do SystemImager-specific tasks. This script will use either a configuration file (`/local.cfg`) or a combination of DHCP and the `/var/lib/systemimager/scripts` file pulled from the image server to determine the autoinstall client's IP address and hostname information.

  If DHCP is used, the client parses the hosts file which was retrieved from the image server to find its IP address and determine its hostname. Finally, the client retrieves an autoinstall script from the image server based on its hostname and executes it. The autoinstall script is image specific, determining which image a client will receive. Following is a summary: IP address -> hostname -> image specific autoinstall script named with hostname.

3.1.2.6. How to Update an Image

- If you want to update an image on your image server, you can use one of the two following methods:

  1. Directly edit the files in the image directory. The best way to do this is to **chroot** into the image directory. You can then work with the image as if it were a running machine. You can even install packages with **apt-get** or **RPM**, for example.

2. Run the **getimage** command again, specifying a golden client that has been modified in the desired way. Again, only the parts of the files that have changed will be pulled across. Files that have been deleted on the golden client will also be deleted in the image. You have the option to update the master autoinstall script for the image or leave it alone. The advantages of running the **getimage** command are that you can verify that your new configuration works on the golden client and that the master autoinstall script is updated.

- Once a system has been autoinstalled, you can use the **updateclient** command to update a client system to match a new or updated image on the image server. So, for example, if you've installed your company's 300 web servers and a security patch comes out the next day, you can simply update the image on the image server and run **updateclient** on each of your web servers. Only the modified files are pulled over, so your site is patched very quickly. You should create an entirely new image with a new version number so that you have some form of revision control. This way, if you find out that the patch you applied corrupted your entire web farm, you can simply do an **updateclient** back to the last known working image.
- By incorporating some modifications submitted by A. L. Lambert of epicRealm, you can use the **updateclient** command with the `-autoinstall` option to copy the autoinstall kernel and initial ram disk to the local hard drive of an autoinstall client that is currently running but needs to be re-deployed. **updateclient** then modifies the `/etc/lilo.conf` file to include an appropriate entry for the new kernel and initial ram disk and makes this new kernel the default. The next time the client system boots, it loads the SystemImager kernel and initial ram disk, which begins the autoinstall process. You can therefore remotely redeploy any running Linux machine without feeding the machine a floppy or CD and without having to reconfigure the BIOS to boot off the network, which can be quite problematic with some BIOSes.

Chapter 4. HOWTO Use Flamethrower for Multicast Installs

4.1. Install SystemImager v3.2.x or higher

See http://systemimager.org/download for details.

4.2. Install the Flamethrower Package and its Dependencies

Debian users can do an "apt-get update" and "apt-get install flamethrower".

For RPM based distributions, you can download the Flamethrower RPMs from here (http://sourceforge.net/project/showfiles.php?group_id=259).

4.3. Run getimage, mvimage, or cpimage

Flamethrower is a stand-alone utility, but SystemImager maintains a SystemImager seperate copy (/etc/systemimager/flamethrower.conf) of the Flamethrower configuration file (/etc/flamethrower/flamethrower.conf). Each of these three commands (getimage, mvimage, cpimage) will create an appropriate entry in /etc/systemimager/flamethrower.conf for the image you specify.

It is also possible to manually add an entry, if you feel compelled to to so. Take a look at the flamethrower.conf man page, or the comments inside the configuration file (image entries go at the very bottom of the file). However, it's much easier to just do a:

```
"mvimage my_image-v1 tmp_name && mvimage tmp_name my_image-v1"
```

4.4. Start up your Flamethrower daemon

Edit /etc/systemimager/flamethrower.conf, and set the following:

```
START_FLAMETHROWER_DAEMON = yes
```

Then crank it up with:

```
/etc/init.d/systemimager-server-flamethrowerd restart
```

4.5. Tell your autoinstall clients to use Flamethrower

If you use DHCP, you can add the following to your dhcpd.conf file:

```
option option-143 code 143 = string;  # (only for ISC's dhcpd v3)
option option-143 "9000";
```

Then restart your DHCP daemon. This is usually either

```
/etc/init.d/dhcp restart
```

or

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

If you use a local.cfg file, you can add the following to that file:

```
FLAMETHROWER_DIRECTORY_PORTBASE=9000
```

4.6. Autoinstall your clients!

Boot your clients from your favorite autoinstall media (network, floppy, CD, or hard drive) and off you go.

4.7. A Detailed Walk Through the Process

1. When the imageserver is started, the `/etc/init.d/systemimager-server-flamethrowerd` init script is run, which starts the Flamethrower daemon (flamethrowerd). flamethrowerd, in turn, starts up udp-sender processes according to `/etc/systemimager/flamethrower.conf`.
2. A new image is pulled using getimage, and the information in `flamethrower.conf` is updated.
3. Prior to starting each cast, flamethrowerd will check to see if `flamethrower.conf` has been updated. If it has, it will be re-read. This allows new images to be made available dynamically, without having to restart `flamethrowerd`.

4. When an autoinstall client comes up, it checks to see if FLAMETHROWER_DIRECTORY_PORTBASE is set. This can be set via "option-143" in `dhcpd.conf`, or via the FLAMETHROWER_DIRECTORY_PORTBASE variable in a local.cfg file. The default value for FLAMETHROWER_DIRECTORY_PORTBASE is 9000.

   If FLAMETHROWER_DIRECTORY_PORTBASE is set, then the autoinstall client requests directory information from the flamethrower daemon running on port FLAMETHROWER_DIRECTORY_PORTBASE. The imageserver waits a few seconds for other clients to join the directory information multicast, then casts out the directory information. The directory information is a directory of files, one per module, each containing the multicast details needed to access that particular module.

5. When the client receives the Flamethrower directory information, it will look up the multicast information for each of the following modules, then sequentially join the multicast for each:

   a. boel binaries (additional binaries that the autoinstall client needs to continue the install)

   b. The entire autoinstall scripts directory

6. After receiving the scripts cast, it will find it’s autoinstall script, and execute it. The autoinstall script will proceed with the install by sequentially joining the multicasts for the appropriate image, then for any and all override directories.

7. When all casts are received, the client completes the install according to the autoinstall script by configuring the boot loader, and performing it’s post install action.

8. When the flamethrowerd daemon completes the cast session for a module, it checks for changes in the `flamethrower.conf` file, re-reads it if necessary, then listens for new clients asking to join the cast for that module. Casts of seperate modules can happen in parallel.

Chapter 5. HOWTO Use OpenSSH for Secure Installs

5.1. Using OpenSSH for Secure Installs

SystemImager supports the ability to capture and install images over ssh. You might want to use this feature if you are doing installations in an unprotected network, or over the public Internet. Some amount of scalability and installation time degredation is expected, and we currently do not support secure multicast (flamethrower) installations.

5.2. Installing systemimager-ssh

Debian users can do an "apt-get update" and "apt-get install systemimager-ssh-<client-arch>".

For RPM based distributions, you currently need to build your own ssh packages.

Chapter 6. FAQ (Frequently Asked Questions)

Q: Where are the images stored?

A: The images are stored in `/var/lib/systemimager/images`.

 NOTE: If you are short on disk space in this location, move the directory to another location:

mv /var/lib/systemimager/images /home/systemimager_images

Then create a soft link to the new directory.

ln -s /home/systemimager_images /var/lib/systemimager/images

Q: How do I make an autoinstall diskette?

A: Run the **mkautoinstalldiskette** command on the image server.

Q: How do I make an autoinstall CD?

A: Run the **mkautoinstallcd** command on the image server.

Q: How do I make an autoinstall punch card?

A: Run the **mkautoinstallpunchcard** command on the image server - deprecated :)

Q: How does the autoinstall client software know where to get the image?

A: When the autoinstall client software (BOEL) is booted, if the LAST_ROOT append parameter has been used, then BOEL attempts to mount and read a local.cfg file from the device specified in LAST_ROOT (Aka: the device that was root on the machine before it was rebooted). BOEL then tries to read a local.cfg file from the floppy drive (if one exists). Finally, BOEL makes a DHCP request, if it still doesn't have enough information to connect to the network. Once it knows it's IP address, it resolves this to it's host name, then retrieves an autoinstall script from the image server based on it's hostname (Ie: hostname.sh).

Q: How does the netboot system know where to get the image?

A: See "How does the autoinstall client software know where to get the image?"

Q: How do I edit files on the initial ramdisk?

A: In SystemImager 3.0.x, the initial ramdisk is a cramfs filesystem. The following example shows you how to open it, edit a file, and create a new ramdisk. You might want to edit this file if you want to change things in the process prior to the execution of the .master script.

1. gunzip < initrd.img > /tmp/initrd
2. mkdir /mnt/tmp
3. mount /tmp/initrd /mnt/tmp -o loop
4. mkdir /tmp/initrd.new

5. cp -a /mnt/tmp/* /tmp/initrd.new
6. umount /mnt/tmp
7. cd /tmp/initrd.new
8. make your edits here - vi etc/init.d/rcS, for example
9. mkcramfs /tmp/initrd.new /tmp/newinitrd
10. gzip -9 < /tmp/newinitrd > /tmp/initrd.img

Q: When I pass options from dhcp (option-100, etc), the client appears to get and try to use a hexadecimal number instead. How do I make it pass a dotted-quad IP address instead?

A: The hexadecimal address is actually the hexadecimal representation of your IP address (you can verify this with the gethostip command). This is normally a quoting issue. Add quotes around the IP address in the configuration file.

Q: I've got netbootmond running, but it isn't working. Why?

A: In order for netbootmond to do it's thang, you must have the rsync daemon running: "/etc/init.d/systemimager-server-rsyncd start".

Q: How do I configure my server to net boot ia64 clients?

A:

1. Install tftp (tftp-hpa >= 0.28 is recommended) on your boot server.
2. Configure inetd or xinetd to enable tftp.
   - To configure inetd, find the tftp entry in `/etc/inetd.conf` and change it to:

     tftp dgram udp wait root /usr/sbin/in.tftpd -v -v -v -s /tftpboot

     Change "/usr/sbin/in.tftpd" to be the full path to your tftp server, if you installed it in a different directory.

     The -v's aren't strictly required but make the tftp server more verbose, which makes it easier to diagnose problems.

     Finally, send a HUP signal to inetd (this causes it to reload its configuration file). # killall -HUP inetd
   - To configure xinetd, change:

```
   service tftp
{
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -s /home/tftp
        disable                 = no
}
```

to:

```
service tftp
{
        socket_type             = dgram
        protocol                = udp
        wait                    = yes
        user                    = root
        server                  = /usr/sbin/in.tftpd
        server_args             = -s /tftpboot -r blksize
        disable                 = no
}
```

Finally, send a USR2 signal to xinetd (this causes it to reload its configuration file).

3. Configure your DHCP server so that it provides boot information to the client. Be careful when setting up your DHCP server - if it is set to hand out dynamic addresses and is located on a public subnet, it may give bogus information to other machines on the network, possibly destroying data on those machines. It is recommended that you use a private subnet for doing network installs. If possible, you should also configure your DHCP server to only answer requests from known hosts based on the MAC address.

   Add an entry for the boot client in /etc/dhcpd.conf

   ```
   host mcmuffin {
           hardware ethernet 00:30:6e:1e:0e:83;
           fixed-address 10.0.0.21;
           filename "elilo.efi";
   }
   ```

4. Copy elilo.efi from an IA-64 machine to your tftpboot directory and make them world readable. This file is usually found in a subdirectory under /boot/efi or in /usr/lib/elilo. It can also be found in the elilo package in IA64 distributions.

   You also must create an elilo.conf file in your tftpboot directory. A sample one is provided in /usr/share/doc/systemimager-doc/examples, or you can type in the one below.

5. Edit /tftpboot/elilo.conf:

   ```
   #
   # Sample elilo.conf for netbooting ia64 systemimager clients
   #
   # Inside your tftp directory you may also want to do this:
   #
   #   mkdir -p ia64/standard
   #   cp /usr/share/systemimager/boot/ia64/standard/* ia64/standard/
   #
   default=systemimager

   image=ia64/standard/kernel
       label=systemimager
       initrd=ia64/standard/initrd.img
       root=/dev/ram
   ```

```
append="vga=extended ramdisk_blocksize=4096 console=tty0"
#
# Uncomment APPEND line below, and comment out APPEND line above, to use
# both monitor (tty0) and first serial port (ttyS0) as console at the
# same time.
#
# NOTE: Be sure your serial port speed is appropriate (57600, 9600, etc.)
#
#append="vga=extended ramdisk_blocksize=4096 console=tty0 console=ttyS0,9600n8"
read-only
```

If ABCDEFGH is the client's IP address in hex, elilo.efi will use the first one of the following files that it finds as its configuration file:

- ABCDEFGH.conf
- ABCDEFG.conf
- ABCDEF.conf
- ...
- A.conf
- elilo.conf

You can use the ipcalc utility, which is available in the syslinux package, to calculate the hex representation of an IP address in dotted quad form.

6. Configure the client to support TFTP booting.
   a. Boot to EFI
   b. Enter the Boot option maintenance menu
   c. Add a boot option
   d. Press return on the line saying "Load file [Acpi/.../Mac()]"
   e. Call the entry Netboot or something similar
   f. Save and exit, Netboot is now available in the boot menu.

Q: How do I set up my autoinstall clients so that the console is available via the serial port?

A: In SystemImager 2.1 and later, mkautoinstallcd and **mkautoinstalldiskette** support an -append option, allowing you to specify additional options for the autoinstall kernel, including serial console options. For example: **mkautoinstallcd -out-file autoinstall.iso -append "console=ttyS0"**

Early versions of SystemImager require additional changes, as cited below:

> Making SystemImager work with a serial console
> Michael S. Fischer, <michael@auctionwatch.com>
> Last modified: 01/04/26 19:20:27
>
> The current version of SystemImager as of this writing (1.4.1)
> does not support PXE booting of clients with serial console
> support. This document describes how to rebuild the kernel and

initrd such that serial console support is possible.

Step 1: Download the sources

Retrieve the source tarball from
http://prdownloads.sourceforge.net/systemimager/va-systemimager-source-1.4.1.tar.bz2
Save it and unpack it to /tmp on your boot server.

```
# bzcat va-systemimager-source-1.4.1.tar.bz2 | tar -C /tmp -xf -
```

Step 2: Prepare the kernel

```
# cd /tmp/va-systemimager-source-1.4.1
# bzcat other_source_and_patches_used_in_this_release.tar.bz2 | tar xf -
# cd other_source_and_patches_used_in_this_release
# cd linux-2.2.18+reiserfs+raid+aic7xxx+VM
```

Now edit the .config file in this directory. Search for a line
that reads:

```
# CONFIG_SERIAL is not set
```

Change this line to read:

```
CONFIG_SERIAL=y
CONFIG_SERIAL_CONSOLE=y
```

Then build the kernel:

```
# make clean && make bzImage
```

Step 3: Install the kernel

Next, you'll need to place the kernel in /tftpboot and
/tftpboot/X86PC/UNDI/linux-install:

```
# cp arch/i386/boot/bzImage /tftpboot/kernel
# cp arch/i386/boot/bzImage /tftpboot/X86PC/UNDI/linux-install/kernel
```

Step 4: Fix the initrd

The initrd, or initial RAM disk containing the root filesystem, needs
to be unpacked, mounted, and have its contents edited as follows:

```
# cp /tftpboot/initrd.gz /tmp
# cd /tmp && gunzip initrd.gz
# mkdir /mnt2
# mount /tmp/initrd /mnt2 -o loop
# cd /mnt2/dev
# rm -f console
```

```
# mknod -m 622 console c 5 1
# mknod -m 600 ttyS0 c 4 64

You should now unmount the initrd, recompress
it, and return it to /tftpboot:

# cd /tmp
# umount /mnt2
# gzip -9 initrd
# cp initrd.gz /tftpboot
# cp initrd.gz /tftpboot/X86PC/UNDI/linux-install


Step 5: Edit syslinux.cfg

The final step is to configure pxelinux to pass the "console="
arguments to the kernel.  Following is a suitable syslinux.cfg
file:

DEFAULT kernel
APPEND console=tty0 console=ttyS0,9600n8 vga=extended load_ramdisk=1
prompt_ramdisk=0 initrd=initrd.gz root=/dev/ram rw
DISPLAY message.txt
PROMPT 1
TIMEOUT 50

Edit the contents of /tftpboot/pxelinux.cfg/syslinux.cfg to reflect
the changes above.  Then, copy this file to
/tftpboot/X86PC/UNDI/linux-install/pxelinux.cfg/syslinux.cfg.
```

Q: Why do installations take so much longer when done using a serial console?

A: The serial console itself often causes the bottleneck. By default, each filename is printed to the console as it is copied from the server. Because serial consoles typically have very low bandwidth, installations may be delayed while the system waits for the names to be printed to the console. You can use the -no-listing option to **getimage** and **mkautoinstallscript** to turn off verbosity.

Q: How can I change the mount options used for the tmpfs filesystem used by BOEL during an autoinstall?

A: tmpfs filesystem mount options can be passed to the BOEL kernel as append parameters by pre-pending "tmpfs_" to the mount option. For example, adding "tmpfs_size=2G tmpfs_nr_inodes=2000" is translated by BOEL to the tmpfs mount options "size=2G,nr_inodes=2000". The following tmpfs mount options are supported: size, nr_inodes, nr_blocks, and mode. See "man mount" for details on these options.

Append parameters can be added with the mkautoinstallcd and mkautoinstalldiskette --append option. For network boot loaders, or local disk boot loaders, you will need to edit the boot configuration file to add the appropriate append parameter(s).

Q: How do I configure my DHCP v3 server?

A: Luca Filipozzi provided details on configuring DHCP v3 servers:

If you are using DHCP v3 from ISC, you will need to use the following configuration directives to make use of option 140:

option systemimager-server code 140 = text; option systemimager-server "192.168.1.1";

The first line defines 'systemimager-server' as the name for option 140 and that it shall take as an argument a text string. The second line assigns a value (an IP address) to the name.

Q: Does the DHCP server have to be on the image server?

A: No. If you are using DHCP, you can use "option-140" and set its value to the IP address of the image server. If you use **mkdhcpstatic** to configure your `dhcpd.conf` file, it will ask you for the IP address of your image server and add the appropriate entry for you.

Because this is not the official use for option-140, work is being done to either get an official number assigned or use a number from the private number range.

Q: With which distributions does SystemImager work?

A: SystemImager is designed to work with _any_ distribution. Post imaging configuration is handled by System Configurator, which uses a "footprinting" technique to identify the style of system configuration files used, and to configure networking, boot, and similar information accordingly. If you find a distribution that SystemImager does not work with, please file a bug report.

Q: How do I add a driver for a special card to the autoinstall kernel?

A: If you have hardware that requires a driver that was not included in the standard flavor boot package, you can build a custom boot package. Unlike earlier (pre-2.9) releases, you will need more than a new kernel binary. However, also unlike earlier releases, you can have multiple boot packages installed simultaneously on your system.

The following instructions describe how to create a new boot package. Use these steps to add a network driver or block device driver that can either be a module or statically linked:

Boot packages are tightly linked to the version of SystemImager you are using. If you make customizations to a boot package for SystemImager 3.0.0, you must forward port those changes when you move to SystemImager 3.2.0.

Creating a custom boot package. If the driver you need to add can be built as a module, you can add it directly onto the autoinstall ramdisk. Use this method if you need to add a driver for the network interface you plan to install over and the driver is only available as a module. This is also the suggested method for adding a module for a device that failed to be discovered at install time. In other cases, you can drop in a patch to be added to the kernel build tree or make modifications to the `.config` file used to build the kernel.

1. Download the source tarball for the release of SystemImager you are using from http://systemimager.org/.
2. Extract the source tarball and cd into the top level of the source tree.

```
# tar xvfj systemimager-3.0.0.tar.bz2
# cd systemimager-3.0.0
```

3. To add a patch to the kernel, drop your patch file in the patches subdirectory of the top level build tree. Follow the naming convention described in the README.patches file in the patches/ subdirectory.
4. If you need to modify the .config file for your kernel (e.g. to turn on a driver), you can find it in the patch directory named linux.<arch>.config. (If you added a patch that includes a new driver, you should be prompted to turn that on in Step 9 of these instructions - the 'make binaries' step).
5. If you are building a binary module from external source to be included in the autoinstall ramdisk, you need the linux source tree to build against. You can generate a source tree in the src/linux directory by running **make patched_kernel-stamp**. Follow the instructions that came with your module source to compile your module against this tree.
6. If you created a binary module, you can copy it into the initrd_source/my_modules directory. Add instructions to the `INSMOD_COMMANDS` file in order to have your module loaded at install time. See comments in that file for details.
7. Edit the `FLAVOR` file in the top level of the source tree. Choose an alphanumeric string that describes your modifications.

```
echo "mygigabitnic" > FLAVOR
```

8. From the toplevel of the source tree, run **make binaries**. See the README file for details on building prerequisites.
9. Run **make install_binaries** to copy the binaries built in the previous step into the appropriate place in /usr/share/systemimager/boot directory. You now have an additional flavor to choose from when creating autoinstall media.
10. File a bug asking for this driver to be added to future releases.

Q: Do I have to do anything to prepare a client from which I will get an image?

A: Yes, you should install the systemimager-client package. If this package is already installed, simply run the **prepareclient** command prior to running **getimage** from the image server.

You should also add any software, configure any files, and do any tweaking to customize the system to your specifications.

Q: Can I use the autoinstalldiskette or autoinstallcd on more than one machine?

A: Yes. The autoinstall media is generic and can be on any machine you want to autoinstall.

Q: How do I push an image to a client?

A: Starting with version 1.5, you can use the **pushupdate** command, which essentially logs into each client and executes the **updateclient** command.

Q: How do I pull an image to a client?

A: If you ran **mkdhcpserver** to configure your dhcp information, and if you answered all the questions you were asked when you did ran **getimage**, including the hostnames and IP addresses, then all you have to do is boot your client with any one of the following three forms of autoinstall media:

1. *autoinstalldiskette* - takes longer to boot, and floppies are often quite volatile
2. *autoinstallcd* - takes slightly less time to boot and is more durable, but you have to have a CD burner and clients that can read CD-R's)
3. *network boot* - boot time is dramatically, but this method requires PXE capable network cards in the clients and additional server-side configuration.

See the entries for **mkautoinstallcd** and **mkautoinstalldiskette** in the command reference chapter in this manual for more information.

Q: How does an autoinstall client know which image to install?

A: In order to better understand the answer, begin by reading the steps the autoinstall client goes through:

1. Boots off the autoinstallmedia
2. Gets an IP address from DHCP
3. Determines the IP address of the image server via DHCP
4. Requests a hosts file from the image server
5. Finds its hostname in the hosts file based on its IP address
6. Requests a script from the image server based on its hostname (for example: `www237.sh`)
7. Executes this script.

The script in question is typically a soft link pointing at the `$image.master` script that was dynamically created when you ran **getimage**. This script explicitly states which image to pull from the image server. Open it and take a look.

These scripts and the `$image.master` script can be found in `/var/lib/systemimager/scripts`.

Q: How do I upgrade to a new version of SystemImager?

A: If you have SystemImager 1.0 or later, you can simply install the new version over the older one. Using a packaged version (.rpm,.deb) is recommended to prevent cruft build-up on your system.

If you have a pre-1.0 version then you can't simply install a new version on top of it. Depending on the version changes, the following may work for you, but it is not guaranteed. If you want to try this method, please check with the `CHANGE.LOG` document to find out what has changed.

Move your images directory out of the SystemImager directory hierarchy and back up your `/etc/rsyncd.conf` file and any other configuration files you may have changed. Then install SystemImager as if you were installing it for the first time, after which you can move your images directory and `/etc/rsyncd.conf` file back.

Q: What if I want to assign static IPs to my clients?

A: You can. **getimage** will ask you if you want to assign static IPs.

Q: I want to use DHCP to assign static IPs to my clients, but I don't want to have to enter my 1000 mac addresses manually. What can I do?

A: SystemImager comes with the **mkdhcpstatic** utility. As you boot your client systems, the DHCP server will assign addresses sequentially. By initially booting your systems in the order you want them to receive their IP addresses, you can ensure that they get the IP address you want them to have.

After booting your systems, run **mkdhcpstatic**. It will re-write your `/etc/dhcpd.conf` file, associating each client's MAC address with its host name. You should then restart your dhcpd daemon. Subsequently, each time your clients request an IP address via DHCP, they will always be assigned their appropriate static IP address.

Note: The client's hostname is used, instead of an explicit IP address, so that you simply have to change the `hosts` file on the DHCP server (or DNS, NIS, etc.) to change the IP address that that client recieves.

Note: Assigning static IP addresses by DHCP is the author's preferred method for administering IP on a large number of systems.

Q: What kind of performance can I expect?

A: Ole Holm Nielsen, Department of Physics, Technical University of Denmark reports:

In our SystemImager installation, we can install 18 clients simultaneously with 1.8 GB images in 6 minutes. Please see The NIFLHEIM SystemImager Page (http://www.fysik.dtu.dk/CAMP/Niflheim/systemimager.html). Our server has Gigabit network, 2 GB of RAM, dual Intel Xeon 2.4 GHz, whereas the clients have Intel P4 and 100 Mbit Ethernet.

James Braid reports:

From a Celeron 700/512Mb server over 100Mbit ethernet, we manage to do a ~1Gb image in about 7 - 10 min. The disks are 5x 120Gb Seagate Barracuda V in one LVM set (non striped), with a ReiserFS filesystem.

Q: How do I update an image on the image server?

A: There are two ways to update an image on the image server:

1. Make the changes to one of your clients and run the **getimage** again.

- You can specify the same image name, in which case the current image will be updated (only changes are pulled across).

- Or you can specify a new image name and have a form of revision control. (This method is highly recommended)

Note: Every time **getimage** is run, it recreates the `$image.master` script. If you have customized your `$image.master` script, be sure to save it before running **getimage** again.

2. Modify the files directly. You can simply cd into the appropriate image directory and edit the files there, or (recommended) you can **cd** into the image directory and run **'chroot . sh'**. This will change your working root directory to the root of the image you want to manipulate. You can then run **rpm** and other commands on the image and not have to worry about getting confused and damaging the image server. When you are done, simply type **exit** and you will be returned to your normal shell.

Q: How do I update a client to match an image?

A: Once you have updated an image on the image server, you can then update your clients to reflect it. (You do not need to do a complete re-autoinstall.) You will find the command, **updateclient**, on your clients, which takes as its parameters the name of the image server and the name of the image you want to update the client to. Run **updateclient -help** to get more information about this command.

Use the revision control method recommended in the "How do I update an image on the image server?" FAQ to bring your production environment back to a known state after doing an **updateclient** to a test image (i.e. do an **updateclient** to the last working image).

The file `/etc/systemimager/updateclient.local.exclude` on your clients is used to exclude files and directories from being updated by the **updateclient** command. You can modify it to suit your own environment.

Q: What is the `updateclient.local.exclude` file used for?

A: It is used by the **updateclient** command. See the "How do I update a client to match an image?," FAQ for more information.

Q: How do I build a new "front-end" server?

A: Another way of stating the question is as follows, "We are in the process of building a new 'front-end' server for our Linux cluster. We'd like to try using SystemImager as our node cloning software if possible.

I've read through the SystemImager FAQ and found the following:

> All of the clients for a particular image should have an identical hardware configuration. They should at least have the same hard drive(s) and the same ethernet card(s).

This is a problem because our front-end from which the image will be captured has a SCSI hard drive and a Gigabit ethernet card. On the other hand, our 64 compute nodes have IDE disk drives and fast ethernet cards. Is it impossible to build a new "front-end" server?

No. But you will need to customize a bit.

Once you have run **getimage**, you will need to edit the `conf.modules` in the image on the image server to load the appropriate module for your ethernet card.

vi /var/lib/systemimager/images/$imagename/etc/conf.modules

Your entry should look like this:

```
alias eth0 tulip
```

You will need to edit the following files in your image:

```
/var/lib/systemimager/images/$imagename/etc/fstab
/var/lib/systemimager/scripts/$imagename.master
/var/lib/systemimager/images/$imagename/etc/lilo.conf
```

Use your editor's search function to change every instance of "sda" to "hda."

Proceed as normal.

Q: How can I use SystemImager to update a small set of files? For instance, I apply a security patch and I want all boxes to reflect that change.

A: Use the **updateclient** command on the client.

1. Choose one of the following methods to update the image on the server:
   a. apply the patch to the image directly
   b. apply the patch to a client and then do another **getimage** specifying the same imagename (won't take long and will update the image)
   c. apply the patch to a client and then do another **getimage** specifying a different imagename. This is preferred as it allows for revision control.
2. Run **updateclient** on the clients that you want to update. Execute **updateclient -help** to get the syntax.

Q: Is there a log file where autoinstall client status is kept?

A: Yes. SystemImager logs can be found on the image server in the directory `/var/log/systemimager`

Q: What other software is SystemImager based on?

A: SystemImager is mostly written in Perl, and makes use of the following software:

- **busybox**
- **bc**
- **devfsd**
- **ISC dhcp**
- **discover**
- **dosfstools**
- **e2fsprogs**
- **jfsutils**
- **xfsprogs**
- **Linux kernel**
- **parted**
- **pxelinux**
- **rsync**
- **syslinux**
- **raidtools**
- **reiserfsprogs**
- **systemconfigurator**
- **uClibc**

Also be sure to take a look at System Installation Suite (SIS), which includes SystemInstaller, SystemImager, and System Configurator. SystemInstaller is a tool that allows you to install images directly to a SystemImager image server. System Configurator, which is also used by the standard SystemImager release, performs configuration of target machine uniquenesses such as IP addresses, network cards, and initial RAM disks needed to boot clients after installation.

Q: I modified my `autoinstallscript.conf` file to configure my non-software RAID image to install as a software RAID image, but the install failed. Why?

A: When you convert a non-software RAID image to a software RAID image, you must manually configure the software RAID information. Modify the image directly or make the changes in an override directory if you want to maintain the original image.

Following are keypoints for manually configuring a computer to use software RAID on Linux. For more detailed information, please refer to The Software RAID HOWTO (http://unthought.net/Software-RAID.HOWTO/) at http://unthought.net/Software-RAID.HOWTO/.

- Drivers in the kernel: You must have the software RAID drivers compiled for the kernel your client will boot from when it finishes installing.

- Device files: For software RAID to work, you must have the proper device files in your image. These device files have names like /dev/md0, /dev/md1, etc. If these device files do not exist, cd into the ./dev directory of your image, then copy and paste the following command to create them. If the software RAID devices are already created, running this command causes no problems.

  ```
  for I in 0 1 2 3 4 5 6 7 8 9; do sudo mknod -m 660 md$I b 9 $I; done
  ```

- /etc/raidtab: You must create this file and/or customize it to meet your needs.
- Other files: You must also modify the files used by your boot loader (lilo, grub, etc.). See your boot loader's documentation for more information on its configuration file format. A quick tip: in lilo.conf, instead of using /dev/md as for the boot= parameter, use the software RAID device that holds your /boot/ directory (where the kernel lives). For example, if /dev/md1 will be mounted on /boot/, then specify boot=/dev/md1.

After manually configuring your image to use software RAID, then make your changes to the autoinstallscript.conf file. Changes that you need to make to this file may include the following. See "**man autoinstallscript.conf**" for more information.

- In the **<disk>** section, add **raid** to the comma separated list of flags for each partition that will participate in a RAID array.
- In the **<fsinfo>** section, set **real_dev** to the software RAID device that will hold each filesystem.

You can now run **mkautoinstallscript** to create a new, software RAID-enabled autoinstallscript for your image.

Q: What's an override directory?

A: An override directory is a directory that gets copied over to your target machines after the main image is transferred. All contents in the override directory are copied over to the root of the target machine's new filesystem. All file attributes are replicated, including directories, permissions, and ownership. This allows you to "over-ride" files in the image. Override directories live in **/var/lib/systemimager/overrides/**.

You don't have to use an override directory, but it can often be useful. For example, you could have two different autoinstall scripts for the same image: one for SCSI machines, and one for IDE machines. You can have your /etc/fstab and /etc/lilo.conf files in override directories with names such as my_image-ide and my_image-scsi.

Simply edit the master autoinstall script and change the overrides variable to include the appropriate override directory. For example, you could change **OVERRIDES="my_image"** to **OVERRIDES="my_image-ide"**.

If using the same overrides on all of your machines, you don't have to change the autoinstall script. Simply put the files that you want to override in the overrides directory that has the same name as your image, and proceed.

You can also use multiple override directories, which are used in the order that you specify them -- each directory overriding the previous directories. You can use this methodology in a highly complex

environment where slight variations exist between several classes of machines but where they all start with the same base image. For example, **OVERRIDES="my_image-ide web_app"**.

Q: How do I expand a filesystem?

A: See "How do I change the size of a partition?"

Q: How do I change the size of a partition?

A:

1. Open your autoinstallscript.conf file in your favourite text editor.

   
   The default autoinstallscript.conf file created by **prepareclient** lives in the `/etc/systemimager` directory in your image.

2. Find the **<disk>** section where **dev** is set to the disk that holds the partition you want to change.
3. Find the **<part>** entry where **num** is the number of the partition in question.
4. Change **size** to the new partition size, keeping in mind that if the size you specify is not sufficient to hold the files stored there, the autoinstall will fail.

   NOTE: Each **<disk>** section can use either MB (megabytes) or % (percentages) to specify partition sizes. See **man autoinstallscript.conf** for more information.

5. Run **mkautoinstallscript** to create a new autoinstall script using the new parameters.

   NOTE: By default, **mkautoinstallscript** uses the autoinstallscript.conf file located in your image's ./etc/systemimager directory. See **man mkautoinstallscript** and **man autoinstallscript.conf** for more information.

Q: How do I change the filesystem(s) that my target machine(s) will use?

A:

1. Make sure that the kernel in your image supports the filesystem(s) you want to use.
2. Open your autoinstallscript.conf file in your favorite text editor.

   NOTE: The default autoinstallscript.conf file created by **prepareclient** lives in the ./etc/systemimager directory in your image.

3. Find the **<fsinfo>** entry where **mp** (mount point) is set to the filesystem that you want to change.
4. Change **fs** to the filesystem you want to use. See **man autoinstallscript.conf** for a list of supported filesystems.

   You must understand the capabilities of your chosen filesystem. Depending on which one you use, you may also need to change the options used to mount the filesystem, which are set by the **options** entry. If you choose unsupported options, your autoinstall may fail.

   In all known cases to date, it has not been necessary to change the **fs** entries in the **<disk>** section when changing filesystem types. The **fs** entries in the **<disk>** section don't actually determine the

filesystem that will be created on those partitions, but the **parted** tool that SystemImager uses for creating disk partitions requires that argument.

5. Run **mkautoinstallscript** to create a new autoinstall script using the new parameters. By default, **mkautoinstallscript** uses the autoinstallscript.conf file located in the ./etc/systemimager directory in your image. See **man mkautoinstallscript** and **man autoinstallscript.conf** for more information.

Q: How do I change the disk type(s) that my target machine(s) will use?

A:

1. Make sure that the kernel in your image has drivers for the disk types you want to use.
2. Open your autoinstallscript.conf file in your favorite text editor.

   
   The default autoinstallscript.conf file created by **prepareclient** lives in the `/etc/systemimager` directory in your image.

3. Find the <**disk**> entry where **dev** (disk device) is set to the disk that you want to change.
4. Change **dev** to the device file for the disk you want to use. For example, changing from SCSI to IDE can be as simple as changing **dev="/dev/sda"** to **dev="/dev/hda"**.
5. Change the **real_dev** entries in the <**fsinfo**> section to match the new devices.
6. Run **mkautoinstallscript** to create a new autoinstall script using the new parameters.

   
   By default, **mkautoinstallscript** uses the `autoinstallscript.conf` file located in the `/etc/systemimager` directory in your image. See **man mkautoinstallscript** and **man autoinstallscript.conf** for more information.

7. Modify the boot loader configuration in your image to work with your target disk configuration.

Q: Can I use a single image across machines with differing disk or partition configurations?

A: Yes. Customize a separate autoinstallscript.conf file for each configuration type, and use **mkautoinstallscript** to create a seperate master autoinstallscript for each configuration.

If a particular configuration's boot device will be different from the boot device in the image, you also need to put a customized copy of your boot loader's configuration file(s) in an override directory.

See the following "Frequently Asked Questions" for more information:

- How do I change the size of a partition?
- How do I change the filesystem(s) that my target machine(s) will use?
- How do I change the disk type(s) that my target machine(s) will use?
- What's an override directory?

Chapter 7. Developing SystemImager®

7.1. Design Goals

The design goal for SystemImager was to create a tool that was both easy and fast to use, allowing system administrators to perform incremental upgrades that minimized system disruptions and provided software independent of any packaging system. The specific requirements follow:

- Images should be pulled from a working system.
- Unattended installs are a must.
- The unattended install system has to be able to repartition the destination drive(s).
- System administrators who don’t completely understand the tool should be able to use it.
- The install should be easy and quick so that it can be useful immediately without a lot of site-specific customization.
- Images should be stored as normal files to allow for incremental upgrades, as opposed to "dd" style block level images of physical disks.
- The SystemImager software should be independent of any and all packaging systems (such as RPM) in order to easily accommodate different distributions.
- SystemImager should be able to store multiple images for different types of systems and for revision control.
- A mechanism should be available to let unattended install clients know which image to install.
- An installed client should be able to update itself to a new or modified image.
- SystemImager should have a command line interface that can easily be wrapped with a GUI.

7.2. Development of SystemImager Architecture

Minimal system requirements were a top priority for SystemImager, which began as a series of utilities written as shell scripts. However, shell scripts were inadequate as SystemImager matured and its utilities became more complex. Perl, part of most Linux installs and possessing minimal system requirements, then replaced shell scripts. Using Perl, developers have been able to generate cleaner, more advanced code.

SystemImager architecture was designed to be open to modification at every level. The protocol for transferring files during installs and updates is currently `rsync(1)`, but the modular code easily allows drop-in replacements as is appropriate. All unicast file transfer mechanisms, including `rsync(1)` are implemented in a "pull" fashion, which is generally considered to be superior to a "push," because it is much easier to monitor the state of the receiving system prior to and during the file transfers. However, multicast may be an option in the future and may need to be implemented as a "push."

7.3. Version Strings

SystemImager releases use an X.Y.Z numbering scheme similar to that used by the Linux kernel. X is a major feature release revision incremented to show significant architecture changes. Y is a minor release revision, incremented to show the addition of new features. Z is a bug-fix revision. If Y is an odd number, the release is a development release. If Y is even, it is a stable release.

Using this scheme, you can see that 2.1.3 is a development release because 1 is odd. You can assume that the only changes between 2.0.3 and 2.0.4 were bug fixes, because only the last digit was incremented.

7.4. Using CVS

SystemImager is maintained in CVS on SourceForge. To submit patches for future releases, generate them against the code in CVS. There are typically two active branches at any given time - the current stable branch and the current development branch. If you are fixing a bug that exists in the current stable release, generate a patch against the stable branch. If you are adding a new feature or fixing a bug that is only in the development branch, use the development branch.

For example, the current stable release is 3.0.0, while the current stable branch is v3_0_x. At some point, the v3_0_x branch will be tagged as v3_0_1, which will be the 3.0.1 release. Except for rare occasions, HEAD is the development branch. Since the current stable branch is 3.0.x, the next development releases will be 3.1.x. At some point, we will put a v3_1_0 tag on HEAD, and that will be 3.1.0.

Instructions for accessing the CVS repository can be found at http://sourceforge.net/cvs/?group_id=259 (http://sourceforge.net/cvs/?group_id=259). There's also a Web interface that allows you to view the files in the repository and look at differences between versions.

7.5. Submitting Patches

Following are suggested guidelines for submitting packages for a future release that make it easier to incorporate the changes.

- Patches should be in unified diff form. If your changes are localized to one file, you can create a patch using a command such as:

  diff -u getimage.orig getimage

  If your changes are in multiple files, you can create a patch using a command such as:

  diff -urN systemimager.orig/ systemimager/
- Review your patches to make sure you aren't changing other things. Even if the additional changes are just whitespace changes, they can make the patch more difficult to review and reduce the chances it can cleanly apply against a slightly different tree.
- It's easiest if patches are against the code in an up-to-date tree of the code branch you're changing. For example, if you've fixed a bug in 3.0.0, check out the v3_0_x branch and create your diff against that tree to prevent you from fixing a bug that has already been fixed and committed or from creating a patch against a version that is different enough that it won't apply cleanly to the current code.

- Please submit patches to SystemImager code that do not affect other components of the System Installation suite to systemimager-devel@lists.sourceforge.net. For patches relevant to the rest of the System Installation Suite, submit to sisuite-devel@lists.sourceforge.net. Use the latter if unsure which one to use.

7.6. Tour of the Source Tree

7.6.1. Makefile Overview

Currently, the make system is a single Makefile that includes make snippets from other places in the tree. This setup allows you to easily depend on other make rules but also means all rules must exist in a flat namespace. Notice that the rules and variables in each .rul file use a common prefix. For example, every rule in the parted.rul file begins with "parted_", and every variable begins with "PARTED_". Another thing to keep in mind when working within this system is that all paths are relevant to the toplevel. We may change to a recursive make style build system at some point.

7.6.2. Third Party Software

The SystemImager build system builds a mini Linux distribution called boel that runs on clients during an autoinstall (See "Additional Installation Information" in Chapter 3 for more information on boel). The `initrd.img` file contains a subset of boel that is small enough to fit on a 1.44MB floppy. After the client has booted and brought up networking, it receives the `boel_binaries.tar.gz` file and extracts it over the root of the filesystem, providing an additional set of commands, libraries, and kernel modules that may be needed during the installation.

Most of the software in boel is built from third party source and is not included in the CVS tree, although it is packaged up with released source tarballs. The build system must therefore be able to access these third party source tarballs to extract them and build the necessary binaries.

Each .rul file for a third party package contains a target for the source tarball. The rules for these targets run the **getsource** script in the toplevel tools/ directory. **getsource** will look around your system for a source tarball (/usr/src, and the toplevel of the build tree), or will, as a last resort, use **wget** to download the source tarball from the Internet. It is highly recommended that you copy any downloaded source to one of those locations so that you only download it once. Although it may not cost you anything to download source repeatedly, there is an associated cost for the people running the servers from which you're downloading. Running **make clean** attempts to remove everything except these third party tarballs, while **make distclean** removes everything including the third party tarballs.

7.6.3. Boel Libraries

On i386, uClibc http://uclibc.org is used within the initrd in order to keep it small enough to fit on a floppy. For the initrds for other architectures and for the binaries in `boel_binaries.tar.gz` on every architecture, the mklibs script copies over the required libraries from the build system. If appropriate pic libraries are found (e.g., the libc6-pic package in Debian), mklibs can take advantage of them and reduce the libraries to only the functions needed by the binaries.

Binaries that will be dynamically linked against need to be built using uClibc wrappers, which is why the PATH variable gets explicitly set for all the initrd binaries.

Chapter 8. Command Reference

This chapter contains the man pages for each of the SystemImager commands.

SI_ADDCLIENTS

<dannf@dannf.org>

dann frazier

November 10, 2002

Name

`si_addclients` — assign clients to an image

Synopsis

si_addclients `[-help][-version][-host` *`HOSTNAME`*`][-domainname` *`DOMAINNAME`*`][-host-range` *`X-Y`*`][-script` *`SCRIPT`*`][-ip-range` *`IP-IP`*`][-interactive` *`YES|NO`*`]`

DESCRIPTION

This manual page documents briefly the **si_addclients** command.

si_addclients is a tool for mapping clients to a SystemImager image. There are three sections to the si_addclients utility:

1. The first section is where you define what a hostname in your cluster looks like, and define a range of hostnames that you will associate with an image in the next section. (You should already have created the image for these clients to use by this point). Hostnames are of the form *`prefixnumber.domainname`* For example, if you specify "bogus.net" as your domainname, "node" as your base hostname, a starting number of 1 and an ending number of 10, you will have defined the following hosts:
   node1.bogus.net
   node2.bogus.net
   node3.bogus.net
   node4.bogus.net
   node5.bogus.net
   node6.bogus.net
   node7.bogus.net
   node8.bogus.net
   node9.bogus.net

node10.bogus.net

Currently, **si_addclients** does not support hostnames with leading zeros (e.g. node09.bogus.net, instead of node9.bogus.net), but this will likely be added in a future release. One reason for making the no-padded-zeros method the default behavior is flexible cluster expansion. For example, if you had initially chosen node01 - node80 as your cluster range, and later decide to add 25 more nodes to your cluster, you will have to renumber all of your nodes to add another leading zero (i.e. node001 - node105).

2. Section 2 is where you specify the image you want to assign to the hosts defined in section 1.
3. Section 3 gives you the option of automatically generating (or just updating) entries in your hosts files. During an autoinstallation, a client will request a hosts file in order to determine its hostname from its IP address. An autoinstall client will attempt other methods for determining it's hostname, including DNS, so this is not always necessary. Once an entry for a host has been added, there is no need to complete this section for that host again (but doing so is harmless).

OPTIONS

A summary of options is included below. By default, if an insufficient set of options is specified, **si_addclients** will fall back to an interactive mode. See `-interactive` below if you wish to alter this behavior.

`-help`

Show summary of options

`-version`

Display version and copyright information.

`-host` *`HOST`*

Base host name of the client(s) to be autoinstalled. Will be combined with "-host-range" to derive actual host names.

`-domainname` *`DOMAINNAME`*

If this option is used, *`DOMAINNAME`* will be appended to the client host name(s).

`-host-range` *`X-Y`*

Number range used to create a series of host names based on the `-client` option. For example, "-client www -range 1-3" will cause si_addclients to use www1, www2, and www3 as host names. If leading zeros are used, then host names will be padded with zeros. For example, "-client www -range 01-03" produces host names www01, www02, and www03. Padding hostnames with zeros is not recommended.

`-ip-range` *IP_0-IP_1*

The range of IP addresses to assign to your autoinstall clients. Where IP_0 is the first IP address in the range and IP_1 is the last ip address in the range.

`-interactive` *YES|NO*

This program will go interactive by default if domainname, host, host-range, and script are all specified.

If you specify YES here, then it will go interactive, even if all of these values are specified.

If you specify NO here, then it will not go interactive, even if it is missing some of the required values.

`-script` *IMAGE*

Master autoinstall script name (typically the imagename). Don't include the path or the .master extension.

SEE ALSO

getimage(8), **mkdhcpserver**(8), **mkdhcpstatic**(8)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<dannf@dannf.org>> for the Debian GNU/Linux system (but may be used by others).

AUTOINSTALLSCRIPT.CONF

<dannf@dannf.org>

dann frazier

November 10, 2002

Name

`autoinstallscript.conf` — specify partition and filesystem options

DESCRIPTION

This manual page documents briefly the `autoinstallscript.conf` file.

The `autoinstallscript.conf` file contains information used by SystemImager to create an autoinstall script. The resultant autoinstall script is used to clone Linux systems.

Disk Section

This section is used to create a disk label and partitions on each of your autoinstall client's disks. This section is indicated with the "disk" tag.

- disk - Partition information about a single disk.
- dev - The device file that represents the disk in question.
- label_type - The kind of disk label used by this disk. Currently, this can be either "msdos" or "gpt".
- unit_of_measurement - The unit of measurement is used with "size" tags to specify the size of each partition. This can be set to MB or %.
- part - Start of information about a new partition.
- num - The minor device number for the partition (partition number).
- size - The size of the partition. The unit used here should correspond to the "unit_of_measurement" setting for the entire disk.

  Size can also hold the special value of "*". This is used to indicate that the specified partition should be growable to the end of the disk. If the last partition is a primary partition, then only that partition should have "*" specified. If the last partition is a logical partition, then both the logical partition, and the extended partition that contains it, should have "*" specified.
- p_type - Partition type. This can be primary, extended, or logical for msdos labelled disks. This can only be primary for gpt labelled disks, as gpt disk labels only support primary partitions.
- id - Partition id. This is a hex number used to specify special information about a partition. It is usually not needed and is optional. In special circumstances where it is needed, **sfdisk** is used to change the partition id to the specified type.
- p_name - This is the name of a partition. Gpt partition tables support naming partitions. If you don't know what this is, you probably don't want to mess with it. A value of "-" works great (gets ignored)!
- flags - A comma separated list (no spaces) of flags used to specify particulars about a partition. Possible values are: boot, hidden, raid, lvm, and lba.

Filesystem Section

This section is used to create filesystems on each of the used partitions, and to create an fstab file on the autoinstall client. Each line of this section is indicated with the "fsinfo" tag.

- fsinfo - Start of a new line of information for the fstab file and/or for filesystem creation information.

- line - This is used to indicate the order that lines should be put into the fstab file on the autoinstall client.
- comment - Indicates a comment that should be included in the fstab file, but is not used for any other processing.
- real_dev - The first field in the fstab file. This is the device where the filesystem resides. It is put into the generated fstab file and is the device used to mount filesystems on the finished machine when it reboots.
- mount_dev - Usually the first field in the fstab file will be an actual device file like "/dev/sda1", but may sometimes look like "LABEL=/boot" or "UUID=c1b9d5a2-f162-11cf-9ece-0020afc76f16". In the case that it is a LABEL or UUID, instead of a real device, we use mount_dev to specify the information that will be put in the fstab file to mount the filesystem after the autoinstall client is imaged. LABEL and UUID information is stored as part of the filesystem, not as part of the partition. See `fstab(5)` for more information. mount_dev will only exist if a LABEL or UUID is used.
- mp - The second field in the fstab file. Mount point. This is the name of the directory where the filesystem should be mounted. See `fstab(5)` for more information.
- fs - The third field in the fstab file. This is where you specify what filesystem you want created on the partition specified on "mount_dev" (or "real_dev"). Valid filesystem types supported by SystemImager are: ext2, ext3, msdos, reiserfs, vfat, jfs, xfs
- mkfs_opts - Certain filesystems may require additional information to properly create a filesystem. Currently, this is only used by msdos and vfat filesystems. Valid values for msdos and vfat filesystems are "-F 12", "-F 16", or "-F 32" to specify the FAT size. If you don’t know what you need, or don’t care, you can leave this blank and defaults will be used.
- options - The fourth field in the `fstab` file. Options needed when mounting the filesystem. If you don’t know that you need any special options, just use "default". See `fstab(5)` for more information.
- dump - The fifth field in the fstab file. This should be a number that is used by **dump(8)**. If you don’t know what to put here, just use "0". See `fstab(5)` for more information.
- pass - The sixth field in the fstab file. This is used by the fsck program to determine the order in which filesystem checks are done at boot time. See `fstab(5)` for more information.
- format - Used to tell SystemImager to not create a filesystem on a partition. If this is set to "no", the filesystem will not be created. If it is absent, or contains any other value, then a filesystem will be created.

BOEL Section

This section contains miscellaneous settings used by BOEL (Brian’s Own Embedded Linux), the embedded linux that performs an install on autoinstall clients.

- devstyle - Tells the autoinstallscript to mount the devfs filesystem in the image prior to running System Configurator, which prepares the client’s boot loader. Many boot loaders need to access the disks at this point, and systems that use devfs won’t have any device files in the freshly copied image for the boot loader to use (at this point, BOEL has done a chroot into the root of the filesystem on the freshly installed client).

LIMITATIONS

If you make a change to this file that could affect the boot process, for example, changing your root device, then you must also modify your boot loader's configuration file. This file can be modified either directly in the image, or in an override directory. For more information on override directories, see the full manual in /usr/share/doc/systemimager-doc/ or at http://systemimager.org/documentation/.

SEE ALSO

si_mkautoinstallscript(8), `fstab(5)`

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<`dannf@dannf.org`>> for the Debian GNU/Linux system (but may be used by others).

SI_CPIMAGE

<dannf@dannf.org>

dann frazier

November 10, 2002

Name

`si_cpimage` — make a copy of a SystemImager image.

Synopsis

si_cpimage `[-help][-version][-verbose][-force][-directory` *`PATH`*`][-server` *`HOSTNAME`*`][-ssh-user` *`USERNAME`*`][`*`SOURCE_IMAGE`*`][`*`DESTINATION_IMAGE`*`]`

DESCRIPTION

This manual page documents briefly the **si_cpimage** command.

si_cpimage makes a copy of a SystemImager image.

OPTIONS

A summary of options is included below.

`-help`

Show summary of options

`-version`

Display version and copyright information.

`-verbose`

Explain what is being done.

`-force`

Don't ask for confirmation before overwriting the destination image or master autoinstall script (if they exist).

`-directory` *`PATH`*

The full path and directory name where you want this image to be stored. The directory bearing the image name itself will be placed inside the directory specified here.

`-server` *`HOSTNAME`*

Hostname or IP address of the imageserver from which you want to copy the image. (Defaults to "localhost".)

`-ssh-user` *`USERNAME`*

Username for ssh connection to the imageserver. Only needed if a secure connection is required.

SEE ALSO

lsimage(8), **mvimage**(8), **rmimage**(8)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<`dannf@dannf.org`>> for the Debian GNU/Linux system (but may be used by others).

SI_GETIMAGE

<dannf@dannf.org>

dann frazier

November 10, 2002

Name

`si_getimage` — pull an image from a golden client

Synopsis

si_getimage [OPTION...] {`-golden-client` *HOSTNAME*} {`-image` *IMAGENAME*}

DESCRIPTION

This manual page documents briefly the **si_getimage** command.

si_getimage is a program that pulls an image to the image server. The source for this image can be either a golden client or an image stored on another image server

OPTIONS

A summary of options is included below.

`-help`

Show summary of options

`-version`

Display version and copyright information

`-golden-client` *HOSTNAME*

Hostname or IP address of the "golden" client

`-image` *IMAGENAME*

Where IMAGENAME is the name to assign to the image you are retrieving. This can be either the name of a new image if you want to create a new image, or the name of an existing image if you want to update an image. If you are using the `-server` option, the same imagename is used for both the source and destination.

`-ssh-user` *`USERNAME`*

Username for ssh connection to the client. Only needed if a secure connection is required.

`-log` *`STRING`*

Quoted string for log file format. See the `rsyncd.conf`(5) man page for options.

`-quiet`

Don't ask any questions or print any output (other than errors). In this mode, no warning will be given if the image already exists on the server.

`-directory` *`PATH`*

The full path and directory name where you want this image to be stored. The directory bearing the image name itself will be placed inside the directory specified here.

`-exclude` *`PATH`*

Don't pull the contents of *`PATH`* from the golden client. *`PATH`* must be absolute (starting with a "/").

To exclude a single file use:

-exclude */DIRECTORYNAME/FILENAME*

To exclude a directory and it's contents use:

-exclude */DIRECTORYNAME/*

To exclude the contents of a directory, but pull the directory itself use:

-exclude "*/DIRECTORYNAME/**"

`-exclude-file` *`FILE`*

Don't pull the *`PATH`*s specified in *`FILE`* from the golden client. See the rsync(1) for details on creating this file.

`-update-script` *`YES|NO`*

Update the $image.master script? Defaults to NO if -quiet. If not specified you will be prompted to confirm an update.

`-no-listing`

Don't show each filename as it is copied over during install. This is useful for times when your console device is slow (e.g. serial console), and is the bottleneck of your installation.

The following options affect the autoinstall client after autoinstalling.

`-ip-assignment` *`METHOD`*

Where METHOD can be:

- static_dhcp -- A DHCP server will assign the same static address each time to clients installed with this image. The DHCP server in this case also assigns the corresponding host name. It is possible therefore, when using static dhcp, to change a machine's hostname and IP address by simply changing one value in the dhcpd.conf file. Some may find this desirable. Also see the **mkdhcpstatic** command.

- dynamic_dhcp -- A DHCP server will assign IP addresses dynamically to clients installed with this image. In dynamic dhcp, the IP address of a machine may change, but you want it to retain the same host name. Therefore the hostname is set by the machine itself. They may be assigned a different address each time.
- static -- The IP address the client uses during autoinstall will be permanently assigned to that client.
- replicant -- Don't mess with the network settings in this image. I'm using it as a backup and quick restore mechanism for a single machine.

`-post-install` *`ACTION`*

ACTION can be:

- beep -- Clients will beep incessantly after successful completion of an autoinstall.
- reboot -- Clients will reboot themselves after successful completion of an autoinstall.
- shutdown -- Clients will halt themselves after successful completion of an autoinstall.

SEE ALSO

addclients(8), **mkdhcpserver**(8), **mkdhcpstatic**(8),

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<`dannf@dannf.org`>> for the Debian GNU/Linux system (but may be used by others).

SI_LSIMAGE

`<dannf@dannf.org>`

dann frazier

November 10, 2002

Name

`si_lsimage` — list the images on an image server

Synopsis

si_lsimage `[-help][-version][-server` *`HOSTNAME`*`][-ssh-user` *`USERNAME`*`]`

DESCRIPTION

This manual page documents briefly the **si_lsimage** command.

si_lsimage lists the available images on a given image server.

OPTIONS

A summary of options is included below.

`-help`

Show summary of options

`-version`

Display version and copyright information.

`-server` *`HOSTNAME`*

Hostname or IP address of the imageserver. Defaults to localhost.

`-ssh-user` *`USERNAME`*

Username for ssh connection to the client. Only needed if a secure connection is required.

SEE ALSO

getimage(8)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<`dannf@dannf.org`>> for the Debian GNU/Linux system (but may be used by others).

SI_MKAUTOINSTALLCD

<dannf@dannf.org>

dann frazier

November 10, 2002

Name

`si_mkautoinstallcd` — generate a bootable iso image for autoinstalling clients

Synopsis

si_mkautoinstallcd `[-help][-version][-arch` *`ARCH`*`][-out-file` *`PATH`*`][-flavor` *`FLAVOR`*`][-kernel` *`PATH`*`][-initrd` *`PATH`*`][-append` *`STRING`*`][-quiet]`

DESCRIPTION

This manual page documents briefly the **si_mkautoinstallcd** command.

si_mkautoinstallcd creates an iso image which can be used to boot an autoinstall client and initiate an autoinstall.

OPTIONS

A summary of options is included below.

`-help`

Show summary of options

`-version`

Display version and copyright information.

`-arch` *`ARCH`*

Create a CD image for autoinstallclients of this architecture. Defaults to the architecture of the host on which **si_mkautoinstallcd** is being run.

`-out-file` *`PATH`*

Name of the file that will hold the resulting ISO image. (Not to worry, this will only be about 3 MB in size.)

`-flavor` *`FLAVOR`*

The flavor of the boot package to use. If this option is not specified, you will be asked to choose a flavor from the available list interactively.

`-initrd` *`PATH`*

Specify an alternate initial ramdisk to use. This is rarely useful, but may be necessary in some cases (e.g. needing to add more special files in /dev). See the SystemImager manual for instructions on creating a custom initial ramdisk.

`-append` *`STRING`*

This option allows you to pass options to the autoinstall kernel. For example, to tell the kernel to use ttyS3 as the serial console:

-append "console=ttyS3"

`-quiet`

Don’t print any output, just provide an appropriate exit code.

SEE ALSO

si_mkautoinstalldiskette(8)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<`dannf@dannf.org`>> for the Debian GNU/Linux system (but may be used by others).

SI_MKAUTOINSTALLDISKETTE

<`dannf@dannf.org`>

dann frazier

November 10, 2002

Name

`si_mkautoinstalldiskette` — creates a bootable floppy for autoinstalling clients

Synopsis

si_mkautoinstalldiskette [-help][-version][-quiet][-floppy *DEVICE*][-out-file *PATH*][-flavor *FLAVOR*][-kernel *PATH*][-initrd *PATH*][-append *STRING*][-config *PATH*][-ssh-key *PATH*]

DESCRIPTION

This manual page documents briefly the **si_mkautoinstalldiskette** command.

si_mkautoinstalldiskette creates a floppy disk which can be used to boot an autoinstall client and initiate an autoinstall.

OPTIONS

A summary of options is included below.

`-help`

Show summary of options

`-version`

Display version and copyright information.

`-quiet`

Don’t print any output, just provide an appropriate exit code. (requires -floppy)

`-floppy` *`DEVICE`*

The 1.44MB floppy drive device containing the disk to format. If not specified, /dev/fd0 is assumed. This command understands "/dev/fd0" through "/dev/fd7". (if you specify -floppy, this command will run non-interactively)

`-out-file` *`FILE`*

Create a 1.44MB floppy image in *`FILE`*. **dd** can be used to write this image to a floppy later. This is useful for some systems that allow you to boot nodes using a "virtual floppy" file that is retrieved via TFTP (some PXE loaders and some remote management boards).

`-kernel` *`PATH`*

Specify an alternate kernel to use. This is useful if you have clients that have network cards or disk controllers that are not supported by the default kernel provided by SystemImager. See the SystemImager manual for instructions on creating a custom autoinstall kernel.

`-flavor` *`FLAVOR`*

The flavor of the boot package to use. If this option is not specified, you will be asked to choose a flavor from the available list interactively.

`-initrd` *`PATH`*

Specify an alternate initial ramdisk to use. This is rarely useful, but may be necessary in some cases (e.g. needing to add more special files in /dev). See the SystemImager manual for instructions on creating a custom initial ramdisk.

`-append` *`STRING`*

This option allows you to pass options to the autoinstall kernel. For example, to tell the kernel to use ttyS3 as the serial console:

-append "console=ttyS3"

`-config` *`FILE`*

Where *`FILE`* contains all the settings necessary for the client to set it's hostname and configure it's networking information without DHCP. This file is copied to `/local.cfg` on the autoinstalldiskette.

`-ssh-key=`*`FILE`*

Where *`FILE`* is the SSH2 id_dsa or id_rsa private key of the user account that the autoinstall client will use to connect to the imageserver. It is usually found in the user's `.ssh` directory.

To use this private key, you must also add the corresponding public key to this user's `.ssh/authorized_keys` file on the imageserver. The public key is usually called `.ssh/id_rsa.pub` or `.ssh/id_dsa.pub`. The `.ssh/authorized_keys` must be readable only by this user. (**chmod 600 .ssh/authorized_keys**)

SEE ALSO

si_mkautoinstallcd(8)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<`dannf@dannf.org`>> for the Debian GNU/Linux system (but may be used by others).

SI_MKAUTOINSTALLSCRIPT

`<dannf@dannf.org>`

dann frazier

November 10, 2002

Name

`si_mkautoinstallscript` — create a SystemImager autoinstallscript

Synopsis

si_mkautoinstallscript `[--help][--version][--quiet][--no-listing][--image` *`IMAGENAME`*`][--script` *`SCRIPTNAME`*`][--force` *`[YES|NO]`*`][--config-file` *`FILE`*`][--ip-assignment` *`METHOD`*`][--post-install` *`ACTION`*`]`

DESCRIPTION

This manual page documents briefly the **si_mkautoinstallscript** command.

si_mkautoinstallscript creates an autoinstallscript for a SystemImager image.

OPTIONS

A summary of options is included below.

`--help`

Show summary of options

`--version`

Display version and copyright information.

`--quiet`

Don’t print any output, just provide an appropriate exit code.

`--no-listing`

Don’t show each filename as it is copied over during install. This is useful for times when your console device is slow (e.g. serial console), and is the bottleneck of your installation.

`--image` *`IMAGENAME`*

Where IMAGENAME is the name of the image for which you want to create a new autoinstall script.

`--script` *`SCRIPTNAME`*

Where SCRIPTNAME is the base name to use for the resultant autoinstall script. If not specified, SCRIPTNAME defaults to IMAGENAME. This is most often used when you want to have multiple

autoinstallscripts for a single image; i.e. when you have clients with different partitioning and/or filesystem schemes that use the same image (same file set).

`--force` *`[YES|NO]`*

Update the $image.master script? Defaults to NO if -quiet. If not specified you will be prompted to confirm an updated.

`--config-file` *`FILE`*

Use FILE to get autoinstallscript configuration information. Defaults to `./etc/systemimager/autoinstallscript.conf` in the image directory.

The following options affect the autoinstall client after autoinstalling.

`--ip-assignment` *`METHOD`*

Where METHOD can be:

- static_dhcp -- A DHCP server will assign the same static address each time to clients installed with this image. The DHCP server in this case also assigns the corresponding host name. It is possible therefore, when using static dhcp, to change a machine's hostname and IP address by simply changing one value in the dhcpd.conf file. Some may find this desirable. Also see the **si_mkdhcpstatic** command.
- dynamic_dhcp -- A DHCP server will assign IP addresses dynamically to clients installed with this image. In dynamic dhcp, the IP address of a machine may change, but you want it to retain the same host name. Therefore the hostname is set by the machine itself. They may be assigned a different address each time.
- static -- The IP address the client uses during autoinstall will be permanently assigned to that client.
- replicant -- Don't mess with the network settings in this image. I'm using it as a backup and quick restore mechanism for a single machine.

`--post-install` *`ACTION`*

ACTION can be:

- beep -- Clients will beep incessantly after successful completion of an autoinstall.
- reboot -- Clients will reboot themselves after successful completion of an autoinstall.
- shutdown -- Clients will halt themselves after successful completion of an autoinstall.

SEE ALSO

getimage(8), **autoinstallscript.conf**(5)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<dannf@dannf.org>> for the Debian GNU/Linux system (but may be used by others).

SI_MKBOOTSERVER

<dannf@dannf.org>

dann frazier

November 10, 2002

Name

`si_mkbootserver` — Configure a server so that other machines can boot from it over the network.

Synopsis

si_mkbootserver
`[-f][--interface=`*`INTERFACE`*`][--localdhcp=`*`[y|n]`*`][--kernel=`*`PATH`*`][--initrd=`*`PATH`*`][--tftpdir=`*`PATH`*`][--p`

DESCRIPTION

This manual page documents briefly the **si_mkbootserver** command.

si_mkbootserver is a tool that assists in configuring a network boot server (currently just PXE servers). It runs in both interactive and non-interactive modes.

OPTIONS

A summary of options is included below.

`-f`

Turn off interactive mode. All other options must be specified for this to work.

`--interface=`*`INTERFACE`*

The interface the PXE daemon will listen on.

`--localdhcp=[y|n]`

This specifies whether or not the DHCP server will run locally or on another machine.

`--kernel=PATH`

The full path to the kernel that clients should boot.

`--initrd=PATH`

The full path to the initial ramdisk that clients should use.

`--tftpdir=PATH`

The full path to the directory from which the tftp server will serve files.

`--pxelinux=PATH`

The full path to the pxelinux bootloader. This is a part of the syslinux package.

SEE ALSO

makedhcpserver(8), **makedhcpstatic**(8)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<dannf@dannf.org>> for the Debian GNU/Linux system (but may be used by others).

SI_MKDHCPSERVER

<dannf@dannf.org>

dann frazier

November 10, 2002

Name

`si_mkdhcpserver` — create a DHCP server configuration file for use with SystemImager

Synopsis

si_mkdhcpserver

DESCRIPTION

This manual page documents briefly the **si_mkdhcpserver** command.

si_mkdhcpserver is an interactive utility for creating a dhcpd configuration file for use with SystemImager. It generates entries commonly needed in PXE configurations, and defines option-100. option-100 is passed to autoinstall clients in order to direct them to the image server they should use.

The PXE configuration that si_mkdhcpserver creates will work for most i386 clients, but will not work for machines of other architectures. Please see the SystemImager manual for instructions on configuring network boot for other supported architectures.

OPTIONS

si_mkdhcpserver currently accepts no options.

SEE ALSO

si_mkdhcpstatic(8)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<dannf@dannf.org>> for the Debian GNU/Linux system (but may be used by others).

SI_MKDHCPSTATIC

<dannf@dannf.org>

dann frazier

November 10, 2002

Name

`si_mkdhcpstatic` — create a DHCP server configuration file with static entries

Synopsis

si_mkdhcpstatic

DESCRIPTION

This manual page documents briefly the **si_mkdhcpstatic** command.

si_mkdhcpstatic is a utility for generating a a dhcpd configuration file for use with SystemImager. Static entries are created for IP addresses that are currently dynamically leased. This utility should be used after autoinstalling clients with an image that uses the dhcp_static method. This method is assigned to the image by **getimage**.

OPTIONS

si_mkdhcpstatic currently accepts to options.

SEE ALSO

si_mkdhcpserver(8), **getimage**(8)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<`dannf@dannf.org`>> for the Debian GNU/Linux system (but may be used by others).

SI_MVIMAGE

`<dannf@dannf.org>`

dann frazier

November 10, 2002

Name

`si_mvimage` — Move a SystemImager image.

Synopsis

si_mvimage `[-help] [-version] [-verbose] [-force] [-directory` *`PATH`*`] [`*`SOURCE_IMAGE`*`] [`*`DESTINATION_IMAGE`*`]`

DESCRIPTION

This manual page documents briefly the **si_mvimage** command.

si_mvimage moves a SystemImager image.

OPTIONS

A summary of options is included below.

`-help`

Show summary of options

`-version`

Display version and copyright information.

`-verbose`

Explain what is being done.

`-force`

Don’t ask for confirmation before overwriting the destination image or master autoinstall script (if they exist).

`-directory` *`PATH`*

The full path and directory name where you want this image to be stored. The directory bearing the image name itself will be placed inside the directory specified here.

SEE ALSO

lsimage(8), **cpimage**(8), **rmimage**(8)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<dannf@dannf.org>> for the Debian GNU/Linux system (but may be used by others).

SI_PREPARECLIENT

<dannf@dannf.org>

dann frazier

November 10, 2002

Name

si_prepareclient — prepare a golden client's image for retrieval

Synopsis

si_prepareclient
[-version][-h][-help][-n][-no-rsyncd][-q][-quiet][-r][-rpm-install][-yes]

DESCRIPTION

This manual page documents briefly the **si_prepareclient** command.

si_prepareclient is a program that prepares a golden client's image for extracting by an image server.

OPTIONS

A summary of options is included below. Only one option is taken.

-version

Display version and copyright information

`-h`, `-help`

Show summary of options

`-n`, `-no-rsyncd`

Do not start the rsync daemon

`-q`, `-quiet`

Run silently. Return an exit status of 0 for success or a non-zero exit status for failure

`-r`, `-rpm-install`

This is only used when building an RPM

`-yes`

Answer yes to all yes/no questions

SEE ALSO

updateclient(8)

More details can be found in the *SystemImager manual.*

AUTHOR

This manual page was compiled by dann <<`dannf@dannf.org`>> for the Debian GNU/Linux system (but may be used by others).

SI_PUSHUPDATE

`<dannf@dannf.org>`

dann frazier

November 10, 2002

Name

`si_pushupdate` — update the image of one or more remote clients

Synopsis

si_pushupdate [-help][-version][-client *HOSTNAME*][-range *X-Y*][-domain *DOMAINNAME*][-clients-file *FILE*][-concurrent-processes *N*][-ssh-user][-continue-install][-reboot]

Options for updateclient: (The following options will be passed on to the **updateclient** command)

[-server *HOSTNAME*][-image *IMAGENAME*][-directory *DIRECTORY*][-nolilo][-autoinstall][-configure-from *DEVICE*][-reboot][-log *"STRING"*]

DESCRIPTION

This manual page documents briefly the **si_pushupdate** command.

si_pushupdate is a tool for updating the image on one or more client nodes. It relies on the availability of ssh as a transport layer for communicating with client nodes.

OPTIONS

A summary of options is included below.

`-help`

Show summary of options

`-version`

Display version and copyright information

`-client HOSTNAME`

Host name of the client you want to update. When used with -continue-install, the name of the client to autoinstall.

`-range X-Y`

Number range used to create a series of host names based on the -client option. For example, "-client www -range 1-3" will cause si_pushupdate to use www1, www2, and www3 as host names. If no -range is given with -client, then si_pushupdate assumes that only one client is to be updated.

`-domain DOMAINNAME`

If this option is used, DOMAINNAME will be appended to the client host name(s).

`-clients-file FILE`

Read host names and images to process from FILE.

`-concurrent-processes N`

Number of concurrent process to run. If this option is not used, N will default to 1.

`-ssh-user`

Username for ssh connection to client.

`-continue-install`

Hosts should be treated as autoinstall clients waiting for further instruction.

`-server HOSTNAME`

Hostname or IP address of the imageserver. (-imageserver is a deprecated option and will go away soon.)

`-image IMAGENAME`

Image from which the client should be updated

`-directory DIRECTORY`

Absolute path of the directory to be updated (defaults to "/")

`-nolilo`

Don’t run lilo after update completes. (lilo is always run unless specified)

`-autoinstall`

Autoinstall this client the next time it reboots. (can’t be run with -nolilo)

`-configure-from DEVICE`

Only used with -autoinstall. Stores the network configuration for DEVICE in the /local.cfg file so that the same settings will be used during the autoinstall process.

`-reboot`

Reboot client after update completes

`-log "STRING"`

Quoted string for log file format. See the rsyncd.conf man page for options.

SEE ALSO

updateclient(8)

More details can be found in the *SystemImager manual.*

AUTHOR

This manual page was compiled by dann <<dannf@dannf.org>> for the Debian GNU/Linux system (but may be used by others).

SI_RMIMAGE

<dannf@dannf.org>

dann frazier

November 10, 2002

Name

si_rmimage — remove a SystemImager image.

Synopsis

si_rmimage [-help][-version][-verbose][-force][*IMAGE*]

DESCRIPTION

This manual page documents briefly the **si_rmimage** command.

si_rmimage removes a SystemImager image from an image server.

OPTIONS

A summary of options is included below.

-help

Show summary of options

-version

Display version and copyright information.

`-verbose`

Explain what is being done.

`-force`

Continue on error (default is to exit on error).

SEE ALSO

lsimage(8), **cpimage**(8), **mvimage**(8)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<dannf@dannf.org>> for the Debian GNU/Linux system (but may be used by others).

SI_MKRSYNCD_CONF

<dannf@dannf.org>

dann frazier

November 10, 2002

Name

`si_mkrsyncd_conf` — generates rsyncd.conf from a rsync stub directory

Synopsis

si_mkrsyncd_conf

DESCRIPTION

si_mkrsyncd_conf is a utility for generating a rsyncd.conf file for SystemImager from an rsync stub directory. The location of the rsync stub directory and the rsyncd.conf file are specified in `/etc/systemimager/systemimager.conf`

Stub files in the rsync stub directory can fit into any of the following categories, specified by the leading numbers in the filename:

- 10*`description`* - comments and global definitions
- 20*`ARCH`* - architecture specific entries
- 30*`DESCRIPTION`* - comments and definitions for the image section
- 40*`IMAGENAME`* - entry for *`IMAGENAME`*
- 70*`DESCRIPTION`* - reserved for local use

All other numeric ranges are reserved for future use.

OPTIONS

No options are currently accepted.

AUTHOR

This manual page was written by dann <<`dannf@dannf.org`>> for the Debian GNU/Linux system (but may be used by others).

SEE ALSO

systemimager.conf(5)

SI_UPDATECLIENT

`<dannf@dannf.org>`

dann frazier

November 10, 2002

Name

`si_updateclient` — update the image running on the local client machine to match that of an image on an image server

Synopsis

si_updateclient [-help][-version][-server *HOSTNAME*][-image *IMAGENAME*][-override *OVERRIDE*...][-directory *DIRECTORY*][-no-bootloader][-autoinstall][-flavor *FLAVOR*][-configure-from *DEVICE*][-ssh-user *USERNAME*][-reboot][-dry-run][-log *"STRING"*]

DESCRIPTION

This manual page documents briefly the **si_updateclient** command.

si_updateclient is a program that brings the local client's image up to date with an image on a image server.

OPTIONS

A summary of options is included below.

`-help`

Show summary of options

`-version`

Display version and copyright information

`-server` *HOSTNAME*

Hostname or IP address of the imageserver. Replaces deprecated `-imageserver` option.

`-image` *IMAGENAME*

Image from which the client should be updated

`-override` *OVERRIDE*

Override module(s) from which to copy additional files. Override directories are useful when you have multiple types of clients that share a subset of files. The image can contain this subset, and you can have an override module for each client type which contains the additional files required for that client type. If the same file exists in an override module and the image, the files in the override module will overwrite those in the image.

You can specify multiple override modules. They will be copied over in the same order the are specified on the command line.

`-directory` *`DIRECTORY`*

Absolute path of the directory to be updated (defaults to "/")

`-nobootloader`

Don't run the bootloader (lilo, elilo, grub, etc) after update completes. (**si_updateclient** always runs the bootloader unless specified).

`-autoinstall`

Autoinstall this client the next time it reboots. This option must be used if you want your partition scheme to be updated to match the partition scheme of the image you are updating too. This is required if the partition scheme of the image you are updating from differs from the partition scheme of the image you are currently running, as your system may be left unbootable otherwise. This option conflicts with `-nolilo`.

Updates that use this option cannot take advantage of rsync's efficient updating mechanism. This process also requires two reboots. This makes `-autoinstall` updates more consuming.

`-flavor` *`FLAVOR`*

The boot flavor to be used for doing an autoinstall (assumes -autoinstall).

`-configure-from` *`DEVICE`*

Only used with -autoinstall. Stores the network configuration for DEVICE in the /local.cfg file so that the same settings will be used during the autoinstall process.

`-ssh-user` *`USERNAME`*

Username for ssh connection from the client. Only needed if a secure connection is required.

`-listing`

List available images on imageserver. This option is now deprecated in favor of **lsimage**.

`-reboot`

Reboot client after update completes

`-dry-run`

Don't actually modify anything, just show what would be done.

`-log` *`STRING`*

Quoted string for log file format. See the `rsyncd.conf`(5) man page for options.

SEE ALSO

si_pushupdate(8), **si_lsimage**(8)

More details can be found in the *SystemImager manual*.

AUTHOR

This manual page was compiled by dann <<dannf@dannf.org>> for the Debian GNU/Linux system (but may be used by others).