SystemImager® v4.1.6 Manual

Andrea Righi
Ari Jort
Austin Gonyou
Ben Spade
Brian Elliott Finley
Curtis Zinzilieta
Dann Frazier
Denise Walters
Greg Pratt
Jason R. Mastaler
Josh Aas

SystemImager® v4.1.6 Manual

by

Andrea Righi

Ari Jort

Austin Gonyou

Ben Spade

Brian Elliott Finley

Curtis Zinzilieta

Dann Frazier

Denise Walters

Greg Pratt

Jason R. Mastaler

Josh Aas

Published $Date: 2008-02-03 16:15:06 +0100 (Sun, 03 Feb 2008) $

SystemImager was created by Brian Elliott Finley. The current maintainer and team lead of the project is Andrea Righi.

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

SystemImager® is a registered trademark of Brian Finley.

Linux® is a registered trademark of Linus Torvalds.

Debian® is a registered trademark of Software in the Public Interest, Inc.

Red Hat® is a registered trademark of Red Hat, Inc. in the United States and other countries.

Solaris® is a registered trademark of Sun Microsystems, Inc.

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 automates GNU/Linux installs, software distribution, and production deployment.

One key feature of SystemImager is that it is distribution-agnostic and is able to support heterogeneous hardware. This allows the deployment of any kind of GNU/Linux distribution (standard or even customized) to any kind of target machine. The main goal of the project is to make deployment of large numbers of computers easy. Typical environments include computer labs and render farms, but it has proven particularly popular in clustered computing environments, such as grid and high performance computing.

Another design feature that facilitates GNU/Linux distribution and hardware independence is that SystemImager works with file based (rather than block based) system images. An image is stored as a directory hierarchy of files representing a comprehensive snapshot of a machine, containing all the files and directories from the root of that machine's file system. Images can be acquired in multiple ways, including retrieval from a sample system (golden client), or by direct generation on the SystemImager server using third-party tools.

The standard method of image creation involves cloning of a pre-installed machine, the golden-client. In this way, the user can customize and tweak the golden-client's configuration according to his needs, verify it's proper operation, and be assured that the image, once deployed, will behave in the same way as the golden-client. Incremental updates are possible by syncing an updated golden-client to the image, then syncing that image to deployed machines using the si_updateclient command. Images are hosted in a central repository on a server, called the image-server, and they can be distributed among the clients using different transports: rsync (the default), multicast (via Flamethrower1), SSL encrypted rsync (using a SSH tunnel), and via BitTorrent.

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

- Organizations involved in complex Linux-based grid-computing environments

- 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 SystemConfigurator (http://wiki.systemimager.org/index.php/System_Configurator) to custom configure autoinstall clients for specific distributions. Through SystemConfigurator, SystemImager supports all major GNU/Linux distributions and most others, including custom or in-house distributions. Using "footprints", SystemConfigurator 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, SystemConfigurator 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.

1.3.2. System Requirements

- Your image server must have enough disk space to hold the images to be installed on your client systems (the default directory used to store images is `/var/lib/systemimager/images`).

- All clients that will use the same image should have the same number of hard drive(s). 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 **si_mkautoinstallscript(8)** command to install the same image on clients with varying disk and/or filesystem 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 server. 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. Usually a running PXE daemon is not necessary if the image server uses a quite recent distributions, since all the recent versions of the DHCP daemon include the PXE functionalities.

- To properly setup a boot server it is strongly suggested the usage of the **si_mkbootserver(8)** tool, included in SystemImager.

1.4. Glossary of Terms

image

A live snaphot of a machine containing files and directories from the root of that machine's filesystem.

An image is a chroot-able filesystem, stored in `/var/lib/systemimager/images/$NAME`.

Examples:

- /var/lib/systemimager/images/RHEL4/
- /var/lib/systemimager/images/Ubuntu_7_04/
- /var/lib/systemimager/images/HPC_1.0/
- ...

override

Overrides can be used to manage differences from images. A typical use is to customize a "vanilla" image adding some additional packages, but they are commonly used to store and distribute configuration files by **si_pushoverrides(8)**.

Overrides are stored by default in /var/lib/systemimager/overrides/ (this directory can be changed in the configuration file **/etc/systemimager.conf**). All the files and directories defined inside an override are distributed exactly as they are, that means preserving *all* the data and also metadata: permissions, ownership, timestamps, etc.

Examples:

- /var/lib/systemimager/overrides/RHEL4/
- /var/lib/systemimager/overrides/Compute_config/
- /var/lib/systemimager/overrides/HPC_1.0_custom/
- ...

image server

A server that has all the images and overrides available for the installation.

client

A machine to be auto-installed with a (single) selected image.

golden client

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

transport

The protocol used to distribute images from the image server to the clients. Different approaches can be used, depending on the particular transport used: push / pull / p2p / ...

Examples: rsync, multicast, rsync over SSH, BitTorrent

autoinstall media

Media that is used to boot an autoinstall client to begin the autoinstall process. Autoinstall media can be a USB disk, a CDROM, the network (via PXE), or the local hard drive of the autoinstall client.

autoinstall script

One or more scripts associated with an image, each unique to a specific partitioning/filesystem/network configuration. The **si_getimage(8)** command creates an initial autoinstallscript, which can be regenerated later, possibly with different options, using the **si_mkautoinstallscript(8)** 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`

si_getimage(8)

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

si_prepareclient(8)

A command you must run on the golden client immediately prior to running **si_getimage(8)** on the image server. **si_prepareclient(8)** 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).

si_clusterconfig(8)

A tool to manage and show the SystemImager cluster topology. By this command is possible to tells your image server which image and overrides to install on the auto-install clients.

si_mkdhcpserver(8)

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

si_mkdhcpstatic(8)

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.

si_mkbootserver(8)

A utility that helps to setup and configure a network boot server for SystemImager.

si_updateclient(8)

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

si_pushoverrides(8)

A command to update and/or keep synchronized clients files, pushing the overrides defined by **si_clusterconfig(8)** from the image server.

Chapter 2. Installing SystemImager

2.1. How Does it Work?

SystemImager uses a centralized server, called *image server* that retrieves a golden client's entire system image and deploys it to any number of different 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 **si_getimage(8)** 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 or debootstrap, used to create a Debian base system from scratch. 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 RPM and DEB format. Official packages can be downloaded from http://www.systemimager.org/ (http://www.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.

An alternative scalable and reliable method to break the bandwidth and performance limits of the image server is the BitTorrent transport. With BitTorrent the upload bandwidth of the clients can be used to distribute the images among the nodes exploiting the advantages of the peer-to-peer networks. For more informations see http://wiki.systemimager.org/index.php/BitTorrent (http://wiki.systemimager.org/index.php/BitTorrent).

2.4. Installing an Image Server

The official packages of SystemImager are distributed in the file release system on SourceForge.net at this link: https://sourceforge.net/project/platformdownload.php?group_id=259 (https://sourceforge.net/project/platformdownload.php?group_id=259)).

SystemImager depends on the System Configurator package that is also available on SourceForge: https://sourceforge.net/project/showfiles.php?group_id=24006 (https://sourceforge.net/project/showfiles.php?group_id=24006)).

A quick way to download all the required packages is to run the sis-install script (replacing i386 with the architecture of your clients):

```
$ mkdir systemimager
$ cd systemimager
$ wget http://download.systemimager.org/pub/sis-install/install
$ chmod u+x install
$ ./install -v --download-only --tag stable --directory . \
> systemconfigurator \
> systemimager-client systemimager-common \
> systemimager-i386boot-standard systemimager-i386initrd_template \
> systemimager-server \
> systemimager-bittorrent systemimager-flamethrower
```

Use `./install --help` for more informations.

Install all the base packages in the image server. In RPM-based distributions run:

```
# rpm -ivh systemconfigurator-*.rpm \
> systemimager-common-*.rpm systemimager-server-*.rpm \
> systemimager-*initrd_template-*.rpm systemimager-*boot-standard-*.rpm
```

In Debian or Debian-like distributions run:

```
# dpkg -i systemconfigurator-*.deb \
> systemimager-common-*.deb systemimager-server-*.deb \
> systemimager-initrd_template-*.deb systemimager-boot-*-standard-*.deb \
```

If you want to use the *BitTorrent* transport install also the systemimager-bittorrent package. RPM package:

```
# rpm -i systemimager-bittorrent-*.rpm
```

DEB package:

```
# dpkg -i systemimager-bittorrent-*.deb
```

If you want to use the *Multicast* transport install also the systemimager-flamethrower package. RPM package:

```
# rpm -i systemimager-flamethrower-*.rpm
```

DEB package:

```
# dpkg -i systemimager-flamethrower-*.deb
```

This package requires udpcast (http://udpcast.linux.lu/source.html) and Flamethrower (http://freshmeat.net/projects/flamethrower), usually included in the common distributions.

If your clients have multiple architectures download also the appropriate systemimager-ARCHboot-standard and systemimager-ARCHinitrd_template in your image server.

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 3.9.6 release should have these packages:

- systemimager-boot-i386-standard_3.9.6-1_all.deb,

- systemimager-boot-x86_64-standard_3.9.6-1_all.deb,

- systemimager-boot-ppc64-ps3-standard_3.9.6-1_all.deb,

- ...

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 multiple boot flavors can be installed simultaneously.

To create a custom flavor perfectly compatible with the distribution you can exploit the UYOK (Use Your Own Kernel) feature. Standard flavors use a general purpose kernel that supports a lot of hardware components, but obviously they can't be able to support all the possible devices that could be present in a totally generic client (see also hardware with proprietary drivers). For this reason starting from 3.6.x the UYOK (UseYourOwnKernel) feature has been introduced. This feature allows you to use the same kernel that runs in your golden client to perform the installation in other clients. In this way it's possible to theoretically support all the hardware/components you could have. Simply if the kernel of your distribution is working well with your clients, it works well too to install them. For more informations see http://wiki.systemimager.org/index.php/UYOK.

2.5. Selecting A Machine To Use As A Golden Client

A golden client is a system manually installed and customized of which you want to make an image (clone).

2.6. 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.

2.7. Installing SystemImager Client Software on a Golden Client

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

1. Download and install the systemimager-client package.

    ```
    $ mkdir systemimager
    $ cd systemimager
    $ wget http://download.systemimager.org/pub/sis-install/install
    $ chmod u+x install
    $ ./install -v --download-only --tag stable --directory . \
    > systemconfigurator \
    > systemimager-client systemimager-common \
    > systemimager-i386initrd_template
    ```

Use `./install --help` for more informations.

Install all the base packages in the image server. In RPM-based distributions run:

```
# rpm -ivh systemconfigurator-*.rpm \
> systemimager-common-*.rpm systemimager-client-*.rpm \
> systemimager-*initrd_template-*.rpm
```

In Debian or Debian-like distributions run:

```
# dpkg -i systemconfigurator-*.deb \
> systemimager-common-*.deb systemimager-client-*.deb \
> systemimager-initrd_template-*.deb
```

2.8. Creating an Image from scratch

An alternative approach to create images is to install all the needed packages into a directory.

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 Images created with SystemInstaller are interchangeable with those created using the SystemImager tools.

You can even create an image using one of the common tools shipped with your distribution, instead of using SystemInstaller. The tool must be able to install a distribution into a directory (ex. Debootstrap for Debian, kickstart or yum for Red Hat, YaST for SuSE, etc.)

2.9. Upgrading SystemImager

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

2.9.1. Regenerating autoinstallscripts

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

si_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

```
# si_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/.

2.9.2. 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 si_mkrsyncd_conf(8) man page for details.

2.9.3. Regenerating boot media

Each time you upgrade Systemimager, you must also upgrade the boot media you use to boot the autoinstall system. Use **si_mkautoinstallcd** or **si_mkautoinstalldisk** to regenerate removable media.

2.9.4. 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 GNU/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 **si_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 **si_getimage** command in the next step; however, you can change these settings later by running the **si_mkautoinstallscript** command.

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

7. Run **si_clusterconfig -e** to define the groups of clients and to associate images and overrides to groups. The command **si_clusterconfig** is available only in SystemImager 3.9.4 or later. Otherwise you can always use the command **si_addclients** (see the next point) to associate the images to your clients.

8. This point can be skipped if **si_clusterconfig** has been used. Otherwise run **si_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.

9. Generate a boot media for your clients. There are four ways to boot the clients for auto-installation:

   - boot from network (PXE) - see **si_mkclientnetboot(8)**

   - boot from an auto-install CD - see **si_mkautoinstallcd(8)**

   - boot from an auto-install disk (USB drive or internal disk) - see **si_mkautoinstalldisk(8)**

   - boot from a running system - see **si_updateclient(8)**

10. Autoinstall the golden image on other machines using one (or more) generated boot media.

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 **si_prepareclient** as root. This will create various files in your /etc/systemimager directory that contain information on your disk partition scheme, filesystem types, etc. **si_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, the **rsync** daemon will be automatically stopped. In case of errors during the image retrieval be sure to manually deactivate the **rsync** daemon by killing the process or by simply rebooting the golden client. This **rsync** server will not start automatically on future reboots. In **rsync over SSH** mode the **rsync** communication is performed opening a SSH tunnel from the image server to the golden client.

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 **si_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 **si_mkdhcpstatic** command after your clients have booted and received an IP address. **si_mkdhcpstatic** will modify your /etc/dhcpd.conf file on the imageserver to include static entries for each of your hosts.

Alternately, you can pass hostname, imageserver, and networking information via installation parameters, in the form of VARIABLE=value. Installation parameters can be defined in /etc/systemimager/pxelinux.cfg/syslinux.cfg or can be passed as argument of **si_mkautoinstalldisk**, **si_mkautoinstallcd**, or **si_mkclientnetboot** using the **--append "STRING"** option. For a complete list of the available installation parameters see http://wiki.systemimager.org/index.php/Installation_Parameters (http://wiki.systemimager.org/index.php/Installation_Parameters).

Alternatively, if you are using a running system's hard drive as the boot media, you can run **si_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.

Example 3-1. Running si_mkdhcpserver

```
[root@imageserver]# si_mkdhcpserver
```

Example 3-2. Running si_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 **si_getimage,** you can abbreviate them to *-im* and *-ip*.

```
[root@server7]# si_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 **si_getimage** on the image server to pull the image from the golden client to the image server.

   With **si_getimage** the image server pulls all the files and directories from the root of the client's filesystem back to the image repository in `/var/lib/systemimager/images`.

   The basic syntax is: "**si_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 si_getimage**."

Example 3-3. Running si_getimage

```
[root@imageserver]# si_getimage -golden-client my-golden-client \
       > -image web_server_image_v1
```

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

7. Run **si_clusterconfig** to manage the groups of your clients in the SystemImager database.

    si_clusterconfig can be used also to show the defined groups with the list of clients that belong to
    each group and/or the image associated to each client or group of clients. In show-mode the
    command accepts as argument a list of hostnames, host-ranges and/or host-group, it resolves them in
    the equivalent list of hostnames and prints them to stdout. The edit-mode can be interactive (option
    -e) or batch (option -u). In interactive edit-mode si_clusterconfig opens an editor in your terminal
    that allows to modify the client group definitions and their properties using a XML syntax. In batch
    edit-mode it only parses the pre-defined XML configuration
    (`/etc/systemimager/cluster.xml`) and refresh the opportune SystemImager internal
    configuration files.

    Do not edit `/etc/systemimager/cluster.xml` directly. Always use **si_clusterconfig -e** to be
    sure that all the files needed by the installations process will be properly synchronized.

    Here is a well-commented example of a simple cluster configuration:

```
<?xml version='1.0' standalone='yes'?>
<!--

******************************** WARNING ****************************************

This file has been generated by si_clusterconfig(8), do not edit manually!

******************************** WARNING ****************************************

       This is the main configuration file to describe the topology of
       your clients and your image server informations.

       This file will be used by all the SystemImager commands to identify the
       logical groups of your clients and their specific configurations.

       See comments below for more details.
-->
<xml>
       <!-- The image server hostname. -->
       <master>master1</master>

       <!--
```

```
        This is the global name:  this name will be used to identify
        all the hosts defined in this file (the global supergroup).

        IMPORTANT: this is a mandatory entry!!!
-->
<name>all</name>

<!--
        This is the global override: all the files stored in this
        overrides will be pushed to all the clients.

        IMPORTANT: this is a mandatory entry!!! If you don't want to use
        it do not create the global override in
        /var/lib/systemimager/overrides/

        Multiple overrides can be specified as a list of multiple XML tags:

        <override>OVERRIDE_NAME_1</override>
        <override>OVERRIDE_NAME_2</override>
        ...
        <override>OVERRIDE_NAME_N</override>

        The OVERRIDE_NAME_1 ... OVERRIDE_NAME_N will be distributed
        preserving the same order as they appear in the XML file. This
        means that in case of file overlaps (more files in multiple
        overrides that should be distributed to the same target
        filename) the first hit wins. In this case OVERRIDE_NAME_1 is
        the most important and OVERRIDE_NAME_N is the least important.
-->
<override>all</override>

<!--
        Following there is an example of a "fake" group. The group
        name is "Local", it uses the image "Local", the override in
        /var/lib/systemimager/overrides/Local and it contains only
        the localhost server. Totally useless, but it's there to
        explain how it works... ;-)
-->
<group>
        <name>Local</name>
        <image>Local</image>
        <override>Local</override>
        <node>localhost</node>
</group>

<!--
        This is a group that contains two nodes: node001 and
        node002. The group is called "Login" and it uses the
        override:

        The clients node001 and node002 will be auto-installed using the
        image:
```

```
                /var/lib/systemimager/images/SuSE10

                And the override:

                /var/lib/systemimager/overrides/SuSE10_frontend

                After the initial installation it will be possible to keep in
                sync the common files for the Login group creating them into the
                "SuSE10_frontend" override and using the command
                si_pushoverride(8).

                It will be even possible to define host-only files creating an
                override using the hostname of the target client. For example
                all the files in /var/lib/systemimager/overrides/node001 will be
                distributed only to node001 (and in case of overlaps these files
                will replace the files that come from the group override and
                from the global override).
-->
<group>
                <name>Login</name>
                <!--
                        If a client belongs to multiple groups, the group with
                        the higher priority will be used to choose the image
                        for that client; for the overrides the groups will be
                        sorted by group priority: in case of file overlaps
                        first hit wins.

                        In this example node001, that belongs to the group
                        Login and Storage, will be auto-installed with the
                        image SuSE10 and it'll receive the overrides in the
                        following order (remember that in case of file overlaps
                        first hit wins):

                            SuSE10_frontend, SuSE10, Storage

                -->
                <priority>10</priority>
                <image>SuSE10</image>
                <!--
                        Also a group can have multiple overrides. The same
                        rules for multiple values of the global overrides are
                        valid also here.
                -->
                <override>SuSE10_frontend</override>
                <override>SuSE10</override>
                <node>node001,node002</node>
</group>

<!--
                Another example. The group Storage contains 16 nodes (from
                node1293 up to node1308). They will be auto-installed using the
                image:
```

```
               /var/lib/systemimager/images/RHEL4

               And the override:

               /var/lib/systemimager/overrides/Storage

               In general the best practice is to use the same name for the
               override and the group name (like this group).
       -->
       <group>
               <name>Storage</name>
               <priority>20</priority>
               <image>RHEL4</image>
               <override>Storage</override>
               <node>node001,node1293-node1308</node>
       </group>

       <!--
               Define your custom groups below (and remember to remove
               or comment the previous entries if you don't want to use
               them).

               ...
       -->

   </xml>
```

8. **si_addclients** creates a symbolic link to the master script for the image to which each specified autoinstall client is assigned. **si_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.

   > You can skip the **si_addclients** step if you've used **si_clusterconfig** as specified at the previous point.

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

   a. In the first configuration screen **si_addclients**, asks you to specify the hostname pattern of your autoinstall clients. Autoinstall client hostnames are defined by a host-range string and a domain name string. For example, if you choose "systemimager.org" as your domain name, and "www07-www11,www20" as your range, you will define the following autoinstall clients:
      www07.systemimager.org
      www08.systemimager.org
      www09.systemimager.org
      www10.systemimager.org
      www11.systemimager.org
      www20.systemimager.org

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

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

c. In the third configuration screen, the **si_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 3-4. Entries in /etc/hosts created by si_addclients

If you give **si_addclients** a range of IPs from "192.168.1.1-192.168.1.99", a hostname range of "server1-server99" and "mydomain.com" as domain name, 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
```

9. Create a boot media to auto-install your clients.

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

- Boot the system from a USB disk or an internal disk.

  Run **si_mkautoinstalldisk** to create an autoinstall USB disk (or a generic disk) that you can use with any machine.

- Boot the system from a CDROM.

Run **si_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.

- Auto-install the images from a running system.

  If your client is already running GNU/Linux you can simply execute the **si_updateclient** command and run it with the `-autoinstall` option.

  Example 3-5. Booting the autoinstall media from a running system's hard drive

  ```
  [root@server7]# si_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.

  SystemImager comes with the **si_mkbootserver** utility to help configure a PXE server. Running **si_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 **si_mkbootserver** detects an error, it will fail out and generate an error message. When you have corrected the error, you can re-execute **si_mkbootserver**, and repeat until it exits successfully. **si_mkbootserver** will probably not work with all PXE clients. If it fails to work with your configuration, please send a mail to sisuite-users@lists.sourceforge.net (mailto:sisuite-users@lists.sourceforge.net).

- Now it's time to install the images on your clients simply booting them with the generated boot media and waiting for the full auto-installation.

3.1.2. SystemImager Tools

3.1.2.1. the si_prepareclient command

- After configuring the golden client, run the **si_prepareclient** command to create a file with the partition informations from your disks that will be put it in `/etc/systemimager/autoinstallscript.conf`.

- **si_prepareclient** will also create a temporary rsync(1) configuration file (in `/tmp` 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. Once the image is successfully pulled from the golden client, this **rsync** daemon will be automatically stopped.

3.1.2.2. The si_getimage command

• After running **si_prepareclient**, run the the **si_getimage** command on the image server. For example :
  si_getimage -golden-client 192.168.1.1 -image my_webserver_image_v1

• **si_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.

• **si_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 **si_getimage** to update an existing image by simply specifying an existing image
  name, for example, **si_getimage -golden-client 192.168.1.1. -image <imagename>. si_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*. **si_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
  si_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/`

  • `/var/lib/systemimager/images/my_webserver_image_v2/`

  • `/var/lib/systemimager/images/my_webserver_image_HEAD/`

  • `...`

3.1.2.3. Autoinstall scripts

• After **si_getimage** has pulled the files to the image directory on the imageserver, it creates an
  auto-install script customized for the image. The auto-install script in this example is named
  "my_webserver_image_v1.master". All auto-install scripts are placed in the
  `/var/lib/systemimager/scripts` directory.

• The disk partitioning information left behind by the **si_prepareclient** command adds the necessary
  commands to re-partition the disk(s) on the autoinstall clients.

• Filesystem informations are taken from the `/etc/systemimager/autoinstallscript.conf` file
  in the image (i.e.
  `/var/lib/systemimager/images/my_webserver_image_v1/etc/systemimager/autoinstallscript.conf`
  and used to determine the appropriate filesystem creation commands and to determine mount points
  for the autoinstall process. Networking informations are added to the autoinstall script based on
  command line options passed to **si_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 si_addclients and si_clusterconfig commands

• After running **si_getimage**, run the **si_clusterconfig** command, which opens an interactive editor where you can define the bindings of your images with your groups of clients.

The same operation can be done by **si_addclients**, which asks you for the hostname range you will be installing, but it doesn't have the concept of host groups, so you can specify only host ranges or single hostnames (the same can be done by **si_clusterconfig**). **si_addclients** prompts you to choose the image that will be installed to these hosts and creates symbolic 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 **si_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, but it is stronly suggested to limit these manual customizations and to always double check them before opening a new bug.

The configurations made by **si_addclients** can ovverride the configurations made by **si_clusterconfig**. In order to exclude unexpected problems it is strongly suggested to always use only the **si_clusterconfig** command, obviously when it is available (that means you're using SystemImager 3.9.4 or greater).

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.img`) can be used to boot from USB disks, CDROMs, the network, or any running Linux system's local hard drive.

The **si_mkautoinstalldisk** and **si_mkautoinstallcd** commands use the **syslinux(2)** utility to create disks 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 standard autoinstall kernel contains all the necessary drivers for the majority of systems. Custom kernels can be generated using UYOK feature to meet special disk and network driver requirements.

To use UYOK and generate `kernel+initrd.img` goes to your golden client and run the following command:

```
# si_prepareclient --server servername --no-rsyncd --my-modules
```

The initrd.img will be generated on the fly from the initrd_template package (eg. systemimager-i386initrd_template). If you specify --no-rsyncd argument, rsyncd will be not restarted. With --my-modules you can save a lot of space in the UYOK initrd, because only the the modules that are currently loaded in your golden client will be included. Without --my-modules all the available modules will be added into the initrd allowing your UYOK kernel+initrd.img to be used also with heterogeneous clients. If all goes well you'll find the UYOK kernel+initrd.img in /etc/systemimager/boot/ in your golden client. When you run si_getimage from your image server (in this case do not specify --no-rsyncd) the UYOK kernel and initrd.img will be transferred to /usr/share/systemimager/boot/<arch>/<name_of_your_image> in your image server. In general, if you have an heterogeneous environment (clients with different hardware and components) it's better to use the standard BOEL kernel+initrd, because the standard kernel is strongly optimized to obtain better performances. In the other cases, in particular if you have 3rd-party or custom kernel modules it's strongly recommended to use UYOK.

- 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`), installation parameters (passed by the kernel boot options, see http://wiki.systemimager.org/index.php/Installation_Parameters for a complete and up to date list of installation parameters), 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**, **aptitude** or **RPM** and **yum** for example.

  2. Run the **si_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 (suggested) or leave it alone. The advantages of running the **si_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 **si_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 **si_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 a **si_updateclient** back to the last known working image.

    In any case it is always suggested to stop the production in the machines before running the update via **si_updateclient**, since some files used by the running applications could be potentially updated or removed while the applications are using them.

- You can also use the **si_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 in this way the image needs to be re-deployed. **si_updateclient** then modifies the boot-loader configuration 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 re-deploy any running Linux machine without feeding the machine any external CD or USB disk and without having to reconfigure the BIOS to boot off the network, which can be quite problematic with some BIOSes.

Chapter 4. HOWTO Use BitTorrent for peer-to-peer Installs

4.1. Install SystemImager v3.7.4 or higher.

Follow the steps described in the installing section for details.

4.2. Configure the BitTorrent transport on the image server.

Open the file `/etc/systemimager/bittorrent.conf` to validate the configuration; in particular check the following parameters:

- `BT_INTERFACE=eth`$_i$ must be the correct interface to reach the client nodes,

- `BT_IMAGES=image1,image2,...,imageN` must contain a comma separated list of the images to be distributed via BitTorrent,

- `BT_OVERRIDES=override1,override2,...,overrideM` must contain a comma separated list of the overrides to be distributed via BitTorrent (you can always decide to distribute an image by BitTorrent and an override by rsync or vice-versa),

- `BT_UPDATE=y|n` set to y to force the synchronization of the BitTorrent data with the images and overrides content every time the daemon on the image server is restart,

- `BT_COMPRESS=y|n` set to y to compress the images and overrides (with gzip) before the deployment (use this option only if your image server is a powerful machine, in particular with a quite recent CPU).

Following there is a typical configuration to deploy 2 images (suse10 and suse10_frontend) with their overrides:

```
#
# "SystemImager"
#
#  Copyright (C) 2006 Andrea Righi <a.righi@cineca.it>
#
#  $Id: bittorrent.conf 3533 2006-04-24 20:25:59Z bli $
#

# The bittorrent tracker port.
BT_TRACKER_PORT=6969

# Tracker state file.
```

```
BT_TRACKER_STATE=/tmp/dstate


# Tracker log file.
BT_TRACKER_LOG=/var/log/systemimager/bittorrent-tracker.log


# Interface used to seed files with bittorrent.
BT_INTERFACE=eth0


# Set to yes if you want to compress the images before distributing
# them via BitTorrent. Set to 'n' if the image server has an old CPU or
# is not powerful in computations.
#
# Allowed values: y|n
BT_COMPRESS=y


# Set to yes if you want to always synchronize the BitTorrent images
# with the chrootable images on /var/lib/systemimager/images when the
# SystemImager BitTorrent daemon starts.
#
# Allowed values: y|n
BT_UPDATE=n


# Comma separated list of images to distribute with BitTorrent
# (ex. BT_IMAGES=RHEL4_base,suse10,frontend,backend...)
# IMPORTANT: no spaces between images!!!
BT_IMAGES=suse10,suse10_frontend


# Comma separated list of overrides to distribute with BitTorrent
# (ex. BT_OVERRIDES=RHEL4_base,suse10,frontend,backend...)
# IMPORTANT: no spaces between overrides!!!
BT_OVERRIDES=suse10,suse10_frontend
```

4.3. Configure the clients to use the BitTorrent transport.

To enable the clients to use the BitTorrent transport you need to specify the boot parameter
`BITTORRENT=y`. Typically this can be done appending this option at the end of the kernel boot string in
`/etc/systemimager/pxelinux.cfg/syslinux.cfg` or passing it via the `--append` option in the
commands that allow it (`si_mkclientnetboot`, `si_mkautoinstallcd`, `si_mkautoinstalldisk`).

Here is a typical `syslinux.cfg` configured to enable all your clients to use auto-installation with the
BitTorrent transport:

```
DEFAULT systemimager


#
# Uncomment next line to send pxelinux boot prompt over serial port 0.
# NOTE: Be sure your serial port speed is appropriate (57600, 9600, etc.)
#
```

```
#SERIAL 0 57600
DISPLAY message.txt
PROMPT 1
TIMEOUT 50

# Add the following to the append line above to use your first serial port
# (ttyS0) as a console in addition to your monitor (tty0).  NOTE: Be sure
# your serial port speed is appropriate (57600, 9600, etc.)
#
#console=ttyS0,57600

# Add the following to the append line above to increase the size of your tmpfs
# filesystem.  About 100MB larger than your image size should suffice.
#
# Other tmpfs mount options are also supported.  See the FAQ for details.
#tmpfs_size=800M

LABEL systemimager
KERNEL kernel
APPEND vga=extended initrd=initrd.img root=/dev/ram BITTORRENT=y MONITOR_SERVER=172.16.36.
```

Last step consists to boot your clients with the proper auto-install boot media and enjoy the auto-installation with BitTorrent.

4.4. Important notes

When you perform any change into an image or override that is deployed via BitTorrent remember to force a synchronization of the BitTorrent repository.

To do so, simply change the value `BT_UPDATE=y` in `/etc/systemimager/bittorrent.conf` and restart the BitTorrent deamon:

```
# /etc/init.d/systemimager-server-bittorrent restart
```

Then if you don't want to re-create the repository at each restart of the daemon change it to `BT_UPDATE=n`.

You can also explicitly remove the tarball and the torrent of the image that you have modified to speed-up the server-side process, instead of setting `BT_UPDATE=y` in the configuration file. In this case a restart of the BitTorrent daemon will re-create only the tarball and the torrent of the single image that you touched.

4.5. A Detailed Walk Through the Process

1. The BitTorrent tracker is started on the image server by the command `/etc/init.d/systemimager-server-bittorrent start`. The tracker allows the distribution of all the torrents that are in the directory `/var/lib/systemimager/torrents`. Torrents needed for the imaging will be generated in the next step.

2. The BitTorrent protocol has been designed to transfer only regular files and directories, but it's not natively able to transfer all the UNIX metadata and special files (like your /dev/ for example). For this reason is necessary to "map" all the files in a image to a single regular file. During the `systemimager-server-bittorrent` startup, SystemImager provides to generate a tarball and a ".torrent" file for each image and override to be distributed by this transport. Tarballs are stored in `/var/lib/systemimager/tarballs` and ".torrent"s in `/var/lib/systemimager/torrents`.

3. After the tarballs and torrents generation, the first seeder is started on the image server (after the tracker) by the script `/etc/init.d/systemimager-server-bittorrent`. The first seeder is the only peer that owns all the chunks of the files to be distributed to the other peers from the beginning. During the first phase of the imaging process it constitutes the only bottleneck (like in the client-server approach), but when the first chunks of data are distributed to some peers (peer set) they begin to talk together, exploiting the advantages of the peer-to-peer network and freeing the image server from the load. In this way, after this transition phase, the image server becomes like another peer in the swarm and it does not constitute a bottleneck anymore.

4. Clients start with a boot package (`kernel + initrd.img`) that includes the BitTorrent client. The first step for a client is to download the needed torrents. This is done using the rsync protocol (torrents are really small compared to the whole image and this doesn't represent a problem in terms of scalability, also with a huge number of clients downloading them at the same time).

5. When the clients have the needed torrents they can start to use the BitTorrent protocol to download anything they need. The first files downloaded via BitTorrent are the BOEL binaries (distributed in a tarball and extracted in the client after the download). The clients continue to seed (upload) the BOEL binaries tarball during the whole auto-installation process, also when they have extracted it, giving their upload band availability to the newer approaching clients.

6. After the download of the BOEL binaries tarball and a first system initialization (module autodetection, disk partitioning, filesystem creation, etc.) it's time to download the image tarball via BitTorrent. This operation works in the same way as the BOEL binaries distribution.

7. The image tarball is extracted in the client's filesystem. During the extraction the client continues to act as a seeder (uploader) for the image tarball. After the extraction the image tarball is removed from the client host and the BitTorrent client is stopped.

8. The overrides are downloaded in the same way.

9. The client is rebooted/kexec-ed/powered-off or starts to beep, according to the post-install action.

4.6. See also

See http://wiki.systemimager.org/index.php/BitTorrent
(http://wiki.systemimager.org/index.php/BitTorrent) for details.

Chapter 5. HOWTO Use Flamethrower for Multicast Installs

5.1. Install SystemImager v3.2.x or higher

Follow the steps described in the installing section for details.

5.2. 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).

5.3. Run si_getimage, si_mvimage, or si_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 (**si_getimage**, **si_mvimage**, **si_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:

```
si_mvimage my_image-v1 tmp_name && si_mvimage tmp_name my_image-v1
```

5.4. 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
```

5.5. 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 the installation parameter approach you can add the following to
/etc/systemimager/pxelinux.cfg/syslinux.cfg (and re-run **si_mkclientnetboot**) or using the
--append "STRING" option with **si_mkautoinstalldisk** or **si_mkautoinstallcd**:

```
FLAMETHROWER_DIRECTORY_PORTBASE=9000
```

5.6. Autoinstall your clients!

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

5.7. 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 si_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.

5.8. See also

See http://wiki.systemimager.org/index.php/Multicast (http://wiki.systemimager.org/index.php/Multicast) for details.

Chapter 6. HOWTO Use OpenSSH for Secure Installs

6.1. Using OpenSSH for Secure Installs (requires SystemImager v3.8.x or higher)

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.

See http://wiki.systemimager.org/index.php/SSH (http://wiki.systemimager.org/index.php/SSH) for details.

6.2. Image server configuration

Uncomment the following 2 lines in `/etc/systemimager/rsync_stubs/10header`:

```
#hosts allow = 127.0.0.1
#hosts deny = 0.0.0.0/0
```

Re-run **si_mkrsyncd_conf**:

```
# si_mkrsyncd_conf
```

Restart the rsync daemon:

```
# /etc/init.d/systemimager-server-rsyncd restart
```

In this way connections to the rsync exported images will be forbidden except from localhost (this will be used by sshd after the SSL tunnel will be opened).

6.3. Client-driven approach

First of all you need to create a boot package (kernel + initrd.img) and include the SSH private key directly into the initrd.img. For this the boot over PXE is strongly discuraged in this case, because kernel and initrd.img are not encrypted during the transmission to the clients with TFTP. To create the boot package with BOEL run the following command:

```
# mkdir /tmp/boot-package
# si_mkbootpackage --destination /tmp/boot-package --kernel \
```

```
/usr/share/systemimager/boot/i386/standard/kernel --filesystem cramfs \
--ssh-key ~foo/.ssh/id_dsa --yes
```

Remember to replace i386 with the architecture of your clients to get the correct kernel (e.g. x86_64). Or if you want to use UYOK:

```
# mkdir /tmp/boot-package
# si_mkbootpackage --destination /tmp/boot-package --image YOUR_IMAGE \
--ssh-key ~foo/.ssh/id_dsa --yes
```

This command will create the boot package in /tmp/boot-package and it will include the SSH private key of the user "foo" into the initrd. To enable the passwordless login for for the user "foo" run the command:

```
$ cat ~foo/.ssh/id_dsa.pub >> ~foo/.ssh/authorized_keys
```

In a similar way you can also use the --ssh-key with **si_prepareclient** in your golden client. In this case you don't need to create the boot package in your image server, simply use kernel + initrd.img generated by **si_prepareclient**.

6.4. Configure the clients to use SSH transport (client-driven)

Create an autoinstall CD with the following command:

```
# si_mkautoinstallcd --out-file /tmp/boot-package/systemimager.iso --kernel \
/tmp/boot-package/kernel --initrd /tmp/boot-package/initrd.img --append \
"MONITOR_SERVER=172.16.36.1 MONITOR_CONSOLE=yes SKIP_LOCAL_CFG=y SSH=y"
```

Remember to replace the address of your monitor server (if you want to use it) and add all the needed installation parameters. If you prefer to use an auto-install USB drive, instead of a CD run:

```
# si_mkautoinstalldisk --device YOUR_USB_DEVICE --kernel \
/tmp/boot-package/kernel --initrd /tmp/boot-package/initrd.img --append \
"MONITOR_SERVER=172.16.36.1 MONITOR_CONSOLE=yes SKIP_LOCAL_CFG=y SSH=y" \
--yes
```

Then boot the clients with the autoinstall CD / USB drive (PXE is not recommended with client-driven SSH) and enjoy the secure auto-installation.

6.5. Server-driven approach

As well as client-driven approach also the server-driven way needs the creation of a client boot package. In this case instead of including the SSH private key (used to connect to the image server), we must include the authorized_keys file, because it's the image server that will open the SSH tunnels to the clients. To create a boot package with BOEL run the following command on your image server:

```
# mkdir /tmp/boot-package
# si_mkbootpackage --destination /tmp/boot-package --kernel \
```

```
/usr/share/systemimager/boot/i386/standard/kernel --filesystem cramfs \
--authorized-keys ~foo/.ssh/id_dsa.pub --yes
```

Remember to replace i386 with the architecture of your clients to get the correct kernel (e.g. x86_64). Or with UYOK:

```
# mkdir /tmp/uyok-boot-package
# si_mkbootpackage --destination /tmp/boot-package --image YOUR_IMAGE \
--authorized-keys ~foo/.ssh/id_dsa.pub --yes
```

After that you will find the kernel+initrd.img to be used for the imaging into the destination directory (/tmp/boot-package).

6.6. Configure the clients to use the SSH transport (server-driven)

Be sure to define the SSH=y installation parameter (for more details see: http://wiki.systemimager.org/index.php/Installation_Parameters). IMPORTANT: server-driven approach is the most secure way to deploy images on the clients, because they never access directly to the image server. For this reason you can forbid every kind of access to the image server (using hosts deny policy or via iptables or using your preferred firewall...). Moreover, since you have to distribute only a public key to the clients, you can ignore the warning of the client-driven approach to not use boot over PXE: in this case the initrd.img doesn't contain private informations and it can be transmitted unencrypted without problems!

6.7. Wait that the clients become ready to accept SSH connection

Wait for the following message, that must appear on the clients console:

```
Started sshd.  You must now go to your imageserver and issue
the following command:

si_pushinstall --hosts ${HOST_OR_IP}.
```

If you are not able to physically watch the clients console you can use monitoring features and check this message from the SystemImager virtual console (see http://wiki.systemimager.org/index.php/Monitoring for more details). The message above means that the clients are ready to accept SSH connection from the image server, so just run the command suggested on the console. You can use also host ranges with si_pushinstall to open all the SSH tunnels in a single shot. For example if you have to image from node01 up to node20 run:

```
# si_pushinstall --hosts node01-node20
```

Then boot the clients with the autoinstall CD / USB drive or via PXE and enjoy the secure auto-installation.

Chapter 7. Monitoring clients installation with SystemImager

7.1. Overview

This document shows how to use the monitoring features of SystemImager. Monitoring allows to watch the details of the installation phases for all your clients. Monitoring is organized in two main components:

- *si_monitor*: a process that runs on the monitor server and collect clients data
- *si_monitortk*: the perl-Tk based monitoring GUI.

7.2. Setting up the monitor server

On the monitor server (typically is the same machine of the image server) simply run:

```
# /etc/init.d/systemimager-server-monitord start
```

This will start the si_monitor daemon. The data sent by the clients will be collected by si_monitor and stored in the XML /var/lib/systemimager/clients.xml. If you cannot execute X applications in your monitor server you can simply periodically look at that file to monitor the installations.

 Never run si_monitor manually! Always use systemimager-server-monitord script!

Then simply run si_monitortk to launch the monitoring GUI.

7.3. Enabling clients to send monitoring data

Define the following parameters to the kernel boot options of the clients:

- *MONITOR_SERVER=IP|HOSTNAME*: IP address or hostname of the monitor server
- *MONITOR_CONSOLE=yes|no*: enable or disable full console view, if enabled it's possible to follow all the installation session of the clients (stdout and stderr) in the monitoring interface (default is *no*)

Usually you need to define the parameters in `/etc/systemimager/pxelinux.cfg/syslinux.cfg`. This works both if you are installing via network boot or by an autoinstall boot media (CD, USB disk, etc.). For example a monitoring-enabled configuration can be the following:

```
LABEL systemimager
KERNEL kernel
APPEND vga=extended initrd=initrd.img root=/dev/ram MONITOR_SERVER=192.168.1.1 MONITOR_CONS
```

Remember to re-run si_mkclientnetboot (for PXE booting), si_mkautoinstallcd (for autoinstall CD booting) or si_mkautoinstalldisk (for autoinstall USB drive booting) to rebuild the configuration files or boot media to accept the new parameters defined above.

7.4. Troubleshooting

7.4.1. Clients do not appear

If you don't see the clients in the monitoring interface when they boot-up check the following issues:

- Do you have a firewall in your image server that filter the port 8181? The clients use that port to contact the image server.

- Try to use the IP address of the monitor server instead of the hostname.

- Are you sure the clients are using the correct boot parameters? (remember that you can check the parameters directly from the clients if you have the access to the real console looking in `/proc/cmdline`)

- Try to increase the verbosity of the si_monitor log changing the value *LOGLEVEL=2* into *LOGLEVEL=3* in `/etc/init.d/systemimager-server-monitord` and restart the monitor daemon (`/etc/init.d/systemimager-server-monitord restart`); all the logs are stored in `/var/log/systemimager/si_monitor.log`).

7.4.2. I've not a X server to see si_monitortk interface...

Clients can be monitored also if an X server is not available, for example if you're connected remotely via ssh on the image server and you've not enabled the X11 forwarding. In this case simply look at the file `/var/lib/systemimager/clients.xml`. It's an XML, but the clients and the attribute names are quite nmemonic. You can periodically cat the file or write your own console monitoring scripts. If you wrote a nice ncurses or a console interface feel free to post the patch to the sisuite-devel (mailto:sisuite-devel@lists.sourceforge.net) list.

Chapter 8. HOWTO Distribute configuration file across a SystemImager cluster

8.1. Install SystemImager v3.9.4 or higher.

Follow the steps described in the installing section for details.

8.2. Overview

si_pushoverrides is a tool to distribute configuration files from the image server to the clients or group of them, using the SystemImager overrides. The command accepts a list of group or node names as arguments and concurrently synchronizes the content of the associated overrides to them using a server-drien approach (the image server copies the files to the clients using rsync over ssh to exploit the advantages of bandwidth optimization and security).

si_clusterconfig is a tool to manage and show the cluster topolgy. In show-mode the command accepts as argument a list of hostnames, host-ranges and/or host-group, it resolves them in the equivalent list of hostnames and prints them to stdout. The edit-mode can be interactive (option -e) or batch (option -u). In interactive edit-mode si_clusterconfig opens an editor in your terminal that allows to modify the client group definitions and their properties using a XML syntax. In batch edit-mode it only parses the pre-defined XML configuration and refresh the opportune SystemImager internal configuration files.

8.3. Define the cluster topology

Run the command *si_clusterconfig -e* as root. There are 3 levels of hierarchy for the overrides:

- *global override*: to be distributed to all the nodes,
- *group override*: to be distributed only in a group of nodes,
- *node override*: to be distributed in a single node.

The files in the global override are distributed to all the nodes. If there is a file with the same path and the same name in a group override, the group override wins. If there is a file with same path and same name in a node override and a group override, then the node override wins.

The required elements are:

- the name of your image server:

- the name of the global group (that identify all the clients):

- the name of the global override (to be distributed in all the clients):

8.4. A simple example

```
<xml>
       <master>master1</master>
       <name>all</name>
       <override>all</override>
       <group>
               <name>Login</name>
               <image>RHEL5</image>
               <override>Login</override>
               <node>node001</node>
               <node>node002</node>
       </group>
       <group>
               <name>Compute</name>
               <image>Ubuntu_gutsy</image>
               <override>Compute</override>
               <node>node003-node010</node>
       </group>
</xml>
```

This is a 10-nodes cluster definition. The hostname of the image server is master1; the cluster has 2 login nodes (node001 and node002) that use the override called Login and 8 compute nodes (node003, node004, node005, node006, node007, node008, node009 and node010), that use the override called Compute.

Example 8-1. Example 1: distribute the passwd, shadow and group to all the nodes

Create the files:

```
# cp -p /etc/passwd /var/lib/systemimager/overrides/all/etc/passwd
# cp -p /etc/shadow /var/lib/systemimager/overrides/all/etc/shadow
# cp -p /etc/group /var/lib/systemimager/overrides/all/etc/group
```

From master1 run the command:

```
# si_pushoverrides -v all
```

Basically when you specify the global override all the nodes defined in `cluster.xml` are updated accordingly to the hierarcy of the overrides.

Example 8-2. Example 2: distribute different access.conf to Login and Compute nodes

Allow root to login on "Login" nodes only from the local domain,
/var/lib/systemimager/overrides/Login/etc/security/access.conf:

```
-:root:ALL EXCEPT LOCAL .localcluster.domain.org
```

Disallow direct login on "Compute" nodes for non-privileged users,
/var/lib/systemimager/overrides/Compute/etc/security/access.conf:

```
-:ALL EXCEPT root wheel:ALL
```

From master1 run the command:

```
# si_pushoverrides -v Compute Login
```

Example 8-3. Example 3: close the second login node (node002) to non-privileged users

/var/lib/systemimager/overrides/node002/etc/security/access.conf:

```
-:ALL EXCEPT root:ALL
-:root:ALL EXCEPT LOCAL .localcluster.domain.org
```

From master1 run the command:

```
# si_pushoverrides -v node002
```

8.5. See also

See http://wiki.systemimager.org/index.php/File_distribution
(http://wiki.systemimager.org/index.php/File_distribution) for details.

Chapter 9. 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 CD?

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

Q: How do I make an autoinstall USB disk?

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

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 si_netbootmond running, but it isn't working. Why?

A: In order for si_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 /var/lib/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 /var/lib/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 /var/lib/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: si_mkautoinstallcd and **si_mkautoinstalldisk** support an -append option, allowing you to specify additional options for the autoinstall kernel, including serial console options. For example: **si_mkautoinstallcd -out-file autoinstall.iso -append "console=ttyS0"**

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 **si_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 with UYOK feature. See
 http://wiki.systemimager.org/index.php/UYOK (http://wiki.systemimager.org/index.php/UYOK)).

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 **si_prepareclient** command prior to running **si_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 autoinstalldisk 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 3.8.0, you can use the **si_pushinstall** command, which opens an SSH tunnel from the image server to the clients. Then each client downloads the image using the SSH tunnel opened by the image server. This is the most secure approach to install clients over insecure networks.

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

A: If you ran **si_mkdhcpserver** to configure your dhcp information, and if you answered all the questions you were asked when you did ran **si_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. *autoinstallcd* - it takes slightly little time to boot and is more durable, but you have to have a CD burner and clients that can read CD-R's)

2. *autoinstalldisk* - it takes slightly little time to boot, but requires that the BIOS of your clients support boot over USB devices.

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 **si_mkautoinstallcd** and **si_mkautoinstalldisk** 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 **si_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: What if I want to assign static IPs to my clients?

A: You can. **si_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 **si_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 **si_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 **si_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 **si_getimage** is run, it recreates the $image.master script. If you have customized your $image.master script, be sure to save it before running **si_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, **si_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 **si_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 **si_updateclient** to a test image (i.e. do an **si_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 **si_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 **si_updateclient** command. See the "How do I update a client to match an image?," FAQ for more information.

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 **si_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 **si_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 **si_getimage** specifying a different imagename. This is preferred as it allows for revision control.

2. Run **si_updateclient** on the clients that you want to update. Execute **si_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: 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/**.

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 **si_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 **si_mkautoinstallscript** to create a new autoinstall script using the new parameters.

   NOTE: By default, **si_mkautoinstallscript** uses the autoinstallscript.conf file located in your image's ./etc/systemimager directory. See **man si_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 **si_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 **si_mkautoinstallscript** to create a new autoinstall script using the new parameters. By default, **si_mkautoinstallscript** uses the autoinstallscript.conf file located in the ./etc/systemimager directory in your image. See **man si_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. Run **si_mkautoinstallscript --autodetect-disks ...** to create a new autoinstall script that will be able to automatically detect disk types at run-time during the imaging of your clients.

3. An alternative method is to manually modify autoinstallscript.conf and re-run **si_mkautoinstallscript**.

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

A: Yes. Be sure to use **--autodetect-disks** with **si_getimage** or **si_mkautoinstallscript** if you have different disk types in your clients and create a different `autoinstallscript.conf` and master script for each partitioning schema you want to use.

9.1. See also

Consult the troubleshooting guide on the SystemImager web site at http://wiki.systemimager.org/index.php/Troubleshooting and the online FAQ at http://wiki.systemimager.org/index.php/FAQ for details.

Chapter 10. Troubleshooting

10.1. What is the "ETHER_SLEEP" variable, and when should I adjust it?

The ETHER_SLEEP variable specifies the number of seconds that your autoinstall client(s) should wait before trying to talk to the network. The default is zero (0), to make installs go faster, as a timeout is not normally needed.

Certain networking equipment, notable switches, may refuse to pass traffic from a new interface that has appeared on a switch port until after a 30+ second delay. This delay is usually a settable option (if your switch even has this capability). Whether or not it is set on your switches is vendor and/or site specific.

If you encounter problems during an autoinstall, such as your autoinstall client not recieving an IP address via DHCP: a) you find that when you ask for a DHCP address from the command line, you get one. b) you manually configure the network interface and can then contact the imageserver; then you may want to change the ETHER_SLEEP variable.

Both of these symptoms can often be explained by the 30+ second timeout passing prior to the manual intervention.

If you decide to change the ETHER_SLEEP variable, a value of 35 has been found to work in most cases (ETHER_SLEEP=35). ETHER_SLEEP can be set in a local.cfg file or by modifying the ./etc/init.d/rcS script in the BOEL source code.

NOTE: The 30+ second timeout at the switch begins with the interface on your autoinstall client is made active (Ie: driver loaded), and is not necessarily tied to when the interface is configured with an IP address.

10.2. si_getimage fails with a "Failed to retrieve `/etc/systemimager/mounted_filesystems` from <golden client>" message.

Two known issues cause this error:

1. Your firewall may be blocking the rsync port. Some Red Hat releases (and possibly other distributions) provide firewall rules as part of a default installation. The **ipchains** and **iptables** utilities have a `-L` that will print a list of active rules.

2. **rsync** relies on the ability to do a reverse lookup of the remote machine. If you don't have reverse DNS setup in your cluster, you can add entries for each machine in your cluster to the `/etc/hosts` file on each machine. (Adding an entry for your image server in your golden client's `/etc/hosts` file should be sufficient for using **si_getimage**.

10.3. My client autoinstallation/update hangs, crashes, or is ridiculously slow.

Goran Pocian reported an instance of unacceptable **si_updateclient** performance that went away when he upgraded from kernel 2.2.17 to 2.2.18.

He also noted that if you mount an NFS filesystem after executing **si_prepareclient**, **si_getimage** will retrieve its contents. As this can heavily increase network load, it can also cause bad performance.

Brian Finley reported other possible causes:

Every once in a while, someone reports some mysterious hanging or transfer interruption issue related to rsync. I had a chance to speak with Andrew Tridgell in person to discuss these issues.

We found two known issues that could be the source of these symptoms. One is a known kernel issue, and one is an rsync issue. The kernel issue is supposedly resolved in 2.4.x series kernels, (SystemImager has not yet been "officially" tested with 2.4.x kernels) and may not be present in all 2.2.x series kernels (I believe).

The rsync bug will be fixed in the rsync 2.4.7 release (to happen "Real Soon Now (TM)"). The rsync bug is caused by excessive numbers of errors filling the error queue which causes a race condition. However, until rsync 2.4.7 has been out for some time, I will still recommend using v2.4.6 unless you specifically experience one of these issues.

Here's a hack that seems to work for Chris Black. Add "--bwlimit=10000" right after "rsync" in each rsync command in the <image>.master script.

```
Change: "rsync -av --numeric-ids $IMAGESERVER::web_server_image_v1/ /a/"
To:     "rsync --bwlimit=10000 -av --numeric-ids $IMAGESERVER::web_server_image_v1/ /a/"
```

Here are some tips on diagnosing the problem:

- If you get an error message in `/var/log/messages` that looks like:

  Jan 23 08:49:42 mybox rsyncd[19347]: transfer interrupted (code 30) at io.c(65)

  You can look up the code number in the `errcode.h` file which you can find in the rsync source code.

- To diagnose the kernel bug: Run **netstat -tn**. Here is some sample output (from a properly working system):

```
$ netstat -tn
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address          Foreign Address State
tcp        1      0 192.168.1.149:1094     216.62.20.226:80 CLOSE_WAIT
tcp        1      0 192.168.1.149:1090     216.62.20.226:80 CLOSE_WAIT
tcp        1      0 192.168.1.149:1089     216.62.20.226:80 CLOSE_WAIT
tcp        0      0 127.0.0.1:16001        127.0.0.1:1029 ESTABLISHED
tcp        0      0 127.0.0.1:1029         127.0.0.1:16001 ESTABLISHED
tcp        0      0 127.0.0.1:16001        127.0.0.1:1028 ESTABLISHED
tcp        0      0 127.0.0.1:1028         127.0.0.1:16001 ESTABLISHED
```

The symptoms are:

- Machine A has data in its Send-Q

- Machine B has no data in its Recv-Q

- The data in machine A's Send-Q is not being reduced

What's happening is:

1. One or both kernels aren't honoring the other's send/receive window settings (these are dynamically calculated)

2. The result is the kernel(s) aren't getting data from machine A to machine B

3. rsync, therefore, isn't getting data on the receive side

4. The process appears to hang.

- Details about the rsync bug:

  What happens:

  1. A large number of errors clogs the error pipe between the receiver and generator

  2. All progress stops.

  3. Again, the process appears to hang.

I hope this information helps...

A possible solution, suggested by Robert Berkowitz, is to add `--bwlimit=10000` to the rsync options in the rsync initscript.

10.4. My autoinstallcd doesn't boot.

Download and install a newer syslinux RPM.

10.5. My client failed to autoinstall, and when I run an rsync command on it manually, it takes forever for the image server to respond.

Be sure that the image server can look up the client's hostname based on its IP address. The easiest way to do this is to have entry in the image server's `/etc/hosts` file for the client system.

10.6. My client fails with the error: `"chroot: cannot execute systemconfigurator: No such file or directory"`

This failure is most commonly associated with a mismatch between the version of SystemImager you used to create your image and the version of SystemImager you used to create the corresponding .master script.

As of SystemImager 2.0, SystemConfigurator is used to make final configuration changes to an image. SystemConfigurator is executed from within the image, so it must be installed within the image on the image server. To insure this, SystemConfigurator must be installed on any golden client before an image is pulled from it. If you have images that were pulled from golden clients that did not have SystemConfigurator installed, you can install SystemConfigurator directly into the image on the imagserver.

Example 10-1. Installing SystemConfigurator into an Image on an Image Serer

1. Download the latest SystemConfigurator package for your system from http://sourceforge.net/projects/systemconfig.

2. Copy the SystemConfigurator package into your image directory. For example:

   # cp systemconfigurator-1.10-1.noarch.rpm /var/lib/systemimager/images/my_image/tmp

3. Chroot into the image directory and install the package.

   # chroot /var/lib/systemimager/images/my_image bash

   # rpm -Uvh /tmp/systemconfigurator-1.10-1.noarch.rpm

   # exit

10.7. My client completes the autoinstall process successfully, but I get an "Invalid Partition Table" error upon reboot, and Linux never boots.

SystemImager 2.0.x and earlier didn't maintain the bootable flag in the partition table. This worked fine in most cases, but in some cases this leads to an unbootable system. To confirm that this is the problem, boot your system from rescue media, and set the bootable flag on your boot partition using cfdisk or another partitioning tool. If this allows your system to boot, then you must upgrade SystemImager and regenerate your autoinstallscript(s). If for some reason you can't upgrade, then check the following:

• Be sure that you are using the latest version of SystemImager and that you are using the autoinstalldiskette image that comes with that version. Note that the version numbers may not match. See the VERSION file.

10.8. See also

Consult the troubleshooting guide on the SystemImager web site at http://wiki.systemimager.org/index.php/Troubleshooting for details.