Oracle® Solaris 11 Express Automated Installer Guide

Part No: 820–6566–14
January 2011

ORACLE

110425@25097

Contents

Preface

The *Oracle Solaris 11 Express Automated Installer Guide* provides instructions for using Automated Installer (AI) to install the Oracle Solaris operating system (OS) on multiple client systems on a network. You can also use AI media to install the Oracle Solaris OS on a single SPARC or x86 platform. All cases require access to a package repository on the network to complete the installation.

Who Should Use This Book

This book is for system administrators who want to perform a hands-free installation of the Oracle Solaris 11 Express operating system over the network.

How This Book Is Organized

This book contains the following chapters and appendices:

- Chapter 1, "Automated Installer Overview," gives a very high level description of the process of installing using Automated Installer.
- Chapter 2, "Setting Up an AI Install Server," describes how to set up and maintain an install server and install services.
- Chapter 3, "Customizing Installations," describes how to accomplish different installations on different clients in one Automated Installer installation.
- Chapter 4, "Specifying Installation Instructions," describes in detail how to specify how you want the clients installed.
- Chapter 5, "Configuring the Client System," describes in detail how to specify post-installation client configuration as part of the automated install.
- Chapter 6, "Setting Up DHCP for AI," describes how to set up both Oracle Solaris DHCP and ISC DHCP. Automated installation requires Dynamic Host Configuration Protocol (DHCP) with Domain Name System (DNS) name resolution.
- Chapter 7, "Installing Client Systems," describes how to associate a particular client with a particular install service and how to initiate the automated installation.
- Chapter 8, "Automated Installations That Boot From Media," explains how to install using AI media.

- Appendix A, "Troubleshooting Automated Installations," gives help with diagnosing and solving problems.
- Appendix B, "Automated Installer Installation Administration Commands," provides the `installadm`(1M) man page.
- Appendix C, "Migrating From JumpStart to Automated Installer," provides information to help you migrate from Solaris Custom JumpStart to Oracle Solaris Automated Installer.

Related Information

In addition to the documentation listed, see the Oracle Solaris 11 Express System Administrator documentation for more information about how to administer Oracle Solaris systems.

- *Getting Started With Oracle Solaris 11 Express* explains how to install the Oracle Solaris 11 Express OS using the live CD or the text installer.
- *Oracle Solaris 11 Express Image Packaging System Guide* explains how to use IPS publishers and package repositories and how to install and update software after the initial system installation. See also `pkg`(1).
- *System Administration Guide: IP Services* discusses Oracle Solaris DHCP.
- *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* discusses Oracle Solaris DNS.
- *Oracle Solaris ZFS Administration Guide* explains how to set up and administer Oracle Solaris ZFS file systems. See also `zfs`(1M) and `zpool`(1M).
- Service Management Facility (SMF) on Oracle Technology Network, including:
  - Service Management Facility (smf) Quickstart Guide
  - Service Management Facility (smf) Service Developer Introduction
  - How to Create a Service Management Facility (login required)

Documentation, Support, and Training

See the following web sites for additional resources:

- Documentation (`http://docs.sun.com`)
- Support (`http://www.oracle.com/us/support/systems/index.html`)
- Training (`http://education.oracle.com`) – Click the Sun link in the left navigation bar.

Oracle Software Resources

Oracle Technology Network (`http://www.oracle.com/technetwork/index.html`) offers a range of resources related to Oracle software:

- Discuss technical problems and solutions on the Discussion Forums (`http://forums.oracle.com`).
- Get hands-on step-by-step tutorials with Oracle By Example (`http://www.oracle.com/technetwork/tutorials/index.html`).
- Download Sample Code (`http://www.oracle.com/technology/sample_code/index.html`).

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

◆ ◆ ◆ CHAPTER 1

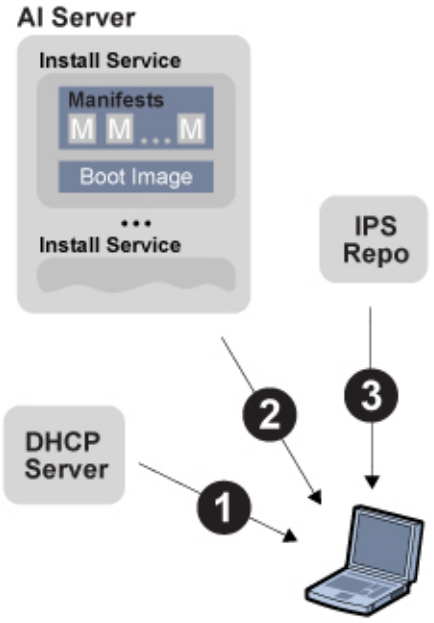
1 Automated Installer Overview

Use Automated Installer (AI) to install the Oracle Solaris operating system (OS) on multiple client systems in a network. AI performs a hands-free installation of both x86 and SPARC systems. You can also use AI media to install the Oracle Solaris OS on a single SPARC or x86 platform. All cases require access to a package repository on the network to complete the installation.

What Is an Automated Installation?

AI automates the installation of the Oracle Solaris OS on one or more SPARC or x86 clients in a network. The clients can differ in architecture, disk and memory capacity, and other characteristics. The installations can differ in network configuration, packages installed, and other specifications.

FIGURE 1–1 Automated Installation of Clients in a Network



An automated installation of a client in a local network consists of the following high-level steps:

1. A client system boots and gets IP information from the DHCP server.
2. Characteristics of the client determine which AI service and which installation instructions are used to install the client.
3. The Oracle Solaris OS is installed on the client, pulling packages from the package repository specified by the installation instructions in the AI service.

How Do I Use the Automated Installer?

To use AI to install client systems over the network, you must set up DHCP and set up an AI service on an AI server. See Chapter 2, "Setting Up an AI Install Server." AI uses DHCP to provide the IP address, subnet mask, router, DNS server, and the location of a boot image to the client machine to be installed. The DHCP server and AI server can be the same machine or two different machines. See Chapter 6, "Setting Up DHCP for AI."

The client machines you want to install must be able to access an Oracle Solaris Image Packaging System (IPS) package repository. The IPS package repository can be on the AI server or on another server on the local network, or the package repository can be on the Internet.

An AI service is associated with a SPARC or x86 AI install image and one or more sets of installation instructions. The AI image is not a complete installation. Client machines must access an IPS package repository to complete their installation. The installation instructions specify one or more IPS package repositories where the client retrieves the packages needed to complete the installation. The installation instructions also include the names of additional packages to install and information such as target device and partition information. See Chapter 4, "Specifying Installation Instructions," for more information. You can also specify instructions for post-installation configuration of the client. See Chapter 5, "Configuring the Client System."

If two client machines are different architectures or need to be installed with different versions of the Oracle Solaris OS, then create two AI services, and associate each AI service with a different AI image.

If two client machines need to be installed with the same version of the Oracle Solaris OS but need to be installed differently in other ways, then create two sets of installation instructions for the AI service. The different installation instructions can specify different packages to install or a different slice as the install target, for example.

The installation begins when you boot the client. When the client boots, DHCP directs the client to the AI install server, and the client accesses the correct install service and the correct installation instructions within that service. Chapter 7, "Installing Client Systems," explains how a client is associated with a particular install service. Chapter 3, "Customizing Installations," explains how a client finds the correct installation instructions to use.

◆ ◆ ◆ CHAPTER 2

Setting Up an AI Install Server

To install clients on a network, AI uses a separate install server. On the AI server, store an AI install image and create an AI service. The AI service specifies instructions for installing the Oracle Solaris OS on different clients.

AI Server Setup Task Map

The following task map summarizes the steps to set up an AI server.

TABLE 2–1 Task Map

Task	Reference
Check whether the server meets the minimum hardware requirements to be an AI server.	See "AI Server Hardware Requirements" on page 16.
Make sure the AI server is running the Oracle Solaris 11 Express operating system. Check other software requirements for the AI server.	See "AI Server Software Requirements" on page 16.
Configure the AI server to use a static IP address.	See "AI Server Software Requirements" on page 16.
Install the AI tool set.	See "Install the AI Installation Tools" on page 17.
Get an AI install image.	See "Set Up an AI Boot Image" on page 18.
Set up an install service.	See "Create an AI Install Service" on page 19. You need a separate install service for each architecture you want to install because an install service is associated with only one install image.
Review default installation specifications.	If the default installation specifications are not suitable for your clients, you can customize the instructions. See "Review the Default Installation Instructions" on page 21.

Install Server Requirements

Any system that meets these requirements can be used as an AI server, including laptops, desktops, virtual machines, and enterprise servers. The AI server can be either an x86 machine or a SPARC machine. An x86 install server can install both SPARC and x86 clients, and a SPARC install server can install both SPARC and x86 clients.

AI Server Hardware Requirements

If you need to install or update the Oracle Solaris 11 Express release on your AI install server, check the installation documentation for memory and disk space requirements.

Memory The minimum requirement to operate as an AI install server is 1 GB of memory.

Disk space After the Oracle Solaris 11 Express release is installed on your AI install server, you need approximately 0.75 GB additional disk space for each AI install service that you plan to create. You need a separate install service for each different client architecture and each different version of the operating system that you plan to install. See "Set Up the Install Image and Install Service" on page 17.

AI Server Software Requirements

Operating system Install the Oracle Solaris 11 Express release on the AI server. To install the Oracle Solaris 11 Express release on the AI server, see *Getting Started With Oracle Solaris 11 Express* or Chapter 8, "Automated Installations That Boot From Media."

Static IP address Configure the AI server to use a static IP address. See "How to Configure an IP Interface" in *System Administration Guide: Network Interfaces and Network Virtualization*.

Default router Ensure that your AI server has a default route set by using the `netstat`(1M) command to show network status. If your AI server does not have a default route set, you can set a static default route by populating the `/etc/defaultrouter`(4) file with the IP address of a static default route for your server's network.

DHCP Set up DHCP. The AI server can also be the DHCP server. Alternatively, you can use a DHCP server that is already set up in this network. You need different DHCP configurations for each client architecture. "Create an Install Service Including Oracle Solaris DHCP Setup" on page 20 and Chapter 6, "Setting Up DHCP for AI," provide information about setting up and configuring DHCP for AI.

Set Up the Install Image and Install Service

Perform the following steps to set up an install image and install service:

- Install the AI installation tool set.
- Download an AI boot image.
- Create an AI install service.

Identify Space for Your AI Image Files

The AI boot image ISO files and the expanded versions of these images must be stored on your AI server. Consider storing these files on a separate ZFS file system so that you can take advantage of ZFS features such as:

- Set separate file system characteristics, such as `compression` or `atime`.
- Directly snapshot and recover specified file systems.

See the *Oracle Solaris ZFS Administration Guide* for more information and best practices for creating ZFS file systems.

Install the AI Installation Tools

The AI installation tools package provides the `installadm`(1M) commands that enable you to create and manage AI install services.

The `installadm` utility enables you to accomplish the following tasks:

- Create and enable install services.
- Set up and update a DHCP server.
- Add custom client installation and configuration instructions.
- Set criteria for clients to use custom installation and configuration instructions.

See "Manage an Install Server" on page 23 and Appendix B, "Automated Installer Installation Administration Commands," for more information about `installadm` commands.

To install the tools package, your AI install server must be able to access an Oracle Solaris Image Packaging System (IPS) package repository. Make sure you are connected to the Internet or to a local IPS package server that contains the `install/installadm` package. The `install/installadm` package that you install must be the same version as the version of the Oracle Solaris OS that the AI install server is running.

Make sure your preferred publisher is set to the repository from which you want to get the AI tools package, or specify the publisher when you install the tools package.

The following example shows that two publishers are defined:

```
# pkg publisher
PUBLISHER                 TYPE    STATUS  URI
solaris      (preferred)  origin  online  http://pkg.oracle.com/solaris/release
example.com  (non-sticky) origin  online  http://pkg.example.com/
```

The following example installs the AI tools package from the preferred publisher, `solaris`:

```
# pkg install install/installadm
```

The following example installs the tools package from the `example.com` publisher, which is not the preferred publisher:

```
# pkg install pkg://example.com/install/installadm
```

See the *Oracle Solaris 11 Express Image Packaging System Guide* for information about how to add a publisher and set a preferred publisher.

Set Up an AI Boot Image

AI uses a minimal boot image to boot the client. After the client boots, the installation of the Oracle Solaris OS continues according to the installation instructions in the install service.

▼ How to Set Up an AI Boot Image

1 Assume the root role:

```
$ su - root
```

2 Create a ZFS file system in the root pool for the boot images.

In this example, the new ZFS file system is named `ai`.

```
# zfs create rpool/ai
```

In many cases, setting the `compression=on` option is a good practice. In this case, you might gain very little from specifying compression because the AI ISO files are already compressed. Similarly, setting `dedup=on` can be a good practice. See "Introducing ZFS Properties" in *Oracle Solaris ZFS Administration Guide* for information on ZFS file system properties. See "The dedup Property" in *Oracle Solaris ZFS Administration Guide* for more information about deduplication.

3 Download the boot image.

To download the AI boot image, go to the following Internet location:
`http://www.oracle.com/technetwork/server-storage/solaris11/downloads/index.html`

Be sure to download the Automated Install image and not the live CD image or the text install image. Download the Oracle Solaris 11 Express SPARC Automated Install image for SPARC clients or the Oracle Solaris 11 Express x86 Automated Install image for x86 clients. For x86, be sure to download the `.iso` file and not the `.usb` file. (The `.usb` file can be used to initiate an AI install by booting a system from a USB device. See Chapter 8, "Automated Installations That Boot From Media." The `.usb` file is not suitable for creating an install service.)

The AI ISO image must be the same version as the Oracle Solaris OS that you plan to install on the client.

Create an AI Install Service

Create an install service to associate an install image with a named install service. Client systems use the install service name to find the correct install image.

An install server can have more than one install service. Each install service is associated with only one boot image. To install both SPARC and x86 clients, for example, you need one install service with a SPARC boot image and a second install service with an x86 boot image.

Use the `installadm create-service` command to create an AI install service. Give the service a name, specify the path to the ISO image to use, and specify the path where you want the ISO image unpacked. The path where the ISO image is unpacked is also called the target or net image.

If you do not supply a name, the name assigned to the install service is `_install_service_`*port_number*.

If you do not specify the path to the ISO image to use, then the path you specify for the net image must already contain the unpacked ISO image files.

The `installadm create-service` command also provides a net install image on a web server running on port 5555. For example, the web server address might be `http://`*ai server*`:5555/export/aiserver/s11-ai-x86/s11-x86`.

The `create-service` command can set up Oracle Solaris DHCP on the AI install server as shown in "Create an Install Service Including Oracle Solaris DHCP Setup" on page 20. See Chapter 6, "Setting Up DHCP for AI," if you want to do any of the following DHCP tasks:

- Set up a separate Oracle Solaris DHCP server.
- Configure an Oracle Solaris DHCP server for use with AI.
- Set up an ISC DHCP server.
- Configure an ISC DHCP server for use with AI.

The DHCP server must be able to provide DNS information to the install clients.

Create an Install Service Without DHCP Setup

The following example creates an AI install service for SPARC clients. (This process is the same for an x86 install service, though the output is different.) The `-n` option specifies the service name, and the `-s` option specifies the path to the AI ISO image file to be used to create this service. In this example, DHCP is already set up on a different server or will be set up later. If the `create-service` command does not detect DHCP on this AI install server, the output of the command displays instructions for creating a DHCP macro by using `dhtadm`(1M) to add the macro to the DHCP configuration table, `dhcptab`(4). See Chapter 6, "Setting Up DHCP for AI," for more information.

```
# installadm create-service -n s11-sparc \
-s /rpool/ai/s11_sparc/iso/s11-ai-sparc.iso \
/rpool/ai/s11_sparc/target
Setting up the target image at /rpool/ai/s11_sparc/target ...
Registering the service s11-sparc._OSInstall._tcp.local

Detected that DHCP is not set up on this server.
If not already configured, please create a DHCP macro
named dhcp_macro_s11-sparc with:
   Boot server IP (BootSrvA) : 10.6.68.29
   Boot file      (BootFile) : http://10.6.68.29:5555/cgi-bin/wanboot-cgi
If you are running the Solaris DHCP server, use the following
command to add the DHCP macro, dhcp_macro_s11-sparc:
   /usr/sbin/dhtadm -g -A -m dhcp_macro_s11-sparc \
-d :BootSrvA=10.6.68.29:BootFile=\"http://10.6.68.29:5555/cgi-bin/wanboot-cgi\":
Note: Be sure to assign client IP address(es) if needed
(e.g., if running the Solaris DHCP server, run pntadm(1M)).
Service discovery fallback mechanism set up
Creating SPARC configuration file
```

This command displays the name and values of a macro, `dhcp_macro_s11-sparc`, that you must add to the DHCP server.

If you are using the Oracle Solaris DHCP server, you can create the macro on your DHCP server by running the `dhtadm` commands shown in the above output on your DHCP server. On systems that support graphic interfaces, you can use the DHCP Manager, `dhcpmgr`(1M), instead of the `dhtadm` commands.

Create an Install Service Including Oracle Solaris DHCP Setup

You can use the `installadm create-service` command to set up an Oracle Solaris DHCP server on this AI install server. (This process is the same for both SPARC and x86 install services, though the output is different.) The following example creates an install service for x86 clients where the network consists of a single subnet and the install server also acts as the DHCP server for the network, using DNS to resolve host names. AI creates a new DHCP macro named `dhcp_macro_s11-x86`. This install service serves five IP addresses (`-c`), starting from 172.1.0.10 (`-i`). See "Configure an Existing DHCP Server for AI" on page 78 for more information.

```
# installadm create-service -n s11-x86 -i 172.1.0.10 -c 5 \
-s /rpool/ai/s11_x86/iso/s11-ai-x86.iso \
/rpool/ai/s11_x86/target
Setting up the target image at /rpool/ai/s11_x86/target ...
Registering the service s11_x86._OSInstall._tcp.local
Creating DHCP Server
Created DHCP configuration file.
Created dhcptab.
Added "Locale" macro to dhcptab.
Added server macro to dhcptab - line1-x4100.
DHCP server started.
Added network macro to dhcptab - 10.0.0.0.
Created network table.
Copying boot file to /tftpboot/pxegrub.I86PC.Solaris-1
Service discovery fallback mechanism set up
```

In this example, you can review the `menu.lst` file in `/tftpboot/menu.lst.s11_x86`.

If DHCP service is already set up on this server, you can use the `-i` and `-c` options to update the DHCP server with new IP addresses for the named AI service.

You can view the DHCP configuration results in the DHCP table by using the DHCP Manager utility, `dhcpmgr`(1M).

Review the Default Installation Instructions

An AI manifest contains installation and configuration instructions that can be used for one or more clients. Each boot image includes a default AI manifest that can be used for clients of any install service that is created using this boot image. The manifest is unpacked along with the other files in the image. For example, if the net image path for a service is `/rpool/ai/s11_sparc/target`, then the default AI manifest is in `/rpool/ai/s11_sparc/target/auto_install/default.xml`.

This is the default AI manifest for all clients that use any install service that uses this install image.

Note – Do not delete, move, or rename this `default.xml` file.

Review this default manifest to determine whether it meets the needs of all the clients that will use an install service based on this image. "Default AI Manifest" on page 34 shows a copy of the default AI manifest. The default manifest might be slightly different in different install images.

To perform different installations on different clients using the same install image, provide customized AI manifests for that install service. More than one AI manifest can be associated with each install service. Clients that do not match the criteria specified to use any custom manifest are installed using the instructions in the default AI manifest.

Note – In general, do not modify the *net_install_image_path*`/auto_install/default.xml` file. The *net_install_image_path*`/auto_install/default.xml` file will be the default AI manifest for any install service created in the future that uses this install image. The default AI manifest must work for any client that does not match a custom manifest, for any service based on this image.

Instead of modifying *net_install_image_path*`/auto_install/default.xml`, consider the following alternatives:

- Add custom manifests to specified install services.

  To create custom installation manifests and associate them with specified clients or types of clients, see Chapter 3, "Customizing Installations."

- Modify the default AI manifest for a specified install service.

  1. Copy *net_install_image_path*`/auto_install/default.xml` to a new location: *new_location*`/default.xml`.
  2. Modify *new_location*`/default.xml`.

     To change installation specifications such as target disk or additional packages to install, see Chapter 4, "Specifying Installation Instructions." To change configuration specifications such as user account or root role password, see Chapter 5, "Configuring the Client System."
  3. Add the modified default manifest to a specified install service.

     The install service uses an internal copy of the default manifest that was in the image when the service was created. Modifying the default AI manifest that is in the image directory does not modify the default AI manifest for any service that has already been created. To modify the internal copy of the default AI manifest for an existing service, add the modified manifest to the service.

     Use the `installadm add-manifest` command to add *new_location*`/default.xml` to a specified install service. See "Add an AI Manifest" on page 24.

     - The new default manifest must be named `default.xml`.
     - The value of the `name` attribute of the `<ai_instance>` tag must be `default`.
     - Do not specify any client criteria in the `add-manifest` command.

Manage an Install Server

After you have set up an AI install server, you might want to perform some of the following tasks. See also the `installadm`(1M) man page.

Add or Delete an Install Service

You need one install service for each AI boot image you want to use. You can create more than one install service associated with the same boot image. To create an install service, use the `installadm create-service` command as described in "Create an AI Install Service" on page 19.

Use the following command to delete an install service:

```
# installadm delete-service [-x] svcname
```

The *svcname* is the name of the service to delete. If you specify the `-x` option, then the associated install image is also deleted.

When you delete an install service, the associated DHCP macros remain in the DHCP table. The output of the `installadm delete-service` command tells you which DHCP macros to remove.

Enable or Disable an Install Service

Use the following command to enable an install service.

```
# installadm enable svcname
```

The *svcname* is the name of the service to enable. This command also enables the web server associated with the service.

Use the following command to disable an install service.

```
# installadm disable svcname
```

This command also disables the web server associated with the service.

Associate Clients With Install Services

The `installadm create-client` command tells a client exactly which install service to use. See "Setting Up an Install Client" on page 85 for more examples and sample output.

Add a Client To an Install Service

Use the `installadm create-client` command to associate a client with an install service. The following command adds the client with MAC address `00:14:4f:a7:65:70` to the `s11-sparc` install service.

```
# installadm create-client -e 00:14:4f:a7:65:70 -n s11-sparc
```

The following example adds an x86 client and redirects installation output to a serial console.

```
# installadm create-client -e c0ffeec0ffee -n s11-x86 -b 'console=ttya'
```

A client can be associated with only one install service. If you run the `installadm create-client` command more than once and specify the same MAC address each time, that client is associated only with the install service that was specified last.

Delete a Client From an Install Service

Use the `installadm delete-client` command to delete a client from an install service. The following command deletes the client with MAC address `00:14:4f:a7:65:70`. You do not need to specify the service name since a client can be associated with only one install service.

```
# installadm delete-client -e 00:14:4f:a7:65:70
```

Associate Client-Specific Installation Instructions With Install Services

You can specify multiple sets of installation and system configuration instructions for each install service, and you can specify which instruction set to use for each client. All of this information is associated with the install service.

Add an AI Manifest

Use the `installadm add-manifest` command to add a custom AI manifest to an install service.

```
# installadm add-manifest -m manifest -n service_name \
[-c criteria="value"|"range" ... \
| -C criteria_file]
```

The value of *manifest* is a full path and file name with `.xml` extension. The *manifest* file contains an AI manifest (installation instructions). The manifest file can also reference or embed an SC manifest (system configuration instructions).

To specify criteria to determine which clients should use the instructions in the specified manifest file, use the `-c` or `-C` option. The value of *criteria_file* is a full path and file name. Example criteria files are shown below and in Chapter 3, "Customizing Installations."

- If you replace the `default.xml` manifest for a service, do not specify client criteria. The default AI manifest is used to install all clients that do not match any custom manifest. See "Review the Default Installation Instructions" on page 21.
- For all other AI manifests that you add, either the `-c` or the `-C` option is required. Only the default AI manifest has no associated client criteria. All other AI manifests are custom AI manifests for specified clients or groups of clients.

The following command adds the `manifest_t200.xml` manifest to the `s11-sparc` install service. The `-c` option specifies that any clients that are using this install service and identify themselves as Sun Fire T200 servers are assigned the `manifest_t200.xml` installation and configuration instructions.

```
# installadm add-manifest -m /rpool/ai/s11_sparc/manifests/manifest_t200.xml \
-n s11-sparc -c platform="SUNW,Sun-Fire-T200"
```

The following command is equivalent to the preceding command if the content of the `criteria_t200.xml` file is as shown.

```
# installadm add-manifest -m /rpool/ai/s11_sparc/manifests/manifest_t200.xml \
-n s11-sparc -C /rpool/ai/s11_sparc/manifests/criteria_t200.xml
```

Following is the content of the `criteria_t200.xml` file.

```
<ai_criteria_manifest>
    <ai_criteria name="platform">
        <value>SUNW,Sun-Fire-T200</value>
    </ai_criteria>
</ai_criteria_manifest>
```

The `installadm add-manifest` command verifies that criteria of the same type do not overlap. For example, if one criteria specification matches IP addresses from 10.0.0.0 to 10.255.255.255, `installadm` exits with an error if you try to add a criteria specification that matches IP address 10.10.10.10. For more information about criteria specifications, see Chapter 3, "Customizing Installations."

Modify Criteria for an Installation Instructions Manifest

Use the `installadm set-criteria` command to update the client criteria associated with an AI manifest that you already added to a service using `installadm add-manifest`.

- Use `installadm set-criteria` with the `-a` option to add more criteria for using the specified manifest. The current criteria are retained, and the new criteria are added.
- Use `installadm set-criteria` with the `-c` or `-C` option to overwrite the criteria for using the specified manifest. The current criteria are removed and replaced by the new criteria.

```
# installadm set-criteria -m manifest -n service_name \
-a criteria="value" | "range" ... \
| -c criteria="value"|"range" ... \
| -C criteria.xml
```

The value of *manifest* is the manifest name that the `installadm list -m` command returns. See "List All Manifests" on page 28.

The following command adds criteria to the `manifest_t200.xml` manifest from "Add an AI Manifest" on page 24.

```
# installadm set-criteria -m manifest_t200.xml -n s11-sparc -a mem="4096-unbounded"
```

The result of the two commands is that the `manifest_t200.xml` manifest is used by any client that is using this install service that is a Sun Fire T200 server and that has at least 4 Gbytes of memory.

You could achieve this same result by using the `-C` option instead of the `-a` option with the following `criteria_t200.xml` file.

```
<ai_criteria_manifest>
    <ai_criteria name="platform">
        <value>SUNW,Sun-Fire-T200</value>
    </ai_criteria>
    <ai_criteria name="mem">
        <range>
            4096
            unbounded
        </range>
    </ai_criteria>
</ai_criteria_manifest>
```

Delete an Installation Instructions Manifest

Use the `installadm delete-manifest` command to remove an AI manifest from an install service. The value of *manifest* is the manifest name that the `installadm list -m` command returns. See "List All Manifests" on page 28.

You cannot delete the default AI manifest.

The following command removes the `manifest_mac1.xml` AI manifest from the `s11-sparc` install service.

```
# installadm delete-manifest -m manifest_mac1.xml -n s11-sparc
```

Show Information About Install Services

Use the `installadm list` command to show information about install services.

List All Install Services on the Install Server

The following command displays all of the install services on this server. In this example, two enabled install services are found. (Disabled services have a Status value of `off`.)

```
# installadm list

Service Name Status Arch  Port  Image Path
------------ ------ ----  ----  ----------
s11-sparc    on     sparc 46501 /rpool/ai/s11_sparc/target
s11-x86      on     x86   46502 /rpool/ai/s11_x86/target
```

Show Information for a Specified Install Service

The following command displays information about the install service specified by the `-n` option:

```
# installadm list -n s11-sparc

Service Name Status Arch  Port  Image Path
------------ ------ ----  ----  ----------
s11-sparc    on     sparc 46501 /rpool/ai/s11_sparc/target
```

List Clients Associated With Install Services

The following command lists all the clients that are associated with the install services on this install server. The clients were associated with the install services by using the `installadm create-client` command. See "Add a Client To an Install Service" on page 24.

```
# installadm list -c

Service Name Client Address    Arch  Image Path
------------ --------------    ----  ----------
s11-sparc    00:14:4F:A7:65:70 sparc /rpool/ai/s11_sparc/target
s11-x86      08:00:27:8B:BD:71 x86   /rpool/ai/s11_x86/target
             01:C2:52:E6:4B:E0 x86   /rpool/ai/s11_x86/target
```

List Clients Associated With a Specific Install Service

The following command lists all the clients that have been added to the specified install service. In the following example, one client is associated with the `s11-sparc` install service.

```
# installadm list -c -n s11-sparc

Service Name Client Address    Arch  Image Path
------------ --------------    ----  ----------
s11-sparc    00:14:4f:a7:65:70 sparc /rpool/ai/s11_sparc/target
```

Show Information About Customized Installations

The commands in this section show which AI manifests are associated with a particular install service. These commands also show which client criteria are associated with each AI manifest. See the `installadm add-manifest` and `installadm set-criteria` commands in "Associate Client-Specific Installation Instructions With Install Services" on page 24.

List All Manifests

The following command lists all custom AI manifests for all install services on this install server. The default AI manifest is not listed. If an install service has no associated custom AI manifests, then that service is not shown in this list.

The Manifest column displays the name of the manifest from the `name` attribute of the `<ai_instance>` tag. In this example, all custom AI manifests are displayed for two install services, `s11-sparc` and `s11-x86`.

```
# installadm list -m

Service Name  Manifest
------------  --------
s11-sparc     manifest_mac1.xml
              manifest_t200.xml
s11-x86       manifest_mac2.xml
              manifest_mac3.xml
              manifest_ipv41.xml
              manifest_ipv42.xml
              manifest_mem1.xml
```

List Manifests Associated With a Specified Install Service

The following example shows all custom AI manifests associated with the install service `s11-x86`. The Manifest column displays the name of the manifest from the `name` attribute of the `<ai_instance>` tag. The Criteria column shows the client criteria that are associated with each manifest.

```
# installadm list -m -n s11-x86

Manifest            Criteria
--------            --------
manifest_mac2.xml   arch = i86pc
                    mac  = 01:C2:52:E6:4B:E0

manifest_mac3.xml   arch = i86pc
                    mac  = 01:C2:52:E6:4B:E6 - 01:C2:52:E6:4B:E9

manifest_ipv41.xml  arch = i86pc
                    ipv4 = 192.168.168.12

manifest_ipv42.xml  arch = i86pc
                    ipv4 = 192.168.168.251

manifest_mem1.xml   arch = i86pc
                    mem  = 2048 MB
```

Administering the AI SMF Service

On the AI server, the SMF service `svc:/system/install/server:default` is the service that represents the overall state of the AI server application and all install services. This SMF service contains the data specific to each install service.

EXAMPLE 2–1 Enabling the AI SMF Service

The AI SMF service is enabled when you run the `installadm create-service` command. The AI SMF service also is enabled when you run any other `installadm` command that affects existing install services. To manually enable the AI SMF service, run the following command:

```
# svcadm enable svc:/system/install/server:default
```

The AI SMF service goes into maintenance mode if no install services are currently enabled on the install server.

EXAMPLE 2–2 Disabling the AI SMF Service

To disable the AI SMF service, run the following command:

```
# svcadm disable svc:/system/install/server:default
```

Do not disable the AI SMF service if any AI install service is still enabled. See "List All Install Services on the Install Server" on page 27 for information about how to see whether any install services are enabled.

◆ ◆ ◆ CHAPTER 3

Customizing Installations

To customize an installation, customize the installation instructions and the system configuration instructions. Then specify client criteria to match the customized installation and configuration instructions with clients that meet the specified criteria. The customized instructions are used instead of the default instructions to install the client.

An AI service has one or more associated AI manifests that contain installation and configuration instructions. You can specify client characteristics such as platform type or amount of memory so that clients that match the specified criteria use the associated AI manifest to complete their installation.

Matching Installation Instructions With Client Characteristics

When you use AI, you first set up a DHCP server and an AI server. The AI server has at least one AI boot image and an AI service that is associated with that AI boot image. When a client boots, DHCP directs the client to the install server. The client is associated with a particular install service. The install service uses the methods described in this chapter to match the client with the correct installation and configuration instructions to use.

To define installations that use different AI images (a SPARC image and an x86 image, or different Oracle Solaris versions), create a separate service for each image. See "Set Up the Install Image and Install Service" on page 17.

To associate a client with a specific install service, add that client to the install service. See Chapter 7, "Installing Client Systems." Specify the MAC address of the client and the name of the install service for this client to use. When the client with this MAC address boots, DHCP directs the client to the install server, and the client uses the specified install service.

To define more than one type of installation for one AI image, create additional AI manifests and add the new AI manifests to the AI service for that AI image. Specify criteria that define which clients should use which AI manifest. See "Associate Client-Specific Installation Instructions With Install Services" on page 24.

To create a custom AI manifest, start with a copy of the default manifest in *net_install_image_path*`/auto_install/default.xml` for the service. See also "Default AI Manifest" on page 34 and "Annotated AI Manifest" on page 61 for examples. To change installation specifications such as target disk or additional packages to install, see Chapter 4, "Specifying Installation Instructions." To change configuration specifications such as user account or root role password, see Chapter 5, "Configuring the Client System."

An AI manifest is selected for a client according to the following algorithm:

- If no custom AI manifests are defined for this install service, the default AI manifest is used. The default AI manifest is not associated with any client criteria. See "Review the Default Installation Instructions" on page 21.
- If custom AI manifests are defined for this install service but the client does not match criteria for any custom AI manifest, then the client uses the default AI manifest.
- If the client matches criteria that have been specified for a custom AI manifest, the client uses that manifest.

  If client characteristics match multiple AI manifests, the client characteristics are evaluated in the order shown in the following table to select a manifest for the installation. The `installadm` tool verifies that criteria of the same type do not overlap. See "Add an AI Manifest" on page 24.

  Multiple non-overlapping criteria are used in the order specified in the following table. For example, if one criteria specification matches the client's MAC address and another criteria specification matches the same client's IP address, the manifest associated with the MAC address criteria specification is used, because `mac` is higher priority for selection than `ipv4`.

TABLE 3–1 Criteria Tags and Criteria Hierarchy

Criteria Name	Description	Examples

`arch`	`uname -m`	`i86pc` or `sun4u` or `sun4v`
		
		`<ai_criteria name="arch">`
		`  <value>i86pc</value>`
		`</ai_criteria>`

`mac`	MAC address, or range of MAC addresses	`<ai_criteria name="mac">`
		`  <value>0:14:4F:20:53:97</value>`
		`</ai_criteria>`
		
		`<ai_criteria name="mac">`
		`  <range>`
		`  0:14:4F:20:53:94`
		`  0:14:4F:20:53:A0`
		`  </range>`
		`</ai_criteria>`

TABLE 3–1 Criteria Tags and Criteria Hierarchy *(Continued)*

Criteria Name	Description	Examples
ipv4	IP address, or range of IP addresses	<ai_criteria name="ipv4"> <value>10.6.68.127</value> </ai_criteria> <ai_criteria name="ipv4"> <range> 10.6.88.1 10.6.68.200 </range> </ai_criteria>
cpu	uname -p	i386 or sparc <ai_criteria name="cpu"> <value>sparc</value> </ai_criteria>
platform	uname -i	i86pc or SUNW,Sun-Fire-T200 or SUNW,Sun-Fire-880 <ai_criteria name="platform"> <value>SUNW,Sun-Fire-T200</value> </ai_criteria>
mem	Memory size in megabytes, or a range of memory size	This example matches clients with at least 2 Gbytes of memory. The unbounded keyword indicates no upper limit in a range. <ai_criteria name="mem"> <range> 2048 unbounded </range> </ai_criteria>

EXAMPLE 3–1 Matching Clients With AI Manifests

In the following example, two custom AI manifests have been added to the same install service. The client criteria associated with those manifests are as shown.

The `manifest_x86.xml` AI manifest was added to the service with the following criteria file that specifies client architecture:

```
<ai_criteria_manifest>
    <ai_criteria name="arch">
        <value>i86pc</value>
    </ai_criteria>
</ai_criteria_manifest>
```

EXAMPLE 3–1 Matching Clients With AI Manifests *(Continued)*

The `manifest_mac1.xml` AI manifest was added to the service with the following criteria file that specifies a client MAC address:

```
<ai_criteria_manifest>
    <ai_criteria name="mac">
        <value>00:14:4f:a7:65:70</value>
    </ai_criteria>
</ai_criteria_manifest>
```

If an x86 client is being installed, it is assigned `manifest_x86.xml`.

If a SPARC client with MAC address `00:14:4f:a7:65:70` is being installed, it is assigned `manifest_mac1.xml`.

If a SPARC system with some other MAC address is being installed, it is assigned the default AI manifest.

Default AI Manifest

When you create a new install service, *install_service_image_path*`/auto_install/default.xml` is the default AI manifest for all clients that use that service.

The default AI manifest is shown below. This default manifest might be slightly different in different install images.

- The default manifest has no target device section. "Defining a Target for the Installation" on page 41 describes how the default target location for the installation is determined.
- The software installation instructions specify the default IPS package repository and install the two packages that are required for every installation.
- The `add_drivers` instructions search each client for any devices that are missing drivers. Any drivers that are found in the default IPS package repository and are not identified as third-party drivers are installed. To also install any third-party drivers that are needed, specify `<search_all addall="true"/>`.
- The system configuration instructions are also called the SC manifest. The embedded SC manifest is enclosed in `<sc_embedded_manifest>` and `</sc_embedded_manifest>` tags. The SC manifest specifies how to configure the client after installation is complete. The SC manifest defines SMF service properties. Notice that the entire SC manifest is inside a comment.

  The default user name is `jack` and the default user password is `jack`. The default root password is `solaris`.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.

-->
<!DOCTYPE auto_install SYSTEM "file:///usr/share/auto_install/ai.dtd">
<auto_install>
  <ai_instance name="default">
    <software>
      <source>
        <publisher name="solaris">
          <origin name="http://pkg.oracle.com/solaris/release"/>
        </publisher>
      </source>
      <!--
    By default the latest build available, in the specified IPS
    repository, is installed.  If another build is required, the
    build number has to be appended to the 'entire' package in following
    form:

    <name>pkg:/entire@0.5.11-0.build#</name>
      -->
      <software_data action="install" type="IPS">
        <name>pkg:/entire</name>
        <name>pkg:/babel_install</name>
        <!--
        The following packages are required by iSCSI and included
        by default to make it easier for users to enable iSCSI if
        desired. They can be deleted from this list if iSCSI isn't
        used. See iscsiadm(1m) man page for more information.
        support for iSCSI.
    -->
        <name>pkg:/network/iscsi/initiator</name>
        <name>pkg:/network/iscsi/iser</name>
      </software_data>
      <!--
        babel_install and slim_install are group packages used to
        define the default installation.  They are removed here so
        that they do not inhibit removal of other packages on the
        installed system.
        -->
      <software_data action="uninstall" type="IPS">
        <name>pkg:/babel_install</name>
        <name>pkg:/slim_install</name>
      </software_data>
    </software>
   <!--
    Add missing driver packages to a booted install image so an
    installation can complete.  Add packages to target as well.
    <search_all> searches and installs from configured repo.
    -->
    <add_drivers>
      <search_all/>
    </add_drivers>
    <sc_embedded_manifest name="AI">
      <!-- <?xml version='1.0'?>
      <!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
      <service_bundle type="profile" name="system configuration">
        <service name="system/install/config" version="1" type="service">
```

```
            <instance name="default" enabled="true">
              <property_group name="user_account" type="application">
                <propval name="login" type="astring" value="jack"/>
                <propval name="password" type="astring" value="encrypted_password"/>
                <propval name="description" type="astring" value="default user"/>
                <propval name="shell" type="astring" value="/usr/bin/bash"/>
                <propval name="uid" type='count' value='101'/>
                <propval name="gid" type='count' value='10'/>
                <propval name="type" type="astring" value="normal"/>
                <propval name="roles" type="astring" value="root"/>
              </property_group>

              <property_group name="root_account" type="application">
                  <propval name="password" type="astring" value="encrypted_password"/>
                  <propval name="type" type="astring" value="role"/>
              </property_group>

              <property_group name="other_sc_params" type="application">
                <propval name="timezone" type="astring" value="GMT"/>
                <propval name="hostname" type="astring" value="solaris"/>
              </property_group>
            </instance>
          </service>
          <service name="system/console-login" version="1" type="service">
            <property_group name="ttymon" type="application">
              <propval name="terminal_type" type="astring" value="sun"/>
            </property_group>
          </service>

          <service name='system/keymap' version='1' type='service'>
            <instance name='default' enabled='true'>
              <property_group name='keymap' type='system'>
                <propval name='layout' type='astring' value='US-English'/>
              </property_group>
            </instance>
          </service>

          <service name="network/physical" version="1" type="service">
            <instance name="nwam" enabled="true"/>
            <instance name="default" enabled="false"/>
          </service>
        </service_bundle>
        -->
      </sc_embedded_manifest>
    </ai_instance>
  </auto_install>
```

◆ ◆ ◆ CHAPTER 4

Specifying Installation Instructions

When you create an AI install service, you get a default AI manifest that specifies how to install the clients. See "Review the Default Installation Instructions" on page 21 for information about the default AI manifest and how to customize the default manifest.

This chapter explains how you can create custom AI manifests for particular clients.

AI Manifest Tags

The AI manifest is an XML file comprised of tag elements, attributes, and values that define how to install a client.

EXAMPLE 4–1 AI Manifest Template

This example shows all the tags that are required for every AI manifest. Many more tags are available, but they are not required. For example, this template does not show any tags for defining an install target or automatically installing missing drivers. Most of the tags that are available for an AI manifest are shown in Table 4–1 and discussed in this chapter.

- You must give the manifest a unique name.
- You must specify at least one IPS package repository.
- Any AI manifest must install at least the `entire` and `babel_install` packages.
- The `uninstall` action is not required. However, it is strongly recommended that any AI manifest should uninstall both the `babel_install` and `slim_install` packages. For the explanation of why these packages should be uninstalled, see Example 4–21.

```
<auto_install>
  <ai_instance name="AI_manifest_name">
    <software>
      <source>
        <publisher name="publisher_name">
          <origin name="repository_URI"/>
```

EXAMPLE 4–1 AI Manifest Template *(Continued)*

```
        </publisher>
      </source>
      <software_data action="install" type="IPS">
        <name>pkg:/entire</name>
        <name>pkg:/babel_install</name>
      </software_data>
      <software_data action="uninstall" type="IPS">
        <name>pkg:/babel_install</name>
        <name>pkg:/slim_install</name>
      </software_data>
    </software>
  </ai_instance>
</auto_install>
```

The following table lists the most common tags that are used in AI manifests. The table tells you where to find information about how to use the tags.

TABLE 4–1 AI Manifest Tags Map

Tag Elements and Attributes	Reference
`<auto_install>` `  <ai_intance name="" http_proxy="" auto_reboot="">`	"Creating a Custom AI Manifest" on page 40

`<target>`
`  <target_device>`
`  <disk>`
`  <disk_name name="" name_type=""/>`
	"Deterministic Target Disk Specifications" on page 42 in "Defining a Target for the Installation" on page 41

`<target>`
`  <target_device>`
`  <disk>`
`  <iscsi name="" source="" target_lun="" target_port="">`
`  <ip>`
	Example 4–6

`<target>`
`  <target_device>`
`  <disk>`
`  <disk_keyword key="boot_disk"/>`
	Example 4–7

`<target>`
`  <target_device>`
`  <disk>`
`  <disk_prop dev_type="" dev_vendor="" dev_size=""/>`
	"Nondeterministic Target Disk Specifications" on page 45

TABLE 4–1 AI Manifest Tags Map *(Continued)*

Tag Elements and Attributes	Reference

`<target>`
`  <target_device>`
`  <disk>`
`  <partition action="">`
`  <partition name="" part_type="">`
`  <size start_sector="" val=""/>`
	"Configuring Partitioning on an x86 Client" on page 46

`<target>`
`  <target_device>`
`  <disk>`
`  <slice action="">`
`  <slice name="" is_root="" force="">`
`  <size val=""/>`
	"Configuring Slices on a Disk" on page 50

`<target>`
`  <target_device>`
`  <swap>`
`  <zvol action="" name="">`
`  <size val=""/>`
	"Configuring Swap and Dump on the Install Device" on page 52

`<target>`
`  <target_device>`
`  <dump>`
`  <zvol action="" name="">`
`  <size val=""/>`
	"Configuring Swap and Dump on the Install Device" on page 52

`<software name="">`
`  <source>`
`  <publisher name="">`
`  <origin name=""/>`
	"Specifying a Source of Packages to Install" on page 54 in "Installing Software" on page 53

`<software name="">`
`  <software_data action="" type="">`
`  <name>`
	"Specifying Packages to Install" on page 56

`<add_drivers>`
`  <search_all addall="">`
`  <source>`
`  <publisher name="">`
`  <origin name=""/>`
	"Identify and Install Missing Drivers on an Install Target" on page 57

Creating a Custom AI Manifest

To create and apply a custom AI manifest, follow these steps:

1. Copy an existing AI manifest. A copy of the default AI manifest that you get when you create a new install service is located at *install_service_image_path*`/auto_install/default.xml`.

   Copy this default manifest or some other AI manifest to a new file name.
2. Modify the new file, adding tags and values according to the information in this chapter.
3. Add the new AI manifest to the appropriate AI install service, specifying criteria that define which clients should use these installation instructions. See "Add an AI Manifest" on page 24 for examples of adding an AI manifest to an install service. See Chapter 3, "Customizing Installations," for information about how to specify client criteria.

Note – If an invalid manifest is provided to a client, the automated installation aborts. To investigate the cause of the validation failure, see `/tmp/install_log` on the client.

The `<auto_install>` and `<ai_instance>` tags enclose the entire AI manifest.

The `<ai_instance>` tag has the following attributes:

- `name` – Required. You must give the manifest a name. The name value must be unique among AI manifests in a particular install service.

  ```
  <auto_install>
    <ai_instance name="AI_manifest_name">
      AI_manifest_contents
    </ai_instance>
  </auto_install>
  ```
- `auto_reboot` – Optional. Omitting the `auto_reboot` attribute is equivalent to setting the value of the attribute to false. By default, AI does not automatically reboot the client after installation. To request automatic reboot of the client after successful installation, specify `auto_reboot="true"`.

  Tip – Setting `auto_reboot` to true is not recommended for x86 clients since boot order is not guaranteed for x86 machines.

  ```
  <auto_install>
    <ai_instance name="AI_manifest_name" auto_reboot="true">
      AI_manifest_contents
    </ai_instance>
  </auto_install>
  ```
- `http_proxy` – Optional. The `http_proxy` attribute helps identify the source of the software to install if DNS is not used. The value of the `http_proxy` attribute is a URI. See "Specifying an HTTP Proxy to Reach an IPS Repository" on page 55.

An AI manifest typically has the following sections inside the <auto_install> and <ai_instance> tags:

- <target> – Optional. See "Defining a Target for the Installation" on page 41.
- <software> – Required. See "Installing Software" on page 53.
- <add_drivers> – Optional. See "Identify and Install Missing Drivers on an Install Target" on page 57.

Defining a Target for the Installation

If you do not specify a target location on a client for installing the Oracle Solaris OS, AI selects a default target.

The default target location for the installation is the first disk found on each client that meets the size requirement. See "Client System Requirements" on page 84. If the size of a disk is greater than or equal to the recommended size, the installer selects that disk as the installation target. If the size of the disk is less than the recommended size, the installer checks the next disk. If no disk is found that meets the size requirement, the automated installation fails for that client. The install log at /tmp/install_log will contain details of the disk selection process.

If you want to specify the target location on a client for installing the Oracle Solaris OS, follow the instructions in this section.

Use the <target> tag to define a target for the installation on each client. The <target> tag encloses the <target_device> tag. The <target_device> tag has the following elements:

- <disk> – Optional. Use this element to define a target disk, partition, or slice for the installation for every client that uses this AI manifest. Each AI manifest can specify only one <target><target_device><disk> section. If you specify a partition or slice, that specification must occur inside a disk specification. A disk can be specified by a name, a set of properties to describe the disk, a keyword, or an iSCSI device. The specified disk is the root device.

  A slice within a disk can be designated to be the root slice. If no slice is specified, then the root slice is 0 and AI installs the Oracle Solaris OS on slice 0.

  For an x86 system, if the disk is partitioned, then the install target must be a Solaris partition. Only one Solaris partition can exist on a disk. You can use an existing Solaris partition or create a new Solaris partition.

AI uses both deterministic and nondeterministic methods to specify an installation target disk.

- "Deterministic Target Disk Specifications" on page 42 – Deterministic specification includes, for example, the disk name or device ID. Specify the keyword `boot_disk` to install on the boot disk. If you specify the target by using any of the deterministic specifiers, or by using the `boot_disk` keyword, then you cannot use any other target specifier since the target is already determined.
- "Nondeterministic Target Disk Specifications" on page 45 – Nondeterministic specification includes, for example, device type, device vendor, or disk size. You can use more than one nondeterministic specifier in one AI manifest.

You can specify a partition or slice on a disk to be the target of the automated installation. You can also use AI to configure disk partitions and slices that are not the target for the automated installation. See "Configuring Partitioning on an x86 Client" on page 46 and "Configuring Slices on a Disk" on page 50.

- `<swap>` and `<dump>` – Optional. You can use AI to configure swap and dump during the automated install. See "Configuring Swap and Dump on the Install Device" on page 52.

Deterministic Target Disk Specifications

Each specification described in this section determines the one particular target on which to install the system. If you use any specification described in this section, then you cannot use any other target specification.

Tag elements in this section enable you to specify a device name, a volume name, a device ID, a device path, an iSCSI device, or the boot disk.

Note – Logical device names (`c#t#d#`) can change across OS releases.

EXAMPLE 4–2 Specifying a Target Device Name

Use a `name_type` of `ctd` to specify a device name. This is the default if `name_type` is not specified.

The following example specifies a logical device name.

```
<target>
  <target_device>
    <disk>
      <disk_name name="c1t0d0" name_type="ctd"/>
    </disk>
  </target_device>
</target>
```

The following example specifies an MPXIO name.

EXAMPLE 4–2 Specifying a Target Device Name *(Continued)*

```
<target>
  <target_device>
    <disk>
      <disk_name name="c0t2000002037CD9F72d0" name_type="ctd"/>
    </disk>
  </target_device>
</target>
```

EXAMPLE 4–3 Specifying a Target Volume Name

Use a `name_type` of `volid` to specify a volume name.

```
<target>
  <target_device>
    <disk>
      <disk_name name="ai-disk" name_type="volid"/>
    </disk>
  </target_device>
</target>
```

This volume name might have been set by using the `format`(1M) command, as shown in the following example.

```
$ format -d c0d0 > /dev/null 2>/dev/null - <<EOF
volname
"ai-disk"
y
quit
EOF
```

EXAMPLE 4–4 Specifying a Target Device ID

Use a `name_type` of `devid` to specify a device ID.

```
<target>
  <target_device>
    <disk>
      <disk_name name="id1,sd@n500000e012596560" name_type="devid"/>
    </disk>
  </target_device>
</target>
```

To get the device ID associated with a device name of the form `c#t#d#`, you can use the `iostat`(1M) command with the `-iEn` options, for example.

```
$ iostat -iEn
c7t0d0             Soft Errors: 0 Hard Errors: 0 Transport Errors: 0
Model: ST31000340NS    Revision:  Device Id:
id1,sd@n500000e012596560
...
```

EXAMPLE 4–4 Specifying a Target Device ID *(Continued)*

The `iostat` command does not report the device ID for LDOM guests. For LDOM guests, use the target discovery test driver as shown in the following example. This test driver is available from the `system/install/tests` IPS package and also is on the AI image. Boot the AI image to run this command on a client that does not have access to IPS packages.

```
# /opt/install-test/bin/tdmgtst -dv | grep ddm_disk_dev_id
ddm_disk_dev_id=id1,vdc@f8498536e4a8ad037000bcb400001
```

Device ID target specification does not work in xVM/PV because the device ID is not available for virtual drives.

EXAMPLE 4–5 Specifying a Target Device Path

Use a `name_type` of `devpath` to specify a device path under `/devices`.

```
<target>
  <target_device>
    <disk>
      <disk_name name="/pci@0/pci@9/pci@0/scsi@1/sd@0,0" name_type="devpath"/>
    </disk>
  </target_device>
</target>
```

The physical path is the path that the related `c#t#d#s#` symbolic link is linked to under `devices`, as shown in the following example.

```
$ ls -l /dev/dsk/c2t0d0s0
/dev/dsk/c2t0d0s0 -> ../../devices/pci@7c0/pci@0/pci@1/pci@0/ide@8/sd@0,0:a
```

EXAMPLE 4–6 Specifying an iSCSI Target

iSCSI technology enables you to install to a disk drive hosted by another computer on the same TCP/IP network.

To specify an iSCSI target as the install target, you must first create an iSCSI boot target. Use the `iscsiadm`(1M) command to create an iSCSI boot target and to get the name and IP values for the `iscsi` tag in the AI manifest.

To specify an iSCSI target for automated installations, specify an iSCSI target name and an IP address as shown in the following example. The target name can be an IQN, EUI, or NAA name. The name value can be a maximum of 233 characters.

```
<target>
  <target_device>
    <disk>
      <iscsi name="c0d2E0001010F68">
        <ip>192.168.1.34</ip>
      </iscsi>
    </disk>
```

EXAMPLE 4–6 Specifying an iSCSI Target *(Continued)*

```
  </target_device>
</target>
```

EXAMPLE 4–7 Specifying the Boot Disk as Target

The `<disk_keyword>` tag accepts `boot_disk` as a keyword. This keyword specifies the current boot disk as the installation target.

```
<target>
  <target_device>
    <disk>
      <disk_keyword key="boot_disk"/>
    </disk>
  </target_device>
</target>
```

Note the following limitations to the `boot_disk` keyword.

- The `boot_disk` keyword does not work in xVM/PV, because information about the boot disk is not available in that environment.
- On some x86 systems, the BIOS does not report the boot disk correctly.
- For SPARC clients, if the `diag-switch?` OBP property is set to `true`, information about the boot disk is not available. To use the `boot_disk` keyword for SPARC clients, make sure `diag-switch?` is set to `false`.

  ```
  # eeprom diag-switch?=false
  ```

Nondeterministic Target Disk Specifications

You can use more than one nondeterministic specifier in one AI manifest, as shown in the following example.

```
<target>
  <target_device>
    <disk>
      <disk_prop dev_vendor="hitachi" dev_size="20gb"/>
    </disk>
  </target_device>
</target>
```

Use the `<disk_prop>` tag to specify a target type, a vendor name, or a target size.

- `dev_type` – The type of the target disk. Possible values include SCSI, ATA, and USB. This value is not case sensitive.
- `dev_vendor` – The manufacturer of the disk drive. Possible values include Sun, Hitachi, EMC, Seagate, Fujitsu. This value is not case sensitive.
- `dev_size` – The size of the target device. This size value must have a units suffix.

Valid units suffixes include the following:

- sectors, secs, sec, s, SECTORS, SECS, SEC, or S
- MB, megabytes, megabyte, mb, m, MEGABYTES, MEGABYTE, M
- GB, gigabytes, gigabyte, gb, g, GIGABYTES, GIGABYTE, G
- TB, terabytes, terabyte, tb, t, TERABYTES, TERABYTE, T

The following examples show two different methods to determine the vendor of a disk. Note that these searches cannot identify the vendor for ATA disks.

Use the `iostat`(1M) command:

```
# iostat -En
c2t1d0           Soft Errors: 0 Hard Errors: 0 Transport Errors: 0
Vendor: HITACHI  Product: HUS10733ASUN72G  Revision: PA05 Serial No: 0602RW159S
```

Use the target discovery test driver that is available in the AI image:

```
# /opt/install-test/bin/test_td -dv
Disk discovery
Total number of disks: 2
--------------------------------------------------------------------------------
 num |    name|    vendor|  ctype| mtype| rem| lbl| bsize|#of blocks|size [MB]|
--------------------------------------------------------------------------------
   1 |* c2t0d0|   SEAGATE|   scsi| FIXED|  No|   V|   512|  71132959|    34732|
   2 |  c2t1d0|   HITACHI|   scsi| FIXED|  No|   V|   512| 143374738|    70007|
--------------------------------------------------------------------------------
```

Configuring Partitioning on an x86 Client

You can use the AI manifest to configure partitions on a disk during the automated installation. You can specify a partition as the install target, and you can delete existing partitions, change the size or type of an existing partition, and create new partitions.

Note – Partition operations are meaningful only for x86 clients.

Use the <partition> tag to configure partitioning of a disk. The <partition> tag must occur inside a <disk> tag. The <partition> tag has the following attributes:

- `action` – Optional. The `action` attribute has the following values:
  - `create` – Create a new partition. This is the default if `action` is not specified.
  - `delete` – Delete the named partition.
  - `use_existing` – Use the existing Solaris partition as the install target. If a Solaris partition exists, that partition is the install target by default. If you specify `use_existing`, and no Solaris partition exists, then the automated installation fails for that client. Only one `use_existing` action can be specified in an AI manifest.

- `name` – Required if the action is `create` or `delete`. The `name` attribute denotes the unique partition identification number in the partition table. For primary partitions, `name` can be 1, 2, 3, or 4. Only one primary partition can be an extended partition (partition type `DOSEXT`). For logical partitions, `name` can be an integer from 5 through 36.

  When the `action` is `create`, then AI creates the partition with the specified `name` during automated installation. When the `action` is `delete`, then AI deletes the partition with the specified `name` during automated installation.

  Note – Partitioning changes made during automated installation occur in the order in which the `delete` and `create` actions are listed in the AI manifest.

- `part_type` – Optional. The `part_type` attribute denotes the type of the partition to be created or deleted. The default value is 191 (Solaris partition) if `part_type` is not specified. Valid values of `part_type` are `SOLARIS` (191), `UNIXOS` (99), `DOSEXT`, `DOS16`, `DOSEXTLBA`, and `FAT32`.

When you create a partition, you can optionally specify the size of the new partition. If only one partition is specified, and the size value is not specified, then the following algorithms are used to calculate the size of the new partition. If more than one partition is specified, then the size value can be omitted for no more than one of the partition specifications.

- For extended or primary partitions, the default size is the remaining free space on the disk.
- For a logical partition in an extended partition, the default size is the largest block of free space in the extended partition for the logical partition.

The `<size>` tag has the following attributes.

- The `start_sector` attribute is the sector number where the partition should start.
- The `val` attribute is the size of the partition. This size value must have a units suffix. See the list of units suffixes in the description of the `dev_size` disk property in "Nondeterministic Target Disk Specifications" on page 45.

Specifying a Partition as the Installation Target for an x86 Client

On an x86 client, the installation target can be a disk partition. If the disk is partitioned, then the install target must be a Solaris partition. Only one Solaris partition can exist on a disk. You can use an existing Solaris partition or create a new Solaris partition.

If a partition exists that is partition type `SOLARIS` or 191, that partition is the install target by default. If no Solaris partition exists, and the installation instructions do not create a Solaris partition, then AI creates a Solaris partition with the remaining disk space. If this space is less than the space required for an AI install, the installation fails for that client.

EXAMPLE 4–8 Specifying an Existing Solaris Partition as Install Target

In this example, the install target is an existing Solaris partition. The install target is the first disk found that contains a Solaris partition (partition type `SOLARIS` or 191). If no existing Solaris partition is found on a particular client, then automated installation fails for that client.

```
<target>
  <target_device>
    <disk>
      <partition action="use_existing"/>
    </disk>
  </target_device>
</target>
```

EXAMPLE 4–9 Specifying a New Partition as Install Target

This example does not specify any action. The default action is create. This example creates two new primary partitions.

```
<target>
  <target_device>
    <disk>
      <partition name="1" part_type="191">
        <size start_sector="200" val="20gb"/>
      </partition>
      <partition name="4" part_type="99">
        <size start_sector="2000" val="20gb"/>
      </partition>
    </disk>
  </target_device>
</target>
```

Modifying Partitions on an x86 Install Client

In addition to specifying a partition as the install target, you can also delete existing partitions and create new partitions during the automated installation. You can create an extended partition, or create logical partitions in a new or existing extended partition.

Note – Partitioning changes made during automated installation occur in the order in which the `delete` and `create` actions are listed in the AI manifest.

EXAMPLE 4–10 Deleting an Existing Partition

You can delete existing partitions during client installation. Specify the `delete` action in the `<partition>` tag and specify the partition number to delete in the `name` attribute.

```
<target>
  <target_device>
    <disk>
      <partition action="delete" name="3"/>
    </disk>
```

EXAMPLE 4–10 Deleting an Existing Partition *(Continued)*

```
  </target_device>
</target>
```

EXAMPLE 4–11 Creating an Extended Partition

You can create an extended partition in an `fdisk` partition table as part of your automated installation. An extended partition is a primary partition (1, 2, 3, or 4) that is partition type `DOSEXT`. Only one primary partition can be an extended partition. An extended partition provides space for one or more logical partitions. Multiple logical partitions can be created in an extended partition.

This example creates a new extended partition of maximum size.

```
<target>
  <target_device>
    <disk>
      <partition name="3" part_type="DOSEXT"/>
    </disk>
  </target_device>
</target>
```

EXAMPLE 4–12 Creating a Logical Partition

This example creates a new logical partition, of the type `SOLARIS`, within the existing extended partition, using any available free space in the extended partition.

If there are no other logical partitions, the entire extended partition is used for the logical partition. If there is more than one logical partition, the total space for all logical partitions in an extended partition cannot exceed the total space in the extended partition. If no space is available for the logical partition because another logical partition is already using all the space, this partition creation fails.

```
<target>
  <target_device>
    <disk>
      <partition name="7" part_type="SOLARIS"/>
    </disk>
  </target_device>
</target>
```

EXAMPLE 4–13 Creating an Extended Partition that Contains Logical Partitions

This example creates an extended partition that contains two new logical partitions. This example accomplishes the following tasks.

- Creates an extended partition using the largest region of free space available on the disk.
- Creates a FAT32 logical partition in the extended partition, using the first contiguous 10 Gbytes of the available space.
- Creates a Solaris logical partition using the largest remaining unused space.

EXAMPLE 4–13 Creating an Extended Partition that Contains Logical Partitions *(Continued)*

```
<target>
  <target_device>
    <disk>
      <!-- Create an extended partition in the largest block of free space -->
      <partition name="4" part_type="DOSEXT"/>

      <!-- Create a 10G FAT32 logical partition on the extended partition.
           This partition consumes the 1st 10G of the extended partition. -->
      <partition name="5" part_type="FAT32">
        <size val="10gb"/>
      </partition>

      <!-- Create a Solaris logical partition using the
           largest unused free space in the extended partition.
           In this example, the partition uses the remaining space
           in the extended partition. -->
      <partition name="6" part_type="SOLARIS"/>
    </disk>
  </target_device>
</target>
```

Configuring Slices on a Disk

You can use the AI manifest to configure slices on a disk during the automated installation. You can specify a slice as the install target, and you can delete existing slices, preserve existing slices, and create new slices.

Slices are similar to partitions. The differences between slices and partitions are the following:

- A slice can be designated as the root slice.
- Slices can be overwritten.
- Slices do not have a `use_existing` action.
- Slices have a `preserve` action that enables you to specify slices to preserve during installation.

Use the `<slice>` tag to configure slices on a disk. The `<slice>` tag must occur inside a `<disk>` tag. The `<slice>` tag has the following attributes:

- `action` – Optional. The `action` attribute has the following values:
  - `create` – Create a new slice. This is the default if `action` is not specified.
  - `delete` – Delete the named slice.
  - `preserve` – Preserve the data on the named slice during installation.
- `name` – Required. The `name` attribute denotes the unique slice identification number in the VTOC table. The valid values of `name` are 0, 1, 3, 4, 5, 6, and 7. Specifying slice 2 is not allowed, since slice 2 represents the whole disk. If no slice is specified, then the root slice is 0 and AI installs the Oracle Solaris OS on slice 0.

- `is_root` – Optional. The `is_root` attribute can be applied to at most one slice. The slice with the `is_root` attribute is included in the root pool. The default value of the `is_root` attribute is false.
- `force` – Optional. The `force` attribute enables overwriting a slice. Creating a slice that already exists is an error if you do not set `force` to `true`. If you create a slice that already exists and you specify `force="true"`, then the newly created slice overwrites the existing slice. The default value of the `force` attribute is false.

When you create a slice, you can optionally specify the size of the slice. If only one slice is specified, and the size value is not specified, then the slice is the whole size of the disk. If more than one slice is specified, then the size value can be omitted for no more than one of the slice specifications. Specify the slice size in the `val` attribute of the slice `<size>` tag. This size value must have a units suffix. See the list of units suffixes in the description of the `dev_size` disk property in "Nondeterministic Target Disk Specifications" on page 45.

Specifying a Slice as the Installation Target

The installation target can be a slice on a disk.

EXAMPLE 4–14 Specifying a New Slice as Install Target

This example creates two new slices. The Oracle Solaris OS will be installed on slice 0. Slice 0 already existed and will be overwritten with the new specification.

```
<target>
  <target_device>
    <disk>
      <slice name="0" is_root="true" force="true">
        <size val="20gb"/>
      </slice>
      <slice name="4">
        <size val="20gb"/>
      </slice>
    </disk>
  </target_device>
</target>
```

Modifying Slices on the Install Client

In addition to specifying a slice as the install target, you can also create, delete, and preserve other slices during the automated installation.

EXAMPLE 4–15 Preserving a Slice

Specify the `preserve` action in the `<slice>` tag to specify that this slice should not be touched during installation. The following example shows how to preserve slice 4.

```
<target>
  <target_device>
    <disk>
```

EXAMPLE 4–15 Preserving a Slice *(Continued)*

```
      <slice name="0" is_root="true">
        <size val="20gb"/>
      </slice>
      <slice action="preserve" name="4"/>
    </disk>
  </target_device>
</target>
```

EXAMPLE 4–16 Deleting a Slice

Specify the `delete` action in the `<slice>` tag to delete this slice during installation. The following example shows how to delete slice 4.

```
<target>
  <target_device>
    <disk>
      <slice name="0" is_root="true">
        <size val="20gb"/>
      </slice>
      <slice action="delete" name="4"/>
    </disk>
  </target_device>
</target>
```

Configuring Swap and Dump on the Install Device

You can use the AI manifest to configure swap and dump during the automated install.

Use the `<swap>` tag to configure swap on the install device. Use the `<dump>` tag to configure dump on the install device.

The `<swap>` tag has an optional `no_swap` attribute. The value of the `no_swap` attribute can be true or false. The default value is false. If `no_swap="true"` is specified, swap is not configured on the install device.

The `<dump>` tag has an optional `no_dump` attribute. The `no_dump` attribute works in the same way for dump configuration as the `no_swap` attribute works for swap configuration.

The `<swap>` and `<dump>` tags use the `zvol` tag. The `zvol` tag has the following attributes:

- `action` – Optional. The default action is `create`, to configure new swap or dump.
- `name` – Required.

To specify the size of the swap or dump, use the `size` tag inside the `zvol` tag. The `val` attribute of the `size` tag must have a units suffix. See the list of units suffixes in the description of the `dev_size` disk property in "Nondeterministic Target Disk Specifications" on page 45.

EXAMPLE 4–17 Configure Swap on the Install Device

```
<target>
 <target_device>
   <swap>
     <zvol action="create" name="swap">
       <size val="20gb"/>
     </zvol>
  </swap>
 </target_device>
</target>
```

EXAMPLE 4–18 Unconfigure Swap on the Install Device

```
<target>
 <target_device>
   <swap no_swap"true"/>
 </target_device>
</target>
```

EXAMPLE 4–19 Configure Dump on the Install Device

```
<target>
  <target_device>
     <dump>
       <zvol action="create" name="dump">
         <size val="2gb"/>
       </zvol>
    </dump>
   </target_device
 </target>
```

Installing Software

The AI install image is not a complete installation. Client machines must access an IPS package repository to complete their installation. The AI manifest must specify at least one package repository location and must specify names of packages to install.

Use the `<software>` tag to define locations of package repositories and names of particular packages to install.

The `<software>` tag has the following elements:

- `<source>` – Required. Use this element to specify where AI should get packages to install. See "Specifying a Source of Packages to Install" on page 54.
- `<software_data>` – Required. Use this element to specify names of packages to install. A particular set of packages is required for every automated installation. You can choose to install additional packages. See "Specifying Packages to Install" on page 56.

An AI manifest is not required to have an equal number of <source> and <software_data> elements. One <software> element could have more <source> elements than <software_data> elements, or one <software> element could have more <software_data> elements than <source> elements.

Specifying a Source of Packages to Install

A *repository* is a location where IPS packages are published and from where packages are retrieved. A repository can be on the local network, or a repository can be on the Internet. A person or corporation who puts a package in a repository is called a *publisher*. Each AI manifest must specify at least one IPS package repository to install the Oracle Solaris OS.

Note – The Oracle Solaris OS release of the AI boot image and the IPS packages must be the same. The IPS repository or repositories specified in the AI manifest must contain packages for that release.

Use the <source> tag to specify a location where AI should get packages to install. One <software> element can contain more than one <source> element. Each <source> element can specify either an IPS package publisher or a directory path.

- <publisher> – Each AI manifest must specify at least one <publisher> element because a particular set of IPS packages is required for every Oracle Solaris automated installation. The <publisher> element has an optional name attribute. The <publisher> element has the following elements:
  - <origin> – Required. The <origin> element has a required name attribute that gives the primary URI of this IPS package repository. A <publisher> element can have only one <origin> element.
  - <mirror> – Optional. The <mirror> element has a required name attribute that gives an additional URI for this same IPS package repository. A <publisher> element can have more than one <mirror> element.

  A mirror is different from multiple repositories. A mirror is a separate location for a repository that has the same content as the <origin> repository in the same <publisher> element. To specify additional repositories with different content, use additional <source> elements.

  Note – The first IPS repository specified in an AI manifest is the preferred source for all IPS packages. If a particular IPS package is not found in the preferred repository, then additional IPS repositories are searched in the order in which they are specified in the AI manifest. See also Example 4–23.

- `<dir>` – Optional. Each `<source>` element must have either a `<publisher>` element or a `<dir>` element. The `<dir>` element has a required `path` attribute that gives the path to one or more packages to install.

EXAMPLE 4–20 Specifying IPS Package Repositories

In this example, `solaris` is the preferred IPS package publisher. This is the repository that will be searched first for IPS packages. This repository must contain the packages that are required to install the Oracle Solaris OS. This repository must contain packages for the Oracle Solaris OS that is the same release as the AI boot image that is associated with this install service.

The *mirror_repo* repository contains exactly the same content as the `solaris` repository. This repository will be used if the connection to the `solaris` repository is too slow.

The *additional_repo* repository contains content that is different from the content in the `solaris` repository. This repository will be used if a specified IPS package is not found in the `solaris` repository. The *additional_repo* repository could be a repository on the local network that contains custom IPS packages, for example.

```
<software name="IPS">
  <source>
    <publisher name="solaris">
      <origin name="http://pkg.oracle.com/solaris/release"/>
      <mirror name="http://pkg.mirror_repo"/>
    </publisher>
  </source>
  <source>
    <publisher>
      <origin name="http://pkg.additional_repo"/>
    </publisher>
  </source>
</software>
```

Specifying an HTTP Proxy to Reach an IPS Repository

A client needs to access an IPS repository to install the Oracle Solaris OS. In the recommended configuration described in Chapter 1, “Automated Installer Overview,” a DHCP server sends DNS information to the client. This DNS information is used to resolve the IPS repository URI to an IP address.

If a client does not have a direct connection to the IPS repository server, the AI manifest can specify a proxy to enable the client to get outside its network. This proxy applies to all publishers.

The value of the `http_proxy` attribute of the `<ai_instance>` element is the URL of the proxy as follows:

```
<auto_install>
  <ai_instance name="AI_manifest_name" http_proxy="http://192.168.0.101:8080">
    AI_manifest_contents
```

```
  </ai_instance>
</auto_install>
```

Specifying Packages to Install

The AI boot image is not a complete installation. Client machines must access an IPS package repository to complete their installation. The default AI manifest shows the minimum packages that must be installed to install the Oracle Solaris OS. List packages to be installed in a `<software_data>` element inside a `<software>` element in the AI manifest.

The `<software_data>` tag has the following attributes:

- `action` – Optional. The value `install` is the default if `action` is not specified. Other possible actions are `uninstall` and `noinstall`. Example 4–21 discusses the use of `uninstall`. "Identify and Install Missing Drivers on an Install Target" on page 57 discusses using `noinstall`.
- `type` – Optional. The value `IPS` is the default if `type` is not specified. `IPS` is the only valid type of package in this section of the AI manifest. "Identify and Install Missing Drivers on an Install Target" on page 57 shows examples of installing different types of packages in the `<add_drivers>` section of the AI manifest.

The `<software_data>` element contains `<name>` elements. Each `<name>` element specifies the name of a package to install. For IPS packages, the `<name>` element can specify the repository to use. See Example 4–23.

EXAMPLE 4–21 Specifying the Minimum IPS Packages To Install

This example shows the minimum IPS packages that must be installed to install the Oracle Solaris OS.

This example also specifies packages to be uninstalled. The packages to be uninstalled are group packages. Group packages are package definitions that are used for convenience to install a set of other packages. You cannot uninstall an individual package that was installed with a group package unless you first uninstall the group package definition. Uninstalling the group package definition does not uninstall all the packages that were installed with the group package. The `babel_install` package is a group package that installs `slim_install`, which is another group package. After installation, both `slim_install` and `babel_install` are uninstalled so that a user can later uninstall other packages that were installed as part of `slim_install` and `babel_install`. The ability to uninstall packages is needed to update and customize the installation. The `babel_install` package definition must be uninstalled first so that the `slim_install` package definition can be uninstalled.

```
<software name="IPS">
  <software_data type="IPS">
    <name>pkg:/entire</name>
    <name>pkg:/babel_install</name>
```

EXAMPLE 4–21 Specifying the Minimum IPS Packages To Install *(Continued)*

```
  </software_data>
  <software_data action="uninstall" type="IPS">
    <name>pkg:/babel_install</name>
    <name>pkg:/slim_install</name>
  </software_data>
</software>
```

EXAMPLE 4–22 Specifying Additional IPS Packages To Install

In this example, Oracle Solaris Studio and the NetBeans DTrace GUI Plug-in are installed.

```
<software name="IPS">
  <software_data type="IPS">
    <name>pkg:/developer/sunstudio12u1</name>
    <name>pkg:/developer/netbeans/plugin/nb-dtrace</name>
  </software_data>
</software>
```

EXAMPLE 4–23 Specifying the IPS Repository in the Package Name

In this example, the OpenOffice suite of tools and the Evolution mail and calendaring utility are installed from the `example.com` IPS repository.

```
<software name="IPS">
  <source>
    <publisher name="solaris">
      <origin name="http://pkg.oracle.com/solaris/release"/>
    </publisher>
  </source>
  <source>
    <publisher name="example.com">
      <origin name="http://pkg.example.com/release"/>
    </publisher>
  </source>
  <software_data type="IPS">
    <name>pkg://example.com/openoffice</name>
    <name>pkg://example.com/mail/evolution</name>
  </software_data>
</software>
```

Identify and Install Missing Drivers on an Install Target

The default installation might not include drivers for all devices on a client. In the AI manifest, you can request AI to do one or both of the following tasks:

- Scan a client for missing device drivers and install appropriate driver packages for that client.
- Install explicitly named driver packages on every client that uses this AI manifest.

Use the <add_drivers> tag to direct AI to do either of these tasks. In both cases, drivers are added to the boot environment first and are installed on the target later in the automated installation process. Adding missing drivers to the boot environment first gives AI access to all client devices during installation.

Note – Packages specified in the <add_drivers> section can install only kernel files. Some file systems on the live image are mounted read-only. Installation to read-only file systems, such as /usr, will fail. On Oracle Solaris live images, all of /kernel is read-write. Most drivers are installed in /kernel.

An AI manifest can specify no more than one <add_drivers> element. The <add_drivers> element can have one or both of the following elements:

- <software> – Optional. Use this element to specify sources and names for packages that are needed to perform the installation. All <software> elements specified in an <add_drivers> element must be specified before the <search_all> element.

  This <software> element includes a <source> element and a <software_data> element. The <source> element specification is affected by the <software_data> type value.

  The <software_data> element has a package type attribute and an action attribute.

  - type – Required. Possible values are P5I, SVR4, or DU. See Example 4–25, Example 4–26, and Example 4–27.
  - action – Optional. The value install is the default if action is not specified. The only other possible action is noinstall. If you set the action attribute to noinstall, the packages are not installed on the target device. The packages are still added to the boot environment and used to perform the installation.
- <search_all> – Optional. An AI manifest can specify no more than one <search_all> element. If a <search_all> element is specified, it must be specified after all <software> items that enumerate explicit driver packages.

  The <search_all> element directs AI to search the client for devices that currently have no driver in the booted install image. Any drivers that are found and that are not identified as third-party drivers are installed into the boot environment and later installed on the target device for this client.

  A *third party* driver in AI is a driver that is not found in the repository but is known to exist elsewhere, such as on a web site.

  If the optional addall attribute is set to true (see Example 4–24), any needed third-party drivers that are found are also added to the boot environment to be installed on the target. The default value of the addall attribute is false. If the addall attribute is omitted or is set to false, missing third-party drivers are not installed.

The <search_all> element has an optional <source> element where you can specify a package repository to search for missing drivers. A <search_all> element can have no more than one <source> element. Only one publisher can be specified within a <search_all> element. The <source> element within a <search_all> element cannot have a <mirror> element.

If the <search_all> element does not specify a <source> element, then the sources specified elsewhere in this AI manifest are used. The database for each repository identifies a driver as third-party or not.

If a missing driver is not found during a search, a warning is displayed but the install is attempted. If the install is successful, the system reboots if the AI manifest requests a reboot (auto_reboot="true").

If an explicitly specified driver package is not found, or a problem occurs during installation of any driver package (whether found by search or explicitly specified), that particular client installation completes and then displays a message about the missing driver and waits at a prompt. The system does not reboot automatically when the installation completes. This behavior provides the opportunity to evaluate the message and determine whether a reboot will succeed. If a reboot can succeed without the missing driver, you can reboot manually.

EXAMPLE 4–24 Searching for Missing Drivers

The <search_all> element directs AI to search for devices that are missing their drivers and search a database for any needed drivers. This example specifies a repository to search for needed drivers. This example specifies addall="true" to install drivers that are identified in the database as third-party drivers, as well as drivers that are not identified as third-party drivers.

```
<add_drivers>
  <search_all addall="true">
    <source>
      <publisher name="example.com">
        <origin name="http://pkg.example.com/release"/>
      </publisher>
    </source>
  </search_all>
</add_drivers>
```

EXAMPLE 4–25 Adding a Package Through a P5I File

This example adds an explicitly specified pkg(5) package given through a P5I file.

A P5I software data type has the following characteristics:

- The <software_data> element cannot have a <name> element.
- The name of the origin of the publisher includes the full path to the P5I file. (The P5I file contains the package name and repository.) The path can be to a local file or to an HTTP or FTP location.

EXAMPLE 4–25 Adding a Package Through a P5I File *(Continued)*

```
<add_drivers>
  <software>
    <source>
      <publisher>
        <origin name="http://myserver.example.com/drivers/p5i/0/mydriver.p5i"/>
      </publisher>
    </source>
    <software_data type="P5I"/>
  </software>
</add_drivers>
```

EXAMPLE 4–26 Adding an SVR4 Package

This example adds an explicitly specified SVR4 package. The full path name in the <origin> tag in the <source> element is the directory that contains the packages. The <name> tag in the <software_data> element is the name of the package. For local packages, this <name> can be the top level directory of a tree of package files, or <name> can be a datastream file. For remote packages, <name> is a datastream file.

This example demonstrates the noinstall action, though the noinstall action can be specified in any software_data tag. The noinstall action installs the package only in the booted environment. The package is available during installation but is not installed on the target device.

```
<add_drivers>
  <software>
    <source>
      <publisher>
        <origin name="/export/package_dir"/>
      </publisher>
    </source>
    <software_data type="SVR4" action="noinstall">
      <name>mydriver2.d</name>
    </software_data>
  </software>
</add_drivers>
```

EXAMPLE 4–27 Adding an Image

This example adds an explicitly specified Driver Update (DU) or Install Time Update (ITU) image. If the image is local, the path name in the <origin> tag in the <source> element is the parent of the DU directory of the install image if the image is expanded. If the install image is not expanded, this path name is the name of the .iso image. If the image is remote, this path name is the full HTTP or FTP path to an image .iso file. All packages in the image will be added.

When the software data type is DU, the <software_data> element cannot have a <name> element.

```
<add_drivers>
  <software>
```

EXAMPLE 4–27 Adding an Image *(Continued)*

```
    <source>
      <publisher>
        <origin name="/export/duimages/mydriverDU"/>
      </publisher>
    </source>
    <software_data type="DU"/>
  </software>
</add_drivers>
```

Annotated AI Manifest

The following file shows an AI manifest that demonstrates most tags and contains comments on usage.

```
<?xml version="1.0"?>
<!--
Copyright (c) 2008, 2010, Oracle and/or its affiliates. All rights reserved.
-->
<!--
================================================================================
DTD sample manifest for Automatic Installer input manifest specification.
================================================================================
-->
<!DOCTYPE auto_install SYSTEM "file:///usr/share/auto_install/ai.dtd">
<auto_install>
  <!--
    "auto_reboot" set to "true" may be an issue for x86 machines.
    The boot order is not guaranteed and may cause unexpected
    behavior. If auto_reboot is not desired, removing this
    attribute, e.g. <ai_instance name="sample_ai_manfiest">
    will work. "auto_reboot" is set to false by default.
    -->
  <ai_instance name="sample_ai_manifest" auto_reboot="true">
    <!--
      ======================================================================
      <target/target_device> - selections for AI target Device specification

      Disk criteria are divided into three mutually exclusive groups:

      G1 - deterministic disk criteria
      ...............................
        * target_device/disk/iscsi parameters
        * target_device/disk/disk_name, with name_type attribute:
          one of ctd, volid, devpath or devid

      G2 - non-deterministic disk criteria
      ............................
        * target_device/disk/disk_prop: Any of dev_type, dev_vendor or
          dev_size

      G3 - keyword disk criteria
      ............................
```

```
      * target_device/disk/disk_keyword: "boot_disk"

    Schema ai.dtd enforces following policy:

    * criteria in group G1 are mutually exclusive - only
      one can be specified at a time

    * groups G1, G2 and G3 are mutually exclusive - i.e.
      if criteria from G1 is specified, no criteria
      from G2 or G3 are allowed and vice versa

    * multiple criteria from G2 can be specified
    ============================================================================
  -->
  <target>
    <target_device>
      <disk>
        <!-- G1 -->
        <!--
          c#t#d# device name like c0t0d0 or
          MPXIO name like c0t2000002037CD9F72d0
        -->
        <disk_name name="c1t0d0" name_type="ctd"/>
        <!-- volume name set for instance by means
          of format(1M) command
        -->
        <!--
        <disk_name name="ai-disk" name_type="volid"/>
        -->
        <!-- device id - e.g. can be obtained by means of
          iostat(1M) -iEn
        -->
        <!--
        <disk_name name="id1,cmdk@AST31000340NS=_____________9QJ2LNYY" name_type="devid"/>
        -->
        <!-- device path under /devices directory, e.g.
          /pci@1e,600000/pci@0/pci@9/pci@0/scsi@1/sd@0,0
        -->
        <!--
        <disk_name name="/pci@0/pci@9/pci@0/scsi@1/sd@0,0" name_type="devpath"/>
        -->
        <!--
          ISCSI target device

        <iscsi name="c0d2E0001010F68">
          <ip>192.168.1.34</ip>
        </iscsi>
        -->
        <!-- G2 -->
        <!--
        <disk_prop dev_vendor="hitachi" dev_size="20480mb"/>
        -->
        <!-- G3 -->
        <!--
        <disk_keyword key="boot_disk"/>
        -->
        <!--
          Uncomment this to force AI to find an existing Solaris
          partition instead of creating a new one.
```

```
      -->
      <!--
      <partition action="use_existing"/>
      -->
      <partition name="1" part_type="99">
        <size start_sector="200" val="20480mb"/>
      </partition>
      <partition name="4" part_type="99">
        <size start_sector="2000" val="20480mb"/>
      </partition>
      <slice name="0" is_root="true">
        <size val="20480mb"/>
      </slice>
      <slice name="4">
        <size val="20480mb"/>
      </slice>
    </disk>
  </target_device>
</target>
<software name="ips">
  <source>
    <publisher name="solaris">
      <origin name="http://pkg.oracle.com/solaris/release"/>
    </publisher>
  </source>
  <!--
    By default the latest build available, in the
    specified IPS repository, is installed.
    if another build is required, the build number has
    to be appended to the 'entire' package in following
    form:
  <name="entire@0.5.11-0.build#"/>
  -->
  <software_data type="IPS">
    <name>pkg:/entire</name>
    <name>pkg:/babel_install</name>
  </software_data>
  <!--
      babel_install and slim_install are group packages used to
      define the default installation.  They are removed here so
      that they do not inhibit removal of other packages on the installed
      system
  -->
  <software_data action="uninstall" type="IPS">
    <name>pkg:/babel_install</name>
    <name>pkg:/slim_install</name>
  </software_data>
</software>
<add_drivers>
  <!--
    Driver Updates: This section is for adding driver packages to the
        boot environment before the installation takes place.  The
        installer can then access all devices on the system.  The
        packages installed in the boot environment will also be installed
        on the target.

        A <search_all> entry performs a search for devices which are
        missing their drivers.  A repository publisher and location
        may be specified, and that repository and its database will
```

```
            be used.  If no publisher and location is specified, the
            configured repositories will be used.
            (See pkg publisher command.)  If <addall> is specified as
            "true", then drivers the database says are third-party drivers
            will be added like all others; otherwise third-party drivers
            will not be added.

                <search_all addall="true">
                    <source>
                        <publisher name="solaris">
                            <origin name="http://pkg.oracle.com/solaris/release"/>
                        </publisher>
                    </source>
                </search_all>

            <software> entries are user-provided specifications of packages
            needed in order to perform the install.  types are P5I, SVR4, DU.
            A <software_data> action of "noinstall" inhibits adding to target.

            P5I: A pkg(5) P5I file, full path is in the source/publisher/origin.
            Path may be to a local file or an http or ftp specification.
                <software>
                    <source>
                        <publisher>
                            <origin
                name=
    "http://pkg.oracle.com/solaris/release/p5i/0/driver/firewire.p5i"/>
                        </publisher>
                    </source>
            <software_data type="P5I"/>
                </software>

            SVR4: An SVR4 package spec. The source/publisher/origin corresponds
            to the directory containing the packages.  The
        software/software_data/name refers tp the package's top level
        directory or the package's datastream file.

                <software>
                    <source>
                        <publisher>
                            <origin name="/export/package_dir"/>
                        </publisher>
                    </source>
                    <software_data type="SVR4">
                        <name>my_disk_driver.d</name>
                    </software_data>
                </software>

            DU: An ITU (Install Time Update) or Driver Update image.
            The source/publisher/origin refers to the path just above the
        image's DU directory (if expanded) or the name of the .iso image.
        All packages in the image will be added.

                <software>
                    <source>
                        <publisher>
                            <origin name="/export/duimages/mydriver.iso"/>
                        </publisher>
                    </source>
```

```
                    <software_data type="DU"/>
                </software>
      -->
      <search_all/>
    </add_drivers>
  </ai_instance>
</auto_install>
```

◆ ◆ ◆ CHAPTER 5

Configuring the Client System

This chapter describes how to specify information needed to configure the client system after installation. You can specify configuration of anything that is configurable via `smf`(5) properties.

Creating a Custom SC Manifest

The system configuration manifest (SC manifest) specifies client system configuration as a set of configuration parameters in the form of an SMF (Service Management Facility) profile. The SC manifest sets SMF properties for appropriate SMF services.

An SMF profile carrying the system configuration is applied during the first boot of the system after installation. SMF services responsible for particular configuration areas process SMF properties and configure the system accordingly.

- You can embed an SC manifest in your AI manifest. See "Default AI Manifest" on page 34 for an example of an SC manifest embedded in an AI manifest.

```
<auto_install>
  <ai_instance name="default">
      AI_manifest_content
    <sc_embedded_manifest name="AI">
      <!-- <?xml version='1.0'?>
        SC_manifest_content
      -->
    </sc_embedded_manifest>
  </ai_instance>
</auto_install>
```

- You can reference a separate SC manifest file within the AI manifest file.

```
<auto_install>
  <ai_instance name="default">
      AI_manifest_content
    <sc_manifest_file name="AI" URI="./sc_manifest1.xml"/>
  </ai_instance>
</auto_install>
```

Specifying Configuration in an SC Manifest

You can specify configuration of anything that is configurable via `smf`(5) properties. For example, the SC manifest can configure a root account, an initial user, keyboard layout, terminal type, an IPv4 network interface (static or DHCP) and default route, an IPv6 network interface (static or `addrconf`) and default route, and DNS (nameserver list, search list, domain).

Root and User Accounts

The `svc:/system/install/config` SMF service configures user and root accounts. This service recognizes two property groups:

- The `root_account` property group includes SMF properties that configure the root account.
- The `user_account` property group includes SMF properties that configure user accounts.

Tip – One method of generating encrypted passwords for the Oracle Solaris OS is to create a user of the intended name and password, copy the password from the `/etc/shadow` file between the first and second colons of the user's record, and add that information into the password values in the manifest.

Configuring the Root Account

The `root_account` property group can contain the following properties.

TABLE 5–1 `root_account` Property Group Properties

Property	Type	Required	Description
`password`	`astring`	yes	Encrypted root password.
`type`	`astring`	no	Account type: `normal` or `role`. The default is `normal`.
`expire`	`string`	no	Expiration date for login. If set to 0 (zero), the user will be forced to change the root password at the next login.

EXAMPLE 5–1 Configuring the Root Account Only With Password Expired

```
<service name="system/install/config" version="1" type="service">
    <instance name="default" enabled="true">
        <property_group name="root_account" type="application">
            <propval name="password" type="astring" value="encrypted_password"/>
            <propval name="type" type="astring" value="normal"/>
            <propval name="expire" type="astring" value="0"/>
        </property_group>

        <property_group name="other_sc_params" type="application">
```

EXAMPLE 5–1 Configuring the Root Account Only With Password Expired *(Continued)*

```
            <propval name="timezone" type="astring" value="GMT"/>
            <propval name="hostname" type="astring" value="solaris"/>
        </property_group>
    </instance>
</service>
```

Configuring a User Account

The `user_account` property group can contain the following properties.

TABLE 5–2 `user_account` Property Group Properties

Property	Type	Required	Description
`login`	`astring`	`yes`	User's login.
`password`	`astring`	`yes`	Encrypted user password.
`description`	`astring`	`no`	Usually the user's full name.
`shell`	`astring`	`no`	Full path name of the program used as the user's shell on login.
`uid`	`count`	`no`	UID of the new user. The default UID is 101.
`gid`	`count`	`no`	User's primary group membership. The default GID is 10.
`type`	`astring`	`no`	Account type: `normal` or `role`. The default is `normal`.
`profiles`	`astring`	`no`	One or more comma-separated execution profiles defined in `prof_attr`(4).
`roles`	`astring`	`no`	One or more comma-separated roles defined in `user_attr`(4).
`sudoers`	`astring`	`no`	Entry put along with login into the `sudoers`(4) file.
`expire`	`astring`	`no`	Expiration date for login. If set to 0 (zero), the user will be forced to change the password at the next login.
`home_zfs_dataset`	`astring`	`no`	User's home directory ZFS dataset. The default is *root_pool*`/export/home/`*login*
`home_mountpoint`	`astring`	`no`	User's home directory mount point. The default is `/export/home/`*login*

Terminal Type and Keyboard Layout

The `svc:/system/console-login` SMF service configures terminal type. See the `ttymon`(1M) man page for definition of related SMF properties.

EXAMPLE 5–2 Configuring `vt100` Terminal Type

```
<service name="system/console-login" version="1" type="service">
    <property_group name="ttymon" type="application">
        <propval name="terminal_type" type="astring" value="vt100"/>
    </property_group>
</service>
```

The `svc:/system/keymap` SMF service configures keyboard layout. See the `kbd`(1) man page for definition of related SMF properties.

EXAMPLE 5–3 Configuring Czech Keyboard Layout

```
<service name='system/keymap' version='1' type='service'>
    <instance name='default' enabled='true'>
        <property_group name='keymap' type='system'>
            <propval name='layout' type='astring' value='Czech'/>
        </property_group>
    </instance>
</service>
```

Static IP and DNS

Networking can be configured automatically or manually on an Oracle Solaris 11 Express system. Configurations that you can do manually can also be specified in an SC manifest.

Automated network configuration (NWAM) is activated by enabling `svc:/network/physical:nwam` and disabling `svc:/network/physical:default`. The NWAM framework handles configuration of the network.

Manual network configuration is activated when `svc:/network/physical:nwam` is disabled and `svc:/network/physical:default` is enabled. In manual mode, the following networking parameters can be configured:

- One IPv4 network interface, either with static IPv4 or DHCP configured
- IPv4 default route
- One IPv6 network interface, either with static IPv6 or autoconfigured
- IPv6 default route
- DNS as a naming service

The `svc:/network/install` and `svc:/network/dns/install` SMF services contain properties that can be used by the services to configure an initial physical network interface or an initial

DNS client configuration. These services are initially disabled with property values that do not result in any system configuration. These services can be enabled and appropriate properties configured in the SC manifest.

The `svc:/network/install` service supports configuring one IPv4 interface and one IPv6 interface and, optionally, a default route reachable by these interfaces. The service defines two property groups: one property group for an IPv4 interface and one for an IPv6 interface. The service uses its properties and `ipadm`(1M) to configure the network interfaces. Similarly, the service uses its properties and `route`(1M) to define a default route.

The `install_ipv4_interface` property group contains the following properties.

TABLE 5–3 `install_ipv4_interface` Property Group Properties

Property	Type	Required	Description
`name`	`astring`	`yes`	Name of network interface.
`address_type`	`astring`	`yes`	Value used to construct the `-T` option for the `ipadm`(1M) `create-addr` subcommand. Valid values are `static` or `dhcp`.
`static_address`	`net_address_v4`	`no`	Only required with an `address_type` of `static`. Used to construct the local address for the `ipadm`(1M) `create-addr` subcommand.
`dhcp_wait`	`astring`	`no`	Only applies with an `address_type` of `dhcp`. If defined, this property is used to construct the `-w` *seconds* (or `forever`) portion of the `ipadm`(1M) `create-addr` subcommand.
`default_route`	`net_address_v4`	`no`	Used to define a default route using `route`(1M). **`# /usr/sbin/route \`** **`-p add default`** *default-route* **`\`** **`-ifp`** *ifname* The value of *ifname* is the interface name portion of the `name` property.

The `install_ipv6_interface` property group contains the following properties.

TABLE 5–4 `install_ipv6_interface` Property Group Properties

Property	Type	Required	Description
`name`	`astring`	`yes`	Name of network interface.
`address_type`	`astring`	`yes`	Value used to construct the `-T` option for the `ipadm`(1M) `create-addr` subcommand. Valid values are `static` or `addrconf`.

TABLE 5–4 `install_ipv6_interface` Property Group Properties *(Continued)*

Property	Type	Required	Description
`static_address`	`net_address_v6`	`no`	Only required with an `address_type` of `static`. Used to construct the local address for the `ipadm`(1M) `create-addr` subcommand.
`interface_id`	`net_address_v6`	`no`	Only applies with an `address_type` of `addrconf`. Used to construct the `-i` *interface_id* portion of the `ipadm`(1M) `create-addr` subcommand.
`stateless`	`astring`	`no`	Only applies with an `address_type` of `addrconf`. Used to construct the `-p stateless=yes|no` portion of the `ipadm`(1M) `create-addr` subcommand.
`stateful`	`astring`	`no`	Only applies with an `address_type` of `addrconf`. Used to construct the `-p stateful=yes|no` portion of the `ipadm`(1M) `create-addr` subcommand.
`default_route`	`net_address_v6`	`no`	Used to define a default route using `route`(1M). **`# /usr/sbin/route \`** **`-p add default`** *default-route* **`\`** **`-ifp`** *ifname* The value of *ifname* is the interface name portion of the `name` property.

The `svc:/network/dns/install` service supports the configuration of a DNS client. The service defines one property group: `install_props`. The service uses its properties to construct a DNS `resolv.conf`(4) file.

The `install_props` property group contains the following properties.

TABLE 5–5 `install_props` Property Group Properties

Property	Type	Required	Description
`domain`	`astring`	`no`	Local domain name. Used to construct the domain directive in `resolv.conf`(4).
`nameserver`	`net_address`	`yes`	Used to construct the `nameserver` directives in `resolv.conf`(4). The `net_address_list` should contain IPv4 and IPv6 addresses.
`search`	`astring`	`no`	A value for the search order host name lookup. This value is used to construct the search directive in `resolv.conf`(4). The `astring_list` should contain domain values.

Example SC Manifests

The examples in this section are complete SC manifests that can be embedded in an AI manifest or included by reference.

Specifying Terminal Type and Keyboard Layout

The example SC manifest below configures the following parameters:

- User and root accounts
- Time zone
- Host name
- Terminal type
- Keyboard layout
- NWAM for network configuration

```
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type="profile" name="system configuration">
    <service name="system/install/config" version="1" type="service">
        <instance name="default" enabled="true">
            <property_group name="user_account" type="application">
                <propval name="login" type="astring" value="jack"/>
                <propval name="password" type="astring" value="encrypted_password"/>
                <propval name="description" type="astring" value="default user"/>
                <propval name="shell" type="astring" value="/usr/bin/bash"/>
                <propval name="uid" type='count' value='101'/>
                <propval name="gid" type='count' value='10'/>
                <propval name="type" type="astring" value="normal"/>
                <propval name="roles" type="astring" value="root"/>
            </property_group>

            <property_group name="root_account" type="application">
                <propval name="password" type="astring" value="encrypted_password"/>
                <propval name="type" type="astring" value="role"/>
            </property_group>

            <property_group name="other_sc_params" type="application">
                <propval name="timezone" type="astring" value="GMT"/>
                <propval name="hostname" type="astring" value="solaris"/>
            </property_group>
        </instance>
    </service>

    <service name="system/console-login" version="1" type="service">
        <property_group name="ttymon" type="application">
            <propval name="terminal_type" type="astring" value="sun"/>
        </property_group>
    </service>

    <service name='system/keymap' version='1' type='service'>
        <instance name='default' enabled='true'>
```

```
                        <property_group name='keymap' type='system'>
                            <propval name='layout' type='astring' value='US-English'/>
                        </property_group>
                    </instance>
                </service>

                <service name="network/physical" version="1" type="service">
                    <instance name="nwam" enabled="true"/>
                    <instance name="default" enabled="false"/>
                </service>
            </service_bundle>
```

Specifying Static Network Configuration

The example SC manifest below configures the following parameters:

- `bge0` with IPv4 static address 10.0.0.10, netmask 255.0.0.0
- 10.0.0.1 IPv4 default route
- `bge1` with IPv6 `addrconf` address type
- DNS 8.8.8.8 nameserver
- `example1.com` as local DNS domain name
- `example2.com`, `example3.com` as DNS search list for host name lookup

```
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type="profile" name="system configuration">
    <service name="system/install/config" version="1" type="service">
        <instance name="default" enabled="true">
            <property_group name="user_account" type="application">
                <propval name="login" type="astring" value="jack"/>
                <propval name="password" type="astring" value="encrypted_password"/>
                <propval name="description" type="astring" value="default user"/>
                <propval name="shell" type="astring" value="/usr/bin/bash"/>
                <propval name="uid" type='count' value='101'/>
                <propval name="gid" type='count' value='10'/>
                <propval name="type" type="astring" value="normal"/>
                <propval name="roles" type="astring" value="root"/>
            </property_group>

            <property_group name="root_account" type="application">
                <propval name="password" type="astring" value="encrypted_password"/>
                <propval name="type" type="astring" value="role"/>
            </property_group>

            <property_group name="other_sc_params" type="application">
                <propval name="timezone" type="astring" value="GMT"/>
                <propval name="hostname" type="astring" value="solaris"/>
            </property_group>
        </instance>
    </service>

    <service name="system/console-login" version="1" type="service">
        <property_group name="ttymon" type="application">
            <propval name="terminal_type" type="astring" value="sun"/>
```

```
        </property_group>
    </service>

    <service name='system/keymap' version='1' type='service'>
        <instance name='default' enabled='true'>
            <property_group name='keymap' type='system'>
                <propval name='layout' type='astring' value='US-English'/>
            </property_group>
        </instance>
    </service>

    <service name="network/physical" version="1" type="service">
        <instance name="nwam" enabled="false"/>
        <instance name="default" enabled="true"/>
    </service>

    <service name='network/install' version='1' type='service'>
        <instance name='default' enabled='true'>
            <property_group name='install_ipv4_interface' type='application'>
                <propval name='name' type='astring' value='bge0/v4'/>
                <propval name='address_type' type='astring' value='static'/>
                <propval name='static_address' type='net_address_v4' value='10.0.0.10/8'/>
                <propval name='default_route' type='net_address_v4' value='10.0.0.1'/>
            </property_group>

            <property_group name='install_ipv6_interface' type='application'>
                <propval name='name' type='astring' value='bge1/v6'/>
                <propval name='address_type' type='astring' value='addrconf'/>
                <propval name='stateless' type='astring' value='yes'/>
                <propval name='stateful' type='astring' value='yes'/>
            </property_group>
        </instance>
    </service>

    <service name='network/dns/install' version='1' type='service'>
        <instance name='default' enabled='true'>
            <property_group name='install_props' type='application'>
                <property name='nameserver' type='net_address'>
                    <net_address_list>
                        <value_node value='8.8.8.8'/>
                    </net_address_list>
                </property>
                <propval name='domain' type='astring' value='example1.com'/>
                <property name='search' type='astring'>
                    <astring_list>
                        <value_node value='example2.com'/>
                        <value_node value='example3.com'/>
                    </astring_list>
                </property>
            </property_group>
        </instance>
    </service>
</service_bundle>
```

◆ ◆ ◆ CHAPTER 6

Setting Up DHCP for AI

AI uses DHCP to provide the IP address, subnet mask, router, DNS server, and location of a boot image to the client machine to be installed.

- If you want to set up an Oracle Solaris DHCP server or configure it for use with AI, see "Oracle Solaris DHCP for AI" on page 77.
- If you want to set up an ISC DHCP server or configure it for use with AI, see "ISC DHCP for AI" on page 80.

Oracle Solaris DHCP for AI

You can set up your AI install server to also be an Oracle Solaris DHCP server, or you can set up a separate DHCP server.

If you already have an Oracle Solaris DHCP server set up, you might need to configure it to work with AI.

Set Up a DHCP Server on the AI Install Server

If you do not have a DHCP server set up, you can use the `installadm create-service` command with the `-i` and `-c` options to set up a pool of IP addresses to be used with this install service. See "Create an Install Service Including Oracle Solaris DHCP Setup" on page 20 for an example. See also the `installadm`(1M) man page. The `-i` option sets up a new DHCP server starting from the specified IP address. The `-c` option set up the specified number of IP addresses in the DHCP table.

Set Up a Separate DHCP Server

If you want to set up a separate Oracle Solaris DHCP server, see Part II, "DHCP," in *System Administration Guide: IP Services*. Then continue with the configuration for AI described in "Configure an Existing DHCP Server for AI" on page 78.

Configure an Existing DHCP Server for AI

If you do not use the `-i` and `-c` options, the `installadm create-service` command provides some instructions for you to configure DHCP. This section provides additional information about configuring DHCP for AI.

Configure the Netmasks Table

Ensure that the DHCP server's `netmasks` table has correct entries for any networks for which it will be providing DHCP service.

```
# cat >>/etc/netmasks
192.168.0.0 255.255.255.0
# getent netmasks 192.168.0.0
192.168.0.0 255.255.255.0
```

Initialize the DHCP Server Service

Run `dhcpconfig`(1M) to initialize the DHCP SMF service. The following command creates a file-based repository and a host macro with a default lease of one day. This command also enables the `svc:/network/dhcp-server` SMF service.

```
# /usr/sbin/dhcpconfig -D -r SUNWfiles -p /var/dhcp
# dhtadm -P
Name            Type    Value
================================================
example-host    Macro :Include=Locale:DNSdmain="example.com":DNSserv=192.168.0.1:
Locale          Macro   :UTCoffst=-25200:
```

Add Network Information Macros and Tables

To add the macros for each site network, use `dhcpconfig` to add the site macro and populate network tables. The `-t` option gives the router for this network. You can use this to also add new network information to DHCP servers hosting existing networks.

```
# dhcpconfig -N 192.168.0.1 -m 255.255.255.0 -t 192.168.0.1
```

Create and Add a DHCP Macro for AI

To configure your Oracle Solaris DHCP server for use with AI, create a DHCP macro and use `dhtadm` to add the macro to the DHCP configuration table, `dhcptab`(4). See the sample `installadm create-service` output in "Create an Install Service Without DHCP Setup" on page 20.

The first macro created for an architecture is the default for all clients of that architecture.

Assign Client IP Addresses

Assign client IP addresses if needed.

Use the `pntadm`(1M) command to add networks to the DHCP network table, `dhcp_network`(4).

Verify network tables:

```
# pntadm -L
```

Give the Client a DHCP Lease

Give the client a DHCP lease and set its DHCP macro to use the correct AI install image. Run the commands in this section on the DHCP server.

Provision the DHCP Lease

The `-i` option of the `pntadm` command is the ID of the client to be installed. The `-m` option specifies the name of this macro. In this example, the name of the macro is the same as the MAC address of the client. This ensures you have only one lease per host.

```
# pntadm -A 192.168.0.2 -i 01C0FFEEC0FFEE -m 01C0FFEEC0FFEE \
-f "PERMANENT+MANUAL" 192.168.0.0
```

Check That the Lease Is There

The Flags value `03` represents `PERMANENT+MANUAL`: The lease is permanent and the DHCP server cannot allocate a different address to the client.

```
# pntadm -P 192.168.0.0 | grep 01C0FFEEC0FFEE
Client ID       Flags   Client IP     Server IP      Lease Expiration  Macro           Comment
01C0FFEEC0FFEE  03      10.41.30.42   172.30.95.10   Forever           01C0FFEEC0FFEE
```

Provision the Macro

```
# dhtadm -g -A -m 01COFFEECOFFEE \
-d ":Include=‘uname -n‘:BootSrvA=192.168.0.1:BootFile=install_test_ai_x86:"
```

Check That the New Macro Is in the Table

```
# dhtadm -P |grep 01C0FFEEC0FFEE
01C0FFEEC0FFEE Macro :Include=example-host:BootSrvA=192.168.0.1:BootFile="install_test_ai_x86":
```

ISC DHCP for AI

This section describes how to configure ISC DHCP to serve the information you need for AI installations.

Basic Network Configuration

Define a basic network in your `dhcpd.conf` file. In the following example, the 192.168.0.0/24 network is defined to serve out IP addresses between 2-100 with a router of 192.168.0.1:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.2 192.168.0.100;
  option routers 192.168.0.1;
}
```

DNS Configuration

Similarly, add DNS information to your `dhcpd.conf` file. If you want each subnet served to get different DNS information, either put the directives in the subnet block or put the directives at the beginning of the file to affect all subnets served.

```
option domain-name "example.com";
option domain-name-servers 192.168.0.1;
```

Boot Server and Boot File Configuration

AI requires boot server and boot file configuration in DHCP. When you boot a client machine for automated installation, that client needs to know where to get a boot file (from the AI server) and the name of the boot file. These pieces of information are provided by the `installadm create-service` command when you set up an AI service. The following example shows partial output from an `installadm create-service` command:

```
Boot server IP (BootSrvA) : 192.168.0.1
Boot file      (BootFile) : install_test_ai_x86
GRUB Menu      (GrubMenu) : menu.lst.install_test_ai_x86
```

In ISC DHCP terms, the boot server is the `next-server` directive and the boot file is the `filename` directive. Add these directives to your subnet group, as shown in this example:

```
subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.2 192.168.0.100;
  option routers 192.168.0.1;
  filename "install_test_ai_x86";
  next-server 192.168.0.1;
}
```

For a SPARC AI service, use the same `filename` directive for the `BootFile` object. SPARC installations do not need a `next-server` (`BootSvrA`) directive.

If you are using AI to install both SPARC and x86 machines, use the ISC DHCP `class` directive to provide boot file information to the SPARC clients and provide boot file and boot server information to the x86 clients. Providing boot-specific information in separate `class` directives enables you to have a default service for each architecture on the network.

PXE Boot Class

The following example is a `class` definition for an x86 hardware boot:

```
class "PXEBoot" {
  option dhcp-class-identifier "PXEClient";
  filename "install_test_ai_x86";
  next-server 192.168.0.1;
}
```

SPARC Boot Class

The following example is a `class` definition for SPARC hardware boot. Note that you do not need a `next-server` directive for SPARC:

```
class "SPARC" {
  match if ( substring (option vendor-class-identifier, 0, 5) = "SUNW." ) and not
           ( option vendor-class-identifier = "SUNW.i86pc" );
  filename "http://192.168.0.1:5555/cgi-bin/wanboot-cgi";
}
```

Sample Configuration File

With these class definitions, SPARC clients request a lease and get SPARC specific information, and x86 clients request and get information specific to x86. The following examples shows the entire `dhcpd.conf` file:

```
# option definitions common to all supported networks...
option domain-name "example.com";
option domain-name-servers 192.168.0.1;

default-lease-time 600;
max-lease-time 86400;

# If this DHCP server is the official DHCP server for the local
```

```
# network, the authoritative directive should be uncommented.
authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
log-facility local7;

# This is an easy way to discriminate on SPARC clients
class "SPARC" {
  match if ( substring (option vendor-class-identifier, 0, 5) = "SUNW." ) and not
           ( option vendor-class-identifier = "SUNW.i86pc" );
  filename "http://192.168.0.1:5555/cgi-bin/wanboot-cgi";
}

# This is a class to discriminate on PXE booting x86 clients
class "PXEBoot" {
  option dhcp-class-identifier "PXEClient";
  filename "install_test_ai_x86";
  next-server 192.168.0.1;
}

# This is a very basic subnet declaration
subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.2 192.168.0.100;
  option routers 192.168.0.1;
}
```

◆ ◆ ◆ CHAPTER 7

Installing Client Systems

Installation begins when you boot the client. When the client boots, DHCP directs the client to the AI install server.

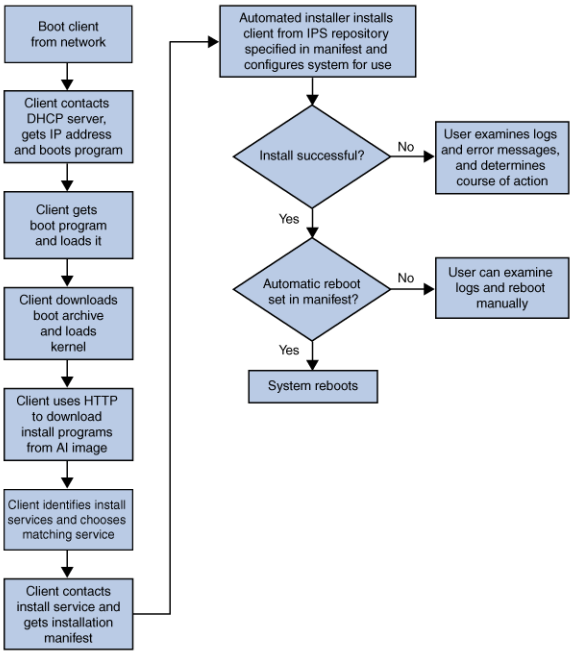
This chapter gives the system requirements for automated installation clients and describes how to associate each client with the correct install service.

How a Client Is Installed

When you set up your install server, you created at least one install service for each client architecture and each version of the Oracle Solaris OS that you plan to install. When you created each install service, you created customized installation instructions and post-installation configuration instructions for different clients as needed. To start the automated installation, you just need to boot the client.

The following flowchart illustrates how a client system is installed. On booting, the client uses specifications in the boot arguments to identify the correct install service to use. The installation is performed on the client system using a boot image, installation specifications, and post-installation configuration specifications provided by the install service.

FIGURE 7–1 Client Installation Steps



Client System Requirements

The client systems for automated installation must meet the following requirements. Any system that meets these requirements can be used as an automated install client, including laptops, desktops, virtual machines, and enterprise servers.

SPARC and x86 Client System Requirements

SPARC and x86 clients of AI installation over the network must meet the following requirements:

Memory **x86:** 512 MB minimum

SPARC: 1 GB minimum

The minimum memory requirement if the AI manifest specifies device driver installation is 1.5 GB. See "Identify and Install Missing Drivers on an Install Target" on page 57 for information about automated device driver installation.

Disk space **SPARC and x86:** 13 GB minimum

Network access. Client systems must be able to access the following resources during the installation:

- A DHCP server that provides network configuration information
- The AI install server
- An IPS repository that contains the packages to be installed on the client system

Additional SPARC Client System Requirements

WAN boot SPARC clients of AI installation over the network must support WAN boot. See "Does My SPARC Client Support WAN Boot?" on page 85.

Firmware The firmware on SPARC clients must be updated to include the current version of the Open Boot PROM (OBP) that contains the latest WAN boot support.

Does My SPARC Client Support WAN Boot?

To boot over the network, AI requires WAN boot support for SPARC clients. You can check whether your client Open Boot PROM (OBP) supports WAN boot by checking whether `network-boot-arguments` is a valid variable that can be set in the `eeprom`.

If the variable `network-boot-arguments` is displayed, or if the command returns the output `network-boot-arguments: data not available`, the OBP supports WAN boot and the client can be installed over the network.

```
# eeprom | grep network-boot-arguments
network-boot-arguments: data not available
```

If the command results in no output, then WAN Boot is not supported and the client cannot be installed over the network. See Chapter 8, "Automated Installations That Boot From Media."

```
# eeprom | grep network-boot-arguments
```

Setting Up an Install Client

On the install server, use the `installadm create-client` command to associate a particular client with a particular install service.

The `installadm create-client` command requires the following information:

- MAC address for the client
- Name of the install service for the client to use for installation

For x86 clients, you can optionally specify boot properties.

Setting Up an x86 Client

The following example associates the x86 client with MAC address `0:e0:81:5d:bf:e0` with the `s11-x86` install service. The `-b` option sets the console value, `console=ttya`, in the client-specific `menu.lst file` in `/tftpboot`.

```
# installadm create-client -b "console=ttya" \
-e 0:e0:81:5d:bf:e0 -n s11-x86
Setting up X86 client...
Service discovery fallback mechanism set up

Detected that DHCP is not set up on this server.
If not already configured, please create a DHCP macro
named 0100E0815DBFE0 with:
   Boot server IP (BootSrvA) : 10.6.68.29
   Boot file      (BootFile) : 0100E0815DBFE0
If you are running the Solaris DHCP server, use the following
command to add the DHCP macro, 0100E0815DBFE0:
   /usr/sbin/dhtadm -g -A -m 0100E0815DBFE0 -d \
:BootSrvA=10.6.68.29:BootFile=0100E0815DBFE0:GrubMenu=menu.lst.0100E0815DBFE0:

Note: Be sure to assign client IP address(es) if needed
(e.g., if running the Solaris DHCP server, run pntadm(1M)).
```

This command displays the name and values of a macro, `0100E0815DBFE0`, which needs to be added to the DHCP server.

Use the `/usr/sadm/admin/bin/dhcpmgr` utility to open the DHCP Manager to view the DHCP configuration results in the DHCP table.

You can also view the results of the `installadm create-client` command in the `/tftpboot` directory:

```
# cd /tftpboot
# ls -l
lrwxrwxrwx  13:23 0100E0815DBFE0 -> pxegrub.I86PC.Solaris-1
drwxr-xr-x  13:26 I86PC.Solaris-1
-rw-r--r--  13:23 menu.lst.0100E0815DBFE0
-rwxr-xr-x  13:23 pxegrub.I86PC.Solaris-1
-rw-r--r--  13:23 rm.0100E0815DBFE0

# cat menu.lst.0100E0815DBFE0
default=0
timeout=30
min_mem64=1000
title Oracle Solaris 11 Express boot image
    kernel$ /I86PC.Solaris-1/platform/i86pc/kernel/$ISADIR/unix
-B install_media=http://135.134.0.10:5555/export/aiserver/s11-ai-x86,
install_service=s11-ai-x86,install_svc_address=135.134.0.10:46501
    module$ /I86PC.Solaris-1/platform/i86pc/$ISADIR/boot_archive
title Oracle Solaris 11 Express Automated Install
    kernel$ /I86PC.Solaris-1/platform/i86pc/kernel/$ISADIR/unix
-B install=true,install_media=http://135.134.0.10:5555/export/aiserver/s11-ai-x86,
```

```
install_service=osol-1003-ai-x86,install_svc_address=135.134.0.10:46501
    module$ /I86PC.Solaris-1/platform/i86pc/$ISADIR/boot_archive
```

Setting Up a SPARC Client

The following example associates the SPARC client with MAC address `00:14:4f:a7:65:70` with the `s11-sparc` install service.

```
# installadm create-client -e 00:14:4f:a7:65:70 -n s11-sparc
Setting up SPARC client...
Creating SPARC configuration file

Detected that DHCP is not set up on this server.
If not already configured, please create a DHCP macro
named 0100E0815DBFE0 with:
   Boot server IP (BootSrvA) : 10.6.68.29
   Boot file      (BootFile) : http://10.6.68.29:5555/cgi-bin/wanboot-cgi
If you are running the Oracle Solaris DHCP server, use the following
command to add the DHCP macro, 0100E0815DBFE0:
   /usr/sbin/dhtadm -g -A -m 0100E0815DBFE0 \
-d :BootSrvA=10.6.68.29:BootFile="http://10.6.68.29:5555/cgi-bin/wanboot-cgi":
Note: Be sure to assign client IP address(es) if needed
(e.g., if running the Oracle Solaris DHCP server, run pntadm(1M)).
```

This command displays the name and values of a macro, `0100E0815DBFE0`, which needs to be added to the DHCP server. If you are using the Oracle Solaris DHCP server, you can create the macro on your DHCP server either by using the DHCP Manager or by running the `dhtadm` commands provided in the output on your DHCP server.

Deleting a Client From a Service

Use the `installadm delete-client` command to delete a client from an install service.

```
# installadm delete-client -e macaddr
```

You do not need to specify the service name since a client can be associated with only one install service.

Installing Clients

Boot the client to start the installation. This section shows you exactly how to boot a SPARC or x86 client. This section also describes how you can monitor installation progress remotely.

Using Secure Shell to Remotely Monitor Installations

You can enable network access to an automated install client by using `ssh`. You can use this access to remotely observe an installation in progress.

Enable remote access by setting the option `livessh` to `enable` in the installation configuration file. When this access is enabled, you can log in to the AI client by using the username `jack` and password `jack`.

Monitoring x86 Client Installations

For x86 systems, the `menu.lst` configuration file is created in the `/tftpboot/` directory with one of the following filename formats.

- If you used the `installadm create-client` command, the filename is `menu.lst.01`*MAC_address*, where *MAC_address* is the MAC address that was specified in the `installadm create-client` command.
- If you did not use the `installadm create-client` command, the filename is `menu.lst.`*service_name*, where *service_name* is the install service name that was specified in the `installadm create-service` command.

In this file, options are provided as kernel parameters. In the following example, the `livessh` and `install_debug` options are set to `enable`.

```
kernel$ ... -B install_media=...,livessh=enable,install_debug=enable
```

Monitoring SPARC Client Installations

For SPARC systems, the `install.conf` file is populated in the `ai_image_dest` directory when an install service is created by using an `installadm create-service` command.

In the `install.conf` file, the options are defined as name-value pairs. In the following example, the `livessh` option is set to `enable`.

```
$ cat ai_image_dest/install.conf
...
livessh=enable
...
```

Installing a SPARC Client

Use the following command to boot SPARC clients from the OBP prompt:

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

SPARC Client Network Boot Sequence

The following events occur during AI boot of a SPARC client:

1. The client boots and gets an IP address and the boot file, `wanboot-cgi`, from the DHCP server.
2. The `wanboot-cgi` boot file reads `wanboot.conf` and sends the location of the WAN boot binary to the client.
3. The WAN boot binary is downloaded using HTTP, and the client boots the WAN boot program.
4. WAN boot gets the `boot_archive` file, and the Oracle Solaris OS is booted.
5. Image archives, `solaris.zlib` and `solarismisc.zlib`, are downloaded using HTTP.
6. The AI SMF service uses multicast DNS (mDNS) to contact the HTTP server and get the AI manifest.
7. The AI install program is invoked with the AI manifest to perform the installation of the Oracle Solaris OS to the client.

Sample SPARC Network Boot Output

The following output from the boot sequence is displayed:

```
{3} ok boot net:dhcp - install

...
OpenBoot 4.23.4, 8184 MB memory available, Serial #69329298.
Ethernet address 0:14:4f:21:e1:92, Host ID: 8421e192.

Rebooting with command: boot net:dhcp - install
Boot device: /pci@7c0/pci@0/network@4:dhcp  File and args:
1000 Mbps FDX Link up
1000 Mbps FDX Link up
<time unavailable> wanboot info: WAN boot messages->console
<time unavailable> wanboot info: Starting DHCP configuration
<time unavailable> wanboot info: DHCP configuration succeeded
<time unavailable> wanboot progress: wanbootfs: Read 366 of 366 kB (100%)
<time unavailable> wanboot info: wanbootfs: Download complete
Tue Aug  5 17:12:09 wanboot progress: miniroot: Read 165251 of 165251 kB (100%)
Tue Aug  5 17:12:09 wanboot info: miniroot: Download complete
SunOS Release 5.11 Version snv_151 64-bit
...
Hostname: solaris
Remounting root read/write
Probing for device nodes ...
Preparing automated install image for use
Downloading solaris.zlib archive
--11:09:11--  http://10.6.35.226:5555//export/home \
/images/s11-ai-sparc//solaris.zlib
          => ‘/tmp/solaris.zlib’
Connecting to 10.6.35.226:5555... connected.
HTTP request sent, awaiting response... 200 OK
```

```
Length: 82,679,296 (79M) [text/plain]

100%[======================================>] 82,679,296    33.39M/s

11:09:13 (33.36 MB/s) - ‘/tmp/solaris.zlib’ saved [82679296/82679296]

Downloading solarismisc.zlib archive
--11:09:13--  http://10.6.35.226:5555//export/home/images \
/s11-ai-sparc//solarismisc.zlib
           => ‘/tmp/solarismisc.zlib’
Connecting to 10.6.35.226:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 620,032 (606K) [text/plain]

100%[======================================>] 620,032       --.--K/s

11:09:13 (36.48 MB/s) - ‘/tmp/solarismisc.zlib’ saved [620032/620032]

--11:09:13--  http://10.6.35.226:5555//export/home/images/ \
s11-ai-sparc//install.conf
           => ‘/tmp/install.conf’
Connecting to 10.6.35.226:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 39 [text/plain]

100%[======================================>] 39            --.--K/s

11:09:13 (953.58 KB/s) - ‘/tmp/install.conf’ saved [39/39]

Done mounting automated install image
Configuring devices.
Reading ZFS config: done.

Automated Installation started
The progress of the Automated Installation can be followed by
viewing the logfile at /tmp/install_log
```

Installing an x86 Client

Initiate the x86 client installation by using one of the following methods to boot from the network:

- Press the appropriate function key. For example, some systems use F12 to boot from the network
- Change the boot order in the BIOS.

When the client boots, select the network device to boot from.

x86 Client Network Boot Sequence

The following events occur during AI boot of an x86 client:

1. The client boots and gets an IP address, and the boot file, `pxegrub`, is downloaded from the location provided by the DHCP server.
2. The `pxegrub` boot file is loaded and reads a `menu.lst` file.
3. The `pxegrub` boot file gets the `boot_archive` file and the Oracle Solaris OS is booted using TFTP.
4. The net image archives, `solaris.zlib` and `solarismisc.zlib`, are downloaded using HTTP as provided by the GRUB menu.
5. The AI manifest is downloaded from an HTTP server specified in the GRUB menu or from the mDNS lookup performed by the AI SMF service.
6. The AI install program is invoked with the AI manifest to perform the installation.

Sample x86 Network Boot Output

The first screen displays the following message when the client receives the correct DHCP response:

```
Intel(R) Boot Agent PXE Base Code (PXE-2.1 build 0.86)
Copyright(C) 1997-2007, Intel Corporation

CLIENT MAC ADDR 00 14 4F 29 04 12 GUID FF2000008 FFFF FFFF FFFF 7BDA264F1400
CLIENT IP: 10.6.68.29   MASK: 255.255.255.0    DHCP IP:  10.6.68.49
GATEWAY: 10.6.68.1
```

The GRUB menu appears with two menu entries. Select the second entry to start an automated installation:

```
Oracle Solaris 11 Express boot image
Oracle Solaris 11 Express Automated Install
```

The default GRUB menu entry, “Oracle Solaris 11 Express boot image,” boots the image without starting a hands-free automated installation. Select the second entry in the GRUB menu, “Oracle Solaris 11 Express Automated Install,” to initiate an automated installation.

Once the image is selected, the following messages are displayed:

```
SunOS Release 5.11 Version snv_151 64-bit
...
Hostname: solaris
Remounting root read/write
Probing for device nodes ...
Preparing automated install image for use
Downloading solaris.zlib archive
--11:09:11--  http://10.6.35.226:5555//export/home \
/images/s11-ai-x86//solaris.zlib
```

```
          => ‘/tmp/solaris.zlib’
Connecting to 10.6.35.226:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 82,679,296 (79M) [text/plain]

100%[====================================>] 82,679,296    33.39M/s

11:09:13 (33.36 MB/s) - ‘/tmp/solaris.zlib’ saved [82679296/82679296]

Downloading solarismisc.zlib archive
--11:09:13--  http://10.6.35.226:5555//export/home/images \
/s11-ai-x86//solarismisc.zlib
          => ‘/tmp/solarismisc.zlib’
Connecting to 10.6.35.226:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 620,032 (606K) [text/plain]

100%[====================================>] 620,032       --.--K/s

11:09:13 (36.48 MB/s) - ‘/tmp/solarismisc.zlib’ saved [620032/620032]

--11:09:13--  http://10.6.35.226:5555//export/home/images/ \
s11-ai-x86//install.conf
          => ‘/tmp/install.conf’
Connecting to 10.6.35.226:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 39 [text/plain]

100%[====================================>] 39            --.--K/s

11:09:13 (953.58 KB/s) - ‘/tmp/install.conf’ saved [39/39]

Done mounting automated install image
Configuring devices.
Reading ZFS config: done.

Automated Installation started
The progress of the Automated Installation can be followed by
viewing the logfile at /tmp/install_log
```

Client Installation Messages

The following messages are common to both SPARC and x86 installations.

Automated Installation Started Message

If the client is able to successfully boot and download the install files, then the following message is displayed:

```
Automated Installation started
   The progress of the Automated Installation can be followed by viewing the
   logfile at /tmp/install_log
```

You can login as `root` with the password `solaris` to monitor the installation messages in `/tmp/install_log`. Once the installation of packages from IPS has started, you might not see updates to this log file for an extended period.

Automated Installation Succeeded Message

If you see the following message, the installation is successful:

```
Auto install succeeded. You may wish to reboot the system at this time.
```

If you have set up automatic reboot in the AI manifest, the system reboots at this time. To specify automatic reboot after successful installation, set the `auto_reboot` attribute of the `<ai_instance>` tag to `true`. The default value is `false`: The client does not automatically reboot after successful installation.

◆ ◆ ◆ CHAPTER 8

Automated Installations That Boot From Media

You can initiate an automated installation of the Oracle Solaris 11 Express OS on a SPARC or X86 system by booting an AI Image on media rather than booting over the network. This chapter discusses reasons to boot an AI client from media and how to perform the installation in that mode.

Overview of Installation Using AI Media

Installation using AI media enables you to accomplish the following tasks:

- Install the system that will be your AI install server.
- Install a SPARC system that does not have WAN boot capability.
- Troubleshoot an ailing system. Boot the system from the removable media and then inspect the installed system and run diagnostics.

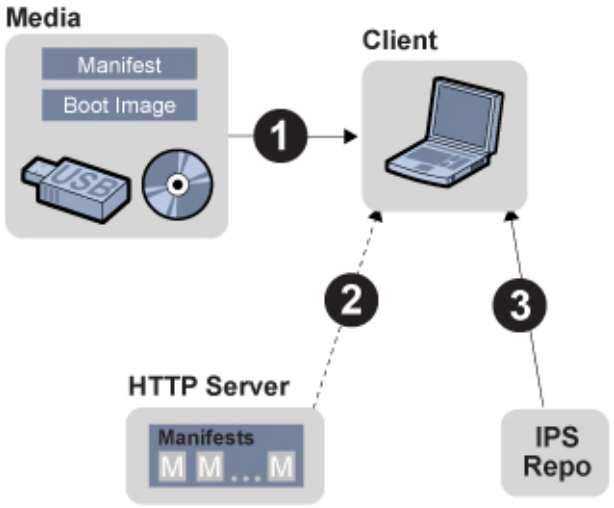
Installation using AI media has the following characteristics:

- You do not need to set up an install server or an install service.
- The system does not need to be able to boot over the network.

Installing Using AI Media

You can boot an AI image from a CD, DVD, or USB device to initiate a hands-free installation of only that system. An AI manifest provides installation and configuration instructions. The system to be installed must have network access. To complete the installation, software packages are retrieved from an IPS repository on the Internet or on the local network. Review the default AI manifest as described in "Creating a Custom AI Manifest" on page 97.

FIGURE 8–1 AI Install Using Media



System Requirements for Installing Using AI Media

Both SPARC and x86 systems must meet the following requirements.

TABLE 8–1 System Requirements for Installation Using AI Media

Requirement	Specifications
Memory	Minimum memory: 512 MB
Disk Space	Minimum disk space: 13 GB
Network Access	The system to be installed must be able to access the following resources during the installation: ■ A DHCP server that provides network configuration information ■ An IPS repository that contains the packages to be installed on the client system If you create a custom AI manifest, the system must be able to access that manifest on an HTTP server.

How To Install Using AI Media

To install by booting from an AI image on media, perform the following steps:

1. **Download the AI boot image.**

   To download the AI boot image, go to the following Internet location:
   `http://www.oracle.com/technetwork/server-storage/solaris11/downloads/index.html`

   - **SPARC systems**

     For SPARC systems, download the SPARC AI `.iso` file.

   - **x86 systems**

     For x86 systems, download the x86 AI `.iso` file or the x86 AI `.usb` file.

2. **Review the default AI manifest.**

   You can use the default manifest that is provided in the AI image, or you can create a custom manifest and provide the location of this custom manifest when the client boots. See "Creating a Custom AI Manifest" on page 97.

3. **Create bootable media.**

   - **SPARC and x86 ISO images**

     Burn the `.iso` file to a CD or DVD.

   - **x86 USB images**

     Use `usbcopy`(1) or a similar utility to create bootable media using the `.usb` file.

4. **Boot from the media.**

   Boot the system from the device that contains the boot image. See "Booting a SPARC System From AI Media" on page 98 and "Booting an x86 System From AI Media" on page 99 for instructions about how to specify the default AI manifest or a custom AI manifest and continue the hands-free installation.

Creating a Custom AI Manifest

You can install the system using the installation specifications in the AI manifest provided in the AI boot image, or you can create custom installation specifications. If you create a custom AI manifest, store the manifest on an HTTP server, and provide the location of the manifest when you boot the system to be installed.

If you download the `.iso` AI image, you can use the following commands to inspect the AI manifest in that image. In this example, `/tmp` is the directory where you downloaded the AI image, and `/home/username` is the directory where you want to copy and edit the AI manifest. The AI manifest is in `auto-install/default.xml` in the image.

```
# lofi_dev=$(/usr/sbin/lofiadm -a /tmp/sol-11-ai-sparc.iso)
# /usr/sbin/mount -o ro -F hsfs ${lofi_dev} /mnt
```

```
# cp /mnt/auto_install/default.xml /home/username/custom.xml
# /usr/sbin/umount /mnt
# /usr/sbin/lofiadm -d ${lofi_dev}
```

Review your copy of the default manifest file (`/home/username/custom.xml` in this example), and decide whether these specifications are satisfactory for this installation.

Alternatively, you can use the manifest shown in "Default AI Manifest" on page 34 as the base to create a custom manifest.

To change installation specifications such as target disk or additional packages to install, see Chapter 4, "Specifying Installation Instructions." To change configuration specifications such as user account or root password, see Chapter 5, "Configuring the Client System."

When you are finished modifying the AI manifest, copy the custom manifest to an HTTP server. Note the URL to the custom AI manifest so that you can provide that URL when you boot the system to be installed. For example, the URL might be `http://example.com/custom.xml`.

Booting a SPARC System From AI Media

You can specify the default AI manifest or a custom AI manifest when you boot the system from the AI media.

Use the Default AI Manifest

To use the default AI manifest that is in the AI boot image, enter the following command at the OBP prompt:

```
ok> boot cdrom - install
```

The automated installation proceeds, using the specifications in the default manifest.

Use a Custom AI Manifest

To use a custom AI manifest, enter the following command at the OBP prompt:

```
ok> boot cdrom - install prompt
```

The following prompt displays:

```
Enter the URL for the AI manifest [HTTP, default]:
```

Type the URL to your custom manifest. For example, type `http://example.com/custom.xml`.

The automated installation proceeds, using the specifications in the custom manifest.

Boot a SPARC Image Without Installing

You might want to boot from media but not install. For example, you might want to troubleshoot or examine the system.

To boot the AI image but not start an automated installation, use the following command:

```
ok> boot cdrom
```

The system boots and a login screen displays, but the installation does not begin.

Booting an x86 System From AI Media

On an x86 system, choose an automated installation option from the GRUB menu. The GRUB menu selection or boot command that you use specifies whether the installation will use the default manifest on the media or a custom manifest that you have stored on an HTTP server.

Your GRUB menu selections should look similar to the following example:

```
GNU GRUB version 0.97 (639K lower / 2078660K upper memory)

Oracle Solaris 11 Express Automated Install custom
Oracle Solaris 11 Express Automated Install
Oracle Solaris 11 Express Automated Install custom ttya
Oracle Solaris 11 Express Automated Install custom ttyb
Oracle Solaris 11 Express Automated Install ttya
Oracle Solaris 11 Express Automated Install ttyb
Boot from Hard Disk

Use the arrow keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.
```

Use the Default AI Manifest

To use the default AI manifest that is in the AI boot image, use the arrow keys to choose one of the following options:

```
Oracle Solaris 11 Express Automated Install
Oracle Solaris 11 Express Automated Install ttya
Oracle Solaris 11 Express Automated Install ttyb
```

The `ttya` option sends the screen output during the installation to serial console `ttya (COM1)`. The `ttyb` option sends the screen output during the installation to serial console `ttyb (COM2)`.

The automated installation proceeds, using the specifications in the default manifest.

Use a Custom AI Manifest

To use a custom AI manifest, choose one of the following options:

```
Oracle Solaris 11 Express Automated Install custom
Oracle Solaris 11 Express Automated Install custom ttya
Oracle Solaris 11 Express Automated Install custom ttyb
```

When you select one of these custom options, the following prompt displays:

```
Enter the URL for the AI manifest [HTTP, default]:
```

Type the URL to your custom manifest. For example, type `http://example.com/custom.xml`.

The automated installation proceeds, using the specifications in the custom manifest.

Boot an x86 Image Without Installing

You might want to boot from media but not install. For example, you might want to troubleshoot or examine the system.

In general, if `install=true` is specified in the kernel line for the GRUB entry that you use, the installation automatically begins. If you want to boot the x86 system without immediately starting an automated installation, check the GRUB menu entry that you plan to choose. If `install=true` is specified in the kernel line for that GRUB entry, edit the line to remove `install=true`. Then when you choose that option, the system boots and a login screen displays, but the installation does not begin.

Viewing the Installation Log Files

When the automated installation is complete, the output states whether the installation succeeded or failed.

- If the installation failed, you can review the installation log at `/tmp/install_log`.
- If the installation succeeded, you can find the log at `/tmp/install_log` before you reboot the system or at `/var/sadm/system/logs/install_log` after you reboot.

◆ ◆ ◆ APPENDIX A

Troubleshooting Automated Installations

This appendix discusses some possible failures and how to recover.

Client Installation Fails

This section suggests actions to take if a client installation fails.

Check the Installation Logs

If an installation to a client system failed, you can find the log at `/tmp/install_log`.

Check IPS Repository

The install client needs to reach the IPS repository defined in the AI manifest in order to install the Oracle Solaris OS. In a normal configuration, the DHCP server sends the DNS information to the client. This DNS information is used to resolve the IPS repository name to an IP address.

A failed installation can result in an error message. See the following sample error message:

```
<OM Nov  2 06:56:32> Creating and configuring pkg(5) image area...
    <TRANSFER_MOD Nov  2 06:56:32> pkg cmd: /usr/bin/pkg image-create
-f -F -p example.com=http://pkg.example.com/release /a
    <TRANSFER_MOD_E Nov  2 06:56:35>
    <TRANSFER_MOD_E Nov  2 06:56:35>
    <TRANSFER_MOD_E Nov  2 06:56:35> pkg image-create: The URI
'http://pkg.example.com/release' does not appear to point to a valid pkg server.
    <TRANSFER_MOD_E Nov  2 06:56:35> Please check the server's address and client's
network configuration.
    <TRANSFER_MOD_E Nov  2 06:56:35> Additional details:
    <TRANSFER_MOD_E Nov  2 06:56:35>
    <TRANSFER_MOD_E Nov  2 06:56:35> Unable to contact valid package server
```

```
    <TRANSFER_MOD_E Nov  2 06:56:35> Encountered the following error(s):
    <TRANSFER_MOD_E Nov  2 06:56:35> Unable to contact any configured publishers.
This is likely a network configuration problem.
    <TRANSFER_MOD_E Nov  2 06:56:35> Unable to initialize the pkg image area at /a
    ...
    <AI Nov  2 06:56:36> Automated Installation failed in Transfer module
    <AI Nov  2 06:56:36> Transferring the files from the source failed.
Please see previous messages for more details
```

This error indicates that the client could not resolve the name of the IPS repository. On the failed machine, try to reach the IPS repository. This output shows that the repository this client installation is trying to reach is `http://pkg.example.com/release`.

Check whether the client can ping the repository:

```
$ ping pkg.example.com
```

If the `ping` command returns the following output, the error might be a connectivity problem:

```
no answer from pkg.example.com
```

If the `ping` command returns the following output, the error might be a DNS problem:

```
ping: unknown host pkg.example.com
```

Check DNS

Check whether DNS is configured on your client by verifying that a non-empty `/etc/resolv.conf` file exists.

If `/etc/resolv.conf` does not exist or is empty, check that your DHCP server is providing DNS server information to the client:

```
# /sbin/dhcpinfo DNSserv
```

If this command returns nothing, the DHCP server is not set up to provide DNS server information to the client. Contact your DHCP administrator to correct this problem.

If an `/etc/resolv.conf` file exists and is properly configured, check for the following possible problems and contact your system administrator for resolution:

- The DNS server might not be resolving your IPS repository server name.
- No default route to reach the DNS server exists.

Check Client Boot Errors

Review the following additional information about errors that occur when the client system is booting.

SPARC Network Booting Errors and Possible Causes

This section describes errors or problems you might see when booting a SPARC client over the network and possible causes.

Timed out Waiting for BOOTP/DHCP Reply

If a DHCP server is not responding to a SPARC client's request, the following messages display:

```
...
OpenBoot 4.23.4, 8184 MB memory available, Serial #69329298.
Ethernet address 0:14:4f:21:e1:92, Host ID: 8421e192.
Rebooting with command: boot net:dhcp - install
Boot device: /pci@7c0/pci@0/network@4:dhcp  File and args:
1000 Mbps FDX Link up
Timed out waiting for BOOTP/DHCP reply
Timed out waiting for BOOTP/DHCP reply
Timed out waiting for BOOTP/DHCP reply
Timed out waiting for BOOTP/DHCP reply
```

The timeout message indicates that the client is sending a DHCP request, and no response has been made to that request. This error is probably caused by a DHCP configuration problem. Check whether your client is configured correctly in the DHCP server.

Boot Load Failed

If the AI client starts downloading the `boot_archive`, but then fails with the error, "Boot load failed," that indicates that the client DHCP information is configured incorrectly.

```
Rebooting with command: boot net:dhcp - install
   Boot device: /pci@7c0/pci@0/network@4:dhcp  File and args:
   1000 Mbps FDX Link up
   HTTP: Bad Response: 500 Internal Server Error
   Evaluating:

   Boot load failed
```

This error could happen if another DHCP server is responding to the client. Check the DHCP configuration for this client. If the configuration appears to be correct, determine whether there is another DHCP server in the subnet. If you are running the Oracle Solaris DHCP server, you can run the DHCP daemon in debug mode with the command:

```
# /usr/lib/inet/in.dhcpd -dv
```

Internal Server Error or WAN Boot Alert

After the AI client has obtained the IP address and initial parameters to start downloading the boot archive, the client might be unable to find or download the `boot_archive`.

- If the client cannot find the `boot_archive`, the following error is displayed.

```
Rebooting with command: boot net:dhcp - install
      Boot device: /pci@7c0/pci@0/network@4:dhcp  File and args:
      1000 Mbps FDX Link up
      <time unavailable> wanboot info: WAN boot messages->console
      <time unavailable> wanboot info: Starting DHCP configuration
      <time unavailable> wanboot info: DHCP configuration succeeded
      <time unavailable> wanboot progress: wanbootfs: Read 366 of 366 kB (100%)
      <time unavailable> wanboot info: wanbootfs: Download complete
      Tue Aug  5 20:46:43 wanboot alert: miniinfo: Request returned code 500
      Tue Aug  5 20:46:44 wanboot alert: Internal Server Error \
(root filesystem image missing)
```

- If the AI client finds the `boot_archive` file but cannot access the file, then the following error is displayed.

```
Rebooting with command: boot net:dhcp - install
      Boot device: /pci@7c0/pci@0/network@4:dhcp  File and args:
      1000 Mbps FDX Link up
      <time unavailable> wanboot info: WAN boot messages->console
      <time unavailable> wanboot info: Starting DHCP configuration
      <time unavailable> wanboot info: DHCP configuration succeeded
      <time unavailable> wanboot progress: wanbootfs: Read 366 of 366 kB (100%)
      <time unavailable> wanboot info: wanbootfs: Download complete
      Tue Aug  5 20:53:02 wanboot alert: miniroot: Request returned code 403
      Tue Aug  5 20:53:03 wanboot alert: Forbidden
```

For both of these problems, fix the `boot_archive` file configured for this client. Check the path name and permissions of the `boot_archive` at `$IMAGE/boot/boot_archive`.

Error Message 403: Forbidden or 404 Not Found

These messages, "ERROR 403: Forbidden" and "ERROR 404: Not Found" can be seen if the AI client successfully downloads the `boot_archive` and boots the Oracle Solaris kernel, but fails to get one of the image archives. An error message is displayed indicating which file is causing the problem. For example, in the following output, the `solaris.zlib` file does not exist or is not accessible at the specified location.

```
Rebooting with command: boot net:dhcp - install
   Boot device: /pci@7c0/pci@0/network@4:dhcp  File and args:
   1000 Mbps FDX Link up
```

```
1000 Mbps FDX Link up
<time unavailable> wanboot info: WAN boot messages->console
<time unavailable> wanboot info: Starting DHCP configuration
<time unavailable> wanboot info: DHCP configuration succeeded
<time unavailable> wanboot progress: wanbootfs: Read 366 of 366 kB (100%)
<time unavailable> wanboot info: wanbootfs: Download complete
Tue Aug  5 21:43:37 wanboot progress: miniroot: Read 165251 of 165251 kB (100%)
Tue Aug  5 21:43:38 wanboot info: miniroot: Download complete
SunOS Release 5.11 Version snv_151 64-bit
...
Hostname: solaris
Remounting root read/write
Probing for device nodes ...
Preparing automated install image for use
Downloading solaris.zlib archive
--15:40:37--  http://10.6.35.226:5555//export/home/images/ai_sparc_111/solaris.zlib
        => '/tmp/solaris.zlib'
Connecting to 10.6.35.226:5555... connected.
HTTP request sent, awaiting response... 403 Forbidden
15:40:37 ERROR 403: Forbidden.

FAILED
Requesting System Maintenance Mode
(See /lib/svc/share/README for more information.)
Console login service(s) cannot run
```

This problem can be caused by one of the following conditions.

- The image path configured in WAN boot is not correct.
- The image path does not exist or is incomplete.
- Access is denied due to permission issues.

Check your DHCP configuration or the contents of the target directory you specified when you ran `installadm create-service`. Check your WAN boot configuration.

Automated Installer Disabled

When installing the Oracle Solaris OS on your client system, you need to include the `install` argument when you boot, as follows, in order to initiate an installation.

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

If you booted without the `install` boot argument, the SPARC client boots into the automated installer boot image, but the installation does not start. The following message is displayed.

```
Auto-installer disabled. Enable the auto-installer service
by running the following command:
svcadm enable svc:/application/auto-installer:default
```

To start an automated installation, you can log in and enable the install service as shown in the message, or you can reboot your system using the command shown above with the `install` argument.

x86 Network Booting Errors and Possible Causes

This section describes errors or problems you might see when booting an x86 client over the network and possible causes.

No DHCP or Proxy DHCP Offers Were Received

If a DHCP server is not responding to an x86 client's request, you see the following messages:

```
Intel(R) Boot Agent PXE Base Code (PXE-2.1 build 0.86)
   Copyright(C) 1997-2007, Intel Corporation

   CLIENT MAC ADDR 00 14 4F 29 04 12 GUID FF2000008 FFFF FFFF FFFF 7BDA264F1400
   DHCP......... No DHCP or ProxyDHCP offers were received
   PXE-MOF: Exiting Intel Boot Agent
```

The timeout message indicates that the client is sending a DHCP request and not getting a response. This issue is probably due to an error in the DHCP configuration. Check to see if your client is configured correctly in the DHCP server.

TFTP Error or System Hangs After GATEWAY Message

The DHCP server provides an IP address and a location of the initial boot program as part of the DHCP response.

- If the boot program does not exist, then the AI client boot cannot proceed. The following message is displayed:

  ```
  Intel(R) Boot Agent PXE Base Code (PXE-2.1 build 0.86)
       Copyright(C) 1997-2007, Intel Corporation

       CLIENT MAC ADDR 00 14 4F 29 04 12 GUID FF2000008 FFFF FFFF FFFF 7BDA264F1400
       CLIENT IP: 10.6.68.29   MASK: 255.255.255.0    DHCP IP:  10.6.68.49
       GATEWAY: 10.6.68.1
       TFTP.
       PXE-T02:    Access Violation
       PXE-E3C: TFTP Error - Access violation
       PXE-MOF: Exiting Intel Boot Agent
  ```

- If the boot program exists, but it is an incorrect program, the AI client hangs after displaying this message:

  ```
  Intel(R) Boot Agent PXE Base Code (PXE-2.1 build 0.86)
       Copyright(C) 1997-2007, Intel Corporation

       CLIENT MAC ADDR 00 14 4F 29 04 12 GUID FF2000008 FFFF FFFF FFFF 7BDA264F1400
       CLIENT IP: 10.6.68.29   MASK: 255.255.255.0    DHCP IP:  10.6.68.49
       GATEWAY: 10.6.68.1
  ```

System Hangs After GRUB Menu Entry is Selected

If the client is able to do the initial boot, but the kernel cannot be booted, the system hangs after you select the entry from the GRUB menu.

On the install server, check whether the `menu.lst` file for this client is pointing to a valid boot archive. The boot directory of the image on the server should be loop-back mounted under the `/tftpboot` directory as shown in this sample excerpt from `df -k`:

```
/export/home/images/osol-1003-ai-x86/boot \
60450439 21678071 38772368 36% /tftpboot/I86PC.Solaris-12
```

`I86PC.Solaris-12` is a sample. The number might vary on your system.

If you know the name of the target directory that you used in the `installadm create-service` command, then you can use that information to determine whether that target directory is mounted. Also, check whether you can access the `/tftpboot/I86PC.Solaris-12/boot_archive` file.

HTTP Request Sent Results in 403 Forbidden or 404 Not Found

On the install server, if one of the install programs is inaccessible or does not exist in the location specified in the `menu.lst` file under `/tftpboot`, then the client is able to boot, but is not able to download that file. An error message is displayed indicating which file is causing the problem. For example, in the following output, the `solaris.zlib` file does not exist at the specified location.

```
SunOS Release 5.11 Version snv_151 64-bit
   ...
   Hostname: solaris
   Remounting root read/write
   Probing for device nodes ...
   Preparing automated install image for use
   Downloading solaris.zlib archive
   --15:40:37--  http://10.6.35.226:5555//export/home/images/ai_x86_111/solaris.zlib
           => ‘/tmp/solaris.zlib’
   Connecting to 10.6.35.226:5555... connected.
   HTTP request sent, awaiting response... 403 Forbidden
   15:40:37 ERROR 403: Forbidden.

   FAILED
   Requesting System Maintenance Mode
   (See /lib/svc/share/README for more information.)
   Console login service(s) cannot run
```

Check the contents of the target directory that you specified when you ran the `installadm create-service` command.

Automated Installer Disabled

When installing the Oracle Solaris OS on x86 client systems for installations that boot over the network, you must select the second entry in the GRUB boot menu to initiate an automated installation. Typically, the menu entries display as follows:

```
Oracle Solaris 11 Express boot image
Oracle Solaris 11 Express Automated Install
```

If you selected the first GRUB menu entry, or allowed the prompt to time out, the system boots into the automated install boot image, but the installation does not start. The following message is displayed:

```
Auto-installer disabled. Enable the auto-installer service
by running the following command:
svcadm enable svc:/application/auto-installer:default
```

To start an automated installation, you can log in and enable the install service as shown in the message, or you can reboot your system and select the second menu entry.

SPARC and x86 Error Messages

The following errors are common to both SPARC and x86 installations.

- "Automated Installation Failed Message" on page 108
- "Unable to Contact Valid Package Server Message" on page 108
- "Install Log Reports Missing Package" on page 109

Automated Installation Failed Message

If there is a failure during installation, then the following message is displayed:

```
Automated Installation failed. Please refer to /tmp/install_log file for
   details
   Apr  9 14:28:09 solaris svc.startd[7]: application/auto-installer:default
   failed fatally: transitioned to maintenance (see 'svcs -xv' for details)
```

Unable to Contact Valid Package Server Message

If there is a failure during installation, check the log file, `/tmp/install_log`. If the AI client could not access the package server for installing packages, you might see the following error messages:

```
installation will be performed from http://pkg.oracle.com/ (solaris)
   list of packages to be installed is:
               entire
               babel_install
   pkg: The URL 'http://pkg.oracle.com/' does not appear to point to a
   valid pkg server.
```

```
Please check the server's address and client's network configuration.
Additional details:

Unable to contact valid package server: http://pkg.oracle.com/
Encountered the following error(s):
Transport errors encountered when trying to contact depot server.
Reported the following errors:
Could not retrieve versions from 'solaris'
URLError, reason: (8, 'node name or service name not known')
Unable to initialize the pkg image area at /a
```

The problem might be caused by one of the following issues:

- The `pkg_server` is down. Try to access the `pkg_server` and check status.
- If you are using DNS, check whether the AI client is set up with DNS. See "Check DNS" on page 102.

Install Log Reports Missing Package

If one of the packages specified in the AI manifest cannot be located in the IPS repositories, then the installer fails before installing any packages on the disk. In the following example, the installer could not find the package, `SUNWTestPkg`, in the IPS repository. You might see the following error message in the `install_log`:

```
<AI Feb 12 20:15:40> installation will be performed from
   http://pkg.oracle.com/solaris/release (solaris)
   <AI Feb 12 20:15:40> list of packages to be installed is:
   <AI Feb 12 20:15:40> entire
                babel_install
   <AI Feb 12 20:15:40> SUNWTestPkg
   <OM Feb 12 20:15:40> Set zfs root pool device
   <OM Feb 12 20:15:40> creating zpool
   <OM Feb 12 20:15:43> /usr/sbin/zfs get -Hp -o value available rpool
   <OM Feb 12 20:15:43> Creating swap and dump on ZFS volumes
   <OM Feb 12 20:16:00> TI process completed successfully
   <OM Feb 12 20:16:00> Transfer process initiated
   <TRANSFERMOD Feb 12 20:16:59> IPS initialization finished
   <TRANSFER_MOD Feb 12 20:52:33> pkg missing
   <TRANSFERMOD Feb 12 20:52:33> IPS transfer failed
   <TRANSFERMOD Feb 12 20:52:33> IPS transfer finished
   <OM Feb 12 20:52:33> Transfer failed with error 15
   <AI Feb 12 20:52:42> om_perform_install failed with error 114
   <AI Feb 12 20:52:42> Auto install failed
```

Check whether the package in question is a valid package. If this package is available on a different IPS repository, add that IPS repository in the AI manifest by using the `<source>` tag.

Run Automated Installations in Debug Verbose Mode

You can run the installer in verbose debug mode, in order to capture more information about the installation in the `install_log` file.

Boot SPARC Systems in Debug Mode

To boot SPARC systems in debug mode, boot the system with the `install_debug` boot argument.

For network installations:

```
ok> boot net:dhcp - install install_debug
```

For installations using media:

```
ok> boot cdrom - install install_debug
```

Boot x86 Systems in Debug Mode

To boot x86 systems in debug mode, edit the GRUB menu and add the `install_debug=enable` boot property to the kernel line of the entry you want to boot.

```
kernel$ ... -B install_media=...,install_debug=enable
```

Alternatively for x86 systems, you can preset debug mode on the AI server by using the `-b <property=value>` option with the `installadm create-service` and `installadm create-client` commands. This option presets that boot property in the GRUB menu for that service or client.

```
# installadm create-client -e MAC_address -n service_name \
-b install_debug=enable
```

Boot the Install Environment Without Starting an Installation

To boot the install environment without starting an installation, boot without specifying the `install` flag as a boot argument.

- For a SPARC system, boot using one of the following two commands:

  ```
  ok> boot net:dhcp
  ok> boot cdrom
  ```

  Since you did not include the `install` argument in the command, the automated installation will not start.

- For an x86 system, boot using one the following options.
  - For x86 installations that boot over the network, the GRUB menu displays as follows.

    ```
    Oracle Solaris 11 Express boot image
    Oracle Solaris 11 Express Automated Install
    ```

    The default GRUB menu entry, "Oracle Solaris 11 Express boot image," boots the image without starting a hands-free automated installation.

    Note – To ensure the system boots without starting the installation, make sure the entry you choose to boot does not have the `install=true` boot property specified in its kernel line.

  - If you are booting an x86 system from media, edit the GRUB menu and remove the `install=true` boot property from kernel line of the entry you want to boot.

    Note – In general for x86 installations, if the `install=true` boot property is specified in the kernel line of the GRUB entry you are booting from, the installation automatically starts. If you want to boot your x86 based system without initiating an automated installation, check that the GRUB boot entry does not specify the `install=true` boot property. If the property is specified, edit the kernel line of boot entry and remove the property.

Start Installation After Booting Without Initiating an Installation

If you selected a boot option that does not initiate an installation, then use the `svcadm` command to enable the auto-installer service.

```
# svcadm enable auto-installer:default
```

◆ ◆ ◆ APPENDIX B

Automated Installer Installation Administration Commands

This appendix shows the `installadm`(1M) man page.

The `installadm`(1M) Man Page

```
System Administration Commands                installadm(1M)

NAME
     installadm - Manages automated installations on a
     network

SYNOPSIS
     /usr/sbin/installadm

     installadm create-service [-b <property>=<value>,...]
     [-f <bootfile>] [-n <svcname>] [-i <dhcp_ip_start>
     -c <count_of_ipaddr>] [-s <srcimage>] <targetdir>

     installadm delete-service [-x] <svcname>

     installadm list [-n <svcname>] [-c] [-m]

     installadm enable <svcname>

     installadm disable <svcname>

     installadm add-manifest -m <manifest> -n <svcname>
     [-c <criteria=value|range> ... | -C <criteria.xml>]

     installadm delete-manifest -m <manifest> -n <svcname>

     installadm set-criteria -m <manifest> -n <svcname>
     -a|-c <criteria=value|range> ... | -C <criteria.xml>

     installadm create-client [-b <property>=<value>,...]
     [-t <imagepath>] -e <macaddr> -n <svcname>

     installadm delete-client <macaddr>
```

```
     installadm help [<subcommand>]

DESCRIPTION

     The automated installer (AI) is used to automate the
     installation of the Oracle Solaris OS on one or more
     SPARC and x86 systems over a network. The
     installations can differ in architecture, packages
     installed, disk capacity, and other parameters.

     The minimal configuration necessary to use the
     automated installer is to have one system as the
     server and one client on which to install. On the
     server, you set up an installation service, which
     is associated with manifests, or specifications,
     for specific x86 installations or SPARC installations.

     Manifests can include information such as a target
     device, additional packages, partition information,
     and other parameters. When the client boots, a
     search is initiated for a manifest that matches the
     client's machine specifications. When a matching
     manifest is found, the client is installed with the
     Oracle Solaris release according to the specifications
     in the manifest file.

     Use the installadm create-service command to set up
     an install server and create an install service.

     An install service is a network entity that specifies
     the parameters for a particular type of installation.
     These specifications are defined in XML manifest
     files.

     The automated installer uses AI ISO images to create
     the install services. An AI ISO image is a collection
     of software in a single file. This image is unpacked
     when an install service is created and used to create
     a net image that enables client installations.

     Once an install server and an installation service are
     set up, you can install the Oracle Solaris release to a
     client on the network, per the default specifications
     in the install service, by booting the client system.

     If you want to perform various types of installations
     in a network, you can create and manage additional
     install services tailored for each type of
     installation by using the installadm create-service
     command. For example, you can set up an install
     service that installs the Oracle Solaris OS to x86
     clients and a service that installs the Oracle Solaris
     OS to SPARC clients.

     If you have clients with varying machine specifications,
     you can manually create or modify manifests so that the
     manifests cover those specific machine specifications.
     Then, you can use the installadm add-manifest command
     to add your new manifests to an install service. You can
```

```
     also use add-manifest to specify criteria to be used in
     determining which manifest should be selected for an
     installation. You can change criteria already associated
     with a manifest with the installadm set-criteria command.

     If you want a specific client to use a specific
     install service, you can associate a service to a
     client by using the installadm create-client command.

     The installadm utility can be used to accomplish the
     following tasks:
     - Set up install services
     - Set up installation images
     - Set up or remove clients
     - Add or delete manifests
     - Specify or modify criteria for a manifest
     - Enable or disable install services
     - List install services
     - List manifests for an install service

SUBCOMMANDS AND OPTIONS
     The installadm command has the subcommands and options
     listed below. Also, see EXAMPLES below.

     installadm

          Displays command usage.

     installadm create-service [-b <property>=<value>,...]
     [-f <bootfile>] [-n <svcname>] [-i <dhcp_ip_start>
     -c <count_of_ipaddr>] [-s <srcimage>] <targetdir>

          Creates an install service.

          The command provides the following functionality:
          - Takes an AI ISO image (<srcimage>), unpacks it,
            and sets up a net image in a target directory
            (<targetdir>. The net image enables client
            installations.
          - Creates an install service and makes it
            available on the network.

          Note the following specifications:
          - For SPARC install services, the first install
            service created on an install server is the
            service that will be used for all client
            installations that use the install server. If
            you want a client to use a different
            install service from this install server, you
            must use the installadm create-client command
            to create a client-specific configuration.
          - By default, both a net image and an install
            service are created.
          - If an existing install service name is provided,
            that existing service is used.
          - If the -s option is not used, and the
            <targetdir> contains a valid net image, then a
            new install service is created with the
            existing net image.
```

```
        - If the -i option and the -c option are used,
          and a DHCP server is not yet configured, a DHCP
          server is configured.
        - If an already-configured DHCP server exists,
          that DHCP server is updated.
        - If DHCP is running on a remote system, updates
          can happen through secure shell. User must
          provide authentication.

    -b  <property>=<value>,...
        For x86 clients only, sets a property value
        in the service-specific menu.lst file in /tftpboot.
        Use this option to set boot properties that are
        specific to this service. This option can accept
        multiple property=value pairs.

    -f  <bootfile>
        Uses this boot file for the install service. If
        boot file doesn't exist, it is created. If this
        option is not specified, a boot file is created
        with a default name.

    -n  <svcname>
        Uses this install service name instead of default
        service name.

        Note: If the -n option is not used, a unique name
        for the install service is automatically assigned
        using the format,
        "_install_service_<port_number>".

        For example, if the port number that installadm
        selected for an install service is 46510, and
        the -n option was not used to provide a custom
        name for the install service, then the install
        service name will be "_install_service_46510."

    -i  <dhcp_ip_start>
        Sets up a new DHCP server. The IP addresses,
        starting from dhcp_address_start, are set up.

    -c  <count_of_ipaddr>
        Sets up a total number of IP addresses in the
        DHCP table equal to the value of the
        count_of_ipaddr. The first IP address is the
        value of dhcp_ip_start that is provided by the
        -i option.

    -s  <srcimage>
        Specifies location of AI ISO image to use for
        setting up the install service.

    <targetdir>

          Required: Specifies location to set up net image.

    installadm delete-service [-x] <svcname>

          Deletes an install service. Accomplishes the
```

```
      following:
      - Removes install service from the network.
      - Stops the web server that is running for this
        install service.
      - Removes the manifest files and web server
        configuration for this install service.

-x    Deletes the install service and also removes the
      associated target net image.

<svcname>

      Required: Specifies the install service name.

installadm list [-n <svcname>] [-c] [-m]

    Lists all enabled install services on a server.

    -c
        Lists the clients of the install services on a local
        server.

    -m
        Lists the manifests associated with the install ser-
        vices on a local server.

    -n <svcname>
        Lists information about the specific install service
        on  a  local server.  Or, if the -c option is speci-
        fied, lists the client information  associated  with
        the specified install service.  Or, if the -m option
        is specified, lists the  manifests  associated  with
        the specified install service.

installadm enable <svcname>

     Enables a specified install service. Also,
     enables the web server associated with the
     service.

<svcname>

      Required: Specifies the name of the install
      service to be enabled.

installadm disable <svcname>

     Disables a specified install service. Also,
     disables the web server associated with the
     service.

<svcname>

     Required: Specifies the name of the install
     service to be disabled.

installadm add-manifest -m <manifest> -n <svcname>
[-c <criteria=value|range> ... | -C <criteria.xml>]
```

```
          Associates manifests with a specific install
          service, thus making the manifests available on
          the network, independently from creating a
          service. When publishing a non-default manifest,
          it is required to associate criteria either via
          criteria entered on the command line (-c) or
          via a criteria XML file (-C).

     -m   <manifest>
          Required: Specifies the path name of the manifest
          to add.

     -n   <svcname>
          Required: Specifies the name of the install
          service this manifest is to be associated with.

     -c   <-c <criteria=value|range> ...>
          Optional: Specifies criteria to be associated with the
          added non-default manifest. When publishing a default
          manifest, criteria must not be specified. When
          publishing a non-default manifest, criteria must be
          specified.

     -C   <criteria.xml>
          Optional: Specifies the path name of a criteria
          XML file containing criteria to be associated with the
          added non-default manifest. When publishing a default
          manifest, criteria must not be specified. When
          publishing a non-default manifest, criteria must be
          specified.

     installadm delete-manifest -m <manifest> -n <svcname>

          Deletes a manifest that was published with a
          specific install service.

     -m   <manifest>
          Required: Specifies the name of an AI manifest
          as output by installadm list with -n option.

     -n   <svcname>
          Required: Specifies the name of the install
          service this manifest is associated with.

     installadm set-criteria -m <manifest> -n <svcname>
     -a|-c <criteria=value|range> ... | -C <criteria.xml>

          Updates criteria of an already published manifest.
          Criteria can be specified via the command line or
          or via a criteria xml file. Criteria must be
          specified with one of the mutually exclusive
          options, -a, -c, or -C.

     -m   <manifest>
          Required: Specifies the name of a manifest.

     -n   <svcname>
          Required: Specifies the name of the install
          service this manifest is associated with.
```

```
-c   <-c <criteria=value|range> ...>
     Optional: Specifies criteria to replace all existing
     criteria for the manifest.

-a   <-a <criteria=value|range> ...>
     Optional: Specifies criteria to be appended to the
     existing criteria for the manifest. If the criteria
     specified already exists, the value/range of that
     criteria is replaced by the specified value/range.

-C   <criteria.xml>
     Optional: Specifies the path name of a criteria
     XML file containing criteria to replace all existing
     criteria for the manifest.

installadm create-client [-b <property>=<value>,...]
[-t <imagepath>] -e <macaddr> -n <svcname>

     Accomplishes optional setup tasks for a specified
     client, in order to provide custom client
     settings that vary from the default settings used
     by the installadm create-service command. Enables
     user to specify a non-default service name and
     image path for a client:
     - Specifies installation service for that client.
     - Sets up DHCP macro, if it doesn't exist.

-b   <property>=<value>,...
     For x86 clients only, sets a property value
     in the client-specific menu.lst file in /tftpboot.
     Use this option to set boot properties that are
     specific to this client. This option can accept
     multiple property=value pairs.

-e   <macaddr>
     Required: Specifies a MAC address for the client.

-n   <svcname>
     Required: Specifies the install service for client
     installation.

-t   <imagepath>
     Specifies the path of the net image to be used with
     automated installer.

installadm delete-client <macaddr>

     Deletes an existing client's specific service
     information that was previously set up using the
     installadm create-client command.

<macaddr>

     Required: Specifies a MAC address for the client.

installadm help [<subcommand>]

     Displays the syntax for the installadm utility.
```

```
     <subcommand>

          If subcommand is provided, the command provides
          the syntax for that subcommand.

CRITERIA FILES

     A criteria XML file allows you to specify criteria for a
     manifest by passing the file to the add-manifest or
     set-criteria commands. Criteria can be specified as a
     value or a range, by using the following tags.

     For a criterion with a specific value:

     <ai_criteria_manifest>
         <ai_criteria name=XXXX>
             <value>yyyy</value>
         </ai_criteria>
     </ai_criteria_manifest>

     where XXXX is the name of the criterion (e.g. MAC, IPV4,
     MEM, or ARCH) and yyyy is the value of the criterion.

     For a criterion with a range:

     <ai_criteria_manifest>
         <ai_criteria name=XXXX>
             <range>
                 yyyy1
                 yyyy2
             </range>
         </ai_criteria>
     </ai_criteria_manifest>

     where XXXX is the name of the criterion (e.g. MAC, IPV4,
     or MEM) and yyyy1 and yyyy2 are the lower and uppper
     bounds of the range.

     Multiple criteria may be specified in the file between
     the <ai_criteria_manifest> and </ai_criteria_manifest>
     tags.

EXAMPLES

     Example 1: Set up an install server and an install
     service for the first time. The command includes a
     starting IP address and total count of IP addresses,
     in order to configure the DHCP server.

     Example% # installadm create-service -n 0906x86 \
     -i 10.6.68.201 -c 5 -s \
     /export/aiimages/osol-0906-ai-x86.iso \
     /export/aiserver/osol-0906-ai-x86

     In this example, the terminal displays the progress
     as follows:

     Setting up the target image at
```

```
/export/aiserver/osol-0906-ai-x86 ...
Registering the service 0906x86._OSInstall._tcp.local
Creating DHCP Server
Created DHCP configuration file.
Created dhcptab.
Added "Locale" macro to dhcptab.
Added server macro to dhcptab - line1-x4100.
DHCP server started.
Added network macro to dhcptab - 10.0.0.0.
Created network table.
copying boot file to
/tftpboot/pxegrub.I86PC.Solaris-1
Service discovery fallback mechanism set up

The AI ISO image is at
/export/aiimages/osol-0906-ai-x86.iso. The command
sets up a net image and an install service that is
based on the AI ISO image.

The installation net image is created in the
/export/aiserver/osol-0906-ai-x86 target directory.
This net image enables client installations.

The progress display shows that the install service,
named 0906x86, is created.

A boot file is created, also named 0906x86,
under /tftpboot. The client will get this file name
through DHCP. The command also creates a link from
the net image at /export/aiserver/osol-0906-ai-x86
to a web server that is running on port 5555.

The DHCP server and macro is created. The starting
IP address is 0.6.68.201. Five IP addresses are
allocated for clients. The command results
identify the macro as dhcp_macro_0906x86.

Example 2: Use the following sample command to set
up a client that references a specific install service
and a specific net image location.

The install service and net image should already exist.

# installadm create-client -b "console=ttya" \
-e 0:e0:81:5d:bf:e0 -t \
/export/aiserver/osol-0906-ai-x86 -n 0906x86

In this example, the terminal displays the
following output:

---
Setting up X86 client...
Service discovery fallback mechanism set up

Detected that DHCP is not set up on this server.
If not already configured, please create a DHCP macro
named 0100E0815DBFE0 with:
Boot server IP (BootSrvA) : 10.6.68.29
Boot file      (BootFile) : 0100E0815DBFE0
```

```
     If you are running the Solaris DHCP Server, use the
     following command to add the DHCP macro,
     0100E0815DBFE0:
     /usr/sbin/dhtadm -g -A -m 0100E0815DBFE0 -d \
     :BootSrvA=10.6.68.29:BootFile=0100E0815DBFE0:\
     GrubMenu=menu.lst.0100E0815DBFE0:

     Note: Be sure to assign client IP address(es) if needed
     (e.g., if running the Solaris DHCP Server, run pntadm(1M)).
     ----

     In this example, the command creates a
     client-specific setup for the system with MAC address
     of 0:e0:81:5d:bf:e0.

     This client will use the install service previously
     set up, named 0906x86, and the net image at
     /export/aiserver/osol-0906-ai-x86.

     Using the -b option, the command sets the console
     value, <console=ttya>, in the client-specific
     menu.lst file in /tftpboot.

     As shown above, this command outputs the name of
     the client-specific macro, 0100E0815DBFE0,
     and its values that need to be added to the DHCP
     server. If you have a Sun DHCP server, create the
     above macro on your DHCP server by running the dhtadm
     command from the output on your DHCP server.

     The pntadm(1M) command may need to be called if you
     set up a Sun DHCP server and client IP addresses
     need to be assigned.  See the pntadm(1M) manpage for
     more information.

     On systems which support graphic interfaces, the
     DHCP Manager may be used instead of the dhtadm or
     pntadm commands. See the dhcpmgr(1M) manpage for more
     information.

     Example 3: Use the following sample command to replace
     the default manifest for an existing install service,
     service_092910, with a custom manifest, my_manifest.xml.
     The manifest contains a name attribute, name="default",
     which designates it as the default manifest.

     # installadm add-manifest -m my_manifest.xml \
     -n service_092910

     Example 4: Use the following  sample  command  to  list  the
     install services on a local server:

         # installadm list

     In this example, the terminal displays the following output:

Service Name Status       Arch  Port  Image Path
------------ ------       ----  ----  ----------
svc-2008-11  off          x86   45602 /export/server/osol-0811-ai-x86
```

```
svc-2009-06  on           x86   45601 /export/server/osol-0906-ai-x86
svc-bld-127  on           x86   45603 /export/server/osol-b127-ai-x86
```

Example 5: Use the following sample command to list the clients for a specific install service of a local server:

```
# installadm list -c -n svc-2009-06
```

In this example, the terminal displays the following output:

```
Service Name Client Address    Arch  Image Path
------------ --------------    ----  ----------
svc-2009-06  01:C2:52:E6:4B:E1 x86   /export/server/osol-0906-ai-x86
```

Example 6: Use the following sample command to list the manifests associated with a specific install service on a local server:

```
# installadm list -m -n svc-2009-06
```

In this example, the terminal displays the following output:

```
Manifest          Criteria
--------          --------
devpublisher.xml  arch     = i86pc
                  ipv4     = 010.000.002.015
                  mac      = 01:C4:51:E6:4B:E6 - 01:C4:51:E6:4B:E9
                  mem      = 2048 MB
```

Example 7: Use the following sample command to add manifest1 to svc1 with a criteria of MAC address equaling "aa:bb:cc:dd:ee:ff":

```
# installadm add-manifest -m manifest1 -n svc1
  -c MAC="aa:bb:cc:dd:ee:ff"
```

Example 8: Use the following sample command to add manifest2 to svc1 with a criteria of an IPv4 range between 10.0.2.100 to 10.0.2.199:

```
# installadm add-manifest -m manifest2 -n svc1
  -c IPV4="10.0.2.100-10.0.2.199"
```

Example 9: Use the following sample command to add manifest3 to svc1 with a criteria of 2048MB memory or greater and an architecture of i86pc:

```
# installadm add-manifest -m manifest3 -n svc1
  -c MEM="2048-unbounded" -c ARCH=i86pc
```

Example 10: Use the following sample command to append to the criteria of manifest2 of svc1, a criterion of 4096MB memory or greater:

```
# installadm set-criteria -m manifest2 -n svc1
  -a MEM="4096-unbounded"
```

Example 11: Use the following sample command to replace the criteria of manifest2 of svc1 with the criteria specified by

```
     the file, /tmp/criteria.xml:

         # installadm set-criteria -m manifest2 -n svc1
           -C /tmp/criteria.xml

     See the CRITERIA FILES section for more information on the
     contents of the criteria xml file.

ATTRIBUTES
     See attributes(5) for descriptions of the  following  attri-
     butes:

     _____________________________________________________________
    |       ATTRIBUTE TYPE        |       ATTRIBUTE VALUE        |
    |_____________________________|_____________________________|
    | Availability                | install/installadm            |
    |_____________________________|_____________________________|
    | Interface Stability         | None / Under Development      |
    |_____________________________|_____________________________|

SEE ALSO
     pntadm(1M), dhcpmgr(1M), attributes(5)

     Oracle Solaris Automated Installer Guide on
     http://docs.sun.com/

     Getting Started with Oracle Solaris on
     http://docs.sun.com/

        Last Changed October 12, 2010
```

◆ ◆ ◆ APPENDIX C

Migrating From JumpStart to Automated Installer

This appendix provides information to help you migrate from Solaris Custom JumpStart to Oracle Solaris Automated Installer.

Comparing JumpStart and Automated Installer

Both Solaris Custom JumpStart and Oracle Solaris Automated Installer (AI) provide hands-free installation of multiple systems on a network. Clients are booted over the network, and once the client is booted, the installer takes over.

JumpStart installs the Oracle Solaris 10 OS and earlier versions of the Oracle Solaris OS. Automated Installer installs the Oracle Solaris 11 Express OS.

JumpStart and AI share the following characteristics:

- Provide hands-free network installation of multiple clients by storing system configurations on an install server
- Provide for different kinds of installations on different clients in one automated installation
- Install both x86 and SPARC clients

TABLE C–1 Comparing JumpStart and AI Tasks

Task	JumpStart	AI
Set up an install server.	Use the `setup_install_server`(1M) command.	Use the `installadm create-service` command. See Chapter 2, "Setting Up an AI Install Server."
Add clients to the installation.	Use the `add_install_client`(1M) command.	Use the `installadm create-client` command. See Chapter 7, "Installing Client Systems."

TABLE C–1 Comparing JumpStart and AI Tasks *(Continued)*

Task	JumpStart	AI
Specify installation instructions.	Use profile files.	Use AI manifest files. See Chapter 4, "Specifying Installation Instructions."
Specify client customizations.	Use rules files to associate clients with profile files.	Use the `installadm set-criteria` command to associate clients with AI manifests. See Chapter 3, "Customizing Installations."
Specify post-installation client configuration.	Use finish scripts and `sysidcfg`(4) files.	Use SMF manifest files that are associated with particular AI manifests. See Chapter 5, "Configuring the Client System."

Converting JumpStart Rules to AI Criteria

AI client criteria can be specified in the following two ways. The directives in the following table apply in both cases. See Chapter 3, "Customizing Installations."

- As `-c` options of the `installadm add-manifest` command or as `-c` or `-a` options of the `installadm set-criteria` command
- In criteria files

Comparing Rules Keywords and Criteria Directives

The following table compares JumpStart rules keywords with AI criteria directives. AI uses these criteria to apply the correct AI manifest and SC (system configuration) manifest information to a particular client. JumpStart rules keywords that are not listed in this table do not have a comparable AI directive, or their functionality does not apply to AI.

TABLE C–2 Comparing JumpStart Rules File Keywords and AI Criteria Directives

JumpStart Rules File Keyword	AI Criteria File Directives
`any`	For client systems that do not match any selection criteria, the AI install service provides a default AI manifest.

`arch sparc`	Command option: `-c cpu=sparc`
	Criteria file:
	`<ai_criteria name="cpu">`
	`  <value>sparc</value>`
	`</ai_criteria>`

TABLE C–2 Comparing JumpStart Rules File Keywords and AI Criteria Directives *(Continued)*

JumpStart Rules File Keyword	AI Criteria File Directives

`hostaddress 10.6.68.127`	Command option: `-c ipv4=10.6.68.127`
	Criteria file:
	`<ai_criteria name="ipv4">`
	`  <value>10.6.68.127</value>`
	`</ai_criteria>`

`hostname solaris`	To uniquely identify a host in AI, use either the IP address as shown above, or use the MAC address.
	Command option: `-c mac=0:3:ba:33:9d:b6`
	Criteria file:
	`<ai_criteria_name="mac">`
	`  <value>0:3:ba:33:9d:b6</value>`
	`</ai_criteria>`

`karch i86pc`	Command option: `-c arch=i86pc`
	Criteria file:
	`<ai_criteria name="arch">`
	`  <value>i86pc</value>`
	`</ai_criteria>`

`memsize 2048`	Command option: `-c mem=2048`
	Criteria file:
	`<ai_criteria name="mem">`
	`  <value>2048</value>`
	`</ai_criteria>`

`model SUNW,Sun-Fire-T200`	Command option: `-c platform=SUNW,Sun-Fire-T200`
	Criteria file:
	`<ai_criteria_name="platform">`
	`  <value>SUNW,Sun-Fire-T200</value>`
	`</ai_criteria>`

`network 10.0.0.1`	Use `ipv4` with a range.
	Command option: `-c ipv4=10.0.0.1-10.0.0.64`
	Criteria file:
	`<ai_criteria name="ipv4">`
	`  <range>10.0.0.1 10.0.0.64</range>`
	`</ai_criteria>`

Converting Rules Files to Criteria

This section shows some sample JumpStart rules converted to AI criteria. The `add-manifest` and `set-criteria` subcommands associate client criteria with specified AI manifests. See "Associate Client-Specific Installation Instructions With Install Services" on page 24.

Sample JumpStart Rules File

In a JumpStart rules file, the first column contains rule keywords and rule values that identify client systems, the second column is the begin script, the third column is the profile, and the fourth column is the finish script.

The following sample JumpStart rules file identifies three groups of systems, and each group is assigned a different profile. In this example, no begin or finish scripts are specified for any of the three client groups. This sample rules file states the following rules:

- Any SPARC systems with a memory size between 2048 MB and 4096 MB will be installed using specifications in the `lx_prof` profile.
- Any x86 systems will be installed using specifications in the `prog_prof` profile.
- Any systems not covered in the above rules will be installed using specifications in the `generic_prof` profile.

```
memsize 2048-4096 && model SUNW - lx_prof -
karch i86pc - prog_prof -
any - generic_prof -
```

Equivalent AI Criteria Specifications

This section shows AI commands that are comparable to the three rules from the previous JumpStart sample. In these examples, the `lx_prof` profile has been converted to the `manifest_lx.xml` AI manifest and the `prog_prof` profile has been converted to the `manifest_prog.xml` AI manifest. Clients that do not match any of these specified criteria use the default AI manifest.

In the following example, any SPARC systems with a memory size between 2048 MB and 4096 MB will be installed using specifications in the `manifest_lx.xml` AI manifest.

```
# installadm add-manifest -m ./manifest_lx.xml -n s11-sparc \
-c arch="sparc" -c mem="2048-4096"
```

You can also put the criteria in a file and reference the file. The following command is equivalent to the preceding command if the content of the `criteria_lx.xml` file is as shown.

```
# installadm add-manifest -m ./manifest_lx.xml -n s11-sparc \
-C ./criteria_lx.xml
```

```
<ai_criteria_manifest>
  <ai_criteria name="cpu">
    <value>sparc</value>
  </ai_criteria>
  <ai_criteria name="mem">
    <range>
      2048
      4096
    </range>
  </ai_criteria>
</ai_criteria_manifest>
```

In the following example, any x86 systems will be installed using specifications in the `manifest_prog.xml` AI manifest.

```
# installadm set-criteria -m ./manifest_prog.xml -n s11-x86 \
-c arch="i86pc"
```

Converting a JumpStart Profile to an AI Manifest

The following table compares JumpStart profile keywords with AI manifest directives. AI uses XML manifest files to define the client installation. See Chapter 4, "Specifying Installation Instructions." JumpStart profile keywords that are not listed do not have a comparable AI directive, or their functionality does not apply to AI.

TABLE C–3 Comparing JumpStart Profile File Keywords and AI Manifest Directives

JumpStart Rules File Keyword	AI Manifest Directives

`boot_device c0t0d0s0 update`	`<target>`
	`  <target_device>`
	`  <disk>`
	`  <disk_name name="c0t0d0" name_type="ctd"/>`
	`  <slice name="0" is_root="true" force="true"/>`
	`  </disk>`
	`  </target_device>`
	`</target>`
	
	The second token value (`update` for SPARC systems and `preserve` for x86 systems) for this keyword is not supported in AI. In AI, the EEPROM on SPARC systems is always updated to the specified target device, so that the installed system automatically boots from that device. On x86 systems, the firmware is never updated.

`bootenv`	A boot environment is automatically created on installation of the Oracle Solaris OS.

TABLE C–3 Comparing JumpStart Profile File Keywords and AI Manifest Directives *(Continued)*

JumpStart Rules File Keyword	AI Manifest Directives

`cluster SUNWCXall`	The Oracle Solaris 11 ExpressOS uses `pkg`(5) group packages. Group packages are specified just as any other package is specified in the manifest. The default AI manifest includes the packages needed for a standard Oracle Solaris 11 Express installation. You can customize this list of packages.
	`<software_data action="install" type="IPS">`
	`  <name>pkg:/entire</name>`
	`  <name>pkg:/babel_install</name>`
	`</software_data>`

`fdisk c0t3d0 solaris maxfree`	`<target>`
	`  <target_device>`
	`  <disk>`
	`  <disk_name name="c0t3d0" name_type="ctd"/>`
	`  <partition name="1" part_type="SOLARIS">`
	`  </disk>`
	`  </target_device>`
	`</target>`
	See "Configuring Partitioning on an x86 Client" on page 46.

`filesys`	In JumpStart, this `filesys` keyword creates the VTOC slice and the UFS file system on that slice. AI creates ZFS file systems, not UFS file systems. To create VTOC slices in AI, use the `<slice>` element. See "Configuring Slices on a Disk" on page 50.
`geo`	Geographic regions for language support are specified through `pkg`(5) group packages. Group packages are specified just like any other package is specified in the manifest. See the JumpStart `cluster` keyword above for more information.
`locale`	Locale support is specified through `pkg`(5) group packages. Group packages are specified just like any other package in the manifest. See the JumpStart `cluster` keyword above for more information.

`package SUNWmysql51 add nfs golden:/packages/Solaris_10`	`<software name="IPS">`
	`  <software_data action="install" type="IPS">`
	`  <name>pkg:/entire</name>`
	`  <name>pkg:/babel_install</name>`
	`  <name>pkg:/database/mysql-51</name>`
	`  </software_data>`
	`</software>`

`package SUNWmysql delete`	`<software_data action="uninstall" type="IPS">`
	`  <name>pkg:/database/mysql-4</name>`
	`</software_data>`

TABLE C–3 Comparing JumpStart Profile File Keywords and AI Manifest Directives *(Continued)*

JumpStart Rules File Keyword	AI Manifest Directives

`partitioning default`
`partitioning existing`
`partitioning explicit`
	If the `target_device` directives are omitted from the AI manifest, a default partitioning is used. See "Configuring Partitioning on an x86 Client" on page 46.

`usedisk`	`<target>`
	`  <target_device>`
	`  <disk>`
	`  <disk_keyword key="boot_disk"/>`
	`  </disk>`
	`  </target_device>`
	`</target>`

Converting Profile Files to AI Manifests

This section shows some sample JumpStart profile specifications converted to AI manifest directives. See Chapter 4, "Specifying Installation Instructions," for more information about AI manifest directives.

Sample JumpStart Profile File

In a JumpStart profile, the first column contains profile keywords, and the second column contains profile values.

The following sample JumpStart profile specifies:

- Systems will receive an initial installation.
- Systems will be set up as standalone systems.
- On each system, a Solaris `fdisk` partition is created on the largest contiguous free space on the `c0t0d0` disk.
- The Entire Distribution software group, `SUNWCall`, is installed on the system.
- The StarOffice suite of tools is installed on the system.
- The Evolution email and calendar utilities are installed on the system.

```
install_type initial_install
system_type standalone
fdisk c0t0d0 solaris maxfree
cluster SUNWCall
cluster SUNWCstaroffice
cluster SUNWCevo
```

Equivalent AI Manifest Directives

The following excerpts from AI manifests show how you could convert the previous JumpStart profile into directives in AI manifests.

The following example directs the installer to install the Oracle Solaris 11 Express release on the `c0t0d0` disk and create a `SOLARIS` partition on the biggest chunk of contiguous free space. See "Configuring Partitioning on an x86 Client" on page 46. Next the example specifies a source of packages to install and installs the two mandatory group packages. See "Specifying Packages to Install" on page 56 for an explanation for why these same group packages must be uninstalled. Finally, this example installs OpenOffice and Evolution.

```
<?xml version="1.0"?>
<!DOCTYPE auto_install SYSTEM "file:///usr/share/auto_install/ai.dtd">
<auto_install>
  <ai_instance name="sample_ai_manifest" auto_reboot="true">
    <target>
      <target_device>
        <disk>
          <disk_name name="c0t0d0" name_type="ctd"/>
          <partition name="1" part_type="SOLARIS">
        </disk>
      </target_device>
    </target>
    <software name="ips">
      <source>
        <publisher name="solaris">
          <origin name="http://pkg.oracle.com/solaris/release"/>
        </publisher>
      </source>
      <software_data type="IPS">
        <name>pkg:/entire</name>
        <name>pkg:/babel_install</name>
      </software_data>
      <software_data action="uninstall" type="IPS">
        <name>pkg:/babel_install</name>
        <name>pkg:/slim_install</name>
      </software_data>
      <software_data type="IPS">
        <name>pkg:/office/openoffice</name>
        <name>pkg:/mail/evolution</name>
      </software_data>
    </software>
  </ai_instance>
</auto_install>
```