# HOW to USE the AUTOMATED INSTALLER

> **OpenSolaris™** How To Guides

Michael Barrett, OpenSolaris Technical Marketing Engineer

Version 1.0 | *Last updated: 05/22/09*

## About This OpenSolaris How To Guide

There is an exciting new OS provisioning technology in OpenSolaris called Automated Installer (AI). Much like Solaris Jumpstart, Red Hat Kickstart, and Novell AutoYast, AI allows an user to install OpenSolaris remotely and in a non-interactive manner with tremendous freedoms around customization and configuration. This OpenSolaris How To Guide will briefly explain the concepts involved in the new network installation service and show how to quickly get started by walking the reader through two examples involving a SPARC and an Intel/AMD targeted machine. AI involves many technologies from DHCP, DNS, IPS, and others. Those topics will not be the focus of this brief. For a more in depth discussion of AI, see *http://dlc.sun.com/osol/docs/content/dev/AIinstall/.*

Starting with OpenSolaris 2009.06, users will be able to install OpenSolaris on SPARC based machines in a supported manner. At first release, the only way to install on SPARC will be via Automated Installer (AI). AI only supports the remote provisioning of OpenSolaris and not Solaris. Jumpstart, the AI predecessor, does not support the remote provisioning of OpenSolaris. Thus there will be a large set of new users experiencing AI for the first time that are especially interested in SPARC. Hopefully this How To Guide will expedite their experimentation with OpenSolaris.

## Contents

# Automated Installer How To Guide

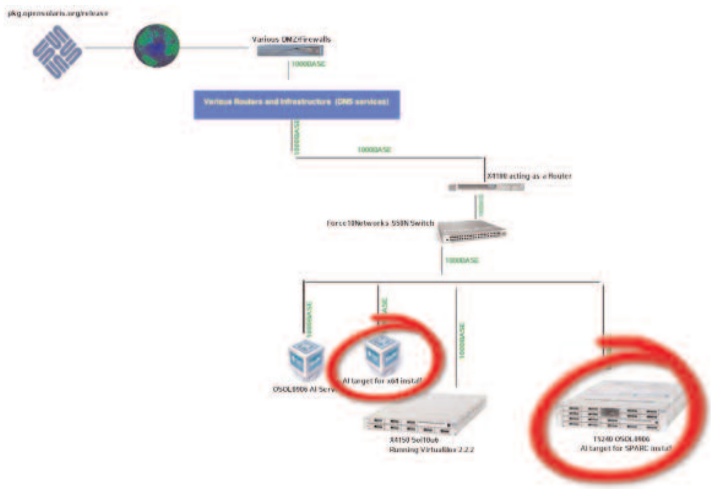
## Overview

Automated Installer represents an evolution in network based OS provisioning. Many of the operating systems on the market today allow for software life cycle administration over the network through a web service. In terms of OpenSolaris, this innovation is represented in the Image Packaging System (IPS). At the same time tasks or services the operating system offers to its application stack and end users have become more and more complex and mobile across the network. Again, OpenSolaris leads the way through it's creation of the Service Management Facility (SMF). Lastly, network based identification assignment of machine assets has become much cleaner and efficient through the use of SLP, mDNS, and DNS. OS installation services in OpenSolaris now fully take advantage of WANboot and PXE boot methods streamlining the process users once had to go through while pre-configuring a compute asset's identify before installation. Automated Installer borrows from all of these innovations and incorporates modern day xml data models throughout it's configuration.

The combination of these innovations are especially useful in an OS provisioning application. The design of Automated Installer allows for the boot strap miniroot to be extremely small. The two-part install of boot strap then IPS pull down allows for the AI server to service more requests faster and allows you to disperse network load to multiple IPS mirrors within your organization. This lowers overall installation time. Lastly, the booting/installing OS is a live object while it is installing. While it is installing you can use it for other activities (such as installing more targets). Automated Installer is a break through technology for operating system provisioning.

### Environment

In this How To Guide, we will remotely install a VirtualBox guest that is sitting on a X4150 and the primary logical domain on a T5240. In the diagram below, our targets are circled in red:



As you can see in the diagram, I will be using my lab's local network behind a corporate infrastructure of network service and firewalls. All three boxes (the AI server, the Intel target, and the SPARC target) will talk to each other over a local 1GB switch in my lab, but all three will also be pulling software down from Sun Microsystems' IPS repositories over the world wide web. In order to use less physical equipment, I've decided to run my AI server in a VirtualBox instance that is on the same physical box as the target Intel based machine I will be PXE booting and remotely installing. For the sake of clarity, from now on I will use the following terminology to describe these three boxes:

**AI server** = machine where the Automated Installer service is running
**x86 target** = Intel/AMD machine I will be remotely installing
**SPARC target** = SPARC machine I will be remotely installing

## Design Concepts

### 2 Primary Actions

The Automated Installer will run as a SMF service on the AI server. You create named **services** that will be served out from this box. These installation services can be a simple default OS installation, ruled by the auto-generated default service manifests, or you may wish to name them after the business service the targets boxes will provide (i.e. MySQL-HR-server or Oracle-CRM-node) then further edit the installation manifests to configure the application layer of the hosts. This is the payload construction part of the action. You have freedom here to match the payload to the business need.

Now that we have services being served out, we need clients. Automated Installer allow you to create **client** configurations to help you target specific boxes and their OS configuration needs. So instead of having a service target a network location of many boxes, you have the granular control to have a service tailored to a compute asset.

The **service** and **client** actions seem abstract, but it's really just setting up a logical order you will use in the **installadm** UNIX commandline. (installadm <action>-service <options> or installadm <action>-client <options>)

### The Language

In order to describe your service, the client, or the relationship between the service and the client, you need to edit/create manifests. There are 3 types of manifests:

**ai_manifest** = the what gets installed, where it get installed on the host, network/IPS information
**sc_manifest** = the historic sysidcfg type of information
**criteria manifest =** mapping between the clients and the correct ai/sc manifests.

## Prerequisites, Assumptions, and Defaults

This How To Guide will be using the environment in the diagram above. It is important to check the following is true in your personal environment to assure success:

- That the AI server and the machines it will be installing have DNS configured and can resolve opensolaris.org.
- AI server has the correct file configuration for a DNS enabled client:
  - > /etc/hosts
  - > /etc/nsswitch.conf
  - > svcs -a | grep dns
  - > /etc/resolv.conf
- That the SPARC box you selected supports WAN boot in the OBP.

- That the x86/x64 box you selected supports PXE boot.
- That you have some familiarity with DHCP.
- That the AI server and the targets can pull software down from the opensolaris.org repositories and are not blocked by firewall rules or web proxy enforcements. You can configure a IPS mirror repository on your private internal network, but the AI server and clients still need to be able to reach the opensolaris.org repository to pull down metadata and installation order lists. This requirement will be lifted as soon as a full mirror support is available in a later release of OpenSolaris.
- You are using or have access to the OpenSolaris 2009.06 or higher AI miniroot ISO image. Only a hard requirement for the SPARC box AI service. You may download the AI images from here: *http://opensolaris.org/os/downloads/*

## Installing Automated Installer Software Packages

Make sure you have the software installed on the AI server:

```
# pkg install SUNWinstalladm-tools
DOWNLOAD                                    PKGS       FILES     XFER (MB)
Completed                                    9/9     1112/1112   6.22/6.22
PHASE                                            ACTIONS
Install Phase                                   1474/1474
```

This IPS package is available from the opensolaris.org/release IPS repository.

```
# svcs -a | grep install
online         12:26:09 svc:/system/install/server:default
```

Notice the install server SMF service that now running and facilitates the Automated Installer tasks.

## Using the Automated Installer

Now that you have the software installed on the AI server, let us now create a service for the SPARC box. Here I will, from the installadm command, tell the AI framework that I would like to create a service called **0906-111b2-sparc** that I will use to install my SPARC boxes. I choose to call the -i and -c options to setup the DHCP configuration for me. It will serve out ipaddress in the the range of 10 (**-c 10**) starting at 10.6.49.80 (**-i 10.6.49.80**). I'm using the ISO image below and the manifests will end up in **/aiserver/osol-0906-111b2-sparc.** **/aiserver** is a ZFS file system with compression turned on and atime modification turned off (these are optional setting, but will aid ZFS performance).

```
# installadm create-service -n 0906-111b2-sparc -i 10.6.49.80 -c 10 -s /iso-
images/osol-0906-111b2-ai-sparc.iso /aiserver/osol-0906-111b-sparc

Setting up the target image at /aiserver/osol-0906-111b-sparc ...
Registering the service 0906-111b2-sparc._OSInstall._tcp.local
Creating DHCP Server
Created DHCP configuration file.
Created dhcptab.
Added "Locale" macro to dhcptab.
Added server macro to dhcptab - dcsw-aiserver.
DHCP server started.
Added network macro to dhcptab - 10.6.49.0.
Created network table.
Service discovery fallback mechanism set up
Creating SPARC configuration file
```

That command above automatically sets up not only the DHCP but also the webserver and its configuration. It will serve out the installation configuration, the /tftpboot, and all other OS services where users in the past had to manually configure while setting up jumpstart. **Installadm** is a great task consolidator.

Now it could be as easy as that single command. I could now `**boot net:dhcp**` any WANboot enabled box on the 10.6.49.0 subnet and it would begin to pull down the OpenSolaris 2009.06 miniroot, install it, boot up, and then begin installing the rest of the OpenSolaris 2009.06 from the opensolaris.org/release IPS repository.

## On the SPARC Target:

```
{0} ok boot net:dhcp
Boot device: /pci@500/pci@0/pci@8/network@0:dhcp  File and args:
/pci@500/pci@0/pci@8/network@0: 1000 Mbps full duplex link up
Timed out waiting for BOOTP/DHCP reply
<time unavailable> wanboot info: WAN boot messages->console
<time unavailable> wanboot info: configuring /pci@500/pci@0/pci@8/network@0:dhcp

pci@500/pci@0/pci@8/network@0: 1000 Mbps full duplex link up
<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 May 12 18:47:43 wanboot progress: miniroot: Read 157890 of 157890 kB (100%)
Tue May 12 18:47:43 wanboot info: miniroot: Download complete
WARNING: Unexpected token '.' on line 101 of /etc/system
SunOS Release 5.11 Version snv_111b 64-bit
Copyright 1983-2009 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
Hostname: opensolaris
Remounting root read/write
Probing for device nodes ...
Preparing automated install image for use
Downloading solaris.zlib archive
--11:37:12--  http://10.6.49.27:5555/export/aiserver/osol-0906-111b2-ai-
sparc/solaris.zlib
           => `/tmp/solaris.zlib'
Connecting to 10.6.49.27:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 83,519,488 (80M) [text/plain]

100%[====================================>] 83,519,488    22.12M/s    ETA 00:00

11:37:16 (20.83 MB/s) - `/tmp/solaris.zlib' saved [83519488/83519488]

Downloading solarismisc.zlib archive
--11:37:16--  http://10.6.49.27:5555/export/aiserver/osol-0906-111b2-ai-
sparc/solarismisc.zlib
          => `/tmp/solarismisc.zlib'
Connecting to 10.6.49.27:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3,147,776 (3.0M) [text/plain]
```

```
100%[=====================================>] 3,147,776      19.63M/s

11:37:16 (19.61 MB/s) - `/tmp/solarismisc.zlib' saved [3147776/3147776]

--11:37:16--  http://10.6.49.27:5555/export/aiserver/osol-0906-111b2-ai-
sparc/install.conf
          => `/tmp/install.conf'
Connecting to 10.6.49.27:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 65 [text/plain]

100%[=====================================>] 65             --.--K/s

11:37:16 (1.64 MB/s) - `/tmp/install.conf' saved [65/65]

Done mounting automated install image
Configuring devices.
Reading ZFS config: done.
Service discovery phase initiated
Service name to look up: 0906-111b2-sparc
Service discovery over multicast DNS failed
Service located at 10.6.49.27:46501 will be used
Service discovery finished successfully
Process of obtaining configuration manifest initiated
Configuration manifest obtained
Automated Installation started
The progress of the Automated Installation can be followed by viewing the logfile
at /tmp/install_log

opensolaris console login:
```

At this point I can login to the OS and follow along with the rest of the installation as it pulls the configuration down from the AI server and packages from the IPS repositories.  I'll be able to see the alternate root mount on /a and I'll be able to see the packages installing.  I'll be able to see the user getting created.  I like to **`tail -f /var/svc/log/*auto-install*`** and **`tail -f /tmp/install_log'** to follow along.  But regardless of whether or not I'm watching these files, I will get a “Failed” or “Success” message to the console upon competition.

Key files to watch on the target node:

```
tail -f /var/svc/log/*auto-install*
tail -f /tmp/install_log
```

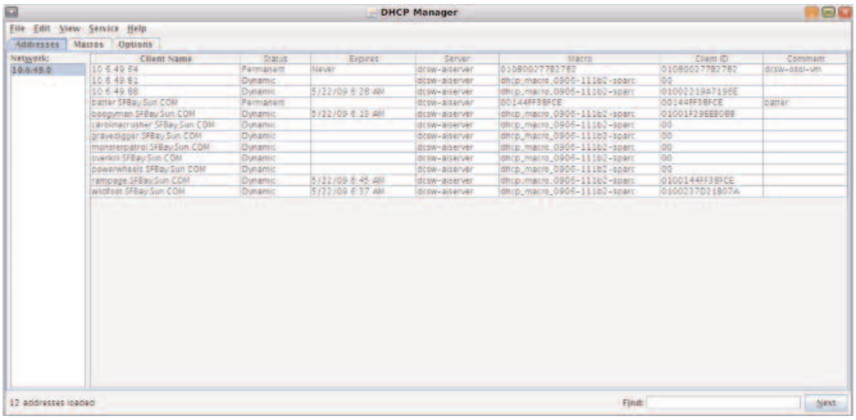
## Back On the AI Server:

On the AI server, you can observer the DHCP configuration through the dhcpmgr GUI and you can watch the webserver access and error logs in this location **/var/ai/image-server/logs.**

```
access_log
error_log
```

Once I launch dhcpmgr, I can look to see how the ipaddress allocation was assigned.  I can see which ones have been used and which MAC address are assigned to which DHCP macros.

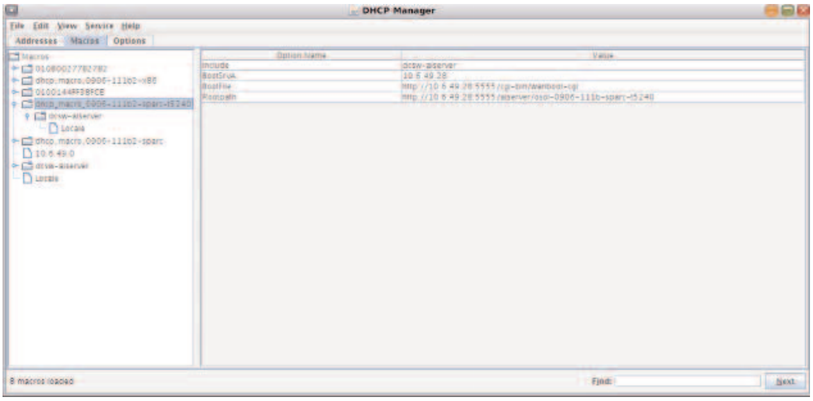
```
/usr/sbin/dhcpmgr
```



| Client Name | Status | Expires | Server | Macro | Client ID | Comment |
|---|---|---|---|---|---|---|
| 10.6.49.64 | Permanent | Never | dcsw-aiserver | 010800277B27B2 | 010800277B27B2 | dcsw-osol-vm |
| 10.6.49.81 | Dynamic | | dcsw-aiserver | dhcp_macro_0906-111b2-sparc | 00 | |
| 10.6.49.88 | Dynamic | 5/22/09 6:28 AM | dcsw-aiserver | dhcp_macro_0906-111b2-sparc | 01002219A7196E | |
| batter.SFBay.Sun.COM | Permanent | | dcsw-aiserver | 00144FF3BFCE | 00144FF3BFCE | batter |
| boogyman.SFBay.Sun.COM | Dynamic | 5/22/09 6:13 AM | dcsw-aiserver | dhcp_macro_0906-111b2-sparc | 01001F29EEB088 | |
| carolinacrusher.SFBay.Sun.COM | Dynamic | | dcsw-aiserver | dhcp_macro_0906-111b2-sparc | 00 | |
| gravedigger.SFBay.Sun.COM | Dynamic | | dcsw-aiserver | dhcp_macro_0906-111b2-sparc | 00 | |
| monsterpatrol.SFBay.Sun.COM | Dynamic | | dcsw-aiserver | dhcp_macro_0906-111b2-sparc | 00 | |
| overkill.SFBay.Sun.COM | Dynamic | | dcsw-aiserver | dhcp_macro_0906-111b2-sparc | 00 | |
| powerwheels.SFBay.Sun.COM | Dynamic | | dcsw-aiserver | dhcp_macro_0906-111b2-sparc | 00 | |
| rampage.SFBay.Sun.COM | Dynamic | 5/22/09 6:45 AM | dcsw-aiserver | dhcp_macro_0906-111b2-sparc | 0100144FF3BFCE | |
| wildfoot.SFBay.Sun.COM | Dynamic | 5/22/09 6:37 AM | dcsw-aiserver | dhcp_macro_0906-111b2-sparc | 0100237D21807A | |

12 addresses loaded

I can also click into the macros to discover the webserver file location and other useful details.



| Option Name | Value |
|---|---|
| Include | dcsw-aiserver |
| BootSrvA | 10.6.49.28 |
| BootFile | http://10.6.49.28:5555/cgi-bin/wanboot-cgi |
| Rootpath | http://10.6.49.28:5555/aiserver/osol-0906-111b-sparc-t5240 |

8 macros loaded

Now let's imagine I have a T5240 that I would like to install additional software, change the default user, etc.  I can quickly create a new service called 0906-111b2-sparc-t5240.  This time I do not specify the DHCP triggering flags (ii and -c) because DHCP is already setup.

```
# installadm create-service -n 0906-111b2-sparc-t5240  -s /iso-images/osol-0906-
111b2-ai-sparc.iso /aiserver/osol-0906-111b-sparc-t5240
Setting up the target image at /aiserver/osol-0906-111b-sparc-t5240 ...
Registering the service 0906-111b2-sparc-t5240._OSInstall._tcp.local
Service discovery fallback mechanism set up
Creating SPARC configuration file
```

Now I can either create new manifests from a text editor and associate them to this service, or I can cd into **/var/ai/<port>/AI_data** and edit the **default.xml** file. Maybe I would like to change the IPS repository for this host to /dev:

```
<ai_pkg_repo_default_authority>
        <main url="http://pkg.opensolaris.org/dev" authname="opensolaris.org"/>
```

Or maybe I would like to change the default user from jack to mike:

```
<property_group name="ai" type="application">
         <propval name="username" type="astring" value="mike"/>
         <propval name="userpass" type="astring" value="9Nd/cwBcNWFZg"/>
         <propval name="description" type="astring" value="default_user"/>
         <propval name="rootpass" type="astring"
         value="$5$VgppCOxA$ycFmYW4ObRRHhtsGEygDdexk5bugqgSiaSR9niNCouC"
         <propval name="timezone" type="astring" value="US/Eastern"/>
         </property_group>
```

Now I would like to tell Automated Installer to only serve that specific service out to one one of my hosts. Here I will use the MAC address of the T5240 NIC port 0 as the identifier:

```
# installadm create-client -e 0:14:4f:f3:8f:ce -n 0906-111b2-sparc-t5240 -t
/aiserver/osol-0906-111b-sparc-t5240
Setting up SPARC client...
Creating SPARC configuration file
Enabled network boot by adding a macro named 0100144FF38FCE
to DHCP server with:
  Boot server IP      (BootSrvA) : 10.6.49.28
  Boot file           (BootFile) : http://10.6.49.28:5555/cgi-bin/wanboot-cgi
  Root path           (Rootpath) : http://10.6.49.28:5555/aiserver/osol-0906-111b-
sparc-t5240
```

Notice above when it created the new DHCP macro for the client, it appended 01 in front of the MAC address.

## AI Server Tasks for the x86 Target:

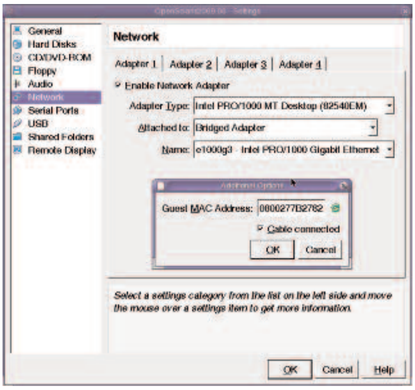
Next I would like to create a service for my x86 host. Here I will issue the same commands as before but with a new ISO and service name. Still no need to call the DHCP flags (-i and -c).

```
installadm create-service -n 0906-111b2-x86 -s /iso-images/osol-0906-111b2-ai-
x86.iso /aiserver/osol-0906-111b-x86
Setting up the target image at /aiserver/osol-0906-111b-x86 ...
Registering the service 0906-111b2-x86._OSInstall._tcp.local
adding tftp to /etc/inetd.conf
Converting /etc/inetd.conf
copying boot file to /tftpboot/pxegrub.I86PC.OpenSolaris-1
Service discovery fallback mechanism set up
```

Now we have two architectural images (SPARC and Intel/AMD) in the same DHCP configuration being served out to anything that can WANboots or PXE boots on the 10.6.49.0 subnet. This configuration will not work. When the x86 target boots, it will get served out to the SPARC image. I need to either connect the specific x86target to the service name like I did above with the **installadm create-client** command, or I can leverage the DHCP configuration directly. In order to show both ways, I will use the DHCP method this time. Here I will call the **pntadm** command and connect the MAC address of my VirtualBox guest to the correct x86 macro name. This will also assure that the guest receives a static ipaddress:

```
pntadm -A 10.6.49.64 -c dcsw-osol-vm -f PERMANENT -i 010800277B2782 -m  osol-
0906-111b-x86 10.6.49.0
```

Note I found the MAC address of the VirtualBox guest via this screen menu choice in VirtualBox:



Network

Enable Network Adapter

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Attached to: Bridged Adapter

Name: e1000g3 - Intel PRO/1000 Gigabit Ethernet

Guest MAC Address: 0800277B2782

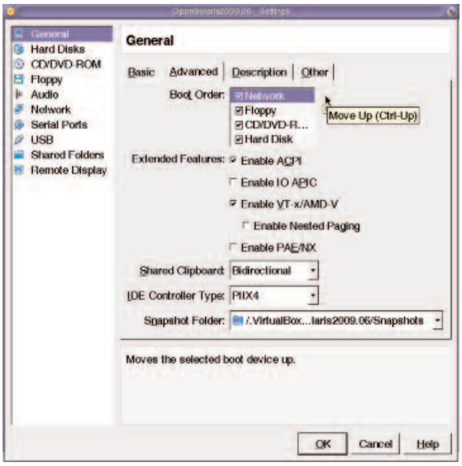
Cable connected

*Select a settings category from the list on the left side and move the mouse over a settings item to get more information.*

Another way around this issue is to modify the DHCP macros the **installadm** command created for me and add an option field declaring the architecture.

## On the x86 Target:

Since I choose to use a VirtualBox guest as my x86target, I must tell VirtualBox that I would like the guest to boot from the network via PXEboot. I can do that from this menu location:



General

Boot Order: ☑ Network, ☑ Floppy, ☑ CD/DVD-R..., ☑ Hard Disk

Extended Features: ☑ Enable ACPI; ☐ Enable IO APIC; ☑ Enable VT-x/AMD-V; ☐ Enable Nested Paging; ☐ Enable PAE/NX

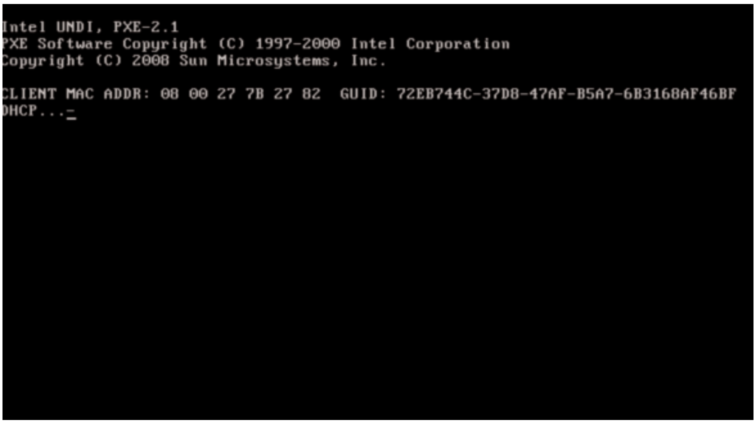
Shared Clipboard: Bidirectional

IDE Controller Type: PIIX4

Snapshot Folder: /.VirtualBox...laris2009.06/Snapshots

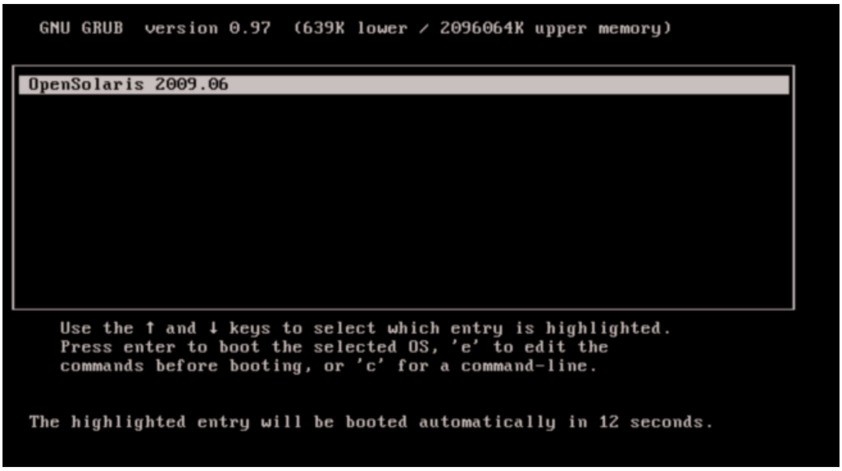
Moves the selected boot device up.

Once the guest boots, I will see it advertise for a DHCP service:



```
Intel UNDI, PXE-2.1
PXE Software Copyright (C) 1997-2000 Intel Corporation
Copyright (C) 2008 Sun Microsystems, Inc.

CLIENT MAC ADDR: 08 00 27 7B 27 82  GUID: 72EB744C-37D8-47AF-B5A7-6B3168AF46BF
DHCP..._
```

It will then find my AI server and be given the AI miniroot to boot from:

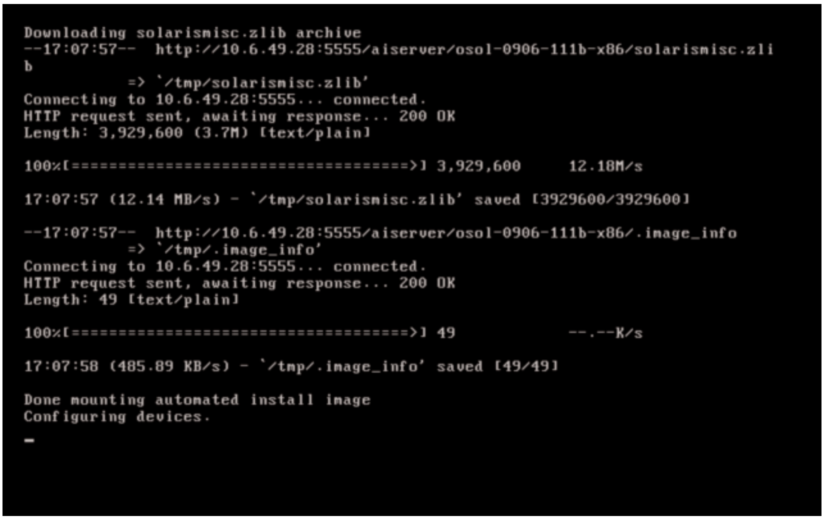


```
 GNU GRUB  version 0.97  (639K lower / 2096064K upper memory)

 OpenSolaris 2009.06

    Use the ↑ and ↓ 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.

 The highlighted entry will be booted automatically in 12 seconds.
```

Next it will download the miniroot and boot back up from it:

```
..................................._
```

Finally, it will begin to download the rest of it's IPS packages from the assigned IPS repository.



```
Downloading solarismisc.zlib archive
--17:07:57--  http://10.6.49.28:5555/aiserver/osol-0906-111b-x86/solarismisc.zli
b
           => `/tmp/solarismisc.zlib'
Connecting to 10.6.49.28:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3,929,600 (3.7M) [text/plain]

100%[====================================>] 3,929,600     12.18M/s

17:07:57 (12.14 MB/s) - `/tmp/solarismisc.zlib' saved [3929600/3929600]

--17:07:57--  http://10.6.49.28:5555/aiserver/osol-0906-111b-x86/.image_info
           => `/tmp/.image_info'
Connecting to 10.6.49.28:5555... connected.
HTTP request sent, awaiting response... 200 OK
Length: 49 [text/plain]

100%[====================================>] 49            --.--K/s

17:07:58 (485.89 KB/s) - `/tmp/.image_info' saved [49/49]

Done mounting automated install image
Configuring devices.
-
```

After that, the key files to observe and overall characteristic of it's installation is as same as the SPARC target discussed earlier in the guide.

## Command Summary

All I needed to do was the following:

### AI Server Tasks for the x86 Target

```
# pkg install SUNWinstalladm-tools
# installadm create-service -n 0906-111b2-sparc -i 10.6.49.80 -c 10 -s /iso-
images/osol-0906-111b2-ai-sparc.iso /aiserver/osol-0906-111b-sparc
# installadm create-service -n 0906-111b2-sparc-t5240  -s /iso-images/osol-0906-
111b2-ai-sparc.iso /aiserver/osol-0906-111b-sparc-t5240
# installadm create-client -e 0:14:4f:f3:8f:ce -n 0906-111b2-sparc-t5240 -t
/aiserver/osol-0906-111b-sparc-t5240
# installadm create-service -n 0906-111b2-x86 -s /iso-images/osol-0906-111b2-ai-
x86.iso /aiserver/osol-0906-111b-x86
# pntadm -A 10.6.49.64 -c dcsw-osol-vm -f PERMANENT -i 010800277B2782 -m  osol-
0906-111b-x86 10.6.49.0
```

### On the SPARC Target

```
# # ssh <service-processor> -l root
Password:
--> cd /SP/console
--> start

ok boot net:dhcp
```

### On the x86 Target:

I used a VirtualBox guest, but if you were on a physical server you could have also done the following to enable PXE boot in the boot order.

```
# ipmitool -H service-processor -U username chassis bootdev pxe
Password:
Set boot Device to pxe
# ipmitool -H service-processor -U username chassis power reset
Password:
Chassis Power Control: Reset
```

## For More Information

For more information about configuring the Automated Installer and OpenSolaris, see the following manuals and community resources:

| Description | |
|---|---|
| OpenSolaris Automated Installer Guide | http://dlc.sun.com/osol/docs/content/dev/AIinstall/ |
| OpenSolaris Installation and Packaging community | http://opensolaris.org/os/community/install/ |
| OpenSolaris Automated Installion Project | http://opensolaris.org/os/project/caiman/auto_install/ |
| OpenSolaris Home Page | http://www.opensolaris.com/ |

opensolaris.com

**Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA Phone 1-650-960-1300 or 1-800-555-9SUN Web sun.com**