

# SystemImager v2.0.1

## Documentation

**Greg Pratt**  
VA Linux Systems  
[gpratt@valinux.com](mailto:gpratt@valinux.com)

**Dann Frazier**  
Hewlett Packard  
[dannf@ldl.fc.hp.com](mailto:dannf@ldl.fc.hp.com)

This document has been prepared to aid in the installation, configuration and on-going administration of the SystemImager product.

SGML source ([systemimager-manual.sgml](#)) for this document is available and should be consulted while reading.

## 1. Acknowledgments and Thanks

Thanks to everyone that gave comments as I was writing this. This includes (in alphabetical order):

- Paonia Ezrine <[paonia@home.welcomehome.org](mailto:paonia@home.welcomehome.org)>
- Brian Finley <[brian@systemimager.org](mailto:brian@systemimager.org)>
- Dann Frazier <[daniel\\_frazier@hp.com](mailto:daniel_frazier@hp.com)>
- Austin Gonyou <[austin@coremetrics.com](mailto:austin@coremetrics.com)>
- Ari Jort <[ajort@valinux.com](mailto:ajort@valinux.com)>
- Jason R. Mastaler <[jasonrm@lanl.gov](mailto:jasonrm@lanl.gov)>
- Ben Spade <[spade@radik.com](mailto:spade@radik.com)>
- Curtis Zinzilieta <[czinzilieta@valinux.com](mailto:czinzilieta@valinux.com)>

SystemImager was conceived and developed by Brian Finley. It's initial implementation was known as Pterodactyl and was used for software and password updates to Solaris boxes of varying hardware and OS versions across a nation wide enterprise network. Over time it evolved into the Linux specific

autoinstall and software distribution tool that it is today. Many of the design decisions for SystemImager were based on perceived short comings in other automated install tools for systems such as Solaris, RedHat Linux, and Windows.

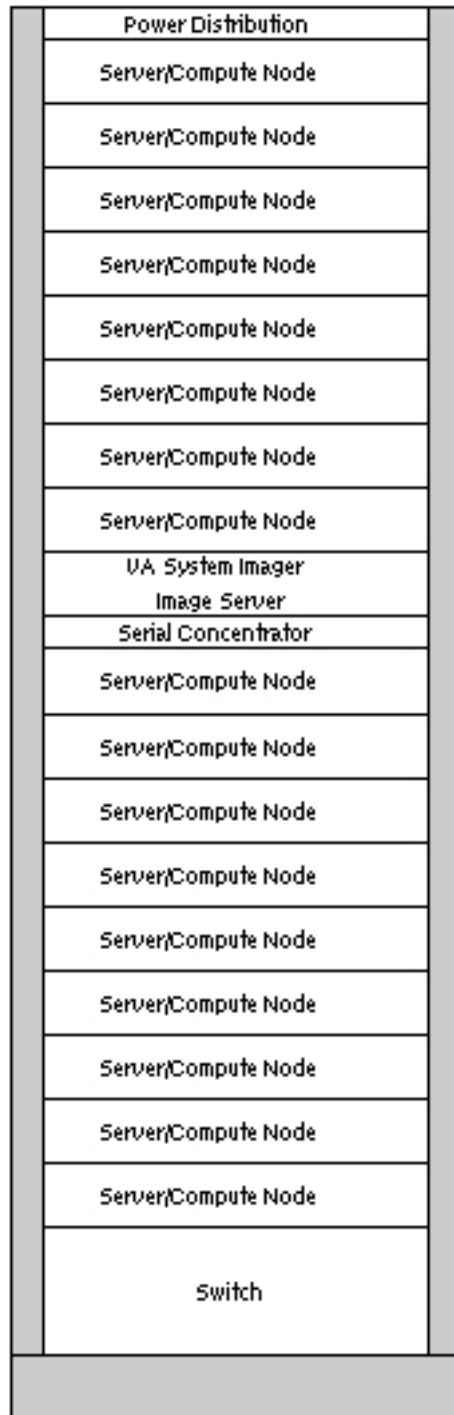
Be sure to view the CREDITS file for a listing of other people who have contributed code or documentation that has been incorporated. Many thanks go to these people as their relentless pursuit in the discovery of bugs and the occasional code contribution are invaluable.

## **2. Introduction**

SystemImager is software that makes the installation of Linux to masses of similar machines relatively easy. It also makes software distribution, configuration, and operating system updates easy. You can even update from one Linux release or version to another! SystemImager can also be used for content management on web servers.

It is most useful in environments where you have large numbers of identical machines. Some typical environments include: Internet server farms, high performance clusters, computer labs, or corporate desktop environments where all workstations have the same basic hardware configuration.

**Example 1. A typical Cluster or Server Farm**



### 3. Who Should Use This Guide

This guide is for system administrators whose responsibility it is to install and configure one or more systems in a networked environment.

Typical organizations that see a benefit from using SystemImager include:

- organizations that have Internet server farms
- organizations that manage many workstations or servers for a department or for the entire company
- organizations that are doing super-computing/clustering with Linux
- anyone who needs to maintain identical configurations on a large number of machines
- manufacturing organizations who must automate the software preload process for Linux based machines

### 4. Supported Distributions

SystemImager is built on top of SystemConfigurator (<http://sourceforge.net/projects/systemconfig>), which takes care of all of the distribution specific details. Therefore, SystemImager should work with any distribution that SystemConfigurator supports (and some that it doesn't :). If you are using an untested distribution, you may find that the networking and hostname information are not configured properly after doing an autoinstall. If you try SystemImager with your favourite distribution and it doesn't work properly, please submit a bug report ([http://sourceforge.net/bugs/?group\\_id=259](http://sourceforge.net/bugs/?group_id=259)), preferably with patches you've created to fix the problem. Appropriate credit is given to all who find and submit valid bugs or contribute code or documentation that is used.

Here is an incomplete list of distributions known to work with SystemImager. If you are successfully using SystemImager with a distribution or version not listed here, please submit a bug report ([http://sourceforge.net/bugs/?group\\_id=259](http://sourceforge.net/bugs/?group_id=259)), so we can add them to the list. (Not a bug I know, but this is the best way to report success!)

**Table 1. Author Tested and Supported Distributions**

Distribution	Versions
Debian	2.1 ('slink' with 2.2.x kernel), 2.2 ('potato'), ('woody')
Kondara	1.1
RedHat	6.0, 6.1, 6.2, 7.0
RedHat with VA Linux Enhancements	6.0.x, 6.1.x, 6.2.x, 7.0.x
Storm	1.4

**Table 2. Other Distributions Reported to Work**

Distribution	Versions	Reported by
Mandrake	7.1	Ben Spade (spade@radik.com)

Distribution	Versions	Reported by
TurboLinux Server	6.0	unknown

Linux-2.0.x based distributions are not supported at this time. Currently there are no plans to add support for Linux 2.0.x based distributions.

## 5. System Requirements

- Your image server must have enough disk space to hold the images to be installed on your client systems.
- All of the clients that will use the same image should have hardware that is as similar as possible. Most importantly they should use the same chipset on the network device(s) and the same number and kind of hard drive(s) (IDE, SCSI, Mylex Hardware RAID, etc.) The hard drives may be of different capacities. Devices may be larger with no problem and can be smaller within reason.
- For PXE installations, you will need a compatible tftp server running on the boot server (usually the image/DHCP server). H. Peter Anvin created the tftp-hpa package, which provides the required functionality. RedHat 7.0 includes this server in their tftp-server package, but the tftp package included with earlier RedHat releases doesn't provide the required functionality. Debian provides this server in the tftpd-hpa package. The atftpd server is also reported to work.

PXE network-based installations may also require a pxe daemon to be running on your image server. This requirement depends on the firmware used on the client side, and the capabilities of your DHCP server.

## 6. How Does it Work? (summary)

The basic idea behind SystemImager is to retrieve the entire system image from a "golden client" to an "image server". This image can then be replicated to any number of client systems.

Once your client systems have had the initial image replicated to them, they can be updated/upgraded by syncing them to an updated image on the image server. During an update, only the modified parts of files are pulled to the client. This makes for a fast, efficient, and accurate mass update.

## 7. SystemImager Terminology

- **autoinstall media** - The media that is used to boot an autoinstall client in order to begin the autoinstall process. This media can be a floppy, a CDROM, the network, or the local hard drive of the autoinstall client.

- `local.cfg` - A configuration file that can be used for autoinstall clients in lieu of DHCP and the `/tftpboot/systemimager/hosts` file on the image server.
- `updateclient` - A command that is executed on client systems allowing them to be updated or synchronized to a new or updated image after the initial autoinstall. **updateclient** enables software and content distribution.
- image server - The machine that will hold and distribute the images.
- golden client - A machine from which an image is taken. Golden clients are manually installed and customized to taste.
- `getimage` - This command is run from the image server to pull a system image from a golden client.
- `prepareclient` - This command is run on the golden client immediately prior to running **getimage** on the image server.
- `mkdhcpserver` - An easy way to create a SystemImager appropriate `/etc/dhcpd.conf` file. DHCP can be used to assign IP addresses to autoinstall clients.
- `mkdhpstatic` - Used 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.
- autoinstall script - A unique autoinstall script is created for each image and is used by the autoinstall client as part of the autoinstall process. The names of autoinstall scripts begin with the image name and end in `.master`. For example: `"my_webserver_image_v1.master"`
- `addclients` - Tells your image server which image to install on your autoinstall clients. It does so by creating soft links to the master autoinstall script with the name of each host that will receive that image. It also allows you to populate the `/etc/hosts` file with sequential host names and IP addresses. This information in `/etc/hosts` is necessary for certain SystemImager operations.
- daemon (<http://www.dictionary.com/cgi-bin/dict.pl?term=daemon>) - This is not specifically a SystemImager term, but it really bugs me when people pronounce this word as "daymen". It is just another, more interesting, less evil appearing way of spelling demon. And just to be clear, in computer terms, a daemon is a program that lies in wait for something to trigger it into action. An example of this is a web server daemon waiting for someone to request a web page. -Brian

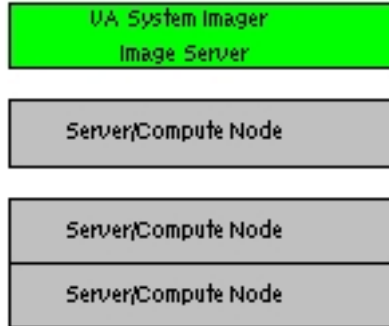
## 8. Overview of SystemImager Installation and Usage

1. Install the SystemImager server package on the machine you have chosen as your image server.
2. Install Linux on your "golden client", and customize as you desire.
3. Install the SystemImager client software on your "golden client" and run the **prepareclient** command.
4. Run **getimage** on the image server to pull the image from the "golden client" to the image server.
5. Run **addclients** on the image server to tell it which clients will receive which image, and to populate the image server's `/etc/hosts` and `/tftpboot/systemimager/hosts` files.
6. Choose and configure the method for assigning IP addresses to your autoinstall clients.
7. Autoinstall the "golden" image on other machines.

## 9. The Actual Step-by-Step HOWTO

1. Install the SystemImager server package on the machine you have chosen as your image server.

**Example 2.** Often times, a machine that is already performing some kind of management function is a good choice for your image server.



You can find the SystemImager packages and links to some of the dependent packages that may not be a part of your Linux distribution at <http://systemimager.org/>.

The SystemImager server software is available in tar ball format and may be available in other distribution specific packaged formats. The example below is for installing from a tar ball. The software is the same regardless of the packaging format.

### Example 3. Installing the SystemImager server software

```
[root@imageserver]# bzcat va-systemimager-server-x.x.x.tar.bz2 | tar -x
[root@imageserver]# cd va-systemimager-server-x.x.x
[root@imageserver]# ./install
Welcome to SystemImager.
```

This install script may modify the following files and/or directories:

```
/tftpboot/systemimager/ -- create if necessary and add appropriate files/links
/tftpboot/pxelinux.cfg/ -- create if necessary and add appropriate files/links
/etc/services           -- add /sync and/or tftp entries if necessary
/etc/inetd.conf         -- remove rsync entry if necessary and
                        add or modify tftp entry if necessary
/etc/rsyncd.conf        -- it is assumed that SystemImager will manage this
                        file and that it will not be used for anything else
```

All modified files will be backed up with the `.beforesystemimager` extension.

See "install -help" for command line options.

```
Install SystemImager? (y/[n]) y
Ok. Installing SystemImager...
```

```
Installing files in /usr/sbin/
```

```

Installing files in /etc/init.d/
Installing files in /tftpboot/
Installing files in /tftpboot/pxelinux.cfg/
Installing files in /tftpboot/systemimager/
Installing files in /var/spool/systemimager/images/
rsync entries already enabled in /etc/services...
tftp entry already enabled in /etc/inetd.conf...
SystemImager brand /etc/rsyncd.conf file already exists...
creating soft links to rsync init script...
running rsync init script...
Stopping rsync daemon: rsync.
Starting rsync daemon: rsync.
Installing files in /usr/share/doc/va-systemimager-x.x.x/

All done! Please read the manual before using SystemImager.
See /usr/share/doc/va-systemimager-x.x.x/ for all documentation.

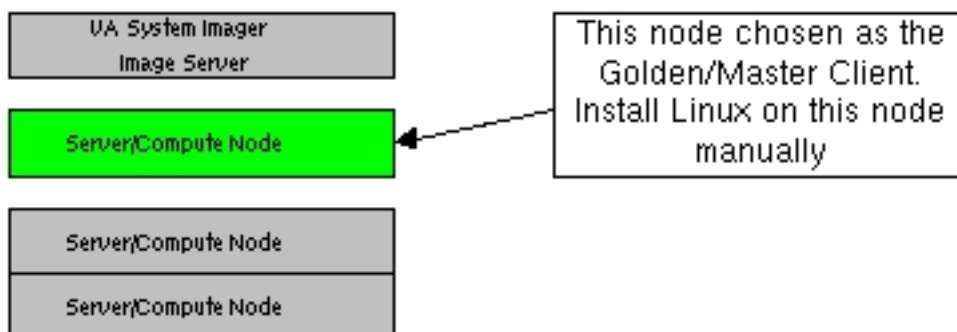
```

## 2. Install Linux on your "golden client", and customize as you desire.

Select a machine which will be the "golden client". This machine's image is the image you will replicate to other machines.

Configure this "golden client" as you normally would any machine. Install Linux on this node with the understanding that the software installed will eventually constitute the image for all other nodes installed with SystemImager. Don't worry too much about getting it exactly right the first time, since you can easily use SystemImager to make incremental changes to your image and distribute those changes without doing a complete re-install.

### Example 4. Install the "golden client"



## 3. Install the SystemImager client software on your "golden client" and run the **prepareclient** command.

The SystemImager client software is available in tar ball format and may be available in other distribution specific packaged formats. The example below is for installing from a tar ball. The software is the same regardless of the packaging format.



### Example 5. Installing the SystemImager client software

```
[root@imageserver]# bzipcat va-systemimager-client-x.x.x.tar.bz2 | tar -x
[root@imageserver]# cd va-systemimager-client-x.x.x
[root@imageserver]# ./installclient
Install SystemImager client? (y/[n]) y
Ok. Installing SystemImager client...
```

Installing files in /usr/share/doc/va-systemimager-x.x.x/

All done! Please read the manual before using SystemImager.  
See /usr/share/doc/va-systemimager-x.x.x/ for all documentation.

You must run "prepareclient" before you can retrieve this clients image.  
Do you want to run "prepareclient" now? (y/[n]) n  
"prepareclient" not run.  
Be sure to run "prepareclient" before you run "getimage" on the imageserver.

### Example 6. Running prepareclient

```
[root@imageserver]# prepareclient
```

Welcome to the SystemImager prepareclient command. This command may modify the following files to prepare your client for having it's image retrieved by the imageserver. It will also create the /etc/systemimager directory and fill it with information about your golden client, such as the disk partitioning scheme(s).

```
/etc/services      -- add rsync line if necessary
/etc/inetd.conf    -- comment out rsync line if necessary
                    (rsync will run as a daemon until shutdown)
/tmp/rsyncd.conf   -- create a temporary rsyncd.conf file with a
                    [root] entry in it.
```

All modified files will be backed up with the .beforesystemimager extension.

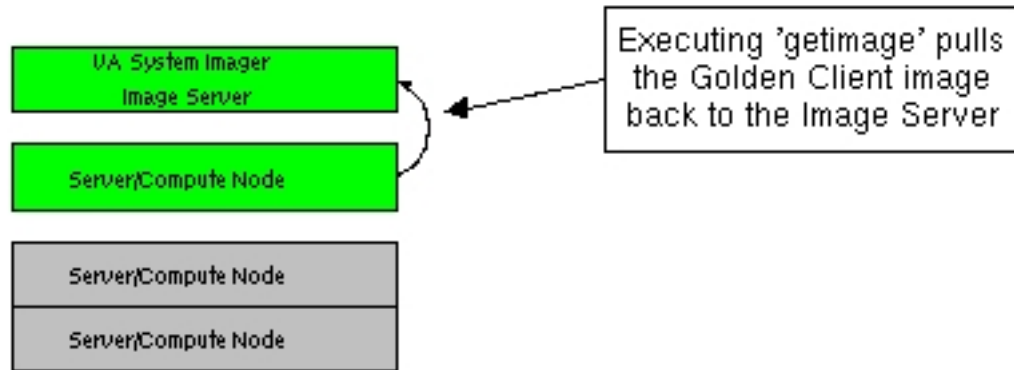
See "prepareclient -help" for command line options.

```
Prepare client for SystemImager? (y/[n]): y
Ok. Preparing client for SystemImager...
```

```
Creating /tmp/rsyncd.conf ...
Starting or re-starting rsync as a daemon.....done!
```

This client is ready to have it's image retrieved.  
You must now run the "getimage" command on the imageserver.

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

**Example 7. Pull the "golden client's" image to the image server**

The basic syntax is: "**getimage -golden-client** [client\_hostname] **-image** [image\_name]"

Where [client\_hostname] is the hostname or IP address of the "golden client" and [image\_name] is the name that you want to give to this image. There are many other options that can be seen with "**getimage -help**".

**Example 8. Running getimage**

```
[root@imageserver]# getimage -g my-golden-client -image web_server_image_v1
```

This program will get the "web\_server\_image\_v1" system image from "my-golden-client" making the assumption that all filesystems considered part of the system image are using ext2, ext3, or reiserfs.

This program will not get /proc, NFS, or other filesystems not mentioned above.

See "getimage -help" for command line options.

```
Continue? ([y]/n): y
```

```
Retrieving /etc/systemimager/mounted_filesystems from my-golden-client to check for moun
----- my-golden-client mounted_filesystems RETRIEVAL PROGRESS -----
receiving file list ... done
/var/spool/systemimager/images/web_server_image_v1/etc/systemimager/mounted_filesystems
wrote 132 bytes read 294 bytes 852.00 bytes/sec
total size is 180 speedup is 0.42
----- my-golden-client mounted_filesystems RETRIEVAL FINISHED -----
```

```
Retrieving image web_server_image_v1 from my-golden-client
----- web_server_image_v1 IMAGE RETRIEVAL PROGRESS -----
receiving file list ... done
./
bin/
bin/mt -> /etc/alternatives/mt
bin/pidof -> ../sbin/killall5
```

```
bin/rbash -> bash
bin/sh -> bash
boot/
boot/lost+found/
cdrom/
dev/
dev/MAKEDEV -> /sbin/MAKEDEV
dev/agpgart
dev/atibm
dev/audio
dev/audio1
dev/audio2
dev/audio3
dev/audioc1
dev/aztcd0
dev/bpcd

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

var/log/
var/log/exim/
var/log/ksymoops/
var/log/news/
var/run/
var/spool/
var/spool/cron/
var/spool/cron/atjobs/
var/state/
var/state/apt/
var/state/apt/lists/
var/state/logrotate/
var/tmp/
wrote 117490 bytes  read 134577117 bytes  413808.32 bytes/sec
total size is 134072667  speedup is 1.00
----- web_server_image_v1 IMAGE RETRIEVAL FINISHED -----
```

Press <Enter> to continue...

#### IP Address Assignment -----

There are four ways to assign IP addresses to the client systems on an ongoing basis:

- 1) `static_dhcp` -- A DHCP server will assign the same static address each time to clients installed with this image. Also see the "`mkdhcpstatic`" command.
- 2) `dynamic_dhcp` -- A DHCP server will assign IP addresses dynamically to clients installed with this image. They may be assigned a different address each time.

- 3) `static` -- The IP address the client uses during `autoinstall` will be permanently assigned to that client.
- 4) `replicant` -- Don't mess with the network settings in this image. I'm using it as a backup and quick restore mechanism for a single machine.

Which method do you prefer? [1]: 3  
You have chosen method 3 for assigning IP addresses.

Are you satisfied? ([y]/n): y  
Would you like to run the "addclients" utility now? (y/[n]): n

- 5. Run **addclients** on the image server to tell it which clients will receive which image, and to populate the image server's `/etc/hosts` and `/tftpboot/systemimager/hosts` files.

#### **Example 9. Running addclients**

```
[root@imageserver]# addclients
```

```
Welcome to the SystemImager "addclients" utility
-----
```

This utility has 3 sections.

"Section 1" will ask you for your hostname information.

"Section 2" will allow you to create softlinks from each client hostname to your "master" script in the `/tftpboot/systemimager/` directory.

Example: `www297.sh -> web_server_image_v1.master`

"Section 3" will ask you for IP address information that will be combined with the hostname information provided in Section 1 to create entries in `/etc/hosts` for each of these same clients. New entries will be appended to the end of `/etc/hosts`. If you specify new hostnames for existing IP addresses, those entries will be re-written in place to reflect the new host names.

Continue? ([y]/n):

```
addclients -- Section 1 (hostname information)
```

-----  
The next series of questions will be used to create a range of hostnames. You will be asked for your domain name, the base host name, a beginning number, and an ending number.

For example, if you answer:

```
domain name      = systemimager.org
base host name   = www
starting number  = 7
ending number    = 11
```

Then the result will be a series of hostnames that looks like this:

```
www7.systemimager.org
www8.systemimager.org
www9.systemimager.org
www10.systemimager.org
www11.systemimager.org
```

```
What is your domain name? []: mydomain.com
What is the base host name that you want me to use? []: server
What number should I begin with? []: 1
What number should I end with? []: 99
```

```
I will work with hostnames:  server1 through server99
                           in the domain:  mydomain.com
```

```
Are you satisfied? (y/[n]): y
```

```
addclients -- Section 2 (soft links to master script)
-----
```

Would you like me to create soft links to a "master" script so that hosts:

```
server1 through server99
```

```
can be autoinstalled with that image? ([y]/n):
```

Here is a list of available images:

```
debian_ide_2.2
debian_hwraid_ext2
web_server_image_v1
```

```
Which image would you like these hosts to receive? [web_server_image_v1]:
```

Your soft links have been created.

Press <Enter> to continue...

```
addclients -- Section 3 (adding or modifying /etc/hosts entries)
```

```
-----
```

It is necessary to have an entry for each client in "/etc/hosts".

I will ask you for your clients' IP addresses one subnet at a time.

Would you like me to make these entries for you? ([y]/n):

```
addclients -- Section 3 (adding or modifying /etc/hosts entries -- continued...)
```

```
-----
```

subnet 1

The first host in subnet 1 will be: server1

What is the starting IP address for subnet 1? []: 192.168.1.1

What is the ending IP address? []: 192.168.1.99

I will work with IP addresses: 192.168.1.1 through 192.168.1.99

Are you satisfied? (y/[n]): y

These entries have been added to /etc/hosts.

Press <Enter> to continue...

#### Example 10. Entries in /etc/hosts created by addclients

```
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
```

6. 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 easify the configuration of the DHCP configuration file (`/etc/dhcpd.conf`), SystemImager includes a utility called **mkdhcpserver**. This utility asks you for all the information it needs to create a DHCP configuration file that is appropriate for your installation of SystemImager. It is also possible to use DHCP to assign static IP addresses to your clients on an ongoing basis after installation. If you choose to do so, simply run the **mkdhcpstatic** command after all of your clients have had a chance to boot and be assigned an IP address. It will modify your `/etc/dhcpd.conf` file on the imageserver to include static entries for each of your hosts.

Alternately, hostname, imageserver, and networking information can be put in a configuration file on a floppy diskette. Or if you are using a running system's hard drive as the boot media, you can run **"updateclient -autoinstall -server <imageserver> -config 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 `/tftpboot/systemimager/hosts` file on the image server. A `local.cfg` file on a floppy will work with any of the autoinstall media. The configuration file can even be put on the autoinstall floppy itself! If you use a `local.cfg` file on a hard drive and on a floppy, the settings on the floppy will override the settings on the hard drive.

### Example 11. Running mkdhcpserver

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

```
Welcome to the SystemImager "mkdhcpserver" command.  This command
will prepare this computer to be a DHCP server by creating the
following file:
```

```
/etc/dhcpd.conf
```

```
If there is an existing file, it will be backed up as:
```

```
/etc/dhcpd.conf.beforesystemimager
```

```
Continue? (y/[n]): y
```

```
Type your response or hit <Enter> to accept [defaults].  If you don't
have a response, such as no first or second DNS server, just hit
<Enter> and none will be used.
```

```
What is your domain name? [localdomain.domain]: mydomain.com
What is your network number? [192.168.1.0]:
What is your netmask? [255.255.255.0]:
What is the starting IP address for your dhcp range? [192.168.1.1]:
What is the ending IP address for your dhcp range? [192.168.1.100]: 192.168.1.99
What is the IP address of your first DNS server? []: 192.168.1.200
What is the IP address of your second DNS server? []: 192.168.1.201
What is the IP address of your third DNS server? []:
What is the IP address of your default gateway? [192.168.1.254]:
What is the IP address of imageserver? [192.168.1.254]: 192.168.1.203
```

What is the air speed velocity of the common swallow? []:

Ahh, but seriously folks...

Here are the values you have chosen:

```
#####
DNS Domain Name:                mydomain.com
network number:                 192.168.1.0
netmask:                        255.255.255.0
starting IP address for your dhcp range: 192.168.1.1
ending IP address for your dhcp range:   192.168.1.99
first DNS server:               192.168.1.200
second DNS server:              192.168.1.201
third DNS server:
default gateway:                192.168.1.254
imageserver:                    192.168.1.203
#####
```

Are you satisfied? (y/[n]): y

The dhcp server configuration file (/etc/dhcpd.conf) file has been created for you. Please verify it for accuracy.

If this file does not look satisfactory, you can run this command again to re-create it: "mkdhcpserver"

WARNING!: If you have multiple physical network interfaces, be sure to edit the init script that starts dhcpd to specify the interface that is connected to your DHCP clients. Here's an example:

Change "/usr/sbin/dhcpd" to "/usr/sbin/dhcpd eth1".

Also, be sure to start or restart your dhcpd daemon. This can usually be done with a command like "/etc/init.d/dhcpd restart" or similar.

### Example 12. Running updateclient with the "-autoinstall" and "-config" options

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

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

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



```
Ignoring entry 'delay'
Ignoring entry 'default'
Added linux
Added systemimager *
```

[illegible]

```
#
# "SystemImager" - Copyright (C) 1999-2001 Brian Elliott Finley <brian@systemimager.org>
#
# 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
```

7. Autoinstall the "golden" image on other machines.

There are four methods for autoinstalling the clients:

- Boot the system from a floppy diskette.

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

- Boot the system from a CDROM.

Run **mkautoinstalled** to create an ISO image that can be burned to CDROM. This CDROM can then be used to boot your autoinstall clients and can also be used with any machine.

- Boot the system from it's own hard drive.

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

**Example 13. Copy over the updateclient command**

```
[root@server7]# rsync -av imageserver::tftpboot/systemimager/updateclient /usr/sbin
receiving file list ... done
updateclient
wrote 115 bytes  read 14560 bytes  29350.00 bytes/sec
total size is 14445  speedup is 0.98
```

**Example 14. Booting the autoinstall media from a system's hard drive**

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

- Boot the system from the network using PXE if your system supports it. This is usually enabled through a BIOS setting.

PXE booting can be flaky and client side firmware is less than consistent. If you must boot without physically touching your machines, may I suggest that you try the *-autoinstall* option to **updateclient** first and see if it meets your needs.

SystemImager comes with the **mkbootserver** utility to help configure a PXE server.

**mkbootserver**, is new with the 2.0.0 release, and will likely not work with all PXE clients. If it fails to work with your configuration, please submit a bug report.

## 10. I Want More Details! What were the Design Goals?

- Images should be pulled from a working system.
- Completely unattended installs were a must.
- The unattended install system had to be able to repartition the destination drive(s).
- One of the main design goals was ease of use. This had to be a tool that could be used by a system administrator that didn't necessarily understand how it worked.
- It was also necessary for it to install easily and quickly so that it could be useful right away without a lot of site specific customization.

- Images should be stored as normal files to allow for incremental upgrades as opposed to "dd" style block level images of physical disks.
- It had to be independent of any and all packaging systems (such as RPM) in order to easily accomodate different distributions.
- It should be able to store multiple images, for different types of systems and for revision control.
- It must provide a mechanism for unattended install clients to know which image to install.
- Once a client was installed, it should be able to update itself to a new or updated image.
- It should have a command line interface that lent itself well to being wrapped with a GUI.

## 11. I Want More Details! Please Describe the Architecture.

SystemImager began as a series of utilities written as shell scripts. Minimal system requirements were considered a top priority. As SystemImager matured, and the utilities became internally more complex, it became clear that shell scripts were falling short of the need. Perl was chosen to pick up the yoke and has allowed for cleaner, more advanced code. It was determined that Perl is installed as part of most "base" Linux installs and therefore was a reasonable choice from a minimal requirements perspective.

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

There are other methods available for doing automatic installs, such as RedHat's KickStart which installs systems based on a list of pre-defined packages. But package based installs are very limiting in that they generally don't have an automated way for dealing with non-packaged files. If you re-compile your kernel, add a piece of non-packaged software, or modify certain configuration files, you are usually required to do some sort of scripting or programming to deal with these "special cases".

In order keep imaging simple, SystemImager uses images that are based on a working installed system. We call this system a "golden client". Just get one of your machines working exactly the way you want and pull it's image to the imageserver with the "getimage" command. You can re-compile your kernel, install custom software, and do any configuration file tweaking you like. SystemImager will get it all.

Now that you have your golden client configured, we need to run the "prepareclient" command. prepareclient will collect the partition information from your disks and put it in the `/etc/systemimager/partitionschemes` directory. A file will be created in this directory for each of your disks and will contain that disk's partition information. prepareclient will also create an rsync(1) configuration file (`/tmp/rsyncd.conf`) and start rsync in server mode (**rsync --daemon**). This 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. This helps avoid security concerns of sharing a golden client's root filesystem via rsync.

On the imageserver we now run the getimage command. Here's an example:

```
"getimage -golden-client 192.168.1.1 -image my_webserver_image_v1"
```

getimage contacts the golden client and requests its `/etc/systemimager/mounted_filesystems` file. This file 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. Unsupported filesystems are everything else including things like proc, devpts, iso9660, etc. getimage then pulls the golden client's entire system image, excluding the filesystems in the exclusion list. The files are pulled 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.

getimage can also be used to update an existing image. By simply specifying an existing image name, you are asking getimage to update that image to match the files on your golden client. In this case, 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. This is one way to keep an image updated when new security patches or other system updates come out. However, the recommended method is to never overwrite a known working image, so that you have a form a revision control. This is not true revision control, where individual file revisions are tracked on a line by line basis. It is revision control on an image by image basis. This form of revision control also ties in to the updateclient command which will be discussed later. 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/`.

After getimage has pulled the files to the image directory on the imageserver it creates a customized autoinstall script. The autoinstall script in our example would be named `"my_webserver_image_v1.master"`. All autoinstall scripts are placed in the `/var/lib/systemimager/scripts` directory. The disk partitioning information left behind by the prepareclient command is used to add the necessary commands to re-partition the disk(s) on the autoinstall clients. File system information is taken from the `/etc/fstab` file in the image (i.e.: `/var/spool/systemimager/images/my_webserver_image_v1/etc/fstab`) and is used to determine the appropriate file system creation commands and to determine mount points for the autoinstall process. Based on command line options passed to getimage or questions it has asked, certain networking information is added to the autoinstall script. 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.

After running getimage, you will run the addclients command. addclients will ask you for the series of hostnames that you will be installing by combining a base host name and a number range. For example, if your base host name is "www", and your number range is from "1" to "3", then the resultant host names would be "www1, www2, www3". It will then prompt you to choose the image that will be installed to these hosts and will create soft links for each hostname that point to the master autoinstall script for that image. For example: `"www3.sh -> web_server_image_v1.master"`. If the image is updated and you choose to allow getimage to also update the master autoinstall script, then each of the associated soft links therefore 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.

addclients will then prompt you for the IP address information for these hosts and will re-write the imageserver's `/etc/hosts` file accordingly, then copy this file to `/var/lib/systemimager/scripts/hosts`. The latter file is used during the autoinstall process for clients using DHCP to determine their host names.

The unattended install portion is flexible and can work with most any hardware available. It is also easily modified to work with new or special hardware. A miniature Linux distribution (Brian's Own Embedded Linux) is used for autoinstalls. It consists of a customized kernel and an initial ram disk which contains only the specific commands and utilities necessary to perform autoinstalls. The same kernel and initial ram disk (initrd.gz) can be used to boot from floppy disks, CDROMs, the network, or any running Linux system's local hard drive. The commands "mkautoinstalldiskette" and "mkautoinstallcd" make use of the syslinux(2) utility to create floppies and CDROMs that will boot the SystemImager kernel and initial ram disk. pxelinux(2), which is a sister tool to syslinux, allows the same kernel and initial ram disk to boot PXE capable machines from the network. A configuration file is needed by syslinux and by pxelinux, but the two tools are able to use the same configuration file and SystemImager handles this for you.

The autoinstall kernel is compiled to contain all the necessary drivers for the majority of systems. Custom kernels can be compiled to meet special disk and network driver requirements. To use a custom kernel, simply copy it to /usr/share/systemimager/i386-boot/kernel. All of the autoinstall media is created from /usr/share/systemimager/i386-boot/kernel and /usr/share/systemimager/i386-boot/initrd.gz on the image server.

Once the kernel has booted, it mounts the initial ram disk as it's root filesystem. It then executes an initialization script on the ram disk that has been written to do SystemImager specific things. This script will use either a configuration file (/local.cfg), DNS, or a combination of DHCP and the /var/lib/systemimager/scripts/hosts 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 retrieved from the image server to find it's IP address and can therefore determine it's hostname. Finally it retrieves an autoinstall script from the image server based on it's hostname and executes it. The autoinstall script is image specific. This is how a client determines which image it will receive. Here is a summary: IP address -> hostname -> image specific autoinstall script named with hostname.

The most common way to assign IP addresses to autoinstall clients is DHCP. To easify the configuration of the DHCP configuration file (/etc/dhcpd.conf), SystemImager includes a utility called mkdhcserver. This utility asks you for all the information it needs to create a DHCP configuration file that is appropriate for your installation of SystemImager. It is also possible to use DHCP to assign static IP addresses to your clients on an ongoing basis after installation. If you choose to do so, simply run the mkdhcstatic command after all of your clients have had a chance to boot and be assigned an IP address. It will modify your /etc/dhcpd.conf file on the imageserver to include static entries for each of your hosts.

Alternately, hostname, imageserver, and networking information can be put in a configuration file on a floppy diskette. Or if you are using a running system's hard drive as the boot media, you can run **"updateclient -autoinstall -server <imageserver> -config eth0"** which will create a local.cfg file at the root of the client's hard drive containing the existing live network settings.

When the autoinstall client boots, it will look for this file and use the provided values instead of getting them from DHCP and the /var/lib/systemimager/scripts/hosts file on the image server. A local.cfg file on a floppy will work with any of the autoinstall media. The configuration file can even be put on the autoinstall floppy itself! If you use a local.cfg file on a hard drive and on a floppy, the settings on the floppy will override the settings on the hard drive.

The format of this configuration file is simply VARIABLE=value for all the appropriate settings. The name of this file must be local.cfg and it must exist on the root of the floppy or hard drive. The floppy can be formatted with either ext2 or fat. An example local.cfg file can be found with the documentation files which are usually installed in /usr/share/doc.

Sometimes you will want to update an image on your image server. There are a couple of ways to do this.

The first way is to directly edit the files in the image directory. The best way to do this is to chroot into the image directory. Once you have done the chroot, you can work with the image as if it were a running machine. You can even install packages with apt-get or RPM, for example. The second way is to run the getimage command again, specifying a golden client that has been modified in the desired way. Again, only the parts of the files that have changed will be pulled across. Files that have been deleted on the golden client will also be deleted in the image. You are also given the option to update the master autoinstall script for the image or to leave it alone. The advantages of this method 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, the updateclient command can be used to update a client system to match a new or updated image on the image server. Let's say that you've installed your company's 300 web servers and a security patch comes out the next day. You simply update the image on the image server and run updateclient on each of your web servers. Only the modified files are pulled over, so it takes very little time, and your entire site is patched! It is recommended that you 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 hosed up your entire web farm, you simply do an updateclient back to the last known working image!

By incorporating some modifications sent in by A.L. Lambert of epicRealm, using the "updateclient" command with the -autoinstall option will copy the autoinstall kernel and initial ram disk to the local hard drive of an autoinstall client that is currently running, but needs to be re-deployed. It will then modify the /etc/lilo.conf file to include an appropriate entry for the new kernel and initial ram disk and will make this new kernel the default. The next time the client system is booted, it will load the SystemImager kernel and initial ram disk, which will begin the autoinstall process! This means that you can remotely re-deploy any running Linux machine without having to have someone feed the machine a floppy or CD and without having to reconfigure the BIOS to boot off the network (which can be quite squirrely (<http://www.dictionary.com/cgi-bin/dict.pl?term=squirrely>) with some BIOSes).

## 12. Troubleshooting

### 12.1. PXE installations work fine for a while, but eventually clients no longer boot the autoinstall kernel.

Are you using xinetd? xinetd 2.1.8.9pre11 (and assumably earlier versions) had a race condition which causes the tftp server to suddenly stop responding. this has supposedly been fixed in version 2.1.8.9pre13. Also, xinetd and inetd have default limits on how many connections can be spawned in a 60 second period. See the inetd or xinetd manpage for details on increasing this limit.

### 12.2. My client fails to assign a hostname to itself, making the autoinstall fail.

Ryan Braby reported the following problem:

[The install] fails when you have addresses like the following:

```
###.###.###.2
###.###.###.21
###.###.###.22
###.###.###.23
###.###.###.24
etc.
```

For the node with the `###.###.###.2` address, the script tries to assign multiple hostnames, and then fails to get any install script.

This should be fixed in the next release (1.4.2).

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

Goran Pocian reported an instance of horrible **updateclient** performance which went away when he upgraded from kernel 2.2.17 to 2.2.18.

Also, he noted that if you mount an NFS filesystem after executing **prepareclient**, **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 today to discuss these issues.

There are 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 it's Send-Q
- machine B has no data in it's 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 numbers 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.

## 12.4. My autoinstallcd doesn't boot.

There was a problem with the following RPM: `syslinux-1.48-1.i386.rpm` Download and install a newer syslinux RPM from <http://systemimager.org/>



## 12.5. When making the autoinstalldiskette, my system gives me an error involving "dd" or "mount".

You are using a pre v0.19 version of SystemImager. Please download the latest version from <http://systemimager.org/>.

If you must use a pre v0.19 version for some reason, be sure that your kernel has "ramdisk" support. Or that you have ramdisk support with a module. If you are using a module, be sure that it is loaded with the **modprobe** command.

But it's probably easier to just get the latest version...

## 12.6. 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.

## 12.7. My autoinstall client booted up and said "dhcp didn't work", but when I do an ifconfig eth0 it has an IP address.

Are you using a pre 1.0 version of SystemImager? If so, please upgrade.

If for some reason you can't upgrade, then:

Are you connected to a switch? Most switches will wait a period of time (usually 30 seconds) after a connected system's interface has come up before transmitting on that port. Newer versions of the autoinstalldiskette bring the ethernet interface up and wait 45 seconds or so before making a DHCP request. It will then wait 35 seconds or so to give the system time to receive an address. You could be using an autoinstalldiskette that does not wait the proper time for your switch and is giving up before it should.

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.

# 13. Frequently Asked Questions

## 13.1. Where are the images stored?

The images are stored in `/var/lib/systemimager/images`

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

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

And create a soft link to the new directory.

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

## 13.2. How do I make an autoinstall diskette?

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

## 13.3. How do I make an autoinstall CD?

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

## 13.4. How do I make an autoinstall punch card?

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

## 13.5. How do I set up my autoinstall clients so that the console is available via the serial port?

Making SystemImager work with a serial console  
Michael S. Fischer, <michael@auctionwatch.com>  
Last modified: 01/04/26 19:20:27

The current version of SystemImager as of this writing (1.4.1)  
does not support PXE booting of clients with serial console support.  
This document describes how to rebuild the kernel and initrd such  
that serial console support is possible.

Step 1: Download the sources

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

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

Step 2: Prepare the kernel

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

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

```
# CONFIG_SERIAL is not set
```

and change it to read

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

Then, build the kernel:

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

### Step 3: Install the kernel

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

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

### Step 4: Fix the initrd

The initrd, or initial RAM disk containing the root filesystem, needs to be unpacked, mounted, and its contents edited. Here's how:

```
# cp /tftpboot/initrd.gz /tmp  
# cd /tmp && gunzip initrd.gz  
# mkdir /mnt2  
# mount /tmp/initrd /mnt2 -o loop  
# cd /mnt2/dev  
# rm -f console  
# mknod -m 622 console c 5 1  
# mknod -m 600 ttyS0 c 4 64
```

Once you've finished this, you should unmount the initrd, recompress it, and return it to /tftpboot:

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

### Step 5: Edit syslinux.cfg

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

```

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

```

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

## 13.6. Does the DHCP server have to be on the image server?

No. If you are using DHCP, you can use "option-100" and set it's value to the IP address of the image server. If you use "mkdhpstatic" 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.

Yes, we now know that option-100 has an official use. We are working on either getting an official number assigned or using a number from the private number range.

## 13.7. How do I add a driver for a special card to the autoinstall client?

All you have to do is start with the linux kernel source and the `.config` file that you can find in the CVS repository on SourceForge:

SystemImager CVS Repository  
(<http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/~checkout~/systemimager/systemimager/>)

The 1.5.0 autoinstall media is based on Linux 2.2.18 and has various patches applied. You can find this source on the SystemImager web site.

Put the `.config` file in your `/usr/src/linux` directory as `.config` and run **make menuconfig** or your favourite kernel config utility. Choose the driver for the card you need.

However! This will probably make the kernel too large for the floppy. So you will want to de-select drivers for many of the cards that will not be used. I would recommend de-selecting all of the Ethernet cards except for the ones you need.

Do the standard:

```

make dep
make clean
make bzImage

```

Please refer to the Kernel-HOWTO (<http://www.caldera.com/LDP/HOWTO/Kernel-HOWTO.html>) for more information on recompiling a kernel.

Then copy the resultant kernel over the kernel ( `/usr/share/systemimager/i386-boot/kernel`) installed in `/usr/share/systemimager/i386-boot` by SystemImager.

You should now be able to do a **mkautoinstalldiskette**, **mkautoinstalled**, or boot off the network.

### 13.8. How do I exclude files or directories during a **getimage**?

As of 2.0.0 there is not support for this. Look for this feature soon.

### 13.9. Do I have to do anything to prepare a client from which I will get an image?

Yes. Install the **systemimager-client** package. If this package is already installed, all you have to do is run the **prepareclient** command prior to running **getimage** from the image server.

You will also want to add any software, configure any files, and do any tweaking to make the system just how you like it!

### 13.10. Can I use the **autoinstalldiskette** or **autoinstalled** on more than one machine?

Yes. The **autoinstall** media is generic, and can be on any machine that you want to **autoinstall**.

### 13.11. How do I push an image to a client?

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

### 13.12. Ok, so how do I pull an image to a client?

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

They are:

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

Search this document for **mkautoinstalled** and **mkautoinstalldiskette** for more information.

### 13.13. How does an autoinstall client know which image to install?

In order to better understand the answer, let me lead up to it with the steps the autoinstall client goes through:

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

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

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

### 13.14. How do I upgrade to a new version?

If you installed SystemImager as of 1.0 or later, you can simply install the new version on top. Using a packaged version (`.rpm`, `.deb`) is recommended as this will prevent cruft build-up on your system.

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

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

### 13.15. But I want to assign static IPs to my clients!

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

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

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

After booting your systems, run **mkdhcpstatic**. It will re-write your `/etc/dhcpd.conf` file associating each clients 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 receives.

Note: Static IP addresses, assigned by DHCP, are the author's preferred method for administering IP on a large number of systems, if static IP addresses are actually required.

## 13.17. What kind of performance can I expect?

Here are some unofficial numbers that I came up with. Your experience may be quite different based on the speed of your network, the performance of your image server and clients, and the size of the image.

If you can send me statistics from your experience, I'll add them below.

**Table 3. SystemImager Performance Statistics**

Size of Image	Clients	Network Type	Switched	Elapsed Time
2.5 GB	21	100BaseTx	yes	1.5 Hours
1.5 GB	20	100BaseTx	yes	.5 Hours

## 13.18. How do I update an image on the image server?

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

1. Make the changes to one of your clients and run the **getimage** again.
  - You can specify the same image name, in which case the current image will be updated (only changes are pulled across).
  - Or you can specify a new image name and have a form of revision control. (This is by far the recommended method)

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

2. Modify the files directly. You can simply `cd` into the appropriate image directory and edit the files there, or (recommended) you can `cd` into the image directory and run **'chroot . sh'**. This will change your working root directory to the root of the image you want to manipulate. You can then run **rpm** command 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.

## 13.19. How do I update a client to match an image?

Once you have updated an image on the image server, you can then update your clients to reflect this new or modified image. (It is not necessary to do a complete re-autoinstall.) You will find the command, **updateclient**, on your clients. This command takes as its parameters the name of the image server and the name of the image that you want to update the client to. Run **updateclient -help** to get more information about this command.

If you use the revision control method recommended in "How do I update an image on the image server?" FAQ, then you will be able to bring your production environment back to a known state (by doing an **updateclient** to the last working image) after having done an **updateclient** to a not quite so thoroughly tested image.

The file `/etc/systemimager.exclude` on your clients is used to exclude files and directories from being updated by the **updateclient** command. Please take a look at this file and modify it to suit your local environment.

## 13.20. What is `systemimager.exclude` used for?

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

## 13.21. How do I build a new "front-end" server?

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

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

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

This is a problem since our front-end which the image will be captured from has a SCSI hard drive and a Gigabit ethernet card. Our 64 compute nodes on the other hand have IDE disk drives and fast ethernet cards.

Are we out of luck?

Nope. Just a little customization ahead of you...

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

**vi /var/spool/systemimager/images/\$imagename/etc/conf.modules**

I believe your entry will look like this:

```
alias eth0 tulip
```



You will need to edit the `fstab` file in the image:

```
vi /var/spool/systemimager/images/$imagername/etc/fstab"
```

Be sure it says `/dev/hdaN` instead of `/dev/sdaN`.

You will need to specify the partition information in the `$imagername.master` script that was created when you did your **getimage**.

```
vi /var/spool/systemimager/installstuff/$imagername.master
```

You will find a section that looks like this:

```
# Here's an example of how to customize a disk:
# start at the beginning 0, give 20M /dev/sda1 for /boot, assume partition type 83
# start where left off, 512M /dev/sda2 for swap, partition type 82
# start where left off, give the rest of the disk to /dev/sda3 for /, assume partition type
#sfdisk -uM /dev/sda <<EOF
#0,20
#,512,82
#,
#;
#EOF
```

Uncomment the lines from `sfdisk` down to `EOF` (`EOF` must be flush left) and change the values to specify your partitions according to the example.

If you don't want to have to think so much, then you can simply do the following command on a client that is already partitioned correctly:

```
"sfdisk -d /dev/hda" (Assuming you only have one disk and that disk is /dev/hda.)
```

Then copy the output into the `$install.master` script and delete the existing entries for the SCSI disk.

Search the `$imagername.master` script for all entries of your old drive `/dev/sda` and be sure that they are changed to the appropriate drive `/dev/hda`. This will be for the **mke2fs** and **mkswap** sections.

Proceed as normal.

## 13.22. How do I edit the scripts so only the FIRST disk is `sfdisk`'ed? I MUST leave the other disks alone.

After you run **prepareclient**, look in the `/etc/partitionschemes` directory on that client. You will find a file that has the partition information for each of that clients disks. Simply delete all except for the primary disk.

Now you can do your **getimage** and everything else like normal.

The only other thing to verify is that the `fstab` file in the image has no entries for the other disks.

**vi /var/spool/systemimager/images/\$imagename/etc/fstab**

### **13.23. 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.**

This is accomplished with the **updateclient** command on the client.

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

### **13.24. Is there a log file where autoinstall client status is kept?**

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

### **13.25. What other software is this based on?**

SystemImager is based on **rsync(1)** and also makes use of:

- **systemconfigurator(1)**
- **busybox(1)**
- **syslinux(2)**
- **pxelinux(2)**

Versions 0.23 and earlier made use of `tomsrtbt(4)`.

# SystemImager Glossary

## **addclients**

Tells your image server which image to install on your autoinstall clients. It does so by creating soft links to the master autoinstall script with the name of each host that will receive that image. It also allows you to populate the `/etc/hosts` file with sequential host names and IP addresses. This information in `/etc/hosts` is necessary for certain SystemImager operations.

## **autoinstall media**

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

## **autoinstall script**

A unique autoinstall script is created for each image and is used by the autoinstall client as part of the autoinstall process. The names of autoinstall scripts begin with the image name and end in `.master`. For example: `my_webserver_image_v1.master`

## **daemon (<http://www.dictionary.com/cgi-bin/dict.pl?term=daemon>)**

This is not specifically a SystemImager term, but it really bugs me when people pronounce this word as "daymen". It is just another, more interesting, less evil appearing way of spelling demon. And just to be clear, in computer terms, a daemon is a program that lies in wait for something to trigger it into action. An example of this is a web server daemon waiting for someone to request a web page.

## **getimage**

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

## **golden client**

A machine from which an image is taken. Golden clients are manually installed and customized to taste.

## **image server**

The machine that will hold and distribute the images.

**local.cfg**

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

**mkdhcpserver**

An easy way to create a SystemImager appropriate `/etc/dhcpd.conf` file. DHCP can be used to assign IP addresses to autoinstall clients.

**mkdhcpstatic**

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

**mkbootserver**

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

**prepareclient**

This command is run on the golden client immediately prior to running `getimage` on the image server.

**updateclient**

A command that is executed on client systems allowing them to be updated or synchronized to a new or updated image after the initial autoinstall. `updateclient` enables software and content distribution.