Configuring JumpStart™ Servers to Provision Sun™ x86-64 Systems

Pierre Reynes, Network Systems Group

Sun BluePrints™ OnLine—February 2005

http://www.sun.com/blueprints

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650 960-1300

Part No. 819-1692-10
Revision 1.0, 2/4/05
Edition: February 2005

Copyright 2005 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, California 95054 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at `http://www.sun.com/patents` and one or more additional patents or pending patent applications in the U.S. and in other countries.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Java, Sun BluePrints, SunSolve, SunSolve Online, docs.sun.com, JumpStart, N1, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government Rights—Commercial use. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2005 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, California 95054 Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuels relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plus des brevets américains listés à l'adresse `http://www.sun.com/patents` et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats - Unis et dans les autres pays.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Certaines parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque enregistree aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, Java, Sun BluePrints, SunSolve, SunSolve Online, docs.sun.com, JumpStart, N1, et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REPONDRE A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.

Please
Recycle

1

Introduction

Organizations are constantly challenged to deploy systems throughout the enterprise with consistent and reliable configurations. Solaris JumpStart technology provides a mechanism for fully automating the Solaris™ Operating System (Solaris OS) installation process. With the ability to locate installation information over the network or from a local CD-ROM drive, and use customized profiles, JumpStart™ facilitates the rapid and consistent deployment of Solaris OS-based systems.

Many organizations have relied on UltraSPARC®/Solaris platforms for years, and use JumpStart technology for operating system deployment. With the introduction of Sun x86-64 systems, organizations are now seeking ways to use existing JumpStart servers to deploy the Solaris OS and Linux operating environment on Sun x86-64 platforms. This Blueprint article describes how to modify existing JumpStart servers to support the deployment of the Solaris OS and Linux operating environment on Sun x86-64 platforms, as well as how to use standard Linux installation tools for configuring Sun x86-64 platforms.

How This Article is Organized

- Chapter 2, “Creating a Standard JumpStart Server on SPARC/Solaris Systems” provides an overview of JumpStart server component installation for the Solaris OS running on Sun UltraSPARC-based systems.
- Chapter 3, “JumpStart and the Dynamic Host Configuration Protocol,” describes best practices for supporting DHCP with JumpStart.
- Chapter 4, “The Preboot eXecution Environment,” provides an overview of the Preboot eXecution Environment and describes the differences between JumpStart stages on SPARC- and x86-64 based platforms.
- Chapter 5, “Configuring JumpStart for Solaris x86 Provisioning,” explains how to configure existing JumpStart servers for Solaris x86 provisioning.

- Chapter 6, “Configuring Red Hat Enterprise Linux Provisioning (Kickstart),” outlines the differences between Solaris JumpStart and Red Hat Kickstart, and explains how to install Red Hat Enterprise Server AS on Sun Fire™ V20z and Sun Fire V40z servers using Kickstart.
- Chapter 7, “Configuring SuSE Linux Provisioning (AutoYAST),” outlines the differences between Red Hat Kickstart and SuSE AutoYaST, and explains how to install SuSE Linux Enterprise Server 9 on Sun Fire V20z and Sun Fire V40z servers with AutoYaST.
- Chapter 8, “Special Notes on Sun x86-64 Platforms,” provides notes on configuration specific to Sun x86-64 platforms and software.
- Chapter 9, “Additional Tools and Technologies,” describes additional tools that take advantage of JumpStart technology.
- Chapter 10, “Summary,” provides background on the authors and links to other sources of information.

This article assumes a basic understanding of the JumpStart technology framework and the services it provides. It is also assumes familiarity with the LU 2.0 framework and its various uses.

Typographic Conventions

TABLE 1-1 shows the typographic conventions used in this article.

TABLE 1-1 Typographic Conventions

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output	**`% su`** `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized	Read Chapter 6 in the *User's Guide*. These are called *class* options. You *must* be superuser to do this.
AaBbCc123	Command-line placeholder text; replace with a real name or value	To delete a file, type `rm` *filename*.

2

Creating a Standard JumpStart Server on SPARC/Solaris Systems

This chapter provides an overview of the installation process for JumpStart server components in the Solaris OS running on Sun UltraSPARC-based systems. More information on JumpStart server installation can be found in the *Solaris Installation Guide: Network-Based Installations* (817-5504-10) on *docs.sun.com*.

JumpStart Server Components

A JumpStart Server consists of several components:

- *Install Client*, the target system to be installed or upgraded.
- *Boot Server*, the network providing a failsafe operating system to the installing client. The boot image is architecture independent, providing basic operating system services to all hardware supported by that operating system release. The Boot Server provides RARP, TFTP and bootparam services.
- *Configuration Server*, a system that helps client systems determine unique profile information. Partition sizes, lists of software components to install, begin and finish scripts, and more are specified in a profile served by the Configuration Server.
- *Install Server*, the source of the software packages to be installed on the client.

Example JumpStart Server Configuration

Configuring a JumpStart server requires two UltraSPARC-II based systems or newer (one as server, one as client), each configured with 9 GB disk drives or larger. The Solaris 9 Operating System 04/04 or later must be installed on the server.Access to the Solaris Operating System installation CDs is required for the configuration process.

For simplicity, the following configuration example consolidates the Boot, Install, and Configuration servers onto a single Solaris JumpStart Server. Table 2-1 provides important hardware and network information to be used in the configuration process.

TABLE 2-1 Basic hardware and network configuration information

JumpStart Server	• IP Address	`192.168.2.2`
	• Hostname	`jumpstart`
JumpStart Client	• MAC Address	`8:0:20:a8:d4:22`
	• IP Address	`192.168.2.20`
	• Hostname	`client`
Network Topology	• Flat (hub/switch) 10/100 Mbps	
	• Netmask: `255.255.255.0` (class C network)	
Domain Name Service (DNS)	• No naming service, router or gateway	

If a domain name service (DNS) is used, it is important to use a fully-qualified domain name (FQDN) for each host. Each line in the `/etc/hosts` file represents a separate node, and must contain tab- or space-delimited values for the system's IP address, FQDN and hostname. If a naming service is not used, the time server must be set to `localhost`.

Configuring the JumpStart Server

Configuring the JumpStart server consists of several tasks (Figure 2-1).

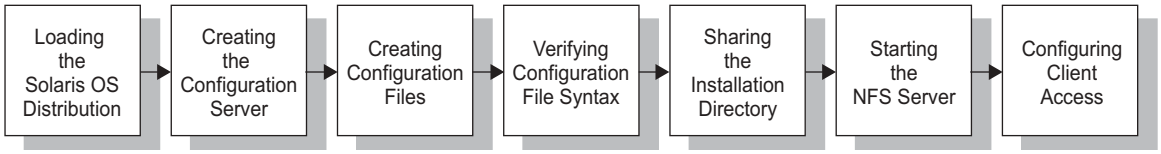


FIGURE 2-1 JumpStart server configuration can be categorized by several steps

Loading the Solaris OS Distribution

The following steps outline the procedure for loading the Solaris OS distribution for a combined Install/Boot Server. Notations are included for changes needed when installing separate Install and Boot servers.

1. Create the parent installation directory for operating system distributions.

```
jumpstart> mkdir /export/install
```

2. Create a subdirectory for the Solaris OS distribution. We recommend including important attributes, such as the update version and architecture, in the directory name. In the following example, a directory is created for Solaris 9 Update 9/04 for the SPARC architecture.

```
jumpstart> mkdir /export/install/s9s_0904
```

3. Insert the Solaris OS Disk 1 of 2 CD-ROM into the CD drive and copy the Solaris OS distribution to the newly created installation directory.

```
jumpstart> cd /cdrom/cdrom0/s0/Solaris_9/Tools
jumpstart> ./setup_install_server /export/install/s9s_0904

Verifying target directory...
Calculating the required disk space for the Solaris_9 product
Calculating space required for the installation boot image
Copying the CD image to disk...
Copying Install Boot Image hierarchy...
Install Server setup complete

jumpstart> cd /
jumpstart> eject cd
```

4. If configuring separate Install and Boot Servers, the **-b** option must be specified on the setup_install_server command line. More information on installing separate Install and Boot servers can be found in the *Solaris 9 9/04 Installation Guide* (817-5768) or the *Solaris 10 Installation Guide: Network-Based Installations* (817-5504) located on *docs.sun.com*.

```
jumpstart> ./setup_install_server -b /export/install/s9s_0904
```

5. Insert the Solaris OS Disk 2 of 2 CD-ROM into the CD drive and copy the remainder of the Solaris OS distribution to the installation directory. If configuring separate Install and Boot Servers, eliminate this step.

```
jumpstart> cd /cdrom/cdrom0/s0/Solaris_9/Tools
jumpstart> ./add_to_install_server /export/install/s9s_0904

The following Products will be copied to
/export/install/s9s_0904/Solaris_9/Product:

Solaris_2_of_2

If only a subset of products is needed enter Control-C and
invoke ./add_to_install_server with the -s option.

Checking required disk space...

Copying the Early Access products...
0 blocks

Copying Top Level installer...
115881 blocks

Copying Tools Directory...
4771 blocks

Processing completed successfully.

jumpstart> cd /
jumpstart> eject cd
```

Creating the Configuration Server

Once the Solaris OS distribution is loaded onto the server, the Configuration Server can be created using the following steps.

1. Create a separate subdirectory to store Solaris JumpStart configuration files.

```
jumpstart> mkdir /export/install/jumpstart
```

2. Insert the Solaris OS Disk 1 of 2 CD-ROM into the CD drive and copy the `check` script from the JumpStart sample directory.

```
jumpstart> cd /cdrom/cdrom0/s0/Solaris_9/Misc/jumpstart_sample
jumpstart> cp check /export/install/jumpstart
```

Creating Configuration Files

While the distribution CD contains several sample files, the following procedures manually create all required configuration files, including the `rules` file, `begin` script (optional), `finish` script (optional), class file or profile, and `sysidcfg` file.

1. Move to the newly created directory for JumpStart configuration files.

```
jumpstart> cd /export/install/jumpstart
```

2. Create the rules file `/export/install/jumpstart/rules`. This text file is used to create the `rules.ok` file needed for a custom JumpStart installation. It contains a lookup table consisting of one or more rules that define matches between system attributes and profiles.

```
jumpstart> cat > rules
any - begin profile finish
CTRL-d
```

3. Create a `begin` script. A `begin` script is run prior to the actual installation. If the optional `begin` script is used, it should be created at this time using the following commands:

```
jumpstart> cat > begin
#!/bin/sh
echo “Begin Script for JumpStart”
echo “System Information: uname -a”
CTRL-d

jumpstart> chmod 755 begin
```

4. Create a `finish` script.A `finish` script runs following the installation. If the optional `finish` script is used, it should be created at this time using the following commands:

```
jumpstart> cat > finish
#!/bin/sh
echo “This is the finish script”
echo “I could set parameters like /etc/defaultrouter”
CTRL-d

jumpstart> chmod 755 finish
```

5. Create a class file, or profile. *Class files* enable automated installations to be performed with custom directives, providing a mechanism for installing Solaris OS with unique package sets, partition tables, and more. A universal profile can be used for all systems, or separate profiles can be created and invoked based on system characteristics, such as platform type or hostname. Custom profile types are defined in rules files and can contain a variety of keywords to help determine which profile should be used. More information on class files and profiles can be found in the *Solaris 9 9/04 Installation Guide* (817-5768) or the *Solaris 10 Installation Guide: Network-Based Installations* (817-5504) located on *docs.sun.com*.

   Example class file, or profile:

```
# install_type MUST be first
install_type    initial_install

# start with the minimal required number of packages
cluster         SUNWCXall
cluster         SUNWCapache     delete
cluster         SUNWCpcmc       delete
cluster         SUNWCpcmcx      delete
cluster         SUNWCthai       delete
cluster         SUNWClp         delete
cluster         SUNWCnis        delete
cluster         SUNWCppp        delete

# want to define how the disk is used - not use defaults
partitioning    explicit
filesys         rootdisk.s1     1024    swap
filesys         rootdisk.s0     free    /

# install system as standalone
system_type     standalone
```

6. Create a `sysidcfg` file. The `sysidcfg` file contains system identification configuration data, including locale, time zone, terminal type and more.

Example `sysidcfg` file:

```
system_locale=en_US
timezone=US/Pacific
timeserver=localhost
terminal=vt100
name_service=NONE
security_policy=NONE
root_password=JdqZ5HrSDYM.o
network_interface=PRIMARY {protocol_ipv6=no netmask=255.255.255.0}
```

Verifying Configuration File Syntax

Once all JumpStart configuration files are created, the `check` script should be run to verify all file contents are syntactically correct.

```
jumpstart> ./check
Validating rules...
Validating profile profile...
The custom JumpStart configuration is ok.
```

Sharing the Installation Directory

Once the distribution is loaded and the JumpStart configuration verified, the `/export/install` directory must be exported to enable read-only access by clients. As a result, the `/etc/dfs/dfstab` file must be modified to share the installation directory with clients. If security is a concern, the entire install directory need not be shared. Instead, only the subdirectories created for the Solaris OS installation can be exported for client access. To share the directory, add the following line to the */etc/dfs/dfstab* file:

```
share -F nfs -o ro,anon=0 -d "Install Server" /export/install
```

Starting the NFS Server

If the NFS server is not running, it must be started to enable network file sharing.

```
jumpstart> /etc/init.d/nfs.server start
jumpstart> showmount -e localhost
export list for localhost:
/export/install (everyone)
jumpstart> shareall
jumpstart> share
/jumpstart ro,anon=0 “JumpStart Server”
```

Configuring Client Access

The final step in configuring the Jumpstart server configures client access to the system.

1. Add the client to the */etc/hosts* file. To add the client, add the following line to the */etc/hosts* file:

```
192.168.2.20 client.intra-connect.net client
```

Add the client to the boot files using the `add_install_client` script. Note that the `add_install_client` script takes the parameters described in Table 2-1.

TABLE 2-1 Options and parameters for the `add_install_client` script

Option	Parameters	Example
`-i`	IP_address	192.168.100.1
`-e`	Ethernet_ID	8:0:20:a8:d4:22
`-s`	Server:Path	jumpstart:/export/install/s9x_0904
`-c`	Server:Path	jumpstart:/export/install/jumpstart
`-p`	Server:Path	jumpstart:/export/install/jumpstart
`-n`	Nameserver:name_service[netmask]	
`-t`	Install_boot_image_path client_name platform_group	

```
jumpstart> cd /export/install/s9x_0904/Solaris_9/Tools
jumpstart> ./add_install_client \
-e 8:0:20:a8:d4:22 \
-s jumpstart:/export/install/s9x_0904 \
-c jumpstart:/export/install/jumpstart \
-p jumpstart:/export/install/jumpstart \
client \
sun4u
Adding Ethernet number for client.intra-connect.net to /etc/ethers
making /tftpboot
enabling tftp in /etc/inetd.conf
starting rarpd
starting bootparamd
updating /etc/bootparams
copying inetboot to /tftpboot
```

Troubleshooting

In the event the JumpStart server does not function as expected, the following steps can be used to help identify the source of the problem.

1. Verify NFS sharing is setup correctly. The installation directory should be shared with the following attributes:

```
-o ro,anon=o /jumpstart
```

2. Verify the contents of the `/etc/ethers` and `/etc/bootparams` files.
3. Verify that `/etc/netmasks` is set correctly.
4. Add the client again using the `add_install_client` script
5. Open two windows on the JumpStart server as `root`.
6. Locate the `in.rarpd` process and terminate it using the `kill` command.

```
jumpstart> ps -ef | grep in.rarpd
21667 pts/1 S 0:00 -in.rarpd
jumpstart> kill -9 21667
```

7. Locate the `in.bootparamd` process and terminate it using the `kill` command.

```
jumpstart> ps -ef | grep in.bootparamd
21668 pts/1 S 0:00 -in.bootparamd
jumpstart> kill -9 21668
```

8. In one window on the JumpStart server, start the `in.rarpd` service. Substitute a different interface name if `hme0` is not in use.

```
jumpstart> in.rarpd -d hme 0
```

9. In the other window on the JumpStart server, start the in.`bootparamd` service.

```
jumpstart> in.bootparamd -d
```

10. Monitor the output for errors.

3

JumpStart and the Dynamic Host Configuration Protocol

Dynamic Host Configuration Protocol

The Dynamic Host Configuration Protocol (DHCP) provides a mechanism for PCs to be moved and automatically re-connected to the enterprise network as needs dictate. It moves the management of client-specific network configuration parameters from individual PCs to a centralized server database. Once the DHCP server is configured, clients automatically receive their configuration parameters during the boot process. Application files typically installed and configured on each PC are now located and available directly from the server.

DHCP provides the automatic, dynamic, and manual allocation of IP addresses. The DHCP daemon automatically allocates, changes, and de-allocates host network addresses, easing the management and allocation of IP addresses. Host-specific configuration parameters are also delivered from DHCP servers to clients, eliminating common administrative hassles that arise from frequent user moves. Based on the Bootstrap Protocol (BOOTP), DHCP is a network protocol that captures the behavior of BOOTP relay agents, allowing DHCP participants to interoperate with new and existing BOOTP systems.

Note – This document assumes the use of the Sun DHCP server included with Solaris 8 OS and later releases. Third-party DHCP servers may be configured, and may behave differently.

Using DHCP with JumpStart

Prior to Solaris 8 Operating System, the JumpStart Boot Server was required to reside in the same IP subnetwork as the Install Client. Because the client bootparam request packet specifies TTL=1, it cannot cross gateways. In addition, the Reverse Address Resolution Protocol (RARP) does not transmit netmask nor router information. As a result, organizations are forced to include a RARP server on every subnet of the network that requires network booting.

BOOTP/DHCP servers can be used across subnets since the most commonly available routers and bridges support DHCP relays — the forwarding of DHCP requests. With the use of DHCP, boot servers for all subnets can be consolidated onto a single server, reducing network complexity and cost.

When a client boots via DHCP, it does not send a RARP request for an IP address, Instead, it broadcasts a `dhcpdiscover` request, looking for a DHCP server aware of its IP address, as well as other information needed to boot or install over the network. An Open Boot PROM (OPB) based client is booted over the network via DHCP using the following command at the `OK` prompt:

```
OK> boot net:dhcp - install
```

DHCP clients obtain their information from DHCP server *macros*, a group of options containing information such as Boot Server hostname, the path to the boot image, Install Server hostname, the path to install image, and more. Macro and option information is stored in the *dhcptab* file located in */var/dhcp*. The `pntadm` command can be used to display much of this information:

- Display the contents of the *dhcptab* file (`pntadm -P`)
- List networks (`pntadm -L`)
- View associated macros (`pntadm -P` *network#)*

Getting Started with DHCP

This section describes best practices for supporting DHCP with JumpStart.

Requirements

Before running the software, it is imperative that the following client and server requirements be met:

- The client system must be a Sun UltraSPARC or x86-64 system running Solaris 8 OS or higher. Note that Solaris 10 OS or higher releases will run in 64-bit mode on x64 systems, while previous releases of Solaris OS will only run in 32-bit mode.
- The Open Boot Prom (OBP) on the client system must be capable of running DHCP. Note that older systems may need patches installed to make this possible.
- There are no requirements on the Boot/Install server. Any Boot/Install server that supports traditional JumpStart functionality is capable of serving as the DHCP Boot/Install server.
- The DHCP server must reside on a system running Solaris 8 OS or higher.

DHCP Commands

Before it can begin accepting commands, the DHCP server must be set up to execute, in either standard or debug mode. Once started the DHCP server can be stopped. Table 3-1 describes the main DHCP commands.

TABLE 3-1 Main DHCP commands

Starting the Sun DHCP Server	`/etc/init.d/dhcp start`
Stopping the Sun DHCP Server	`/etc/init.d/dhcp stop`
Starting the Sun DHCP Server in Debug Mode	`/usr/lib/inet/in.dhcpd start -d`

DHCP Tools and Executables

Several tools are available to help configure and manage DHCP servers. Table 3-2 identifies and provides a brief description of these utilities.

TABLE 3-2 DHCP tools and executables

`dhcpmgr`	Graphical user interface (GUI) for configuring the DHCP server located in `/usr/sadm/admin/bin`. Available on Solaris 8 OS or higher.
`dhcpconfig`	Text-based tool for configuring the DHCP server. Available on Solaris 2.6 OS, Solaris 7 OS and Solaris 8 OS.
`dhtadm`	Utility for displaying, and making changes to, the options and macros in the `dhcptab` file.
`pntadm`	Utility for displaying, and making changes to, DHCP network tables.
`in.dhcpd`	The DHCP server daemon.
`dhcpinfo`	A utility used by client startup scripts to obtain information, such as the hostname, from the DHCP client daemon (`dhcpagent`). The `dhcinfo` utility can also be used to obtain parameter values at the command line.
`dhcpagent`	The DHCP client daemon, which implements the client side of the DHCP protocol.

DHCP Files

The DHCP server uses the files described in Table 3-3 to store configuration, network, and client information.

TABLE 3-3 Main DHCP files

`dhcptab`	Typically located in /var/dhcp, this file contains options with assigned values that are grouped into macros.
`inittab`	Located in /etc/dhcp, this file provides information about all supported DHCP options. It is used by DHCP-related commands to parse and understand DHCP options. Existing options detailed in this file should not be modified. Note that this file is available on Solaris 8 OS or later.
`dhcp_network`	Typically located in /var/dhcp, this file maps the client IP address to the client ID (MAC address) and the macro used to boot and install. This file is named after the network ID of the clients in the network, such as 192_168_2_0.

Setting Up DHCP with JumpStart

The following steps outline how to configure DHCP to work with a JumpStart server.

1. Create a standard JumpStart server following the procedures outlined in chapter 2. Be sure to configure the Install Server (using the `setup_install_server script`) and a profile server, properly setup system identification (via the `sysidcfg` file), and share the JumpStart directories (via the `/etc/dfs/dfstab` file).

2. Once the standard JumpStart server is configured, setup the JumpStart client using the `add_install_client` script with the `-d` option. The script displays the options required on the DHCP server to ensure the client can be configured correctly with JumpStart via DHCP. During this process, a DHCP-capable `inetboot` file is copied to `/tftpboot`.

3. Next, configure the DHCP server using the `dhcpmgr` utility located in `/usr/sadm/admin/bin` on systems running Solaris 8 OS or later, or the `dhcpconfig` utility on systems running Solaris 2.6, 7 or 8 OS. The utility prompts for a variety of network configuration information:

   a. Upon execution, the utility prompts for the type of server to be configured. Configure the system as a DHCP server.

   b. Next, users are prompted for the data storage location. Choose the default location, */var/dhcp.*

   c. Next, the utility asks users to specify the lease policy. Choose the default option by clicking the right arrow.

   d. At the next prompt, specify the DNS domain and DNS servers (optional). Note that these fields are automatically filled in if the */etc/resolv.conf* file exists.

   e. When prompted, specify the network address and subnet mask. If these fields are not filled in automatically, enter the network address and subnet mask of the subnet that the DHCP server will serve.

   f. Next, the utility asks for the network type and router to be specified. Select `LAN` and `Use router`, and enter the default router.

   g. Next, specify the NIS domain and NIS servers. Note that these fields are automatically filled in if the DHCP server is setup as a NIS client. Accept the default values.

   h. Finally, specify the NIS+ domain and NIS+ servers. Note these fields are automatically filled in if the DHCP server is setup as a NIS+ client. Accept the default values.

   i. Review the configuration and select `OK`. At this point, it is important not to start the address wizard. The client will be configured after options and macros are created in the following steps.

4. Create JumpStart DHCP options using the `add_install_client` script with the **`-d`** option:

```
../../Solaris_9_09_04/Solaris_9/Tools/add_install_client -d \
     [-s server:path] [-c server:path] [-p server:path] \
     [-t install boot image path] [-f boot file name]
     platform_name platform_group

../../Solaris_9_09_04/Solaris_9/Tools/add_install_client -d \
     -e ethernet id [-s server:path] [-c server:path] \
     [-p server:path] [-t install boot image path] \
     [-f boot file name] platform_group
```

5. Create macros. Use macros to assign values to options and group options, such as the install server, boot server, path to the boot file, and so on.

6. JumpStart the client. Add the client's IP address to the DHCP server and associate it with the client's macro.

```
ok boot net:dhcp -install
            OR
ok boot net:dhcp -dvs install (debugging mode)
```

4

The Preboot eXecution Environment

The systems in today's heterogeneous computing environments often use different booting mechanisms. While Sun SPARC-based workstations and servers can initiate a network boot sequence by simply issuing a `boot net` command at the Open Boot™ PROM (OPB) prompt, most x86-64 systems can boot directly off a network interface card (NIC) through the Preboot eXecution Environment (PXE). As a result, it is important to understand how Solaris JumpStart and PXE differ and interoperate.

PXE incorporates three technologies that establish a common and consistent set of pre-boot services within the boot firmware of x86-64 systems. With these technologies in place, new x86-64 client systems should be able to enter a heterogeneous network, acquire a network address from a DHCP server, download a bootstrap program, and set up the system.

- A uniform client protocol for requesting the allocation of a network address, as well as the downloading of a Network Bootstrap Program (NBP) from a network boot server.
- A set of application programming interfaces (APIs) in the machine's pre-boot firmware environment, providing a consistent set of services that can be employed by the NBP or the BIOS.
- A standard method of initiating the pre-boot firmware to execute the PXE protocol on the client machine.

PXE Transactions

The PXE stage of the network boot process aims to bootstrap the operating system kernel and pass the kernel installation-specific data. In this effort, PXE utilizes the Dynamic Host Configuration Protocol (DHCP) and Trivial File Transfer Protocol (TFTP) IP service protocols. Clients with a PXE-capable NIC broadcast a `DHCPDISCOVER` request for handling by a DHCP server (or proxy DHCP server) configured to answer install client requests. The DHCP message received by the client contains IP configuration data and the location of a TFTP file.

- PXE on the server

  PXE servers must provide DHCP and TFTP services. The DHCP server itself is allotted a range of IP addresses, and provides macros with name-value pairs. The TFTP server is a repository for PXE-compatible NBP files. At a minimum, a PXE server must be able to provide clients with an IP address and a macro containing the network location of an NBP file. In more complex situations, the server supplies additional IP information, such as the subnet mask, name service information and the gateway. The server may also be configured to give individual hosts specific IP and NBP file configurations, by associating a unique IP address and macro with an install client's MAC (hardware) address.

- PXE on the client

  When a client receives a DHCP reply, it should have enough information for the PXE-capable NIC to configure its IP address and initiate a TFTP transfer of the NBP file using the location provided in the macro. Once the NBP file is loaded, it self-executes and determines the next course of action (behavior differs based on OS type).

  - PXE on the Solaris OS

    The Solaris OS NBP attempts to obtain a boot environment profile (*bootenv.rc*) that defines information such as the I/O interface for the console (serial, keyboard, display) and boot device path (the PCI hardware path to the network card). If this file does not exist, the Solaris OS NBP attempts to start the interactive Solaris Configuration Assistant through the keyboard/display interface, prompting the user for the boot device. Once the boot device is determined (either automatically or interactively), the Solaris kernel defined in the DHCP macro provided is loaded. The PXE stage is complete once the kernel is successfully loaded into main memory.

  - PXE in the Linux operating environment

    Unlike the macro mechanism used in the Solaris OS, the Linux NBP (`pxelinux`) looks for a file providing the boot environment. If the profile is to match a specific host, the `pxelinux` configuration file name matches all or part of the client's IP address (in hexadecimal format). This file contains the location of the desired Linux distribution kernel, console interface, and more. The PXE stage is complete after the Linux kernel is loaded with the parameters supplied by the configuration file.

    PXE does not specify the operational details and functionality of the NBP received by the client from the server. However, the intent is clear — running this executable should result in a system ready for use by users. At a minimum, the operating system should be installed, as well as device drivers and other software appropriate to the client hardware configuration. User-specific system configuration and application installation may also be included.

PXE does specify the protocols by which a client requests and downloads an executable image from a Boot Server, and the minimum requirements on the client execution environment when the downloaded image is executed.

Differences Between JumpStart Stages on SPARC and x86-64 Platforms

Table 4-1 summarizes the differences between JumpStart stages on SPARC- and x86-64 systems.

TABLE 4-1 Differences between JumpStart stages on SPARC- and x86-64 platforms

SPARC/BOOTP JumpStart	SPARC/DHCP JumpStart	x86-64 (PXE)/DHCP JumpStart
Client is booted using the `boot net - install` command.	Client is booted using the `boot net:dhcp - install` command.	Network Boot is configured in BIOS, or press the appropriate key at boot (depends on system, typically F12).
The client broadcasts a reverse address resolution protocol (RARP) request on the network, asking for its Internet address.	The client sends a `DHCPDISCOVER` broadcast requesting an Internet address.	Similar to the SPARC implementation, except the PXE client broadcasts a `DHCPDISCOVER` message (port 67) that contains `PXEClient` extension tags (including client identifier, client UNDI version, client system architecture).
The rarpd daemon (`in.rarpd`) running on the Boot Server responds to the RARP request with an IP address. It uses the /etc/ethers and /etc/hosts databases to map the clients Ethernet MAC address to the IP address. A name service, such as NIS or NIS+, can also be used to map the address.	The DHCP daemon (`in.dhcpd`) running on the Boot Server responds with a `DHCPOFFER`. The DHCP has its own data files, the format and location of which depend on the implementation. For example, the Solaris DHCP server may store this data as text files in /var/dhcp.	Similar to the SPARC implementation. The DHCP server responds with a `DHCPOFFER` (port 68) that contains PXE server extension tags, other DHCP option tags, and the Client IP address.
The client issues a tftp request to get the bootstrap loader from the boot server.	The client issues a tftp request based on the BootSrvA and BootFile parameters it received in its response from the DHCP server. The PXE client sends a `DHCPREQUEST` (port 67) that contains `PXEClient` extension tags and other DHCP option tags. Similtar to the SPARC implementation.	
The `inetd` daemon running on the Boot Server receives the tftp request and spawns the `in.tftp` daemon to handle the request. The `in.tftp` daemon searches for a matching IP address and architecture in the `/tftpboot` directory and returns the JumpStart boot image (a mini root kernel) to the client.		

TABLE 4-1 Differences between JumpStart stages on SPARC- and x86-64 platforms

SPARC/BOOTP JumpStart	SPARC/DHCP JumpStart	x86-64 (PXE)/DHCP JumpStart
The bootfile which OBP loads can be directly booted, unlike PXE.The client now boots from the JumpStart boot image just received.		The PXE client sends a `Boot Service Discover` (port 67 or 4011) that contains `PXEClient` extension tags and other DHCP option tags. The Boot Service on the Boot Server responds with a `Boot Service Ack` (client source port) that contains PXE Server extension tags (including the Network Bootstrap Program file name). The PXE client downloads the executable file using either standard TFTP (port 69) or MTFTP (port assigned in Boot Server Ack packet). The PXE client determines whether an authenticity test on the downloaded file is required. If the test is required, the client sends another `DHCPREQUEST` message to the boot server requesting a credentials file for the previously downloaded boot file, downloads the credentials via TFTP or MTFTP, and performs the authenticity test. The PXE client initiates execution of the downloaded code. If needed, the Solaris Device Configuration Assistant starts, prompting the user for the boot device/bootpath. (The Solaris Device Configuration Assistant can be bypassed with a properly-configured *bootenv.rc* file.) Knowing the bootpath, the bootloader prompts the user to determine if the Solaris installation will be an interactive or Jumpstart install.
The client now boots the kernel received from TFTP.		
The client broadcasts a RARP request asking for an IP address.	The client again issues a `DHCPDISCOVERREQUEST` request asking for an IP address. This request is for the Solaris OS, where the previous request was needed for the client to fetch the net boot image. The Boot Service on the Boot Server responds with a `Boot Service Ack` (client source port) that contains PXE Server extension tags (including the Network Bootstrap Program file name).	

TABLE 4-1 Differences between JumpStart stages on SPARC- and x86-64 platforms

SPARC/BOOTP JumpStart	SPARC/DHCP JumpStart	x86-64 (PXE)/DHCP JumpStart
The rarpd daemon running on the boot server responds to the RARP request with an IP Address. It uses the ethers and hosts databases to map the clients ethernet MAC address to the IP address. The client sends a bootparam request (hostconfig request) to get its hostname. The nameserver/boot server returns to the client the information stored in the bootparams table. The client sends a bootparam request to get the root file system on the boot server. The nameserver/boot server returns to the client the information stored in the bootparams table.	The DHCP daemon on the boot server responds with a `DHCPOFFERACK`, again based on the data stored in the DHCP server configuration files. This response includes: • SrootNM and SrootIP4: hostname and IP address of the boot server • SrootPTH: path to the exported mini-root filesystem on the boot server • SinstNM and SinstIP4: hostname and IP address of the install server • SrootPTH: path to the exported Solaris distribution on the install server • SjumpsCF: path to the Jumpstart configuration • SsysidCF: path to the sysidcfg	
Using the bootparameters information just received, the client uses NFS to mounts the / (root) file system from the boot server and starts the init program.	Using the DCHP information just received, the client uses NFS to mounts the / (root) file system from the boot server and starts the init program.	
Once the boot server is done bootstrapping the client, it then points the client to the configuration server.		
The client uses the bootparameters information to search for the configuration server.	The client uses the DHCP information to locate the configuration server.	
The client mounts the configuration directory and runs sysidtool.		
The client mounts the installation directory (location of OS image) on the install server.		
The client runs SunInstall and installs itself.		

5

Configuring JumpStart for Solaris x86 Provisioning

This chapter describes how to configure a Solaris install server to install Solaris 9 OS on PXE clients. Procedures are included to illustrate how to install Solaris 9 OS on a Sun Fire V20z or Sun Fire V40z x64 server using a network connection to the PXE server.

Configuring JumpStart for Solaris x86 requires:

- A JumpStart server
- A networked x86-based system with a CD-ROM or DVD-ROM drive
- A Sun Fire V20z or Sun Fire V40z server to act as the JumpStart client
- A set of Solaris 9 Operating System HW 04/04 for x86 or later distribution CD-ROMs
- A set of Solaris 9 Operating System 04/04 for x86 (Solaris 9 HW 04/04 for x86, if client is a Sun Fire V40z system) or later distribution CD-ROMs

Copying x86-64 CD Images to SPARC Servers

Preparing the x86-64 System

The UNIX® File System (UFS) is an architecture-specific file system. Because it is not byte-neutral, the mounting of a UFS partition from an x86-64 based machine onto a SPARC-based system (and vice versa) is not possible. To provide this capability, an x86-64 system can be used to share Solaris x86 media on the network, enabling the content located on CDs to be copied to the SPARC-based JumpStart server over the network. As a result, it is important

not to insert the Solaris x86 media directly in the SPARC-based server. This step is necessary only if CD-ROM media used. DVD-ROM media uses an architecture-neutral file system and can be read on SPARC- or x86-64 systems.

The x86-64 system must include a CD-ROM or DVD-ROM drive, and be part of the site's network and name service. If a name service is used, the system must also be in the NIS, NIS+, DNS, or LDAP name service. If a name service is not used, information about this system must be distributed according to site policies. More information on the creation of cross-platform install servers using CD-ROMs can be found in *Creating a Cross-Platform Install Server for CD Media* section in chapter 7 of the *Solaris 10 Installation Guide: Network-Based Installations* manul, document number 817-5504 available on *docs.sun.com*.

The following steps prepare a remote x86-64 system for mounting a CD image on a SPARC-based system:

1. Become superuser on the remote x86-64 system.
2. Insert the Solaris 9 Software x86 Platform Edition 1 of 2 CD into the CD/DVD-ROM drive on the x86-64 system.
3. Share the CD-ROM content by adding the following entries to the */etc/dfs/dfstab* file:

```
share -F nfs -o ro,anon=0 /cdrom/cdrom0/s0
share -F nfs -o ro,anon=0 /cdrom/cdrom0/s2
```

4. Start the NFS daemon.

```
# /etc/init.d/nfs.server start
```

5. Verify that the CD is available to other systems on the network.

```
# share
- /cdrom/sol_9_x86/s0 ro,anon=0 " "
- /cdrom/sol_9_x86/s2 ro,anon=0 " "
```

Mounting the x86-64 CD Image

1. Become superuser on the SPARC system designated as the Solaris x86 install server.

2. Access the Solaris x86 CD by creating two directories for the appropriate mount points, one for the miniroot and one for the product. Note that *directory_name_s0* is the name of the directory that will contain the miniroot from slice 0, and *directory_name_s2* is the name of the directory that will contain the product from slide 2.

```
# mkdir directory_name_s0
# mkdir directory_name_s2
```

3. Mount the remote Solaris x86 CD image.

```
# mount remote_x86_system:/cdrom/sol_9_x86/s0 directory_name_s0
# mount remote_x86_system:/cdrom/sol_9_x86/s2 directory_name_s2
```

Copying the Image to the SPARC System

1. Change to the Tools directory on the mounted disc.

```
# cd /directory_name_s2/Solaris_9/Tools
```

2. Copy the disc to the install server's hard disk in the directory created. The -t option specifies the path to a boot image other than the one in the Tools directory on the Solaris 9 Software 2 of 2 CD. The directory_name_s0 parameter is the name of the directory containing the miniroot from slice 0, and install_dir_path specifies the name of an empty directory to which the disc image is to be copied.

```
# ./setup_install_server -t directory_name_s0 install_dir_path
```

   Note that the setup_install_server command indicates whether enough disk space is available for the Solaris 9 Software disc images. To determine available disk space, use the `df` command with the `-kl` option (`df -kl`).

3. Change to the top-level directory.

```
# cd /
```

4. Unmount both directories.

```
# unmount directory_name_s0
# unmount directory_name_s2
```

Unsharing CD-ROM Slices on the x86-64 System

To unshare CD-ROM slices and eject the CD, perform the following steps:

1. Unshare the CD-ROM slices.

```
# unshare /cdrom/sol_9ia/s0
# unshare /cdrom/sol_9ia/s2
```

2. Eject the Solaris 9 Software x86 Platform Edition 1 of 2 CD.

Complete the SPARC System CD Installation

The following steps complete the CD installation on the SPARC-based system.

1. Insert the Solaris 9 Software x86 Platform Edition 2 of 2 CD into the CD-ROM drive on the SPARC-based system.
2. Change to the Tools directory on the mounted CD-ROM.

```
# cd /cdrom/cdrom0/Solaris_9/Tools
```

3. Copy the CD to the install server's hard disk. The *install_dir_path* parameter specifies the directory to which the CD image is to be copied.

```
# ./add_to_install_server install_dir_path
```

4. Eject the Solaris 9 Software x86 Platform Edition 2 of 2 CD.

5. Insert the Solaris 9 x86 Platform Edition Languages CD into the CD-ROM drive on the SPARC-based system and mount the CD.

6. Change to the Tools directory on the mounted CD.

```
# cd /cdrom/cdrom0/Tools
```

7. Copy the CD to the hard disk on the install server. The *install_dir_path* parameter specifies the directory to which the CD image is to be copied.

```
# ./add_to_install_server install_dir_path
```

8. Eject the Solaris 9 x86 Platform Edition Languages CD.

Add the Install Client to the Install Server

1. Change to the Tools directory and add the install client to the install server. The *add_install_client* script provided with Solaris 10 OS supports command line options, using the **-b** flag, which automatically create the *bootenv.rc* file with the necessary parameters. See page 55 for a discussion of special notes on the Solaris Operating System for more information.

```
# cd /export/install/s9x_0904/Solaris_9/Tools
# ./add_install_client -d SUNW.i86pc i86pc
Adding "share -F nfs -o ro,anon=0 /export/home/boot_server/Solaris_9/Tools/Boot" to /
     etc/dfs/dfstab
making /tftpboot
copying inetboot to /tftpboot
copying nbp to /tftpboot
To Enable SUNW.i86pc in the DHCP server, add an entry to the server with the following
     data:
Install server      (SinstNM)  : server
Install server IP   (SinstIP4) : 192.168.2.2
Install server path (SinstPTH) : /export/home/net_install
Root server name    (SrootNM)  : server
Root server IP      (SrootIP4) : 192.168.2.2
Root server path    (SrootPTH) : /export/home/net_install/Solaris_9/Tools/Boot

To enable PXE boot, create a macro definition called
PXEClient:Arch:00000:UNDI:002001 which has the following values:
Boot file           (BootFile) : nbp.SUNW.i86pc
Boot server IP      (BootSrvA) : 192.168.2.2
```

2. Verify the existence of the */tftpboot* directory.

```
# ls /tftpboot/
```

The following files should be located in the */tftpboot* directory:

- *SUNW.i86pc*
- *nbp.SUNW.i86pc*
- *inetboot.I86PC.Solaris_9-1*
- *rm.SUNW.i86pc*
- *nbp.I86PC.Solaris_9-1*
- *tftpboot*

DHCP Server Configuration

This section describes how to configure DHCP using the dhcpconfig command line interface. The techniques described are intended for experienced Solaris OS system administrators, and enable ease-of-use of this command in scripts. Administrators new to the Solaris OS environment, or those preferring a graphical user interface, can use the DHCP manager (`dhcpmgr`) utility to configure and manage the DHCP service. More information on using the DHCP Manager can be found in the *Solaris 9 System Administration Guide: IP Services*, document number 806-4075-10.

Several man pages provide additional information on the DHCP configuration commands. These man pages include: `dhcpconfig(1M)`, `dhcpmgr(1M)`, `dtdadm(1M)`, and `pntadm(1M)`. These man pages can be accessed by typing `man` *`command-name`* at the Solaris OS system prompt. These man pages are also listed in *man pages section 1M: System Administration Commands*, document number 816-0211-10.

Note – Care should be taken when configuring the DHCP server. Incorrect reconfiguration of an existing DHCP server may adversely affect the network. Experimenting with the configuration of a DHCP server on a private subnet may prove helpful.

The following steps detail how to configure the DHCP server.

1. Ensure variable values are available. Table 5-1 identifies the variables needed to execute the DHCP server configuration commands.

TABLE 5-1 DHCP server configuration variables and values

data_resource	DHCP data storage module type. Valid values are: `SUNWfiles`, `SUNWbinfiles`, `SUNWnisplus`.
dhcp_directory	Path name of the DHCP directory.
dns_ip	IP address of the DNS server.
domain	DNS domain name.
hosts_resource	Resource in which to place hosts data, typically the name service in use on the server. Valid values are: `nisplus`, `files`, or `dns`.
hosts_domain	DNS or NIS+ domain name for hosts data (used only if hosts_resource is `nisplus` or `dns`).
network_ip	IP address of the network.
router	Router IP address.
client_ip	IP address of the install client.
client_name	Name of the install client.
client_id	Client ID (MAC address) of the install client (optional).
macro	Name of the macro containing DHCP options. Sun suggests using the network IP address for the macro name.
install_server_ip	IP address of the install server.
install_server_name	Name of the install server.
install_path	Path name of the install server directory.
boot_path	Path name of the boot server directory.
boot_server_name	Name of the boot server.
boot_server_ip	IP address of the boot server.

2. Configure the DHCP server using the `dhcpconfig` command with the `-D` option. If the `-h` option is used to specify `dns` or `nisplus` for the *hosts_resource*, add the `-y` suboption to the command line.

```
# dhcpconfig -D -r data_resource -p dhcp_directory -a dns_ip
      -d domain -l 84600 -h hosts_resource [-y hosts_domain]

# dhcpconfig -D -r SUNWfiles -p /var/dhcp -a 192.168.2.100
      -d sun.com -l 84600 -h files
```

3. Configure the network for DHCP service and create a macro labeled after the network IP address using the `dhcpconfig` command with the `-N` option. The *network_ip* value can be used to replace the *macro* variable in later steps.

```
# dhcpconfig -N network_ip -t router

# dhcpconfig -N 192.168.2.0 -t 192.168.2.254
```

4. Add the PXE client macro to the DHCP configuration using the `dhtadm` command.

```
# dhtadm -A -m PXEClient:Arch:00000:UNDI:002001 -d
      ':BootFile="nbp.SUNW.i86pc":BootSrvA={boot_server_ip}:'

# dhtadm -A -m PXEClient:Arch:00000:UNDI:002001 -d
      ':BootSrvA=192.168.2.2:'
```

Note – While the output of the `./add_install_client` command specifies placing the `BootFile` parameter in the *PXEClient:Arch:00000:UNDI:002001* macro, this practice is not recommended if providing other releases of Solaris OS or Linux is planned. Instead, the BootFile parameter should be placed in the operating system-specific macro for target install clients.

---

5. Add client information to the DHCP server configuration and assign a macro to the configuration.

```
# pntadm -A -client_ip -c client_name -f DYNAMIC -m macro network_ip

# pntadm -A 192.168.2.10 -c client10 -f DYNAMIC \
      -m 192.168.2.0 192.168.2.0
```

If a permanent, static assignment is desired for a given host, the `-f` option should be set to `PERMANENT` and the `-i` option should be included to specify the MAC address for the client ID. Note that the format for the MAC address is `01XXXXXXXXXXXX`, where the `X`s represent the MAC address in upper-case hexadecimal, with the colons removed.

```
# pntadm -A -client_ip -c client_name -f PERMANENT \
     -i 01xxxxxxxxxxx -m macro network_ip

# pntadm -A 192.168.2.10 -c client10 -f PERMANENT \
     -i 01000347F1FFFF -m 192.168.2.0 192.168.2.0
```

6. Create required DHCP vendor options using the `dhtadm` command.

```
# dhtadm -A -s SinstNM -d 'Vendor=SUNW.i86pc,11,ASCII,1,0'
# dhtadm -A -s SinstPTH -d 'Vendor=SUNW.i86pc,12,ASCII,1,0'
# dhtadm -A -s SinstIP4 -d 'Vendor=SUNW.i86pc,10,IP,1,1'
# dhtadm -A -s SrootNM -d 'Vendor=SUNW.i86pc,3,ASCII,1,0'
# dhtadm -A -s SrootPTH -d 'Vendor=SUNW.i86pc,4,ASCII,1,0'
# dhtadm -A -s SrootIP4 -d 'Vendor=SUNW.i86pc,2,IP,1,1'
# dhtadm -A -s SjumpsCF -d 'Vendor=SUNW.i86pc,14,ASCII,1,0'
# dhtadm -A -s SsysidCF -d 'Vendor=SUNW.i86pc,13,ASCII,1,0'
# dhtadm -A -s SbootURI -d 'Vendor=SUNW.i86pc,16,ASCII,1,0'
```

7. Add the DHCP options to the macro and assign values using the `dhtadm` command. Be sure not to put quotes (`""`) around the *install_server_ip* and *boot_server_ip* parameters. Note that the `BootFile` option omitted from the `PXEClient:00000:UNDI:002001` macro is included in the `192.168.2.0` macro.

```
# dhtadm -M -m macro -e 'SrootPTH="boot_path"'
# dhtadm -M -m macro -e 'SrootIP4=boot_server_ip'
# dhtadm -M -m macro -e 'SinstNM="install_server_name"'
# dhtadm -M -m macro -e 'SinstPTH="install_path"'
# dhtadm -M -m macro -e 'SinstIP4=install_server_ip'
# dhtadm -M -m macro -e 'SrootNM="boot_server_name"'

For example:
# dhtadm -M -m 192.168.2.0 -e 'SinstNM="server"'
# dhtadm -M -m 192.168.2.0 -e 'SinstPTH="/export/install/s9x_0904"'
# dhtadm -M -m 192.168.2.0 -e 'SinstIP4=192.168.2.2'
# dhtadm -M -m 192.168.2.0 -e 'SrootNM="server"'
# dhtadm -M -m 192.168.2.0 -e 'SrootPTH="/export/install/s9x_0904/Tools/Boot"'
# dhtadm -M -m 192.168.2.0 -e 'SrootIP4=192.168.2.2'
# dhtadm -M -m 192.168.2.0 -e 'BootFile=nbp.SUNW.i86pc'
```

When a variety of platforms and operating systems reside on the same network, Sun recommends giving the macro an explicit name rather than simply using the network address. For example, the macro name *s9x_0904* could be used for a generic Solaris OS installation, while *s9x_0904_Web_Srvr* could be use for a more specific configuration.

It is important to note that the standard Solaris OS installation instructions may not document the `BootSrvA`, `BootFile`, and `SbootURI` DHCP options, or clarify their difference and utility. Knowledge of each option can be valuable if multiple releases of the Solaris OS are to reside on the same install server. In this case, `BootSrvA`, `BootFile`, and `SbootURI` should be defined in the macro associated with the respective client. Note that the NBP and bootloader files are release-specific, and cannot be interchanged or reused with other releases.

- `BootSrvA`, the IP address of the boot (TFTP) server.
- `BootFile`, the filename of the NBP executable located on the boot server, downloaded via TFTP, and executed by PXE. While the bootfile is copied automatically when the `add_install_client` command is executed, it may be copied manually from the Solaris OS distribution path *Solaris_X/Tools/Boot/boot/solaris/nbp*.
- `SbootURI`, the actual Solaris OS bootloader. The NBP requests this parameter and executes the next stage of the boot process. This parameter takes the form of a URL, such as *tftp://boot_server_ip/bootloader_filename*. While this file is copied automatically when the `add_install_client` command is executed, it may be copied manually from the Solaris OS distribution path *Solaris_X/Tools/Boot/boot/solaris/boot.bin*.

Note that the latest version of the `add_install_client` command creates symbolic links in the */tftpboot* directory using the client name. For example, a SPARC-based DHCP client named *client* can be set using `BootFile=client`, while an x86-64 Solaris OS DHCP client be set using `SbootURI="tftp://ip_server_address/client"`.

8. Print the macro values using the `dhtadm` command with the `-P` option.

```
# dhtadm -P
```

9. Verify the DHCP daemon is running. Output similar to the following is displayed.

```
# ps -aef | grep in.dhcpd
root 3040 484 0 19:21:49 pts/2 0:00 grep in.dhcpd
root 3017 1 0 19:12:48 ? 0:00 /usr/lib/inet/in.dhcpd
```

Preparing the Client System for a PXE Installation

Once the PXE server is configured, the client system should be prepared. The PXE installation can be performed locally or remotely.

Local PXE Installation

A local PXE installation is accomplished with a keyboard, video and mouse.

1. Boot the machine on which the Solaris 9 OS software will be installed by pressing the ON/OFF button on the front panel.
2. Press the `F12` key when the BIOS information screen appears.
3. Continue with the procedures outlined in the *Installing Solaris OS with a PXE Server* section below.

Remote PXE Installation

A remote PXE installation is accomplished on Sun Fire V20z and Sun Fire V40z systems via the Service Processor.

Before performing a remote Solaris OS installation, the operating system must be instructed to use the serial console provided through the Service Processor (SP). This is accomplished by creating a *bootenv.rc* file associated with the system's client ID. The sample *bootenv.rc* file below uses the Service Processor via serial port `ttya`, and bypasses the *Solaris Configuration Assistant* by explicitly defining the `bootpath` variable. If, for example, the client ID of a Sun Fire V20z install client is `01000347F1FFFF`, then the filename *01000347F1FFFF.bootenv.rc* is created in the */tftpboot* directory with the following contents.

Note – Tips on booting other Sun x86-64 platforms can be found in Chapter 8, “Special Notes on Sun x86-64 Platforms.”

```
#ident  "@(#)bootenv.rc 1.23    00/07/17 SMI"

#       bootenv.rc -- boot environment variables

setprop auto-boot? True
setprop auto-boot-cfg-num -1
setprop auto-boot-timeout 5
setprop boottimeout 0
setprop bshfirst false
setprop output-device ttya
setprop input-device ttya
setprop bootpath /pci@0,0/pci1022,7450@a/pci17c2,10@2
setprop boot-file kernel/unix
setprop target-driver-for-scsi sd
setprop target-driver-for-direct cmdk
setprop target-driver-for-csa cmdk
setprop target-driver-for-dsa cmdk
setprop target-driver-for-smartii cmdk
setprop pciide true
setprop prealloc-chunk-size 0x2000
setprop ata-dma-enabled 1
```

For an unattended Solaris OS installation, the following line must be added to the *bootenv.rc* file:

```
setprop boot-args install
```

Note that this line must be removed from the *bootenv.rc* file on the client disk when installation is complete. This can be done with a `finish` script that takes care of the following tasks:

- Removes the `boot-args` variable definition from the */boot/solaris/bootenv.rc* file.
- Sets the `bootpath` variable to the installed disk. Otherwise, if the `bootpath` variable was used for a network card, the `bootpath` may be kept on the disk file.
- Suppresses the call to the `kdmconfig` command during the first system boot.
- Reboots the system.

```
#!/bin/sh
# ------------------------------------------------------------------
# ident "@(#)finish.cluster 1.13††† 03/04/07 SMI"
# ------------------------------------------------------------------

touch /a/etc/notrouter
touch /a/noautoshutdown

sync
echo "Changing and syncing bootenv.rc"

# clear the boot-args property
echo "setprop boot-args ''" >> /a/boot/solaris/bootenv.rc

# set the bootpath property to boot from the hard disk
STRING=`df | grep '^/a '† | sed 's/).*//' | sed 's/^.* (//'`
STRING=`ls -l ${STRING}`
MYROOT=`echo $STRING | sed 's/.*..\/..\/devices//'`

echo "setprop bootpath ${MYROOT}" >> /a/boot/solaris/bootenv.rc
# disable kdmconfig from running after the first reboot
sysidconfig -b /a -r /usr/openwin/bin/kdmconfig
sync

# Some x86 systems sometimes do not reboot after a jumpstart
sync
reboot
```

Performing the PXE Installation

1. Login to the Service Processor using the `ssh` command.

```
# ssh -l manager sp_ipaddress

# ssh -l admin 192.168.1.10
```

2. Setup the Service Processor console parameters.

```
$ platform set console -s sp --speed 9600 -e
```

3. Boot the platform.

```
$ platform set os state boot -W
```

4. Console into the platform.

```
$ platform console
```

5. When the first boot screen appears, press the escape and ampersand keys (`Esc` and `@`) to enter network boot mode.

Installing Solaris OS with a PXE Server

Once the client system is prepared, the Solaris OS can be installed with a PXE server using the following steps.

1. Choose an option when the system prompts for boot options. If an option is not selected, the server will reboot by default.
2. Once the initial boot images from the PXE server are downloaded, the *Device Configuration Assistant* is displayed. Press the `F2` key to continue.
3. Press the Enter key when the Bus Enumeration screen appears.

```
Determining bus types and gathering hardware configuration data...
Press Enter.
```

4. Devices are now scanned. When the system device scan is complete, the *Identified Devices* screen appears.
5. If changes need to be made in the *Identified Devices* screen, make the appropriate changes and press the `F4` key to continue. If no changes are needed, simply press the `F2` key to continue without making any modifications.
6. The *Loading* screen appears, containing messages about drivers loaded to the boot system.
7. When the *Boot Solaris* screen appear, select the `NET` option that corresponds to the selected Ethernet port and press the `F2` key to continue.
8. The Solaris OS `suninstall` program checks the default boot disk for the requirements to install or upgrade the system.

6

Configuring Red Hat Enterprise Linux Provisioning (Kickstart)

This chapter outlines the differences between Solaris JumpStart and Red Hat Kickstart, and explains how to install Red Hat Enterprise Server AS on Sun Fire V20z and Sun Fire V40z x64 servers using Kickstart.

Differences Between Solaris JumpStart and Red Hat Kickstart

A DHCP-based, Solaris OS network install client relies heavily on DHCP macros to provide a host of information, including IP addresses, paths to the operating system distribution, and the location of the JumpStart profile. In contrast, Red Hat Linux relies less on DHCP macros, instead relying on its pre-programmed PXE loader to seek out specific files that provide analogous information. Table 6-1 summarizes the differences between how Solaris OS and Linux network install clients proceed.

The Solaris OS and Red Hat Linux both enable system administrators to create highly tuned profiles that allow completely hands-off installations. As a result, it is important to note the formats of JumpStart and kickstart files differ. While this document does not cover these differences in great detail, the following qualities are noteworthy:

- Solaris JumpStart and Red Hat kickstart both set parameters using simple name-value sets, each on a separate line. (Note that SuSE AutoYaST uses an XML configuration file.)
- Solaris JumpStart uses separate files for its `begin` and `finish` scripts. Pre- and post-install scripts are appended to the kickstart configuration file.
- The Anaconda installer creates an *anaconda-ks.cfg* file in the */root* directory during the installation, whether interactive or automated. This file can be reused or modified for future automated installs.

Red Hat Linux provides the Kickstart Configurator tool, enabling administrators to create or modify a kickstart file using a graphical user interface. The Kickstart Configurator can be launched by typing one of the following commands in a terminal window, depending on the version of the operating system:

```
$ /usr/sbin/redhat-config-kickstart
OR
$ /usr/sbin/ksconfig
```

TABLE 6-1 Differences between Solaris JumpStart and Red Hat Kickstart

Solaris JumpStart	Red Hat Kickstart
Requests the IP address and location of the NBP TFTP file according to a DHCP response.	Requests the IP address and location of the PXELINUX FTP file, according to a DHCP response.
NBP executes and loads the Solaris OS miniroot environment, according to DHCP `Sroot*` options.	PXELINUX reads the `pxelinux.cfg/*` file, and fetches the Linux kernel (`vmlinux`) and INITial RamDisk of drivers (`initrd`).
Determines the `input-device` and `output-device` from the bootenv.rc file, if available. If the *bootenv.rc* file specifies the bootpath, it continues to the bootloader. Otherwise, the *Solaris Configuration Assistant* prompts the user.	Determines the console from the `pxelinux.cfg/*` file, if available.
Executes the Solaris OS kernel.	Execute the Linux kernel.
Starts the Solaris OS installer. Some or all of the steps may be skipped based on a directive set in the JumpStart file (DHCP `SjumpsCF` option). The path to the network operating system is set in DHCP `Sinst*` options.	Starts the Red Hat Anaconda installer. Some or all of the steps may be skipped based on directives set by the kickstart profile (`ks.cfg`). The path to the network operating system set in the kickstart profile.

Installing Red Hat Enterprise Server 3.0 AS on Sun Fire V20z and Sun Fire V40z Servers

Installing Red Hat Enterprise Linux 3.0 AS on Sun Fire V20z and Sun Fire V40z servers requires:

- A Sun Fire V20z or Sun Fire V40z install client using a serial console through the Service Processor
- Red Hat Enterprise Linux 3.0 Advanced Server

PXELINUX is part of the SYSLINUX suite of boot microapplications. Written specifically for PXE-capable NICs, PXELINUX is responsible for bootstrapping the load of the Linux kernel and its INITial RamDisk (`initrd`) of drivers. PXELINUX is pre-compiled to look for configuration files in the */pxelinux.cfg* directory on TFTP servers. The most recent version searches for configuration files with names in the order specified in Table 6-2.

TABLE 6-2 Configuration file search order

`01-aa-bb-cc-dd-ee-ff`	MAC address, hyphen-delimited, lowercase hexadecimal
`AABBCCDD`	IP address in uppercase hexadecimal
`AABBCCD`	IP address in uppercase hexadecimal without the lowest byte of last octet
`AABBCC`	IP address in uppercase hexadecimal without the last octet
`...`	...
`A`	IP address in uppercase hexadecimal, highest byte of first octet only
`default`	filename simply *default*

Note – PXELINUX is not normally included in Linux distributions. It is available for download at *http://syslinux.zytor.com/pxe.php*.

Setting Up the Install Server

The following steps outline how to create an entry for the Linux boot image on the DHCP server. Note that PXE is used to boot the Sun Fire V20z system from the server.

- Copying the Red Hat distribution from CD to a shared directory on the Install Server
- Downloading and copying the *pxelinux.0* image
- Creating and populating *pxelinux.cfg* with second boot stage configuration files
- Configuring the DHCP macros

1. On the Install Server, copy the Linux distribution to the system. Repeat for each CD.

```
# cd /cdrom/cdrom0
# mkdir /export/install/rhel_3x86_64
# tar cvpf - . | ( cd /export/install/rhel_3x86_64 ; tar -xvpf - )
```

2. Add the following line to the */etc/dfs/dfstab* file to share the distribution, then the `shareall` command. To confirm the sharing is configured correctly, run the `share` command.

```
share -F nfs -o ro,anon=0 /export/install/rhel_3x86_64
```

3. Copy the *vmlinuz* and *initrd.img* files from the Linux distribution. The system will not boot without these files. Note that these files are unique to a distribution, and cannot be interchanged or re-used with other distributions.

```
# mkdir -p /tftpboot/rhel_3x86_64
# cp /export/install/rhel_3x86_64/images/pxeboot/vmlinuz /tftpboot/rhel_3x86_64
# cp /export/install/rhel_3x86_64/images/pxeboot/initrd.img /tftpboot/rhel_3x86_64
```

4. Copy the *pxelinux.0* binary to the */tftpboot* directory. The following assumes the *pxelinux.0* binary was downloaded and extracted from the `PXELINUX` package in the */tmp* directory.

```
# cp /tmp/pxelinux.0 /tftpboot
# chmod -R 755 /tftpboot/rhel_3x86_64 pxelinux.0
```

5. Create the *pxelinux.cfg* directory and set the permissions appropriately.

```
# mkdir /tftpboot/pxelinux.cfg
# chmod -R 755 /tftpboot/pxelinux.cfg
```

6. Create a default configuration file named */tftpboot/pxelinux.cfg/default*. This file should contain the information below. Note that the `append` line is one continuous line. The kickstart file must be defined in the configuration file on the NFS Install Server. In the example below, the name of the kickstart file is *rhel_3x86_64.cfg*

```
label linux
kernel rhel_3x86_64/vmlinux
append ksdevice=eth0 ks=nfs:192.168.2.2:/export/install/kickstart
     rhel_3x86_64.cfg console=ttyS0,9600 load_ramdisk=1
     initrd=rhel_3x86_64/initrd.img network ipappend 1
```

7. Create multiple configuration files for different IP addresses, if required. The file should use hexadecimal notation for the IP address, such as C0A8020A for 192.168.2.10. This file can be generated with the following command.

```
# echo 192.168.2.10 | \
     awk -F"." '{printf("%02x%02x%02x%02x\n",$1,$2,$3,$4)}' | \
     tr "[a-z]" "[A-Z]"
```

Associating the System with an IP Address

Many situations, including server configuration, expect a system to have a permanent IP address. An IP address can be associated with the MAC address of a system to ensure a consistent system identity using a specific `pntadm` command. The first step is determining the MAC address of the server to install. The simplest way to accomplish this task is to start the server. When the server comes up, press the `F12` key or hit the escape key (`ESC`) if a serial connection is being used. Choose `Network` as the boot device. As the system scans for DHCP responses, it displays its MAC address. Pick the *highest* for installing Linux, and connect the server to the network on the bottom Gigabit Ethernet port on the back panel. Once the MAC address is known, continue by adding the MAC address to the DHCP configuration.

1. Create a generic Linux macro on the Install Server using the `dhtadm` command. Name the macro `Linux`, and point the `BootFile` parameter to the *pxelinux.0* binary, and the `BootSrvA` parameter to the Install Server.

```
# dhtadm -A -m Linux -d \
      ':BootFile="pxelinux.0":BootSrvA=192.168.2.2:'
```

2. If the same Install Server is used for installing the Solaris OS and the Linux operating environment, we recommend creating a subdirectory in the */tftpboot* directory for the Linux configuration files. In the example below, the *pxelinux* directory becomes */tftpboot/linux-install/pxelinux.cfg*. This macro can be used for multiple Linux install clients.

```
# dhtadm -A -m Linux -d \
      ':BootFile="linux-install/pxelinux.0":BootSrvA=192.168.2.2:'
```

3. Configure the client to use the macro just created. The following example uses a permanent DHCP lease and associates the lease with a MAC address. The parameters used to configure the DHCP server for this client include:
   - Hostname of the install client, as defined in */etc/hosts* (`client10`)
   - Client ID (`01` followed by the MAC address of the install client) (`01000347F1FFFF`)
   - Macro name, as used in the `dhtadm` command above (`Linux`)
   - The keyword `PERMANENT`, indicating this is a permanent configuration for this DHCP entry
   - Network table to be modified in the DHCP server (`192.168.2.0`).

```
# pntadm -A client10 -i 01000347F1FFFF -m Linux -f \
      PERMANENT 192.168.2.0
```

Note – If the install client was configured previously for a different target installation, then the existing definition must be deleted before configuring the client with the above macro. The deletion can be accomplished with the `-D` option of the `pntadm` command: `pntadm -D client10 192.168.2.0`.

The Kickstart File

In the *pxelinux.cfg/default* file created in the preceding section, the `append` parameter `ks=nfs:192.168.2.2:/export/install/kickstart/rhel_3x86_64.cfg` is passed to kickstart as the pointer to the auto-install configuration file. An example kickstart file follows.

```
lang en_US
langsupport en_US
keyboard us
mouse genericusb
timezone --utc America/Los_Angeles
rootpw abc123
reboot
bootloader --linear --location=mbr
install
nfs --server 192.168.2.2 --dir /export/install/rhel_3x86_64
clearpart --all
#clearpart --linux
part /boot --fstype ext3 --size 64 --ondisk sda
part  swap --size 512 --ondisk sda
part / --fstype ext3 --size 1 --grow --ondisk sda
network --device eth0 --bootproto dhcp
network --device eth1 --bootproto dhcp
auth  --useshadow  --enablemd5
rootpw linux
firewall --disabled
authconfig --enableshadow --enablemd5
timezone America/New_York
bootloader --location=mbr --append console=ttyS0,9600
clearpart --all --drives=sda
part /boot --fstype "ext3" --size=100 --ondisk=sda
part / --fstype "ext3" --size=1024 --grow --ondisk=sda
part swap --size=1000 --grow --maxsize=2000 --ondisk=sda
#Do not configure the X Window System
skipx
text

%packages
#@Everything
@ development-tools
@ kernel-development

%post --nochroot
cp /mnt/source/{lsi,bcm5700}*.rpm /mnt/sysimage/root

%post
if rpm --quiet -q kernel; then
  rpm -Uvh /root/lsi-[0-9]*.rpm
  rpm -Uvh /root/bcm5700-[0-9]*.rpm
fi
if rpm --quiet -q kernel-smp; then
  rpm -Uvh /root/lsi-smp*.rpm
  rpm -Uvh /root/bcm5700-smp*.rpm
fi

# Switch from tg3 to bcm5700 ethernet driver
if [ -f /etc/modules.conf ]; then
  mv /etc/modules.conf /etc/modules.conf.old
  sed -e 's/tg3/bcm5700/' /etc/modules.conf.old > /etc/modules.conf
fi
depmod -a > /dev/null 2> /dev/null
if [ -f /etc/sysconfig/hwconf ]; then
  mv /etc/sysconfig/hwconf /etc/sysconfig/hwconf.old
  sed -e 's/tg3/bcm5700/' /etc/sysconfig/hwconf.old > /etc/sysconfig/hwconf
fi
rm -f /root/{lsi,bcm5700}*.rpm
```

The example kickstart file instructs the client to use DHCP when it reboots after the installation is completed. In order for the client to use a static network configuration instead, two lines must be replaced, with *ipaddress*, *netmask* and *gateway* specifying the values corresponding to the network configuration the system will use once the installation is complete. Note that this file is installed in the */export/install/kickstart directory.*

```
Replace:

network --device eth0 --bootproto dhcp
network --device eth1 --bootproto dhcp

With:

network -device eth0 --bootproto=static -ip ipaddress --netmask netmask -gateway gateway
network -device eth1 --bootproto=static -ip ipaddress --netmask netmask -gateway gateway
```

Creating the Kickstart File

1. Create the *kickstart* directory.

```
# mkdir /export/install/kickstart
```

2. Share the *kickstart* directory. Add the following line to the */etc/dfs/dfstab file*.

```
share -F nfs -o ro,anon=0 /export/install/kickstart
```

3. Run the `shareall` command.

4. Create a file named *rhel_3x86_64.cfg* containing the kickstart profile in the */export/install/kickstart* directory. Note that the example kickstart file has been tested on a Sun Fire V20z server. It contains a post-installation script section (beginning with the `%post` command) that installs the device drivers provided on the Sun Fire V20z/V40z resource CD-ROM. While it may be used as a template, we recommend creating a kickstart file by performing an interactive kickstart installation (do not include a `ks=` option for `append`) and configuring the system manually as desired.

   Red Hat Enterprise Linux creates the kickstart configuration file *anaconda-ks.cfg* in the */root* directory during the installation based on the selected options. This file can be used and modified for future installations. Using interactive kickstart as a configuration tool provides the opportunity to create an error-free starting point for automated installation profiles. The kickstart file generated can be modified via a graphical user interface using the Kickstart Configurator utility. To launch the Kickstart Configurator, type */usr/sbin/redhat-config-kickstart* in a terminal window.

At this point, the following items are configured and the install client is ready for Linux:

- The distribution has been copied to the server and shared via NFS.
- The DHCP server has been configured with a macro for `PXELINUX` and the client is configured to use that macro.
- The TFTP server has the `PXELINUX` binary installed, and a default or client-specific profile has been created in the *pxelinux.cfg* directory.
- A kickstart profile has been created and shared for the client (for automated installs).

The system is now ready to be booted. (Instructions on connecting to the Service Processor on Sun Fire V20z servers can be found in the preceding chapter.) Once the PXE boot option is selected in the BIOS, startup messages should appear that are similar to the following: .

```
PXELINUX 2.11 2004-08-16  Copyright (C) 1994-2004 H. Peter Anvin
UNDI data segment at:   00090940
UNDI data segment size: 4CB0
UNDI code segment at:   000955F0
UNDI code segment size: 49B0
PXE entry point found (we hope) at 955F:00D6
My IP address seems to be C0A8020A 192.168.2.10 ip=192.168.2.10:192.168.2.2:192.168.2.1:255.255.255.0
TFTP prefix:
Trying to load: pxelinux.cfg/01-00-09-3d-00-18-24
Loading rhel_3x86_64/vmlinuz.....
Loading rhel_3x86_64/initrd.img.....
```

Once the kernel and *initrd* files are loaded, a variety of messages are displayed on screen as the system is probed and drivers are loading. Depending on the completeness of the kickstart profile, the system may or may not prompt for locale, disk layout, package selection, password setup, and other information.

7

Configuring SuSE Linux Provisioning (AutoYAST)

The current version of Linux distributed with the Sun Java™ Desktop System is based on SuSE Linux. As a result, the principles for SuSE AutoYaST configurations described in this chapter can be applied to Sun Java Desktop Systems.

Once a server is capable of installing Solaris 9 OS and Red Hat Enterprise Linux 3.0 AS, it can be expanded to support installations of SuSE Linux Enterprise Server 9.

Differences Between Red Hat Kickstart and SuSE AutoYAST

SuSE Linux uses an automated version of the Yet another Setup Tool (YaST), called AutoYAST. The YaST configuration management system is used to create or edit an autoyast control file. As root, run the autoyast utility in a terminal window.

```
# /sbin/yast2 autoyast
```

Both kickstart and AutoYaST use the *pxelinux.0* binary and *pxelinux.cfg* configuration directory and files, although the format of the append parameters in *pxelinux.cfg* may differ slightly. Table 7-1 summarizes the differences between the Red Hat Kickstart and SuSE AutoYaST utilities.

TABLE 7-1 Differences between Red Hat Kickstart and SuSE AutoYaST

Red Hat Kickstart	SuSE AutoYaST
File format is a fairly simple text file with parameter names and values.	Uses an XML-based file format for specifying autoinstall parameters.
Pre- and post-install scripts are located in separate sections in the kickstart configuration file.	Pre- and post-install scripts are embedded in the XML AutoYaST file.
Interactive use of Anaconda provides a kickstart file in */root* at the end of the installation. The Kickstart Configurator can also be used to create a kickstart file using a graphical interface.	AutoYaST is created by the `yast2 autoyast` command after the system is first installed and configured manually.

Installing SuSE Linux Enterprise Server 9 on Sun Fire V20z and Sun Fire V40z Servers

Installing SuSE Linux Enterprise Server 9 on Sun Fire V20z and Sun Fire V40z servers requires a Sun Fire V20z or Sun Fire V40z server and the SuSE Linux Enterprise Server 9 software distribution on CD-ROM.

1. On the Install Server, copy the Linux distribution to the system. Repeat for each CD.

```
# cd /cdrom/cdrom0
# mkdir /export/install/sles_9x86_64
# tar cvpf - . | ( cd /export/install/sles_9x86_64 ; tar -xvpf - )
```

2. Add the following line to the */etc/dfs/dfstab* file to share the distribution, then run the `shareall` command. To confirm the sharing is configured correctly, run the `share` command.

```
share -F nfs -o ro,anon=0 /export/install/sles_9x86_64
```

3. Copy the *linux* and *initrd* files from the Linux distribution. The system will not boot without these files. Note that these files are unique to a distribution, and cannot be interchanged or re-used with other distributions.

```
# mkdir -p /tftpboot/sles_9x86_64
# cp /export/install/sles_9x86_64/boot/loader/linux \
     /tftpboot/sles_9x86_64
# cp /export/install/sles_9x86_64/boot/loader/initrd \
     /tftpboot/sles_9x86_64
# chmod -R 755 /tftpboot/sles_9x86_64
```

4. If a default profile from a previous Linux installation configuration exists, rename it for future use.

```
# mv /tftpboot/pxelinux.cfg/default \
     /tftpboot/pxelinux.cfg/rhel_3x86_64
```

5. Create a default configuration file named */tftpboot/pxelinux.cfg/default*. This file should contain the information below. Note that the `append` line is one continuous line.

```
label linux
kernel sles_9x86_64/linux
append netdevice=eth0 autoyast=nfs://192.168.2.2//export/install/
     autoyast/sles_9x86_64.xml  console=ttyS0,9600 load_ramdisk=1
     initrd=sles_9x86_64/initrd  install=nfs://192.168.2.2//
     export/install/sles_9x86_64 network ipappend 1
```

Because SuSE uses PXELINUX, you can use the same MAC or IP address-specific naming used for Red Hat installations can be used here to provide unique profiles for specific clients. (See the previous chapter for information on PXELINUX.)

As Linux distributions do not rely on DHCP macros beyond the server and filename of PXELINUX, there is no need to reconfigure the DHCP server if changing between Red Hat and SuSE Linux distributions.

The AutoYaST File

In the *pxelinux.cfg/default* file just created, the `append` parameter `autoyast=nfs://192.168.2.2//export/install/autoyast/sles_9x86_64.xml` is passed to AutoYaST as the pointer to the auto-install control file. An example AutoYaST file follows. Note that unlike the kickstart format, AutoYaST uses XML.

```
<?xml version="1.0"?>
<!DOCTYPE profile SYSTEM "/usr/share/autoinstall/dtd/profile.dtd">
<profile xmlns="http://www.suse.com/1.0/yast2ns" xmlns:config="http://
       www.suse.com/1.0/configns">
  <configure>
    <networking>
      <dns>
        <dhcp_hostname config:type="boolean">false</dhcp_hostname>
        <dhcp_resolv config:type="boolean">false</dhcp_resolv>
      </dns>
      <interfaces config:type="list">
        <interface>
          <bootproto>dhcp</bootproto>
          <device>eth0</device>
          <startmode>onboot</startmode>
        </interface>
      </interfaces>
      <modules config:type="list">
        <module_entry>
          <device>static-0</device>
          <module></module>
          <options></options>
        </module_entry>
      </modules>
      <routing>
        <ip_forward config:type="boolean">false</ip_forward>
      </routing>
    </networking>
    <runlevel>
      <default>3</default>
    </runlevel>
  </configure>
<install>
    <bootloader>
      <activate config:type="boolean">false</activate>
      <global config:type="list"/>
      <kernel_parameters>acpi=off 3</kernel_parameters>
      <loader_device></loader_device>
      <loader_type>grub</loader_type>
      <location></location>
      <repl_mbr config:type="boolean">false</repl_mbr>
      <sections config:type="list"/>
    </bootloader>
    <general>
      <clock>
        <hwclock>localtime</hwclock>
        <timezone>US/Eastern</timezone>
      </clock>
      <keyboard>
        <keymap>english-us</keymap>
      </keyboard>
      <language>en_US</language>
      <mode>
        <confirm config:type="boolean">false</confirm>
        <forceboot config:type="boolean">false</forceboot>
      </mode>
      <mouse>
        <id>none</id>
      </mouse>
</general>
    <partitioning config:type="list">
```

```
 <drive>
        <device>/dev/sda</device>
        <initialize config:type="boolean">false</initialize>
        <partitions config:type="list">
          <partition>
            <crypt>twofish256</crypt>
            <filesystem config:type="symbol">ext2</filesystem>
            <format config:type="boolean">true</format>
            <loop_fs config:type="boolean">false</loop_fs>
            <mount>/boot</mount>
            <partition_id config:type="integer">131</partition_id>
            <size>128mb</size>
          </partition>
          <partition>
            <crypt>twofish256</crypt>
            <filesystem config:type="symbol">swap</filesystem>
            <format config:type="boolean">true</format>
            <loop_fs config:type="boolean">false</loop_fs>
            <mount>swap</mount>
            <partition_id config:type="integer">130</partition_id>
            <size>2048mb</size>
          </partition>
<partition>
            <crypt>twofish256</crypt>
            <filesystem config:type="symbol">reiser</filesystem>
            <format config:type="boolean">true</format>
            <loop_fs config:type="boolean">false</loop_fs>
            <mount>/</mount>
            <partition_id config:type="integer">131</partition_id>
            <size>max</size>
          </partition>
        </partitions>
        <use>all</use>
      </drive>
    </partitioning>
    <software>
      <addons config:type="list">
        <addon>Base-System</addon>
        <addon>YaST2</addon>
      </addons>
      <base>Minimal</base>
    </software>
  </install>
</profile>
```

In the following example, the AutoYaST file (*sles_9x86_64.xml*) is installed in the */export/install/autoyast* directory.

1. Create the */export/install/autoyast* directory.

```
# mkdir /export/install/autoyast
```

2. Edit the */etc/dfs/dfstab* file to share the distribution and run the `shareall` command. To confirm the sharing is configured correctly, run the `share` command.

```
share -F nfs -o ro,anon=0 /export/install/autoyast
```

3. Create a file named *sles_9x86_64.xml* containing the AutoYaST profile in the */export/install/autoyast* directory. Note that the example AutoYaST file has been tested on a Sun Fire V20z server. While it may be used as a template, we recommend creating an AutoYaST file on an existing SuSE system that has been installed interactively. Once installed and booted, the `yast autoyast` command can be run to create an AutoYaST profile. Note that the `autoyast2` package may need to be installed prior to this step.

4. Once the AutoYaST profile is configured and saved, transfer it to the install server and point to it in the *pxelinux.cfg* profile using `autoyast=` for the `append` option.

At this point, the following items are configured and the install client is ready for SuSE Linux:

- The distribution has been copied to the server and shared via NFS.
- The DHCP server has been configured with a macro for `PXELINUX` and the client is configured to use that macro.
- The TFTP server has the `PXELINUX` binary installed, and a default or client-specific profile has been created in the *pxelinux.o* directory.
- An AutoYaST profile has been created and shared for the client (for automated installs).

The system is now ready to be booted. Once the PXE boot option is selected in the BIOS, startup messages should appear that are similar to the following: .

```
PXELINUX 2.11 2004-08-16  Copyright (C) 1994-2004 H. Peter Anvin
UNDI data segment at:   00090940
UNDI data segment size: 4CB0
UNDI code segment at:   000955F0
UNDI code segment size: 49B0
PXE entry point found (we hope) at 955F:00D6
My IP address seems to be C0A8020A 192.168.2.10 ip=192.168.2.10:192.168.2.2:192.168.2.1:255.255.255.0
TFTP prefix:
Trying to load: pxelinux.cfg/01-00-09-3d-00-18-24
Loading sles_3x86_64/linux.....
Loading sles_9x86_64/initrd.....
```

Once the kernel and *initrd* files are loaded, a variety of messages are displayed on screen as the system is probed and drivers are loading. Depending on the completeness of the AutoYaST profile, the system may or may not prompt for locale, disk layout, package selection, password setup, and other information.

8

Special Notes on Sun x86-64 Platforms

Serial Consoles

Serial consoles provide a simple and inexpensive means for remote out-of-band management. Sun systems typically use a serial console configured for communication at 9600 bits per second (bps) with no parity, 8 data bits and 1 stop bit.

The serial console instructions in this document assume an infrastructure using a 9600-N81 configuration in the BIOS, terminal servers, and other systems.When the serial console is configured in this manner, function keys in the `dtterm` and `xterm` utilities may not work as desired. To change the behavior of function keys, create a file named *func_vt100* on a Sun workstation with the following contents:

```
*Dtterm*translations: #override\n\
       F1:         string("\033OP")\n\
       F2:         string("\033OQ")\n\
       F3:         string("\033OR")\n\
       F4:         string("\033OS")\n\
       F6:         string("\033[17~")\n\
       F7:         string("\033[18~")\n\
       F8:         string("\033[19~")\n\
       F9:         string("\033[20~")\n\
       F10:        string("\033[21~")\n\
       F11:        string("\033[22~")\n\
       F12:        string("\033[23~")\n\
```

Once the *func_vt100* file is created and saved, the function key mappings can be reapplied with the following sequence of commands. Note that the last line resets all dtsessions, making the one running in the background the default session.

```
$ /usr/openwin/bin/xmodmap -e 'keycode 75 = F11'
$ /usr/openwin/bin/xmodmap -e 'keycode 76 = F12'
$ /usr/openwin/bin/xrdb -l func_vt100
$ /usr/dt/bin/dtterm &
$ /usr/dt/bin/dtsession_res -load -system
```

Alternatively, a keystroke sequence (the escape key (`Esc`) followed by a numbered key) can be used to accomplish the function key remapping.The escape key sequence for each function key remapping is detailed in Table 8-1.

TABLE 8-1 Escape sequences for function key remapping

Function Key	Keystroke Sequence	Function Key	Keystroke Sequence
F1	Esc 1	F7	Esc 7
F2	Esc 2	F8	Esc 8
F3	Esc 3	F9	Esc 9
F4	Esc 4	F10	Esc 0
F5	Esc 5	F11	Esc !
F6	Esc 6	F12	Esc @

Linux

The Linux operating system assigns device names, such as `eth0` and `sda`, when the system is booted. Device names are assigned based on the order in which devices are scanned in the bus tree. As a result, changes in the hardware configuration may result in devices being renamed. If the order in which the platform scans its devices is unknown, it is recommended that the Linux operating system be installed on a Sun x86-64 system before additional I/O cards are added to the system. Unless otherwise stated, Linux assigns `eth0` to the primary NIC and `sda` (or `hda`) to the first storage device on Sun x86-64 platforms as shipped from the factory.

When installing Linux through a serial console, make sure that the `console=` option is configured appropriately in the respective *pxelinux.cfg* profile and the `bootloader` option (through the kickstart or AutoYaST file).

Switch Configurations

Install clients may have difficulty automatically configuring the network interface after PXELINUX has loaded. In this case, the following network switch configurations are recommended:

Cisco Catalyst switches should be enabled for fast port link negotiation.

```
interface mod/port spanning-tree portfast
OR
set spantree portfast mod/port enable
```

- Gigabit (1000 Mbps) connections should be downgraded to 100 Mbps for the duration of the installation process.

Linux Client IDs Under Sun DHCP Servers

Linux may not broadcast client IDs which conform to the 01XXXXXXXXXXXX encoded ethernet address format which the Sun DHCP server uses for static MAC-to-IP mapping. As a result, *kickstart* may fail to retrieve the expected IP address during the startup process. To address this issue, create a unique *kickstart* profile for the client and statically define the IP address. Point to this *kickstart* file using a unique PXELINUX configuration file in the *pxelinux.cfg* directory, using the MAC address or upper-case hexadecimal IP address filename scheme described above.

Solaris Operating System

On systems running the Solaris Operating System, platform-specific hardware configurations can be passed to install clients through the use of the *bootenv.rc* file. While this mechanism is used by certain platforms to boot (usually noted in the accompanying documentation), it also provides the opportunity to completely bypass the Solaris Configuration Assistant. Note that certain platforms require explicit definition of the bootpath in the *bootenv.rc* file in order to boot from the network.

More recent versions of the Solaris `add_install_client` script support command line options (using the `-b` flag) which automatically create the *bootenv.rc* file when configuring the install client. For example, the following command sets the output and input devices to `ttya` and the hardware bootpath to `/pci@0,0/pci8086,340f@3` for a system with a MAC address `00:03:47:F1:FF:FF:.`

```
# ./add_install_client -d -e 00:03:47:f1:ff:ff \
      -b "output-device=ttya" -b "input-device=ttya" \
      -b "bootpath=/pci@0,0/pci8086,340f@3" i86pc
```

The `add_install_client` script creates a file in the */tftpboot* directory associated with the MAC address (client ID) named *01XXXXXXXXXXXX.bootenv.rc*. For example, the above created a file named *01000347F1FFFF.bootenv.rc*. On systems running older Solaris OS releases, the *bootenv.rc* file must be created manually in the */tftpboot* directory. The format for setting hardware configurations takes the following form:

```
setprop property value
```

For example:

```
setprop output-device ttyb
setprop input-device ttyb
setprop bootpath /pci@0,0/pci8086,340f@3
```

Solaris OS typically defaults the output device to `screen` and the input device to `keyboard`. Therefore, proper configuration of the *bootenv.rc* file is essential for systems that are to be installed and run exclusively using a serial console, such as blades and headless servers.

Sun x86-64 Platforms

The following sections provide an overview of hardware-specific notes for Sun x86-64 platforms.

Sun Fire V20z Server

The Sun Fire V20z Server is a general purpose x64 server featuring dual AMD Opteron™ processors that runs either the Solaris OS or Linux environment. For Sun Fire V20z systems running the Solaris OS, the *bootenv.rc* file should be modified to set the serial console and bypass the boot device stage of the Solaris Configuration Assistant, as follows:

- Set `input-device` and `output-device` to `ttya`
- Set `bootpath` to `/pci@0,0/pci1022,7450@a/pci17c2,20@2`

For Sun Fire V20z systems running the Linux environment, the serial console should be set in the *pxelinux.cfg* file, and as an option to the kickstart or AutoYaST bootloader, as follows:

- Set `console=ttyS0,9600n8`

Sun Fire V40z Server

The Sun Fire V40z Server is a 4P x64 server, featuring AMD Opteron 32-/64-bit processors that runs either the Solaris OS or Linux environment. For Sun Fire V40z systems running the Solaris OS, the *bootenv.rc* file should be modified to set the serial console and bypass the boot device stage of the Solaris Configuration Assistant, as follows:

- Set `input-device` and `output-device` to `ttya`
- Set `bootpath` to `//pci@0,0/pci1022,7450@a/pci17c2,20@2`

For Sun Fire V40z systems running the Linux environment, the serial console should be set in the *pxelinux.cfg* file, and as an option to the kickstart or AutoYaST bootloader, as follows:

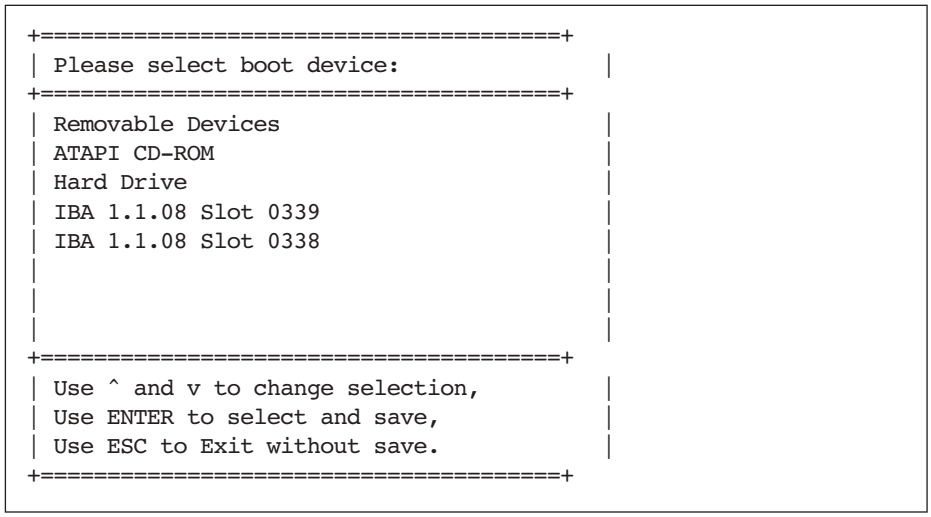
- Set `console=ttyS0,9600n8`

Sun Fire V60x and Sun Fire V65x Servers

The Sun Fire V60x and Sun Fire V65x servers are x86-64 based, general-purpose thin servers. These systems provide cross-platform support for standard Linux distributions, Red Hat Enterprise Linux, SuSE Enterprise Linux Server, or the Solaris Operating System, x86 Platform Edition.

Unlike other Sun x86-64 platforms, network ports 1 and 2 on Sun Fire V60x and V65x servers correspond to `eth1` and `eth0` on Linux, and `e1000g0` and `e100g1` on Solaris OS. we recommend using network port 2 for network-based installations. To select the boot device on the Sun Fire V60x and Sun Fire V65x servers:

- Press the `ESC` key during system boot
- Wait until the boot device selection screen appears
- Select `IBA 1.1.08 Slot 0339`
- Press `ENTER`



For Sun Fire 60x and Sun Fire V65x systems running the Solaris OS, the *bootenv.rc* file should be modified to set the serial console and bypass the boot device stage of the Solaris Configuration Assistant, as follows:

- Set `input-device` and `output-device` to `ttyb`
- Set `bootpath` to `/pci@0,0/pci8086,2545@3/pci8086,1460@1f/pci8086,341a@7`

For Sun Fire 60x and Sun Fire V65x systems running the Linux environment, the serial console should be set in the *pxelinux.cfg* file, and as an option to the kickstart or AutoYaST bootloader, as follows:

- Set `console=ttyS1,9600n8`

Sun Java Workstation W1100z and W2100z Systems

The Sun Java Workstation W100z and Sun Java Workstation W2100z are powerful x64 workstations featuring the AMD Opteron processor. The PXE-bootstrapped network installation is verified for the following operating systems on these systems:

- Solaris 9 Update 7 (32-bit)
- Solaris 10 Pre-release (32- and 64-bit)
- Sun Java Desktop System 2 (32-bit)
- Red Hat Enterprise Linux 3 (*Taroon*) Update 3 Workstation (32- and 64-bit)

Note – While Red Hat Enterprise Linux 3 Update 2 installs correctly, the *initrd.img* file in this release does not contain the necessary drivers to allow for an unattended kickstart auto-install.

The Sun Java Workstation W1100z and the Sun Java Workstation W2100z systems may ship with PXE capabilities disabled. To enable network PXE booting on these systems, perform the following steps:

- Press `F2` on the initial BIOS screen to enter the setup tool
- Change to the *Advanced* screen
- Select *Chipset Configuration*
- Set *Onboard Ethernet* to `Enable`
- Set *Onboard Ethernet PXE* to `Enabled`
- Press `F10` to save and exit
- Press `F8` on the initial BIOS screen upon reboot
- Wait for the I/O devices to configure and a boot device prompt to appear
- Select `MBA vX.X.X Slot XXX` for a PXE network boot (values for X depend on the hardware configuration and version)

Sun Fire B1600 Blade Platform, Sun Fire B100x and B200x Blade Servers

The Sun Fire B1600 Blade Platform facilitates the mixing, matching, and management of SPARC and x86-64 architectures, Solaris and Linux operating systems, and specialty networking blades — all managed through N1™ management software. A layered system management framework enables the management of server and specialty networking blades at multiple levels, easing configuration and management.

The Sun Fire B1600 Blade Platform provides a mechanism for booting all Sun blade platforms without accessing the BIOS of individual blade systems. In order to boot from the network, a boot command must be issued prior to (re)booting the blade system. Note that the parameter `sN` refers to the blade's slot assignment. For example, a blade located in slot 0 would be referenced as `s0` in the command.

```
sc> bootmode bootscript="boot net" sN
```

The hardware used in Sun's blade platforms requires supplemental drivers in addition to the standard Linux distribution packages. See the *Installing Linux from a PXE Boot Install Environment* document located at *http://www.sun.com/products-n-solutions/hardware/docs/html/817-5625-10/Linux_PXE_boot.html#pgfId-1017996* for information on how to configure a Linux distribution for blade installations.

The Sun Fire B100x and B200x Blade Servers can only be installed using a serial console via the Sun Blade 1600 Platform system controller. When running the Solaris OS, the *bootenv.rc* file should be modified to set the serial console and bypass the boot device stage of the Solaris Configuration Assistant, as follows:

- Set `input-device` and `output-device` to `ttya`
- Set `bootpath` to `/pci@0,0/pci108e,16a8@8` on Sun Fire B100x Blade servers
- Set `bootpath` to `/pci@0,0/pci8086,2545@3/pci8086,1460@1d/pci108e,16a8@3` on Sun Fire B200x Blade servers

When running the Linux environment, the serial console should be set in the *pxelinux.cfg* file, and as an option to the kickstart or AutoYaST bootloader, as follows:

- Set `console=ttyS0,9600n8`

Sun Cobalt LX50

For Sun Cobalt LX50 systems running the Solaris OS, the *bootenv.rc* file should be modified to set the serial console and bypass the boot device stage of the Solaris Configuration Assistant, as follows:

- Set `input-device` and `output-device` to `ttyb`
- Set `bootpath` to `/pci@0,0/pci8086,340f@3`

For Sun Cobalt™ LX50 systems running the Linux environment, the serial console should be set in the *pxelinux.cfg* file, and as an option to the kickstart or AutoYaST bootloader, as follows:

- Set `console=ttyS1,9600n8`

Parameter Summary for Sun x86-64 Platforms

For Sun x86-64 platforms running the Solaris OS, the *bootenv.rc* file should be modified to set the serial console and bypass the boot device stage of the Solaris Configuration Assistant. For Sun x86-64 platforms running the Linux environment, the serial console should be set in the *pxelinux.cfg* file, and as an option to the kickstart or AutoYaST bootloader. Table 8-2 identifies the appropriate settings for Sun x86-64 platforms.

More information on setting input and output device types, see the *Installing Solaris with a Redirected Console on x86/AMD Platforms* available on Sun's website at *http://sunsolve.sun.com/search/document.do?assetkey=1-9-79477-1&searchclause=79477*.

TABLE 8-2 Summary of settings for Sun x86-64 platforms

Solaris Operating Systems		
Sun Fire V20z	`input-device=ttya` `output-device=ttya`	`bootpath=/pci@0,0/pci1022,7450@a/pci17c2,20@2`
Sun Fire V40z	`input-device=ttya` `output-device=ttya`	`bootpath=/pci@0,0/pci1022,7450@a/pci17c2,20@2`
Sun Fire V60x	`input-device=ttyb` `output-device=ttyb`	`bootpath=/pci@0,0/pci8086,2545@3/` `pci8086,1460@1f/pci8086,341a@7`
Sun Fire V65x	`input-device=ttyb` `output-device=ttyb`	`bootpath=/pci@0,0/pci8086,2545@3/` `pci8086,1460@1f/pci8086,341a@7`
Sun Fire B100x	`input-device=ttya` `output-device=ttya`	`bootpath=/pci@0,0/pci108e,16a8@8`
Sun Fire B200x	`input-device=ttya` `output-device=ttya`	`bootpath=/pci@0,0/pci8086,2425@3/` `pci8086,1460@1d/pci108e,16a8@3`
Sun Cobalt LX50	`input-device=ttyb` `output-device=ttyb`	`bootpath=/pci@0,0/pci8086,340f@3`
Linux Operating Environment		
Sun Fire V20z	`console=ttyS0,9600n8`	
Sun Fire V40z	`console=ttyS0,9600n8`	
Sun Fire V60x	`console=ttyS1,9600n8`	
Sun Fire V65x	`console=ttyS1,9600n8`	
Sun Blade 1600	`console=ttyS0,9600n8`	
Sun Fire B100x	`console=ttyS0,9600n8`	
Sun Fire B200x	`console=ttyS0,9600n8`	
Sun Cobalt LX50	`console=ttyS1,9600n8`	

9

Additional Tools and Technologies

Sun provides additional tools and technologies that can be used to help configure Sun x86-64 platforms.

Sun Control Station with AllStart Module

Sun Control Station software is a solution designed to simplify the management of volume server deployments. A comprehensive set of preinstalled modules enable the remote management of hundreds of end node systems, including tracking and applying system images and software updates, deploying new services, and monitoring server health and performance.

The AllStart control module is a software provisioning system that enables the Sun Control Station to be used to automate the initial installation of a supported operating system along with its associated software packages. It provides a graphical user interface for SuSE Linux and Java Desktop System (AutoYAST), Red Hat Linux (kickstart), and Solaris OS for x86 (JumpStart) installations.

Using the features of the AllStart module, users can:

- Install a given configuration on a large number of clients
- Perform unattended software installations
- Create and modify software payloads obtained from ISO images, CD-ROM, DVD-ROM, or Flash Archives
- Define client profiles
- Monitor and validate system installations and updates

More information on the Sun Control Station software and the Allstart control module is located at *http://www.sun.com/controlstation*.

JumpStart Enterprise Toolkit and JetPXE

The JumpStart Enterprise Toolkit (JET) provides a framework to simplify and extend the JumpStart functionality provided within the Solaris Operating System. It helps automate the installation and configuration of common applications, and incorporates best practices gained from Sun's extensive experience of installing and configuring Sun servers. This flexible, modular, intuitive toolkit provides basic operating system installations along with frequently used/requested additional product installation, and minimal configuration. These utilities can help:

- Configure the JumpStart server and populate it with media and patches.
- Set up a target server configuration by specifying the products to be installed and the installation parameters for each of them.
- Reduce code repetition through a library of common functions that can be used by both the scripts within the toolkit and additional product modules.

JetPXE allows a DHCP-enabled JumpStart Enterprise Toolkit server to PXEboot x86-64 based clients running the Solaris OS. Currently Solaris OS versions 8, 9 and 10 are supported. JetPXE provides many ready-to-use scripts that simplify the process of setting up and using JumpStart.

The JumpStart Enterprise Toolkit is available for download at *http://www.sun.com/download/products.xml?id=3f5e55d1.html.*

10

Summary

As organizations turn to JumpStart technology to deploy systems throughout the enterprise with consistent and reliable configurations, they are better able to fully automate the Solaris OS installation process. Today, many organizations are seeking new ways to take advantage of existing JumpStart servers and use them to deploy the Solaris OS and Linux operating environment on Sun x86-64 platforms. With the information contained in this Blueprint Article, organizations are better able to utilize the same install server used to JumpStart Sun SPARC-based systems to:

- Add Preboot eXecution Environment (PXE) support to standard JumpStart install servers running on SPARC/Solaris systems
- Perform a remote installation of Solaris OS on Sun x86-64 platforms
- Perform a remote installation of SuSE Linux on Sun x86-64 platforms
- Perform a remote installation of Red Hat Linux on Sun x86-64 platforms

About the Author

Pierre Reynes is currently working in Technology Marketing focusing on x64-based servers, and networking and security solutions. Before joining Sun in 1999, he worked as a Technical Director at DNS Telecom in Paris, France, and was in charge of secured network offerings and distribution of Cobalt Internet appliances. Prior to that he co-founded a multimedia design and hosting company in France. Throughout his career, he has helped define, document and market x86-64 platforms, networking infrastructure components and software, firewalls, and data communication systems.

Acknowledgements

The author would like to recognize the following individuals for their contributions to this article:

- Greg Cross, formerly of Sun Microsystems

Ordering Sun Documents

The SunDocs[SM] program provides more than 250 manuals from Sun Microsystems, Inc. Readers living in the United States, Canada, Europe, or Japan, can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` Web site enables users to access Sun technical documentation online. Users can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: `http://www.sun.com/blueprints/online.html`

References

Solaris DHCP Administration Guide. 806-5529. To access this book online, go to `http://docs.sun.com`.

Solaris Installation Guide. 817-5768. To access this book online, go to `http://docs.sun.com`.

Sun Control Station AllStart Module. 817-3605-11. To access this book online, go to `http://docs.sun.com`.

Sun Fire V60x and Sun Fire V65X Servers Solaris Operating Environment Installation Guide. 817-2875-10. To access this book online, go to `http://www.sun.com/products-n-solutions/hardware/docs/Servers/Workgroup_Servers/Sun_Fire_V60x-V65x/index.html`

JumpStart Blueprints:
`http://www.sun.com/blueprints/browsesubject.html#jumpstart`

Preboot Execution Environment (PXE) Specification Version 2.1:
`ftp://download.intel.com/labs/manage/wfm/download/pxespec.pdf`

Linux Network Install HOWTO:
`http://www.linux.org/docs/ldp/howto/Network-Install-HOWTO.html`

Red Hat System Administration Guide — Installations:
`http://www.redhat.com/docs/manuals/enterprise/RHEL-3-Manual/sysadmin-guide/pt-install-info.html`

SuSE YaST Auto Installer:
`http://www.suse.de/~nashif/autoinstall/`