Booting and Shutting Down Oracle® Solaris 11.4 Systems

Part No: E60978
November 2020

ORACLE®

Booting and Shutting Down Oracle Solaris 11.4 Systems

Part No: E60978

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Référence: E60978

Restrictions de licence/Avis d'exclusion de responsabilité en cas de dommage indirect et/ou consécutif

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf stipulation expresse de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, accorder de licence, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Exonération de garantie

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Avis sur la limitation des droits

Si ce logiciel, ou la documentation qui l'accompagne, est livré sous licence au Gouvernement des Etats-Unis, ou à quiconque qui aurait souscrit la licence de ce logiciel pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique :

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

Avis sur les applications dangereuses

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est pas conçu ni n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer un risque de dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour des applications dangereuses.

Marques

Oracle et Java sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d'autres propriétaires qu'Oracle.

Intel et Intel Inside sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Epyc, et le logo AMD sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée de The Open Group.

Avis d'exclusion de responsabilité concernant les services, produits et contenu tiers

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers, sauf mention contraire stipulée dans un contrat entre vous et Oracle. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation, sauf mention contraire stipulée dans un contrat entre vous et Oracle.

Date de publication et mention de la version préliminaire de Disponibilité Générale ("Pre-GA")

Version préliminaire de Disponibilité Générale ("Pre-GA") : 15.01.2020

Avis sur la version préliminaire de Disponibilité Générale ("Pre-GA") de la documentation

Si ce document est fourni dans la Version préliminaire de Disponibilité Générale ("Pre-GA") à caractère public ou privé :

Cette documentation est fournie dans la Version préliminaire de Disponibilité Générale ("Pre-GA") et uniquement à des fins de démonstration et d'usage à titre préliminaire de la version finale. Celle-ci n'est pas toujours spécifique du matériel informatique sur lequel vous utilisez ce logiciel. Oracle Corporation et ses affiliés déclinent expressément toute responsabilité ou garantie expresse quant au contenu de cette documentation. Oracle Corporation et ses affiliés ne sauraient en aucun cas être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'utilisation de cette documentation.

Mention sur les informations confidentielles Oracle

INFORMATIONS CONFIDENTIELLES ORACLE. Destinées uniquement à un usage autorisé. Ne pas distribuer à des tiers.

Avis sur la reconnaissance du revenu

Si ce document est fourni dans la Version préliminaire de Disponibilité Générale ("Pre-GA") à caractère privé :

Les informations contenues dans ce document sont fournies à titre informatif uniquement et doivent être prises en compte en votre qualité de membre du customer advisory board ou conformément à votre contrat d'essai de Version préliminaire de Disponibilité Générale ("Pre-GA") uniquement. Ce document ne constitue en aucun cas un engagement à fournir des composants, du code ou des fonctionnalités et ne doit pas être retenu comme base d'une quelconque décision d'achat. Le développement, la commercialisation et la mise à disposition des fonctions ou fonctionnalités décrites restent à la seule discrétion d'Oracle.

Accessibilité de la documentation

Pour plus d'informations sur l'engagement d'Oracle pour l'accessibilité de la documentation, visitez le site Web Oracle Accessibility Program, à l'adresse : http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Accès aux services de support Oracle

Les clients Oracle qui ont souscrit un contrat de support ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info ou le site http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs si vous êtes malentendant.

Contents

Using This Documentation

- **Overview** – Describes how to boot and shut down a system
- **Audience** – Technicians, system administrators, and authorized service providers
- **Required knowledge** – Experience administering an Oracle Solaris system

Product Documentation Library

Documentation and resources for this product and related products are available at http://www.oracle.com/pls/topic/lookup?ctx=E37838-01.

Feedback

Provide feedback about this documentation at http://www.oracle.com/goto/docfeedback.

♦ ♦ ♦ CHAPTER 1

Overview of Booting and Shutting Down a System

This book describes boot and shutdown processes on physical machines running single Oracle Solaris instances. The information generally applies to both SPARC and x86 platforms, unless specifically stated otherwise.

This chapter contains the following information:

- "Booting and Shutdown Features in Oracle Solaris"
- "Using Rights Profiles to Administer Boot Features"
- "Service Management Facility and Booting"

Refer also to the following documentation:

- For systems that have service processors or multiple physical domains, boot and shutdown information can be found in their respective product documentation at `https://www.oracle.com/products/oracle-a-z.html`.
- For booting operations that are connected to installations, refer to the following guides:
  - *Manually Installing an Oracle Solaris 11.4 System*
  - *Automatically Installing Oracle Solaris 11.4 Systems*
- For an overview of the boot architecture and boot process in Oracle Solaris, see "Overview of the Oracle Solaris Boot Architecture" in *Booting and Shutting Down Oracle Solaris 11.3 Systems*.

Booting and Shutdown Features in Oracle Solaris

The following are general features of Oracle Solaris booting and shutdown operations:

- On Oracle Solaris x86 platforms, UEFI secure boot combines with the Oracle Solaris Verified Boot feature which confirms signatures on kernel module loads. Together, they provide an end to end chain of trust when you boot the system.

- WAN boot is supported on both SPARC and x86 systems that have UEFI enabled. WAN boot enables you to boot and install software over a wide area network.
- The boot process accommodates different types of devices for booting, such as firmware-inaccessible storage devices, where the boot pool is separate from the root pool. In addition, on some servers whose boot pool might be unavailable, you can boot the system from fallback images with which these servers are preconfigured.

Using Rights Profiles to Administer Boot Features

Oracle Solaris implements role-based access control (RBAC) to control system access. To perform shutdown or booting operations as well as administering booting features, you must be assigned at a minimum the Maintenance and Repair profile. Other profiles are required if you need to perform additional tasks indirectly related to maintenance and repair, such as creating archives or installing software.

An administrator that has the `solaris.delegate.*` authorization can assign the required profiles to users.

For example, an administrator assigns the Maintenance and Repair profile to user `jdoe`. Before `jdoe` executes a privileged boot administration command, `jdoe` must be in a profile shell. The shell can be created by issuing the `pfbash` command. Or, `jdoe` can combine `pfexec` with every privileged command that is issued, for example, `pfexec bootadm` or `pfexec shutdown`.

As an alternative, instead assigning profiles directly to users, a system administrator can create a role that would contain a combination of required profiles to perform a range of tasks.

Suppose that a role `repairua` is created with both the Maintenance and Repair profile and the Unified Archive Administration profile. As an authorized user, `jdoe` uses the `su` command to assume that role. All roles automatically get `pfbash` as the default shell.

For more information about rights profiles, see "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

Service Management Facility and Booting

The Oracle Solaris Service Management Facility (SMF) augments the traditional UNIX startup scripts, init run levels, and configuration files. With SMF serving as an infrastructure to manage booting, the mechanism behind the boot process features the following:

- Boot information is stored in the service's log file in `/var/svc/log`. To display boot messages onscreen, type the `boot -v` command syntax.

- Services are typically restarted if they are terminated. If the service is defective, the service is placed in maintenance mode. Use the `svcadm` command to stop or start SMF processes.
- Many of the scripts in `/etc/init.d` and `/etc/rc*.d` as well as entries in `/etc/inittab` have been removed. Their functionalities are managed by SMF. Scripts and `inittab` entries from ISVs or are locally developed will continue to run. However, in the boot process, these run after the SMF services.
- You can specify SMF milestones when you boot a system. Milestones correspond to boot run levels and are useful for performing administrative tasks that pertain to booting. See "About Run Level Booting" on page 44

♦ ♦ ♦ CHAPTER 2

x86: Administering the GRand Unified Bootloader

This chapter discuses GRUB 2, the default system boot loader for x86 systems since Oracle Solaris 11.2. All information applies to x86 systems only, unless specified otherwise.

The chapter covers the following topics:

- "Administering the GRUB Menu"
- "Administering Contents of the GRUB Menu"
- "Adding Kernel Arguments at Boot Time"
- "Customizing the GRUB Configuration"
- "Installing GRUB 2"

For any task that involves GRUB Legacy, refer to *Booting and Shutting Down Oracle Solaris 11.3 Systems*. The section "Introducing GRUB 2" in that guide also provides an overview of GRUB 2.

Note - To manage GRUB and issue commands described in this chapter, you must have the appropriate rights profiles. See "Using Rights Profiles to Administer Boot Features" on page 14.

Administering the GRUB Menu

All GRUB 2 configuration information is stored in the `grub.cfg` file. **Never** directly edit `grub.cfg`, whose contents can be overwritten by other commands. To configure GRUB2, use `bootadm` subcommands instead.

All `bootadm` subcommands for managing GRUB apply only to x86 systems. However, certain `bootadm` subcommands can be used on SPARC based systems. See, for example, "Managing the Oracle Solaris Boot Archives" on page 67.

The `bootadm` command has the following subcommands:

`add-entry`	Adds a boot entry to the GRUB menu.
`change-entry`	Changes the attributes of a specified boot entry in the GRUB menu.
`generate-menu`	Generates a new boot loader configuration file.
`install-bootloader`	Installs the system boot loader. This subcommand applies to both x86 and SPARC platforms.
`list-menu`	Displays the current boot entries in the GRUB menu.
`remove-entry`	Removes a boot entry from the GRUB menu.
`set-menu`	Maintains the GRUB menu. You can use this subcommand to set a particular GRUB menu entry as the default, to add security protection to the GRUB menu, and to set other menu options and boot loader options. The `-P` option supports changing menus on multiple root pools.
`set-menu-password`	Sets a password to prevent the GRUB menu from being seen.
`show-entry`	Shows a boot entry from the GRUB menu. This subcommand is equivalent to `list-menu`.

For more details, see the `bootadm`(8) man page.

Password-Protecting the GRUB Menu

To control access to the GRUB menu, use `set-menu-password`. The command supports the following options:

- `-s` *password* sets the password for accessing the GRUB menu entries.
- `-r` removes the password lock.
- `-l` verifies whether a password lock is in place and which users have access to each menu entry.

Caution - If the default boot entry is locked, the system would require a password in order to boot. To enable system boots without manual intervention, do not set a password lock.

Authorizing Users to Access the GRUB Menu

To grant specific users access to the GRUB menu, follow these steps:

1. Add users to the authenticated users list.

   ```
   $ bootadm set-menu add-user=username
   ```

   Note - The password used to authenticate access to the GRUB menu is not the same password used by the OS when it is booted.

2. Grant access to specific users.
   - To grant access to the whole GRUB menu:

     ```
     $ bootadm set-menu add-superuser username
     ```
   - To grant access to a specific menu entry:

     ```
     $ bootadm change-entry -i menu-number add-auth=username
     ```

     The *menu-number* corresponds to the menu entry to which access is granted.

Displaying GRUB Menu Entries

To display a system's menu entries, use `list-menu` or `show-entry`.

Used by itself, `list-menu` displays the entire GRUB menu, including the location of the boot loader configuration files and other information.

```
$ bootadm list-menu
the location of the boot loader configuration files is: /rpool/boot/grub
default 0
console graphics
timeout 30
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
2 Oracle Solaris 11.4
```

To obtain information about a specific menu entry, identify the entry with is menu number.

```
$ bootadm list-menu -i 0
     the location of the boot loader configuration files is: /rpool/boot/grub
     title: Oracle Solaris 11 11/11
     kernel: /platform/i86pc/kernel/$ISADIR/unix
```

```
     kernel arguments: -B $ZFS-BOOTFS -v
     boot archive: /platform/i86pc/$ISADIR/boot_archive
     ZFS root pool: rpool
```

The `show-entry` command shows only entry-specific information.

```
$ bootadm show-entry -i 0
     title: Oracle Solaris 11 11/11
     kernel: /platform/i86pc/kernel/$ISADIR/unix
     kernel arguments: -B $ZFS-BOOTFS -v
     boot archive: /platform/i86pc/$ISADIR/boot_archive
     ZFS root pool: rpool
```

Generating the GRUB Menu

To generate a `grub.cfg` file, use `generate-menu`. The file would contain the OS instances currently installed on a system.

```
$ bootadm generate-menu
```

Options direct the way the command is implemented:

- `-f` overwrites an existing `grub.cfg` file.
- `-P` *pool-name* generates a new GRUB menu for a different root pool than the current one.

Note - Verify your changes with the `list-menu` subcommand or by viewing the contents of `grub.cfg`.

Administering Contents of the GRUB Menu

Two commands are available to configure GRUB menu contents: `set-menu` and `change-entry`.

The following sections provide examples of how to use these commands. The examples use the menu listing in “Displaying GRUB Menu Entries” on page 19 as the starting menu on which the changes are implemented.

After making any configuration changes, verify the results with the `list-menu` command or by viewing the contents of `grub.cfg`. In some cases, a system reboot is required for the changes to take effect.

Using the `set-menu` Command

The `set-menu` subcommand changes GRUB menu configuration such as the timeout or the default boot entry:

```
$ bootadm set-menu [-P pool] [-R altroot [-p platform]] argument
```

The *argument* variable consists of one or a series of *key=value* pairs.

EXAMPLE 1 Changing the Default Boot Entry in the GRUB Menu

This example changes the default boot entry to Oracle Solaris 11.4.

```
$ bootadm set-menu default=2

$ bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 30
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
2 Oracle Solaris 11.4
```

The `change-entry` subcommand performs the same function. See "Using the `change-entry` Command" on page 22.

EXAMPLE 2 Changing the Menu Timeout Value in the GRUB Menu

This example changes the `timeout` from 30 seconds to 45 seconds.

```
$ bootadm set-menu timeout=45

$ bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 45
0 Oracle Solaris 11/11
1 Oracle Solaris 11.3
2 Oracle Solaris 11.4
```

EXAMPLE 3 Setting the GRUB Console Type

This example sets the console type to `serial`.

```
$ bootadm set-menu console=serial
```

Other than serial type, `text` and `graphics` console types are also supported. Specify `text` if you are using BIOS serial redirection.

For a serial type console, you can set the following parameters through the `serial_params` argument:

port	Port number from `0` to `3`. Typically, `0` is used for `ttya` or `COM1`, `1` for `ttyb` or `COM2`, and so on.
speed	Speed that the serial port uses. Without a value specified, GRUB 2 uses the speed that the serial port has been initialized to use. But if the serial port has not been initialized, an undetermined speed might cause unpredictable output. To be safe, always provide a speed value.
data bits	Either `7` or `8`.
parity	Can be `e` (even), `o` (odd), or `n` (none).
stop bits	Either `0` or `1`.

All of the serial parameters, with the exception of `port`, are optional.

Using the `change-entry` Command

The `change-entry` subcommand sets certain boot attributes for one or more boot entries in the GRUB menu. If multiple entries have the same title, all of the entries are affected.

A special attribute, `set-default`, sets the default entry to boot from when the timer expires. This attribute has the same function as the `set-menu default=`*value* subcommand. See Example 1, “Changing the Default Boot Entry in the GRUB Menu,” on page 21.

Use the following command syntax:

```
$ bootadm change-entry [-P pool] {[entry-title[,entry-title...]}]
| -i entry-number[,entry-number]...]} { key=value [ key=value ...]
| set-default }
```

EXAMPLE 4 Setting the Title for a Boot Entry in the GRUB Menu

This example shows how to set the title for a specified boot entry. If multiple entries have the same title, all of the entries are affected.

```
$ bootadm change-entry -i 1 title="Oracle Solaris 11-backup1"

$ bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 45
0 Oracle Solaris 11/11
1 Oracle Solaris 11-backup1
2 Oracle Solaris 11.4
```

EXAMPLE 5 Changing a Boot Entry by Specifying Kernel Arguments

Kernel boot settings for a menu entry are set through the `kargs` argument. Specify multiple settings inside quotes.

This example sets boot entry number 2 both to boot in single-user mode and to display messages in verbose mode.

```
$ bootadm change-entry -i 2 kargs="-v -s"
$ bootadm list-menu -i 2
The location of the boot loader configuration files is: /rpool/boot/grub
     title: Oracle Solaris 11.4
     kernel: /platform/i86pc/kernel/$ISADIR/unix
     kernel arguments: -v -s
     boot archive: /platform/i86pc/$ISADIR/boot_archive
     ZFS root pool: rpool
```

EXAMPLE 6 Removing Kernel Arguments From a Boot Entry

This example assigns a null value to `kargs` to remove the previous settings:

```
$ bootadm change-entry -i 1 kargs=
$ bootadm list-menu -i 1
the location of the boot loader configuration files is: /rpool/boot/grub
title: Oracle Solaris 11-backup1
kernel: /platform/i86pc/kernel/amd64/unix
kernel arguments:
boot archive: /platform/i86pc/amd64/boot_archive
ZFS root pool: rpool/ROOT/s11.3.backup
```

Adding and Removing Menu Entries

You can expand or reduce the GRUB menu on a system by adding or removing entries. Adding an entry assumes that the entry's OS instance is installed on the system.

▼ How to Add a Boot Entry to the GRUB Menu

The `add-entry` subcommand creates a new entry to the GRUB menu. If you specify an entry number greater than the total entries, the new entry is added at the end of the list. Otherwise, it is inserted at the specified position in the menu.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **(Optional) List the current menu entries.**

   ```
   $ bootadm list-menu
   ```

2. **Add the new entry.**

   ```
   $ bootadm add-entry -P pool -i [entry-number] entry-title
   ```

3. **Set the `bootfs` property for the newly added entry.**

   ```
   $ bootadm change-entry -i new-entry-number bootfs='pool-name/ROOT/be-name'
   ```

   This step ensures that the new entry does not use the default `bootfs` value that is set for the root pool.

4. **Verify that the boot entry was added.**

   ```
   $ bootadm list-menu
   ```

Note - If you do not see your changes, check the `grub.cfg` file to verify that the change was made.

Example 7 Adding a Linux Entry to the GRUB Menu

The following example shows how to add a menu entry to the GRUB menu by using the `bootadm add-entry` command. In this example, entry number 2 is added.

```
$ bootadm list-menu
```

```
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 45
0 Oracle Solaris 11/11
1 Oracle Solaris 11-backup1
2 Oracle Solaris 11.4

$ bootadm add-entry -i 3 Oracle Linux
$ bootadm change-entry -i 3 bootfs='rpool/ROOT/test'
$ bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 45
0 Oracle Solaris 11/11
1 Oracle Solaris 11-backup1
2 Oracle Solaris 11.4
3 Oracle Linux

$ bootadm list-menu -i 3
The location of the boot loader configuration files is: /rpool/boot/grub
title: Oracle Linux
kernel: /platform/i86pc/kernel/amd64/unix
kernel arguments: -B $ZFS-BOOTFS
boot archive: /platform/i86pc/amd64/boot_archive
ZFS root pool: rpool
bootfs: /rpool/ROOT/test
```

▼ How to Remove a Boot Entry From the GRUB Menu

The `remove-entry` subcommand removes an entry, or a comma-separated list of entries, from the menu. If you specify multiple entries with the same title, all of the entries with that title are removed.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **(Optional) List the current boot entries.**

   ```
   $ bootadm list-menu
   ```

2. **Remove the specified entry from the GRUB menu.**

   ```
   $ bootadm remove-entry [-P pool] [{entry-title [,entry-title...] |
   -i entry-number[,entry-number...]}
   ```

3. **Verify that the entry has been removed.**

```
$ bootadm list-menu
```

Note - If you do not see your changes, check the `grub.cfg` file to verify that the change was made.

Example 8 Removing a Boot Entry From the GRUB Menu

The following example shows the removal of entry number 2 from the GRUB menu.

```
$ bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 45
0 Oracle Solaris 11/11
1 Oracle Solaris 11-backup1
2 Oracle Solaris 11.4
3 Oracle Linux

bootadm remove-entry -i 0
1 entry removed
$ bootadm list-menu
The location of the boot loader configuration file is /rpool/boot/grub
default 2
console graphics
timeout 45
1 Oracle Solaris 11-backup1
2 Oracle Solaris 11.4
3 Oracle Linux
```

Adding Kernel Arguments at Boot Time

On x86 platforms, you can set boot attributes and kernel arguments at boot time. These changes persist until the next time the system is booted.

Follow these general steps:

1. Boot the system.

```
$ reboot -p
```

   The GRUB menu is displayed.

2. Edit your selected entry.
3. Exit the edit menu to proceed with the boot process.

To edit a specific menu entry, select the entry, then type `e` to edit it. On the GRUB edit screen, navigate to the `$multiboot` line, then type the additional boot option or kernel argument at the end of the line.

A `$multiboot` line with added arguments might resemble this example:

`$multiboot /ROOT/transition/@/$kern $kern -B console=graphics -B $zfs_bootfs` *added-arguments*

Note - When parameters are specified by using the `eeprom` utility *and* on the GRUB command line, the GRUB command line settings take precedence.

Supported Kernel Arguments

The following kernel arguments and options can be specified when you edit the GRUB menu at boot time:

`unix`	Specifies the kernel to boot.
`-a`	Prompts the user for configuration information.
`-i altinit`	Specifies an alternative executable as the primordial process. `altinit` is a valid path to an executable.
`-k`	Boots the system with the kernel debugger enabled
`-m` *smf-options*	Controls the boot behavior of the Service Management Facility (SMF) There are two categories of options: recovery options and messages options.
`-r`	Specifies a reconfiguration boot. The system probes all attached hardware devices and then assigns nodes in the file system to represent only those devices that are actually found.
`-s`	Boots the system to a single-user state.
`-v`	Boots the system with verbose messages enabled.

For example, to load the kernel debugger (`kmdb`) at boot time, you would edit the `$multiboot` line as follows. The change is shown in bold.

```
$multiboot /ROOT/transition/@/$kern $kern -B console=graphics -B $zfs_bootfs -k
```

For more information, see the `kernel`(8) man page.

Adding EEPROM Parameters

Aside from kernel arguments, you can pass EEPROM parameters to the kernel to be used at boot time. You must specify these with the `-B` option. Separate multiple property values with a comma.

`acpi-user-options`	Determines whether ACPI will be used or not. By default, ACPI is used.
`console`	Specifies the console device.
`screen-`*#columns* `screen-`*#rows*	If the console is text or graphics type, this parameter sets number of columns and rows of text.

For example, to disable the use of ACPI at boot time, you would edit the `$multiboot` line as follows. The change is shown in bold.

```
$multiboot /ROOT/transition/@/$kern $kern -B console=graphics -B $zfs_bootfs -B acpi-user-options=0x2
```

For accepted values you can specify for these parameters, see the `eeprom`(8) man page.

Customizing the GRUB Configuration

The `grub.cfg` file contains most of the GRUB configuration. To add more complex constructs to the GRUB configuration, create a */pool-name*`/boot/grub/custom.cfg` file to contain these customizations. This file shares the same location as the `grub.cfg` and `menu.conf` files. GRUB then processes the contents of the `custom.cfg` file through code at the end of the `grub.cfg` file.

On a system with 64-bit UEFI firmware, entries in the `custom.cfg` file might look like the following:

```
menuentry "Windows (64-bit UEFI)" {
```

```
   insmod part_gpt
   insmod fat
   insmod search_fs_uuid
   insmod chain
   search --fs-uuid --no-floppy --set=root cafe-f4ee
   chainloader /efi/Microsoft/Boot/bootmgfw.efi
}
```

On a system with BIOS firmware, entries might look like the following:

```
menuentry "Windows" {
   insmod chain
   set root=(hd0,msdos1)
   chainloader --force +1
}
```

Installing GRUB 2

This section discusses procedures for installing GRUB 2 under specific circumstances.

▼ How to Install the Boot Loader

If the boot loader becomes corrupted and the system can no longer boot, you would need to reinstall the boot loader by following these steps.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Boot the system from the Oracle Solaris media.**

2. **Import the root pool.**

   ```
   $ zpool import -f pool-name
   ```

3. **Install the boot loader.**

   ```
   $ bootadm install-bootloader [-f] -P pool-name
   ```

   `-f` Forces the installation of the boot loader and bypasses any version checks.

Note - Do *not* use the `-f` option unless you are sure that you want to overwrite the boot loader with the version that is on the media.

`-P` Specifies the boot configuration for the pool to be used.

4. **Export the root pool.**

   `$ zpool export` *pool-name*

5. **Reboot the system.**

How to Install GRUB in a Location Other Than the Default Location

On systems with BIOS firmware, sometimes installing GRUB 2 into the master boot record might be necessary. After the installation, GRUB 2 is then the default system boot loader, regardless of which DOS partition is marked as the active partition. When DOS partitioning is used on systems with BIOS firmware, and the Solaris partition is a primary partition, the default GRUB 2 installation location is the partition boot record. If the partition is a logical partition, GRUB 2 is always installed in the MBR.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Assume the `root` role.**

   See "Using Your Assigned Administrative Rights" in *Securing Users and Processes in Oracle Solaris 11.4*.

2. **Install the boot loader into the MBR location.**

   `$ bootadm install-bootloader -M`

3. **Reboot the system.**

♦ ♦ ♦ CHAPTER 3

Shutting Down a System

This chapter describes shutdown processes of Oracle Solaris systems. Unless stated otherwise, the information applies to both SPARC and x86 platforms.

The chapter covers the following topics:

- "About the System Shutdown Processes"
- "Turning Off Power to System Devices"

Note - To shut down or boot systems and issue commands described in this chapter, you must have the appropriate rights profiles. See "Using Rights Profiles to Administer Boot Features" on page 14.

About the System Shutdown Processes

Oracle Solaris is designed to run continuously so that the electronic mail and network software can work correctly. However, if required, you can shut these systems down to intermediate levels where only some system services are available, or to a level where you can safely turn the power off.

For information about using your system's power management features, see the `poweradm`(8) man page.

To shut a system down, one of two commands are typically used: `shutdown` or `init`. Both start processes that write all file system changes to disk and terminate all system services, processes, and the OS. System services managed by SMF are shutdown in reverse dependency order.

- `shutdown` is typically used on systems running in the multiuser state. If used without options, the command brings the systems to run level S (single-user) by default.
- `init` is best used on stand-alone systems where other users are not affected by any shutdown. It does not send out notifications about the pending shutdown and completes the shutdown process faster.

Other shutdown-related commands such as `reboot`, `halt`, or `poweroff` are also available. However, these are not the preferred commands to use under normal situations. Avoid using the system's Stop key sequence as well. These methods do not perform clean shutdowns and should be used only as a last resort. Refer to the commands' respective man pages for more information.

For information about shutting down a system for recovery purposes, including using the `halt` command, see "How to Stop a System for Recovery Purposes" on page 70.

You can choose the run level to which the system is shut down and from which you can perform additional actions. For more details, see "How Run Levels Work" on page 44.

▼ How to Shut Down a System

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. With a system with multiple users, check if any users are logged in.

```
$ who
holly       console      May  7 07:30
kryten      pts/0        May  7 07:35   (starlite)
lister      pts/1        May  7 07:40   (bluemidget)
```

The command displays users, their corresponding terminal lines, log in dates and times, and hostnames, if users are remotely logged in.

2. Choose one of the following steps:

- **Use the `shutdown` command:**

  ```
  $ shutdown -iinit-state -ggrace-period -y
  ```

-i*init-state*	Specifies the system's state as a result of the shutdown. The choices are 0, 1, S or s, 5, and 6. Run levels 0 and 5 are reserved for shutting the system down. Run level 6 reboots the system. Run level 2 is available as a multiuser operating state.
-g*grace-period*	Indicates a time in seconds before the system is shut down. The default period is 60 seconds.

-y	Shuts down the system without further prompts after the grace period is reached. If you use this option, then skip to Step 4.

For more information, see the `shutdown`(8) man page.

- **Use the `init` command:**

  ```
  $ init [options]
  ```

  For *options*, you would typically specify a run level, for example `init 5`. Other options besides run levels are also supported. See the `init`(8) man page.

3. **If prompted, type `y` to proceed with the shutdown.**
4. **If prompted, type the `root` password.**
5. **After you have finished performing any system administration tasks, press Control-D to reboot to the default run level.**

Note - By default, the shutdown process the system to the single-user state (S). However, the boot process by default brings the system to the multiuser level (3).

6. **(Optional) Verify that the system is at the run level that you specified in the `shutdown` command.**

Example 9 Bringing a System to a Single-User State

The `shutdown` command is used to bring a system to run level S (the single-user state) after 3 minutes of issuing the command.

```
$ who
root     console      Apr 15 06:20

$ shutdown -g180 -y
Shutdown started.    Fri Apr 15 06:20:45 MDT 2015
.
Broadcast Message from root (console) on portia Fri Apr 15 06:20:46...
The system hostname will be shut down in 3 minutes
.
.
SINGLE USER MODE
.
Enter user name for system maintenance (control-d to bypass):xxxxxx
```

Example 10 Bringing a System to a Shutdown State

The `shutdown` command is used to bring a system to run level 0 in five minutes without requiring additional confirmation.

```
$ who
root       console       Jun 17 12:39...
userabc   pts/4        Jun 17 12:39   (:0.0)

$ shutdown -i0 -g300 -y
Shutdown started.    Fri Apr 15 06:35:48 MDT 2015
.
Broadcast Message from root (console) on murky Fri Apr 15 06:35:48...
The system pinkytusk will be shut down in 5 minutes
.
.
Broadcast Message from root (console) on murkey Fri Apr 15 06:40:38...
THE SYSTEM hostname IS BEING SHUT DOWN NOW ! ! !
Log off now or risk your files being damaged
.
.
Apr 15 06:41:57 The system is down.  Shutdown took 69 seconds.
.
Press any key to reboot.
```

If you are bringing the system to run level 0 to turn off power to all devices, see “Turning Off Power to System Devices” on page 35.

Example 11 Bringing a System to a Multiuser State

By specifying `-i6`, this `shutdown` command syntax reboots a system to run level 3 in two minutes. No additional confirmation is required.

```
$ who
root    console      Jun 14 15:49    (:0)
userabc    pts/4        Jun 14 15:46    (:0.0)

$ shutdown -i6 -g120 -y
Shutdown started.    Fri Apr 15 06:46:50 MDT 2015
.
.
Changing to init state 6 - please wait
.
Apr 15 06:49:40 The system is down.  Shutdown took 50 seconds.
.
rebooting...
.
hostname console login:
```

Example 12 Shutting Down a System With the `init` Command

The `init` command is used to bring a stand-alone system to the run level where it is safe to turn off power.

```
$ init 0
INIT: New run level: 0
The system is coming down.  Please wait.
.
.
The system is down.
.
Press any key to reboot
```

See Also Regardless of the reasons for shutting the system down, you should return to run level 3, where all file resources are available, and users can log in. For instructions on bringing a system back to a multiuser state, see Chapter 4, “Booting a System”.

Turning Off Power to System Devices

You can shut down an x86 based system by pressing the power button. An ACPI event is then sent that alerts the system that the user has requested a shutdown. This method of turning the power off is equivalent to issuing a `shutdown -i0` or an `init 0` command.

For information about turning off power to devices, see the instructions for the specified hardware in the product documentation at `https://www.oracle.com/products/oracle-a-z.html`.

During the shutdown process, if the `console=graphics` option was used to boot the system, and the shutdown is triggered by the `Xorg` server, a progress status indicator is displayed. To bypass the progress status indicator, set the new `splash-shutdown` property of the `svc:/system/boot-config` SMF service to `false`, as follows:

```
$ svccfg -s svc:/system/boot-config:default setprop config/splash_shutdown = false
$ svcadm refresh svc:/system/boot-config:default
```

♦ ♦ ♦ CHAPTER 4

Booting a System

This chapter describes booting and rebooting an Oracle Solaris system. Unless stated otherwise, the information applies to both SPARC and x86 platforms.

The chapter covers the following topics:

- "Displaying and Setting Boot Attributes"
- "About Run Level Booting"
- "Booting From an Alternate Operating System or Boot Environment"
- "Rebooting a System"

Note - To perform the steps issue commands described in this chapter, you must have the appropriate rights profiles. See "Using Rights Profiles to Administer Boot Features" on page 14.

Displaying and Setting Boot Attributes

The section describes ways to display and set boot attributes on SPARC and x86 platforms.

SPARC: Using the OpenBoot PROM

The boot PROM is used to boot a SPARC based system and to modify boot parameters such as the boot device or the default boot file or kernel. You also use the boot PROM if you want to run hardware diagnostics before bringing the system to a multiuser state.

For a complete list of PROM commands, see the `eeprom`(8) man page.

To access the boot PROM, bring the system to run level 0. The system then displays the `ok` prompt.

```
$ init 0
```

```
ok
```

The next sections show tasks you can perform after you have accessed the `ok` prompt.

Identifying the PROM Revision Number of a System

To display the PROM revision number, type the following command:

```
ok banner
```

Determining the Default Boot Device

To obtain information about the default boot device, type the following command:

```
ok printenv boot-device
```

If the boot device is local, the output format would be similar to the following example:

```
boot-device =  /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0,0:a
```

If the boot device is network based, the output format would resemble the following example:

```
boot-device = /sbus@1f,0/SUNW,fas@e,8800000/sd@a,0:a \
/sbus@1f,0/SUNW,fas@e,8800000/sd@0,0:a disk net
```

▼ How to Identify Devices on a System

Identifying the system's devices helps you determine the appropriate devices from which to boot. To view the `probe` commands that are available on your system, use the `sifting probe` command.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Change the PROM `auto-boot?` value to `false`.**

   ```
   ok setenv auto-boot? false
   ```

2. **Clear all system registers.**

   ```
   ok reset-all
   ```

3. **Identify the devices on the system.**

   ```
   ok probe-device
   ```

Alternatively, you can also use the `devalias` command to identify the device aliases and the associated paths that *might* be connected to the system.

4. **(Optional) Restore the `auto-boot?`'s setting to `true`.**

```
ok setenv auto-boot? true
auto-boot? =          true
```

5. **Boot the system to a multiuser state.**

```
ok reset-all
```

Example 13 Displaying IDE Device Information

The following example shows how to identify the devices connected to a system.

```
ok setenv auto-boot? false
auto-boot? =          false
ok reset-all
.
.
ok probe-ide
   Device 0  ( Primary Master )
         Removable ATAPI Model: MATSHITACD-RW  CW-8124

  Device 1  ( Primary Slave )
         Not Present

  Device 2  ( Secondary Master )
         Not Present

  Device 3  ( Secondary Slave )
         Not Present

ok setenv auto-boot? true
auto-boot? =          true

ok reset-all
```

Example 14 Displaying Device Aliases

This example shows a sample output of the `devalias` command.

```
ok devalias
ttya /pci@7c0/pci@0/pci@1/pci@0/isa@2/serial@0,3f8
nvram /virtual-devices/nvram@3
```

```
net3 /pci@7c0/pci@0/pci@2/network@0,1
net2 /pci@7c0/pci@0/pci@2/network@0
net1 /pci@780/pci@0/pci@1/network@0,1
net0 /pci@780/pci@0/pci@1/network@0
net /pci@780/pci@0/pci@1/network@0
ide /pci@7c0/pci@0/pci@1/pci@0/ide@8
cdrom /pci@7c0/pci@0/pci@1/pci@0/ide@8/cdrom@0,0:f
disk3 /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@3
disk2 /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@2
disk1 /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@1
disk0 /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0
disk /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/disk@0
scsi /pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2
virtual-console /virtual-devices/console@1
name aliases
```

▼ How to Change the Default Boot Device

Before You Begin You might need to identify the devices on the system before you can change the default boot device to some other device. For information about identifying devices on the system, see "How to Identify Devices on a System" on page 38.

Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Change the value of the `boot-device`.**

   ```
   ok setenv boot-device device[n]
   ```

   `device[n]` — Identifies the `boot-device` value, such as `disk` or `network`. The *n* can be specified as a disk number. Use one of the `probe` commands if you need help identifying the disk number.

2. **Verify that the default boot device has been changed.**

   ```
   ok printenv boot-device
   ```

3. **Save the new `boot-device` value.**

   ```
   ok reset-all
   ```

   The new `boot-device` value is written to the PROM.

Example 15 Changing the Default Boot Device

In this example, the default boot device is set to disk.

```
ok setenv boot-device /pci@1f,4000/scsi@3/disk@1,0
boot-device =         /pci@1f,4000/scsi@3/disk@1,0
ok printenv boot-device
boot-device           /pci@1f,4000/scsi@3/disk@1,0
ok reset-all
Resetting ...
.
Rebooting with command: boot disk1
Boot device: /pci@1f,4000/scsi@3/disk@1,0  File and args:
```

Example 16 Setting the Boot Device to the Network

In this example, the default boot device is set to become network-based.

```
ok setenv boot-device net
boot-device =         net
ok printenv boot-device
boot-device           net                   disk
ok reset-all
.
host1 console login:
```

Working With EEPROM Parameters

With appropriate privileges, you can use the `eeprom` command to display and modify EEPROM parameter settings.

On x86 systems, when you set `boot-args` or `boot-file` properties, a special entry called `Solaris bootenv rc` is automatically added to the GRUB menu and is marked as the default entry. Thus, on x86 systems, you can also use the GRUB menu to set certain EEPROM properties.

Caution - Properties that are set with the `eeprom` command can be overridden by kernel command lines. For example, if, during boot time, you access the kernel command line from the GRUB menu and set some parameters, those settings are applied to the system instead of the settings in the boot file.

For more detailed information, see the `eeprom`(8) man page.

EEPROM Parameters on UEFI Systems

On UEFI enabled systems, the parameters are stored in two places. Oracle Solaris specific variables are stored in `/boot/solaris/bootenv.rc` file. UEFI specific variables are set in the NVRAM store. Unlike SPARC with OBP, Oracle Solaris variables are not consumed by UEFI firmware.

Most UEFI variables are in a binary format and are translated to a readable format. When translation is not possible, a hexdump is printed.

Viewing EEPROM Parameters

EEPROM parameters vary by platform. To view the available EEPROM parameters for your system type, use the following general syntax:

```
$ eeprom [-u] [parameter]
```

The `-u` option displays UEFI specific parameters on UEFI enabled systems.

Used by itself, the `eeprom` command displays all EEPROM parameters. This example refers to an x86 based system:

```
$ eeprom
keyboard-layout=Unknown
ata-dma-enabled=1
atapi-cd-dma-enabled=1
ttyb-rts-dtr-off=false
ttyb-ignore-cd=true
ttya-rts-dtr-off=false
ttya-ignore-cd=true
ttyb-mode=9600,8,n,1,-
ttya-mode=9600,8,n,1,-
lba-access-ok=1
console=ttya
```

This example displays UEFI EEPROM parameters. Note that the sample shows a partial output.

```
$ eeprom -u
MonotonicCounter=0x1f2
OsaBootOptNum=0xffff
ConOut=/PciRoot(0x0)/Pci(0x1c,0x7)/Pci(0x0,0x0)/Pci(0x0,0x0)/AcpiAdr(2147549440)
 /PciRoot(0x0)/Pci(0x1f,0x0)/Serial(0x0)/Uart(115200,8,N,1)/UartFlowCtrl(None)/
VenPcAnsi()
```

```
ConIn=/PciRoot(0x0)/Pci(0x1f,0x0)/Serial(0x0)/Uart(115200,8,N,1)/UartFlowCtrl(None)/
VenPcAnsi()
 /PciRoot(0x0)/Pci(0x1d,0x0)/USB(0x1,0x0)/USB(0x8,0x0)
BootOrder=Boot0000 Boot0001 Boot0002 Boot0003 Boot0004 Boot0005 Boot0006
Lang=eng
PlatformLang=en-US
Timeout=0x1
Boot0001=description:string=[UEFI]USB:USBIN:USB USB Hard Drive , flags:int=1,
 device_path: \
 string=/PciRoot(0x0)/Pci(0x1a,0x0)/USB(0x1,0x0)/USB(0x2,0x0)/
HD(1,MBR,0x004D5353,0x800,0x3b5800), \
 optional_data:string=AMBO
...
```

Setting EEPROM Parameters

To configure EEPROM parameters or properties, use the following general syntax:

```
$ eeprom [-u] [-d] parameter=value
```

The `-u` option pertains to UEFI specific parameters. The `-d` option deletes the parameter setting.

Note - On x86 platforms, certain properties such as `boot-device` are set through the setup utility for your firmware type, for example, UEFI Boot Manager. Thus setting the boot device with the `eeprom` applies only to SPARC based platforms.

The following are additional examples of setting EEPROM parameter values:

Disable automatic reboot:

```
$ eeprom auto-boot?=false
```

Set kernel boot arguments:

```
$ eeprom boot-args=-k
```

Set the console device:

```
$ eeprom console=graphics
```

Set the boot order on a UEFI enabled system:

```
$ eeprom -u BootOrder="Boot0005 Boot0001 Boot0002 Boot0003 Boot0004 Boot0000"
```

Delete a UEFI EEPROM Parameter

```
$ eeprom -u -d BootOrder
```

About Run Level Booting

The following procedures describe how to boot a system to various states, also known as *run level booting*.

How Run Levels Work

A system's *run level* (also known as an *init state*) defines what services and resources are available to users. A system can be on only one run level at a time.

Oracle Solaris has eight run levels, which are described in the following table. The default run level is specified in the `/etc/inittab` file as run level 3.

TABLE 1 Oracle Solaris Run Levels

Run Level	Init State	Purpose
0	Power-down state	To shut down the operating system so that it is safe to turn off power to the system.
s or S	Single-user state	To run as a single user with some file systems mounted and accessible.
1	Administrative state	To access all available file systems. User logins are disabled.
2	Multiuser state	For normal operations. Multiple users can access the system and all file systems. All daemons are running except for the NFS server daemons.
3	Multiuser level with NFS resources shared	For normal operations with NFS resources shared. This is the default run level.
4	Alternative multiuser state	Not configured by default, but available for customer use.
5	Power-down state	To shut down the operating system so that it is safe to turn off power to the system. If possible, automatically turns off power on systems that support this feature.
6	Reboot state	To stop the operating system and reboot to the state that is defined by the `initdefault` entry in the `/etc/inittab` file. The SMF service, `svc:/system/boot-config:default`, is enabled by default. When the `config/fastreboot_default` property is set to true, `init 6` bypasses certain firmware initialization and test steps, depending on the specific capabilities of the system. See "Accelerating the Reboot Process" on page 52.

In addition, the `svcadm` command can be used to change the run level of a system, by selecting a milestone at which to run. The following table shows which run level corresponds to each milestone.

TABLE 2 Run Levels and SMF Milestones

Run Level	SMF Milestone FMRI
S	`milestone/single-user:default`
2	`milestone/multiuser:default`
3	`milestone/multiuser-server:default`

When to Use Run Levels or Milestones

In general, changing milestones or run levels is an uncommon procedure. If it is necessary, using the `init` command to change to a run level will change the milestone as well and is the appropriate command to use. The `init` command is also good for shutting down a system.

However, booting a system using the `none` milestone can be very useful for debugging startup problems. There is no equivalent run level to the `none` milestone. For more information, see "How to Investigate Problems Starting Services at System Boot" in *Managing System Services in Oracle Solaris 11.4*.

When a system is being booted you can select the milestone to boot to or select the level of error messages to be recorded.

- SPARC based systems

  To specify a milestone to boot, use the following command:

  ```
  ok boot -m milestone=milestone
  ```

  The default milestone is `all` which starts all enabled services. To start only the `init`, `svc.startd` and `svc.configd` services, specify `none`. The `none` milestone enables you to start services manually and is useful for debugging. See "How to Investigate Problems Starting Services at System Boot" in *Managing System Services in Oracle Solaris 11.4* for instructions on how to use the `none` milestone.

  The run-level equivalents `single-user`, `multiuser`, and `multiuser-server` are also available, but are not commonly used. The `multiuser-server` milestone, in particular does not start any services which are not a dependency of that milestone, so may not include important services.

  To specify the level of logging, use the following command:

  ```
  ok boot -m logging-level
  ```

  Specify `quiet`, `verbose` or `debug`. For information about logging levels, see "Specifying the Amount of Startup Messaging" in *Managing System Services in Oracle Solaris 11.4*.
- x86 based systems

To specify a milestone to boot or to choose the level of logging, edit the GRUB menu at boot time. Add the `-m` *smf-options* kernel argument to the end of the `$multiboot` line of the specified boot entry. For example:

```
$multiboot /ROOT/s11.3_18/@/$kern $kern -B $zfs_bootfs -m logging-level
```

Determining a System's Current Run Level

To determine a system's current run level, use the `who -r` command. Output similar to the following example is displayed:

```
$ who -r
      run-level 3  Dec 13 10:10  3  0 S
```

The two rightmost information bits in the output indicate the number of times the system has been at this run level since the last reboot (0) and the system's previous run level (S).

Booting Systems to Specific Run Levels

To boot a system, or to boot it to specific run levels, the commands to use depend on the system's platform. On SPARC systems, you must be on the `ok` prompt to issue boot commands. Also, the boot process might prompt you for a password.

This example boots a system to the default multiuser state.

- ON SPARC systems:

  ```
  $ init 0
  ok boot
  ```

- On x86 platforms:

  ```
  $ reboot
  ```

This example boots a system to a single user state.

- On SPARC systems:

  ```
  $ init 0
  ok boot -s
  ```

- On x86 platforms:

  ```
  $ reboot -p
  ```

The -p option displays the GRUB menu at boot time.

1. From the GRUB menu, select the boot entry that you want to modify, then type e to edit that entry.
2. Add -s at the end of the $multiboot line.
3. Press the appropriate keys to continue booting.

Note - To always display the GRUB menu at boot time, disable the Fast Reboot feature. See "Accelerating the Reboot Process" on page 52. Then you can omit the -p option.

How to Boot a System Interactively

If the original boot file is damaged, boot the system interactively. Then you can choose to boot from a different /etc/system file or boot environment.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Make backup copies of the /etc/system and boot/solaris/filelist.ramdisk files.**

```
$ cp /etc/system /etc/system.bak
$ cp /boot/solaris/filelist.ramdisk /boot/solaris/filelist.ramdisk.orig
```

2. **Add the etc/system.bak file name to the /boot/solaris/filelist.ramdisk file.**

```
$ echo "etc/system.bak" >> /boot/solaris/filelist.ramdisk
```

3. **Depending on the platform, do one of the following:**

- **For SPARC platforms, issue the following command from the ok prompt.**

```
ok boot -a
```

- **For x86 platforms:**

a. **Reboot the system to display the GRUB menu.**

```
$ reboot -p
```

b. **Edit the boot entry by adding -a at the end of the $multiboot line.**

c. **Press the appropriate keys to continue booting.**

4. **At the `Name of system file` prompt, specify the backup file that you created.**
   For example:

```
Name of system file [etc/system]: /etc/system.bak
```

5. **At the `Retire store` prompt, press Return to bypass.**

Note - The `/etc/devices/retire_store` file is the backing store for devices that are retired by the Fault Management Architecture (FMA). The system no longer uses these devices. However, for recovery purposes, simply press Enter to bypass this file.

6. **After the system has booted, fix the original `/etc/system` file.**

7. **If necessary, reboot the system using the fixed file.**

Example 17 SPARC: Booting a System Interactively

In the following example, the screen output is truncated to show only relevant parts in bold.

```
$ init 0
.
ok boot -a
.
Name of system file [/etc/system]:  /etc/system.bak
.
Retire store [/etc/devices/retire_store] (/dev/null to bypass): Press Enter
.
system-28 console login:
```

Example 18 x86: Booting a System Interactively

In the following example, `Oracle Solaris 11.3` is edited to boot interactively.

The sample screen output is truncated to show only the parts related to the steps.

```
$ reboot -p
.
(After some messages, the menu appears.)

 ****************************************************************************
 *Oracle Solaris 11.3                                                       *
 *                                                                          *
```

```
*                                                                            *
******************************************************************************
```

You would select `Oracle Solaris 11.3` and type `e` to edit it. The edit is shown in bold.

```
+----------------------------------------------------------------------------+
| setparams 'Oracle Solaris 11.3'                                            |
| .                                                                          |
| .                                                                          |
| $multiboot /ROOT/s11.3/@/$kern $kern -B $zfs_bootfs -a                     |
| .                                                                          |
+----------------------------------------------------------------------------+
```

After editing, you would press the appropriate keys to continue with the boot process. Additional prompts appear.

```
Name of system file [/etc/system]:  /etc/system.bak
.
Retire store [/etc/devices/retire_store] (/dev/null to bypass):      Press Enter
.
system-04 console login:
```

Booting From an Alternate Operating System or Boot Environment

A *boot environment* (BE) is a ZFS file system that is designated for booting. A boot environment is essentially a bootable instance of the Oracle Solaris OS image, plus any other software packages that are installed into that image. You can maintain multiple boot environments on a single system. Each boot environment can have different OS versions installed. When you install Oracle Solaris, a new boot environment is automatically created during the installation.

For more information, see the `beadm`(8) man page as well as *Creating and Administering Oracle Solaris 11.4 Boot Environments*.

---

x86 only - If the boot device as identified by GRUB contains a ZFS storage pool, then the `grub.cfg` file that is stored in that pool's top level dataset, which shares the same name as the pool. There is always exactly one such dataset in a pool. After the system is booted, this dataset is mounted at /*pool-name* on the root file system.

Multiple bootable datasets or root file systems can exist in a pool. In that pool, the default root file system is identified by the pool' s `bootfs` property. However, the `zfs-bootfs` command enables you to specify any bootable dataset to use. See the `boot`(8) man page.

▼ SPARC: How to Boot From an Alternate Operating System or Boot Environment

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Bring the system to the `ok` PROM prompt.**

   ```
   $ init 0
   ```

2. **Display a list of available boot environments.**

   ```
   ok boot -L
   ```

3. **From the list, select an entry as prompted.**

   Based on your selected entry, a command instruction is provided.

4. **Type the command as provided in the instruction.**

   ```
   ok boot -Z rpool/ROOT/boot-environment
   ```

Example 19 SPARC: Booting From an Alternate Boot Environment

In the example, `s11.3_backup2` is selected to be the alternate boot environment for booting. The corresponding command is then provided. After you type the command at the `ok` prompt, the system is booted from the selected BE and displays the login prompt.

The screen output is truncated to show only relevant parts in bold.

```
$ init 0
.
ok boot -L
.
.
1 Oracle Solaris 11.3 SPARC
2 s11.3_backup
3 s11.3_backup2
Select environment to boot: [ 1 - 3 ]: 3

To boot the selected entry, invoke:
boot [<root-device>] -Z rpool/ROOT/s11.3_backup2

ok boot -Z rpool/ROOT/s11.3_backup2
.
```

```
system-28 console login:
```

▼ x86: How to Boot From an Alternate Boot Environment

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Display the GRUB menu.**

   ```
   $ bootadm list-menu
   ```

2. **Reboot the system from the entry you want to use.**

   ```
   $ reboot index-number
   ```

Example 20 Booting From an Alternate Boot Environment With `bootadm`

In this example, the selected alternate boot environment is `Oracle Solaris 11.s B14`.

```
$ bootadm list-menu
the location of the boot loader configuration files is: /rpool/boot/grub
default 1
timeout 30
0 s11.s.backup
1 Oracle Solaris 11.s B14

$ reboot 1
.
system-04 console login:
```

Rebooting a System

The same commands from previous sections are used to reboot a system, but the `reboot` command is more commonly used.

However, for systems running on multiuser state, you might want to issue the `shutdown` command first. The command sends out notifications to users who are logged in. Thus, they can save their work and log out properly.

After shutting the system down, you can boot it normally to the run level you want, or to the default multiuser state. See Chapter 3, “Shutting Down a System”.

Accelerating the Reboot Process

The Fast Reboot feature of Oracle Solaris is supported on both SPARC and x86 platforms. This feature implements an in-kernel boot loader that loads the kernel into memory and then switches to that kernel, so that the reboot process occurs within seconds.

The feature is configured through the `svc:/system/boot-config:default` service. Two properties of the service control the reboot behavior:

- `config/fastreboot_default` controls normal system reboots.
- `config/fastreboot_onpanic` controls system reboots that occur as a result of system panics.

The two properties are independent of each other and can have different settings without affecting each other's behavior.

On x86 systems, the service's `config/fastreboot_default` property is set to true and therefore Fast Reboot is always enabled by default.

On SPARC systems, the feature is disabled. To perform a single instance of a fast reboot on a SPARC system, use the `reboot -f` command syntax. To enable the feature permanently, change the setting of the `config/fastreboot_default` property. For example:

```
$ svccfg -s "system/boot-config:default" setprop config/fastreboot_default=true
$ svcadm refresh svc:/system/boot-config:default
```

For more information, see the `svcadm`(8) and `svccfg`(8) man pages.

x86: About the `quiesce` Function

The system's capability to bypass the firmware when booting a new OS image has dependencies on the device drivers' implementation of a new device operation entry point, `quiesce`. On supported drivers, this implementation *quiesces* a device, so that at completion of the function, the driver no longer generates interrupts. This implementation also resets the device to a hardware state, from which the device can be correctly configured by the driver's attach routine,

without a power cycle of the system or being configured by the firmware. For more information about this functionality, see the quiesce(9E) and dev_ops(9S) man pages.

Note - Not all device drivers implement the quiesce function. For troubleshooting instructions, see "Conditions Under Which Fast Reboot Might Not Work" on page 83 and "How to Clear a Failed Automatic Boot Archive Update on a System That Does Not Support Fast Reboot" on page 69.

x86: Methods to Reboot a System

Aside from the example in Example 20, "Booting From an Alternate Boot Environment With bootadm," on page 51, several other methods enable you to boot a system to alternative boot environments.

EXAMPLE 21 Using the beadm Command

In this example, you activate a selected boot environment. That boot environment is used the next time you reboot, and subsequently becomes the default BE.

```
$ beadm list
```

List of boot environments displayed

```
$ beadm activate be-name
$ reboot
```

EXAMPLE 22 x86: Specifying Command Arguments

If you know the BE name, you can reboot directly to the BE by specifying it at the command line.

```
$ reboot -- 'rpool/zfsbe2'
```

EXAMPLE 23 x86: Using Combined Options

This example shows how enable a kernel debugger while booting from a specified BE.

```
$ reboot -- 'rpool/zfsbe3 /platform/i86pc/kernel/amd64/unix -k'
```

EXAMPLE 24 x86: Rebooting to a New Kernel

This example shows how to reboot the system to a new kernel named `my-kernel`.

```
$ reboot -- '/platform/i86pc/my-kernel/amd64/unix -k'
```

EXAMPLE 25 x86: Debugging While Booting to a Run Level

This example shows how to debug while rebooting a system to a single-user state.

```
$ reboot -- '-ks'
```

♦ ♦ ♦ CHAPTER 5

Booting a System From the Network

This chapter provides information about booting an Oracle Solaris system from the network. Unless stated otherwise, the information applies to both SPARC and x86 platforms.

The chapter covers the following topics:

- “Requirements for Booting a System From the Network”
- “SPARC: Booting a System From the Network”
- “x86: Booting a System From the Network”

Note - To perform network based booting and issue commands described in this chapter, you must have the appropriate rights profiles. See “Using Rights Profiles to Administer Boot Features” on page 14.

Requirements for Booting a System From the Network

WAN boot is the mechanism that boots an Oracle Solaris system from the network. This capability is available in SPARC and x86 UEFI systems.

Use of WAN boot is associated with installing Oracle Solaris, which require more preparations. See *Automatically Installing Oracle Solaris 11.4 Systems*.

To enable network booting, the following are required:

- A configured DHCP server

  To service x86 based systems that use Preboot eXecution Environment (PXE) boot, the DHCP server's `PXEClient` must have the following information:

  - IP address of the file server
  - Name of the boot file, which is `pxegrub2` for systems with BIOS firmware and `grub2netx64.efi` for systems with UEFI firmware.
- A boot server that provides `tftp` service

For information about DHCP, see *Working With DHCP in Oracle Solaris 11.4*.

SPARC: Booting a System From the Network

The information in this section refers to SPARC systems.

Setting Network Boot Arguments

With the proper authorization, you can set configuration parameters to be used by the PROM during a WAN boot. If set, these parameters take precedence over any default configuration values from the DHCP server for those parameter.

At a minimum, you must provide the following information to the OpenBoot PROM:

- IP address of the booting client
- Name of the boot file
- IP address of the system that is providing the boot file image

The subnet mask and IP address of the default router might also be required.

You specify network parameter settings to PROM through the `network-boot-arguments` variable. The variable supports the following parameters:

`tftp-server`	IP address of the TFTP server
`client-id`	DHCP client identifier: this can be set to any unique value that the DHCP server allows. For AI clients, this value should be set to the hexadecimal hardware address of the client, preceded by the string `01` to indicate an ethernet network. For example, an Oracle Solaris client with the hexadecimal Ethernet address `8:0:20:94:12:1e` uses the client ID `0108002094121E`.
`dhcp-retries`	Maximum number of DHCP retries
`file`	File to download by using TFTP or URL for WAN boot
`host-ip`	IP address of the boot client (in dotted-decimal notation)
`hostname`	Host name to use in the DHCP transaction

`http-proxy`	HTTP proxy server specification (*IPADDR*[:*PORT*])
`router-ip`	IP address of the default router (in dotted-decimal notation)
`subnet-mask`	Subnet mask (in dotted-decimal notation)
`tftp-retries`	Maximum number of TFTP retries

To set `network-boot-arguments` parameters, use the following syntax:

```
$ eeprom network-boot-arguments="protocol,parameters"
```

protocol	Address discovery protocol to be used.
parameters	Any number of comma-separated parameters from the previous table. Each parameter must be in *key=value* format.

For example, for a system with the host name `mysystem.example.com` and where you allow only 3 DHCP retries, you would type the following:

```
$ eeprom network-boot-arguments="dhcp,hostname=mysystem.example.com,dhcp-retries=3"
```

If you supply network boot settings to PROM, then do not use any arguments when issuing the command to boot the system. Otherwise, depending on the arguments you use, some or all of those boot settings you defined would be ignored. From the previous example, if you specify `dhcp` when booting, the process will use DHCP, but will ignore the settings for host name and DHCP retries.

Setting Up an NVRAM Alias

Instead of always specifying to use DHCP when issuing the `boot` command, you can save the instruction at the PROM level by setting up an NVRAM alias.

For example, the following instruction sets a system to use DHCP when booting from a network device.

```
ok nvalias net /pci@1f,4000/network@1,1:dhcp
```

When you subsequently boot the system, instead of typing `boot net:dhcp`, you would only type `boot net`.

Caution - Do not use the `nvalias` command to modify the `NVRAMRC` file unless you are very familiar with this command as well as with the `nvunalias` command.

▼ SPARC: How to Boot From the Network

For the first step in this procedure, see the examples provided in the previous sections.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **(Optional) Perform either or both of the following steps.**
   - **Specify the appropriate values for the `network-boot-arguments` parameter.**
   - **Set up an NVRAM alias to always use DHCP.**
2. **Bring the system to the `ok` PROM prompt.**

   ```
   $ init 0
   ```
3. **Boot the system from the network.**

   ```
   ok boot net
   ```

   Note - If you skipped Step 1, type the following instead:

   ```
   ok boot net:dhcp
   ```

x86: Booting a System From the Network

For x86 systems, two types of networking boots are available:

- WAN boot is supported on x86 UEFI systems.
- For non-UEFI x86 systems, Preboot eXecution Environment (PXE) boot is supported provided that their network adapter firmware supports the PXE specification. The PXE Network Bootstrap Program (NBP) uses GRUB 2 to load the Oracle Solaris kernel to proceed with the boot process.

If you are using PXE boot, the DHCP server must be properly configured. In automatic installations, the AI server and the DHCP server can be in one system, and both are managed by the `installadm` command.

If the DHCP server is separate, you must configure it in the same manner that the `installadm` command normally configures an accessible DHCP server, which is to set up the `BootFile`

based on the client architecture identifier. To help the administrator, the `installadm` command prints out the client arch boot file paths that should be set for manually configured DHCP servers.

For reference, see the `installadm`(8) man page.

▼ x86: How to Boot From the Network

Booting x86 systems from the network involves performing configurations on the firmware.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Reboot the system.**

   ```
   $ reboot -p
   ```

2. **Access the BIOS or UEFI firmware.**

   The specific keys to access the firmware depend on whether your system is UEFI enabled or not.

3. **On the BIOS or UEFI firmware screens, modify the boot priority so that the system would boot from the network.**

   Setting this configuration depends on which utility screen is displayed.

4. **Press the appropriate keys to proceed with the boot process.**

   The process continues by booting from the network.

♦ ♦ ♦ CHAPTER 6

Managing Systems with Boot Pools

This chapter provides information that is relevant to systems that use storage devices inaccessible to firmware such as SPARC M7 series servers.

The chapter covers the following topics:

- "Overview of Booting From Firmware-Inaccessible Storage Devices"
- "Managing a Boot Pool, Boot Pool Datasets and Fallback Images"
- "OpenBoot Properties in Oracle Solaris"
- "`os-root-device` Variable"

Note - To manage these systems and issue commands described in this chapter, you must have the appropriate rights profiles. See "Using Rights Profiles to Administer Boot Features" on page 14.

Overview of Booting From Firmware-Inaccessible Storage Devices

Boot pools are created on systems that can boot from firmware-inaccessible storage devices. On these devices, the boot pool is separate from the root pool and thus, cannot be identified by the firmware. Instead, the boot process itself identifies and imports the boot pool. It also identifies the path to the root pool and verifies that the OS instance in the boot archive and the root file system match.

Managing a Boot Pool, Boot Pool Datasets and Fallback Images

On systems that use these devices, you manage the boot pool with the `bootadm boot-pool` command. The following options are available:

`add`	Adds new devices to the boot pool.
`list`	Displays information about configuration settings for the boot pools.
`remove`	Immediately removes a device from the boot pool.
`resync`	Resynchronizes the boot pool, and creates bootable datasets for particular boot environments.
`set`	Changes a boot pool parameter. Currently, you can change only the `eviction_algorithm` parameter.

For more information see the `bootadm`(8) man page.

By default, boot pool datasets associated with the most recently booted BEs remain in the boot pool. Datasets less used for booting are removed or evicted if the boot pool runs out of space. Only the datasets associated with a BE are removed by this process. The BE is not affected.

You can prevent automatic deletion for a specific BE. Or, you can change the eviction process for all datasets.

How to Prevent a BE From Being Removed

This procedure retains a boot pool dataset for a BE so that the BE will remain bootable.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Identify the BE to retain.**

```
$ beadm list
BE                Flags Mountpoint Space   Policy Created
--                ----- ---------- -----   ------ -------
BE1               -   -          6.13M   static 2014-10-09 17:21
BE2               -   -          52.86M  static 2015-01-03 16:22
BE3               NR   /          313.1M  static 2015-02-04 17:36
```

2. **Change policy for the BE.**

   For example, to retain `BE2`, type the following:

```
$ beadm set-policy -p noevict BE2
```

3. **Verify that policy has changed.**

The retained BE acquires a `noevict` flag.

```
$ beadm list
BE                Flags Mountpoint Space   Policy         Created
--                ----- ---------- -----   ------         -------
BE1                     -          6.13M   static         2014-10-09 17:21
BE2               -     -          52.86M  noevict,static 2015-01-03 16:22
BE3               NR    /          313.1M  static         2015-02-04 17:36
```

Disabling Eviction of All Boot Pool Datasets

To disable automatic eviction for all datasets in the boot pool, type the following command:

```
$ bootadm boot-pool set eviction_algorithm=none
```

The command prevents any dataset removal. However, if the boot pool gets full, activities that add information to the boot pool will fail, including the following:

- Creating a new BE, which is often done by `pkg` operations
- Activating a BE whose dataset is not in the boot pool
- Changing the policy on a BE to `noevict`

▼ How to Make a BE Bootable

If a dataset is evicted, the BE with which it is associated becomes unbootable. This procedure shows how to restore the BE's bootable state.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. Identify unbootable BE.

The `!` flag indicates an unbootable BE. The `N` flag indicates the currently booted or active BE, and the `R` flag indicates the BE to be used at the next reboot. In most cases, the active BE is also the BE for rebooting.

```
$ beadm list
BE                Flags Mountpoint Space   Policy Created
--                ----- ---------- -----   ------ -------
BE1               !-    -          6.13M   static 2014-10-09 17:21
BE2               -     -          52.86M  static 2015-01-03 16:22
BE3               NR    /          313.1M  static 2015-02-04 17:36
```

2. **Activate the unbootable BE.**

```
$ beadm activate BE1
```

3. **Verify the activation.**

An activated BE automatically becomes the BE for the next reboot.

```
$ beadm list
BE                Flags Mountpoint Space   Policy         Created
--                ----- ---------- -----   ------         -------
BE1               R     -          6.13M   static         2014-10-09 17:21
BE2               -     -          52.86M  noevict       2015-01-03 16:22
BE3               N     /          313.1M  static         2015-02-04 17:36
```

4. **(Optional) Reset the `R` flag to the previous BE.**

```
$ beadm activate BE3
$ beadm list
BE                Flags Mountpoint Space   Policy         Created
--                ----- ---------- -----   ------         -------
BE1               -     -          6.13M   static         2014-10-09 17:21
BE2               -     -          52.86M  noevict        2015-01-03 16:22
BE3               NR    /          313.1M  static         2015-02-04 17:36
```

▼ How to Update the Fallback Image

This procedure applies to those systems that have service processors (SPs) with the fallback miniroot installed. Whenever you update to an SRU version on the host, you must also update the fallback image.

Note - The fallback image's `README` file also provides the same instructions for updating the fallback image. Make sure to refer to that document for additional information related to the image.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See “Using Rights Profiles to Administer Boot Features” on page 14.

1. **Access your MOS account at `https://support.oracle.com`.**

On the support web page, search for Doc ID 2045311.1 (*Oracle Solaris 11.3 Support Repository Updates (SRU) Index*). This index lists the different SRU versions with links to their corresponding fallback boot images.

2. **Download the SRU version's fallback image.**

3. **At the ILOM prompt, check the version of the current image to verify that you do not have the latest version installed already.**

```
-> show /SP/firmware/host/miniroot version

  /SP/firmware/host/miniroot
    Properties:
        version = fallback-5.11-0.175.2.9.0.5.0
```

4. **At the ILOM prompt, update the fallback image.**

```
-> load -source  http://webserver.example.com/fallback/fallback.pkg
```

OpenBoot Properties in Oracle Solaris

New OpenBoot properties provide information about the devices that can be used while the system is booting. These properties are automatically maintained.

These properties are read-only and are visible through `/chosen`.

`boot-pool-list`	Lists device paths to OpenBoot accessible storage devices that comprise a boot pool and are the devices that Oracle Solaris will use when booting
`tboot-list`	Lists storage devices that include fallback images

`os-root-device` Variable

The `os-root-device` NVRAM variable defines devices and root file systems for root pools. To view this variable use the `printenv` command at the OpenBoot prompt or the `eeprom` command at a shell prompt. This variable is automatically maintained and normally should not need manual intervention. You can use the following keywords to define root pools stored on a firmware-inaccessible storage device:

```
osroot-path
osroot-type
osroot-iscsi-initiator-id
osroot-iscsi-target-ip
osroot-iscsi-port
osroot-iscsi-partition
osroot-iscsi-lun
osroot-iscsi-target-name
osroot-iscsi-target-name
osroot-subnet-mask
osroot-host-ip
```

When you use os-root-device to set a system to boot from a firmware-inaccessible storage device, that system will continue to use that configuration. To boot the system using another device, clear the current os-root-device setting, then assign a new value. For example:

```
$ eeprom os-root-device=
$ eeprom os-root-device=new-value
```

To display current os-root-device settings, use the following command. The output in this example has been reformatted for readability.

```
$ eeprom os-root-device
os-root-device=osroot-type=ZFS/iSCSI/IPv4/IPoIB;
osroot-iscsi-port=3260;
osroot-iscsi-target-ip=168.168.1.2;
osroot-iscsi-partition=a;
osroot-iscsi-initiator-id=iqn.1986-03.com.sun:01:0010e05db261.550b268b;
osroot-iscsi-lun=5;
osroot-iscsi-target-name=iqn.1986-03.com.sun:02:fb3685b9-883d-460a-b817-8ea0d8c023dc;
osroot-subnet-mask=255.255.255.0;osroot-host-ip=168.168.1.156;
osroot-path=/pci@314/pci@1/pciex15b3,1003@0:port=1,pkey=FFFF,protocol=ip
```

♦ ♦ ♦ CHAPTER 7

Troubleshooting Booting a System

This chapter discusses issues with booting or rebooting and provides troubleshoot information. Unless stated otherwise, the information applies to both SPARC and x86 platforms.

The chapter covers the following topics:

- "Managing the Oracle Solaris Boot Archives"
- "Shutting Down and Booting a System for Recovery Purposes"
- "Forcing a Crash Dump and Reboot of the System"
- "Booting a System With the Kernel Debugger (`kmdb`) Enabled"
- "Troubleshooting Issues With Fast Reboot"
- "Troubleshooting Issues With Booting and the Service Management Facility"
- "Problems Booting After an Installation"

For information about stopping and starting Oracle Solaris for recovery purposes, if you are running a service processor, and instructions on controlling Oracle ILOM service processors, see the hardware documentation at `https://docs.oracle.com/cd/E19694-01/E21741-02/index.html`.

Note - All of the troubleshooting steps require that you have the correct authorization to run the commands. See "Using Rights Profiles to Administer Boot Features" on page 14.

Managing the Oracle Solaris Boot Archives

In addition to administering the boot loader on x86 platforms, the `bootadm` command is also used to perform the following tasks to maintain Oracle Solaris boot archives:

- List the files and directories that are included in a system's boot archive.
- Manually update the boot archive.

The syntax of the command is as follows:

```
$ bootadm [subcommand] [-option] [-R altroot]
```

For example, the following syntax lists the files and directories in the boot archive:

```
$ bootadm list-archive
```

For more information about the `bootadm` command, see the `bootadm`(8) man page.

Managing the `boot-archive` SMF Service

The `boot-archive` service is controlled by SMF. The service instance is `svc:/system/boot-archive:default`.

To check the status of the `boot-archive` service, type this command:

```
$ svcs boot-archive
STATE          STIME    FMRI
online         10:35:14 svc:/system/boot-archive:default
```

The service must always be enabled. Otherwise, automatic recovery of the boot archive upon a system reboot might not occur. As a result, the boot archive could become unsynchronized or corrupted and would prevent the system from booting.

To enable a disabled `boot-archive` service, type the following:

```
$ svcadm enable system/boot-archive
```

For more information, see the `svcadm`(8) and `svcs`(1) man pages.

▼ How to Manually Update the Boot Archive

If the automatic update of the boot archive failed, clear the error with this procedure.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. Update the boot archive.

```
$ bootadm update-archive
```

To update the boot archive on an alternate root, type:

```
$ bootadm update-archive -R alternate-root
```

Caution - Do not reference the root file system of any non-global zone with the `-R` option. Doing so might damage the global zone's file system, compromise the security of the global zone, or damage the non-global zone's file system. See the `zones(7)` man page.

2. **Reboot the system.**

```
$ reboot
```

▼ x86: How to Clear a Failed Automatic Boot Archive Update on a System That Does Not Support Fast Reboot

If a system does not support the Fast Reboot feature, the automatic update of the boot archive might fail. Consequently, the system might be unable to reboot from the same boot environment.

In this case, a warning similar to the following is displayed, and the system goes into maintenance mode:

```
WARNING: Reboot required.
The system has updated the cache of files (boot archive) that is used
during the early boot sequence. To avoid booting and running the system
with the previously out-of-sync version of these files, reboot the
system from the same device that was previously booted.
```

The following steps describe how to recover from this failure.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Reboot the system.**

```
$ reboot
```

2. **If the active BIOS or UEFI boot device and the GRUB menu entries point to the current boot instance, follow these steps to prevent a boot archive update failure:**

   a. **Set the `auto-reboot-safe` property of the `svc:/system/boot-config` SMF service to `true`, as follows:**

```
$ svccfg -s svc:/system/boot-config:default setprop config/auto-reboot-safe = true
```

b. Verify that the `auto-reboot-safe` property is set correctly.

```
$ svccfg -s svc:/system/boot-config:default listprop |grep config/auto-reboot-safe
config/auto-reboot-safe            boolean  true
```

Shutting Down and Booting a System for Recovery Purposes

In the following instances, you must first shut down a system to analyze or troubleshoot booting and other system problems.

- Troubleshoot error messages when the system boots.
- Stop the system to attempt recovery.
- Boot a system for recovery purposes.
- Force a crash dump and reboot of the system.
- Boot the system with the kernel debugger.

You might need to boot the system for recovery purposes.

The following are some of the more common error and recovery scenarios:

- Boot from the installation media or from an AI server on the network to recover from a problem that is preventing the system from booting or to recover from a lost `root` password. This method requires you to mount the boot environment after importing the root pool.
- **x86 only:** Resolve a boot configuration problem by importing the root pool. If a problem with the file exists, you do *not* have to mount the boot environment, just import the root pool, which automatically mounts the `rpool` file system that contains the boot-related components.

▼ SPARC: How to Stop a System for Recovery Purposes

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See “Using Rights Profiles to Administer Boot Features” on page 14.

1. Bring the system to `ok` PROM prompt by using the `shutdown` or `init 0` command.

2. **Synchronize the file systems.**

```
ok sync
```

3. **Type the appropriate `boot` command to start the boot process.**

   For example, to boot the system to a single-user state, type the following:

```
ok boot -s
```

   For more information, see the `boot`(8) man page.

4. **Verify that the system was booted to the specified run level.**

```
$ who -r
 .       run-level s  May  2 07:39     3      0  S
```

5. **If the system does not respond to any input from the mouse, do one of the following:**

   - **Press the Reset key to reboot the system.**
   - **Use the power switch to reboot the system.**

Example 26 Powering Off a System

If you are running Oracle Solaris 11 on a system that uses a service processor, after shutting down the system, you must switch from the system console prompt to the service processor prompt. From there, you can stop the service processor, as shown in this example:

```
$ shutdown -g0 -i0 -y
# svc.startd: The system is coming down. Please wait.
svc.startd: 91 system services are now being stopped.
Jun 12 19:46:57 wgs41-58 syslogd: going down on signal 15
svc.startd: The system is down.
syncing file systems...done
Program terminated
r)eboot o)k prompt, h)alt?
$ o

ok #.
->

-> stop /SYS
Are you sure you want to stop /SYS (y/n)? y
Stopping /SYS
```

```
->
```

If you need to perform an immediate shutdown, use the `stop -force -script /SYS` command. Before you type this command, ensure that all data is saved.

Example 27 Powering On a System

The following example shows how to power on the system that uses a service processor. You must first be logged in to Oracle ILOM. See https://docs.oracle.com/cd/E81115_01/html/E86148/z40019501400145.html.

If you have a modular system, make sure that you are logged into the desired server module.

```
-> start /SYS
Are you sure you want to start /SYS (y/n) ? y
Starting /SYS

->
```

To skip confirmation prompts, type `start -script /SYS`.

▼ x86: How to Stop and Reboot a System for Recovery Purposes

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **If the keyboard and mouse are functional, type `init 0` to stop the system.**

   ```
   $ init 0
   ```

2. **If the system does not respond to any input from the mouse, do one of the following:**

   - **Press the Reset key to reboot the system.**
   - **Use the power switch to reboot the system.**

How to Boot to a Single-User State to Resolve a Bad `root` Shell or Password Problem

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Depending on the platform, do one of the following:**

   - **For SPARC platforms:**

     a. **Bring the system to the `ok` PROM prompt.**

     ```
     $ init 0
     ```

     b. **Boot the system to a single-user state.**

     ```
     ok boot -s
     ```

   - **For x86 platforms:**

     a. **Reboot a running system with the `-p` option of the `reboot` command.**

     ```
     $ reboot -p
     ```

     b. **When the GRUB menu is displayed, select the appropriate boot entry, then type `e` to edit that entry.**

     c. **Using the arrow keys, navigate to the `$multiboot` line, then type `-s` at the end of the line.**

   - **To exit the GRUB edit menu and boot the entry you just edited, press Control-X. If you have a system with UEFI firmware, and you are not using a serial console, pressing F10 also boots the entry.**

2. **Correct the shell entry in the `/etc/passwd` file.**

   ```
   $ vi /etc/password
   ```

3. **Reboot the system.**

How to Boot From Media to Resolve an Unknown `root` Password

Use the following procedure if you need to boot the system to correct an unknown `root` password or similar problem. This procedure requires you to mount the boot environment after importing the root pool. If you need to recover a root pool or root pool snapshot, see "Replacing Disks in a ZFS Root Pool" in *Managing ZFS File Systems in Oracle Solaris 11.4*.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Boot from the Oracle Solaris media by using one of the following options:**

   - **SPARC: Text installation – Boot from the installation media or from the network, then select the Shell option (option 3) from the text installation screen.**
   - **SPARC: Automated installation – Use the following command to boot directly from an installation menu that allows you to exit to a shell:**

     ```
     ok boot net:dhcp
     ```

   - **x86: Text installation – From the GRUB menu, select the Text Installer and command line boot entry, then select the Shell option (option 3) from the text installation screen.**
   - **x86: Automated installation – Boot from an AI server on the network. Select the Text Installer and command line entry from the GRUB menu. Then, select the Shell option (option 3) from the text installation screen.**

2. **Import the root pool.**

   ```
   zpool import -f rpool
   ```

3. **Create a mount point for the boot environment.**

   ```
   $ mkdir /a
   ```

4. **Mount the boot environment on the mount point `/a`.**

   ```
   $ beadm mount solaris-instance|be-name /a
   ```

   For example:

```
$ beadm mount solaris-2 /a
```

5. **If a password or shadow entry is preventing a console login, correct the problem.**

   a. **Set the TERM type.**

   ```
   $ TERM=vt100
   $ export TERM
   ```

   b. **Edit the shadow file.**

   ```
   $ cd /a/etc
   $ vi shadow
   $ cd /
   ```

6. **Update the boot archive.**

   ```
   $ bootadm update-archive -R /a
   ```

7. **Unmount the boot environment.**

   ```
   $ beadm umount be-name
   ```

8. **Halt the system.**

   ```
   $ halt
   ```

9. **Reboot the system to a single-user state, and when prompted for the root password, press Return.**

10. **Reset the root password.**

    ```
    $ passwd -r files root
    New Password: xxxxxx
    Re-enter new Password: xxxxxx
    passwd: password successfully changed for root
    ```

11. **Press Control-D to reboot the system.**

x86: How to Boot From Media to Resolve a Problem With the GRUB Configuration That Prevents the System From Booting

If your x86 based system will not boot, the problem might be caused by a damaged boot loader or a missing or corrupt GRUB menu. Use the following procedure in these types of situations.

Note - This procedure does *not* require you to mount the boot environment.

If you need to recover a root pool or root pool snapshot, see "Replacing Disks in a ZFS Root Pool" in *Managing ZFS File Systems in Oracle Solaris 11.4*.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Boot from the Oracle Solaris media.**

   - **Text installation – From the GRUB menu, select the Text Installer and command line boot entry, then select the Shell option (option 3) from the text installation screen.**
   - **Automated installation – Booting from an AI server on the network requires a PXE boot. Select the `Text Installer and command line` entry from the GRUB menu. Then, select the Shell option (option 3) from the text installation screen.**

2. **Import the root pool.**

   ```
   $ zpool import -f rpool
   ```

3. **To resolve a GRUB configuration issue, do one of the following:**

   - **If the system will not boot, but no error messages are displayed, the boot loader might be damaged. To resolve the problem, see "Installing GRUB 2" on page 29.**
   - **If the GRUB menu is missing, a "cannot open grub.cfg" error message is displayed at boot time. To resolve this problem, see "Generating the GRUB Menu" on page 20.**

- **If the GRUB menu has become corrupted, other error messages might be displayed as the system attempts to parse the GRUB menu at boot time. See also "Generating the GRUB Menu" on page 20.**

4. **Exit the shell and reboot the system.**

```
exit
        1  Install Oracle Solaris
        2  Install Additional Drivers
        3  Shell
        4  Terminal type (currently sun-color)
        5  Reboot

Please enter a number [1]: 5
```

Forcing a Crash Dump and Reboot of the System

Forcing a crash dump and reboot of the system are sometimes necessary for troubleshooting purposes. The `savecore` feature is enabled by default.

For more information about system crash dumps, see "Configuring Your System for Crash Dumps" in *Troubleshooting System Administration Issues in Oracle Solaris 11.4*.

▼ SPARC: How to Force a Crash Dump and Reboot of the System

Use this procedure to force a crash dump of a SPARC based system. The example that follows this procedure shows how to use the `halt -d` command to force a crash dump of the system. You will need to manually reboot the system after running this command.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Bring the system to the `ok` PROM prompt.**

2. **Synchronize the file systems and write the crash dump.**

```
> n
ok sync
```

After the crash dump is written to disk, the system will continue to reboot.

3. **Verify that the system boots to run level 3.**

   The login prompt is displayed when the boot process has finished successfully.

   hostname `console login:`

Example 28 SPARC: Forcing a Crash Dump and Reboot of a System by Using the `halt -d` Command

This example shows how to force a crash dump and reboot of a SPARC based system by using the `halt -d` command.

```
$ halt -d
Jul 21 14:13:37 jupiter halt: halted by root

panic[cpu0]/thread=30001193b20: forced crash dump initiated at user request

000002a1008f7860 genunix:kadmin+438 (b4, 0, 0, 0, 5, 0)
  %l0-3: 0000000000000000 0000000000000000 0000000000000004 0000000000000004
  %l4-7: 00000000000003cc 0000000000000010 0000000000000004 0000000000000004
000002a1008f7920 genunix:uadmin+110 (5, 0, 0, 6d7000, ff00, 4)
  %l0-3: 0000030002216938 0000000000000000 0000000000000001 0000004237922872
  %l4-7: 000000423791e770 0000000000004102 0000030000449308 0000000000000005

syncing file systems... 1 1 done
dumping to /dev/dsk/c0t0d0s1, offset 107413504, content: kernel
100% done: 5339 pages dumped, compression ratio 2.68, dump succeeded
Program terminated
ok boot
Resetting ...

.
.
Rebooting with command: boot
Boot device: /pci@1f,0/pci@1,1/ide@3/disk@0,0:a
File and args: kernel/sparcv9/unix
configuring IPv4 interfaces: hme0.
add net default: gateway 198.51.100.240
Hostname: jupiter
The system is coming up.  Please wait.
NIS domain name is example.com
.
.
.
System dump time: Wed Jul 21 14:13:41 2013
Jul 21 14:15:23 jupiter savecore: saving system crash dump
```

```
in /var/crash/jupiter/*.0
Constructing namelist /var/crash/jupiter/unix.0
Constructing corefile /var/crash/jupiter/vmcore.0
100% done: 5339 of 5339 pages saved
.
.
.
```

▼ x86: How to Force a Crash Dump and Reboot of the System

If you cannot use the `reboot -d` or the `halt -d` command, you can use the kernel debugger (`kmdb`) to force a crash dump. The kernel debugger must have been loaded, either at boot time or with the `mdb -k` command for the following procedure to work.

Note - You must be in text mode to access the kernel debugger. So, first exit any window system.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Access the kernel debugger.**

   The method that is used to access the debugger is dependent upon the type of console that you are using to access the system.

   - If you are using a locally attached keyboard, press F1–A.
   - If you are using a serial console, send a break by using the method appropriate to that type of serial console.

   The `kmdb` prompt is displayed.

2. **To force a crash, use the `systemdump` macro.**

   ```
   [0]> $<systemdump
   ```

   Panic messages are displayed, the crash dump is saved, and the system reboots.

3. **Verify that the system has rebooted by logging in at the console login prompt.**

Example 29 x86: Forcing a Crash Dump and Reboot of the System by Using the `halt -d` Command

This example shows how to force a crash dump and reboot of an x86 based system by using the `halt -d` command.

```
$ halt -d
4ay 30 15:35:15 wacked.<domain>.COM halt: halted by user

panic[cpu0]/thread=ffffffff83246ec0: forced crash dump initiated at user request

fffffe80006bbd60 genunix:kadmin+4c1 ()
fffffe80006bbec0 genunix:uadmin+93 ()
fffffe80006bbf10 unix:sys_syscall32+101 ()

syncing file systems... done
dumping to /dev/dsk/c1t0d0s1, offset 107675648, content: kernel
NOTICE: adpu320: bus reset
100% done: 38438 pages dumped, compression ratio 4.29, dump succeeded

Welcome to kmdb
Loaded modules: [ audiosup crypto ufs unix krtld s1394 sppp nca uhci lofs
genunix ip usba specfs nfs md random sctp ]
[0]>
kmdb: Do you really want to reboot? (y/n) y
```

Booting a System With the Kernel Debugger (kmdb) Enabled

If you need to troubleshoot system problems, running a system under the kernel debugger can be very helpful. The kernel debugger can help you investigate system hangs.

For example, if you are running the kernel while the kernel debugger is active, and you experience a hang, you might be able to break into the debugger to examine the system state. Also, if the system panics, the panic can be examined before the system is rebooted. In this way, you can get an idea of which section of code might be causing the problem.

The following procedures describe the basic steps for troubleshooting system problems by booting with the kernel debugger enabled.

▼ SPARC: How to Boot a System With the Kernel Debugger (kmdb) Enabled

This procedure shows how to load the kernel debugger (`kmdb`) on a SPARC based system.

Note - Use the `reboot` command and the `halt` command with the `-d` option if you do not have time to debug the system interactively. Running the `halt` command with the `-d` option requires a manual reboot of the system afterward. However, if you use the `reboot` command, the system boots automatically. See the `reboot(8)` for more information.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Halt the system, causing it to display the `ok` prompt.**

   To halt the system cleanly, use the `halt` command.

2. **Type `boot -k` to request the loading of the kernel debugger. Press return.**

3. **Access the kernel debugger.**

   The method used to enter the debugger depends on the type of console that is used to access the system:

   - **If you are using a locally attached keyboard, press Stop-A or L1–A, depending on the type of keyboard.**
   - **If you are using a serial console, send a break by using the method that is appropriate for your type of serial console.**

   A welcome message is displayed when you enter the kernel debugger for the first time.

```
Rebooting with command: kadb
Boot device: /iommu/sbus/espdma@4,800000/esp@4,8800000/sd@3,0
.
.
.
```

Example 30 SPARC: Booting a System With the Kernel Debugger (`kmdb`) Enabled

The following example shows how to boot a SPARC based system with the kernel debugger (`kmdb`) enabled.

```
ok boot -k
Resetting...

Executing last command: boot kmdb -d
Boot device: /pci@1f,0/ide@d/disk@0,0:a File and args: kmdb -d
Loading kmdb...
```

x86: How to Boot a System With the Kernel Debugger (kmdb) Enabled

This procedure shows the basics for loading the kernel debugger. The savecore feature is enabled by default.

Before You Begin Ensure that your role has the appropriate rights profiles to perform this procedure. See "Using Rights Profiles to Administer Boot Features" on page 14.

1. **Boot the system.**
2. **When the GRUB menu is displayed, type e to access the GRUB edit menu.**
3. **Use the arrow keys to select the $multiboot line.**
4. **In the GRUB edit menu, type -k at the end of the $multiboot line.**

   To direct the system to stop (break) in the debugger before the kernel executes, include -d option with the -k option.
5. **To exit the GRUB edit menu and boot the entry you just edited, press Control-X. If you have a system with UEFI firmware, and you are not using a serial console, pressing F10 also boots the entry.**

   Typing -k loads the debugger (kmdb), then directly boots the operating system.
6. **Access the kernel debugger.**

   The method used to access the debugger is dependent upon the type of console that you are using to access the system.

   - If you are using a locally attached keyboard, press F1–A.
   - If you are using a serial console, send a break by using the method that is appropriate for that type of serial console.

   To access the kernel debugger (kmdb) before the system fully boots, use the -kd option.

   Using the -kd option loads the debugger and then gives you an opportunity to interact with the debugger before booting the operating system.

   A welcome message is displayed when you access the kernel debugger for the first time.

See Also For more detailed information about interacting with the system by using kmdb, see the kmdb(1) man page.

x86: Troubleshooting Issues With Fast Reboot

The following sections describe how to identify and resolve some common issues that you might encounter with the Fast Reboot feature of Oracle Solaris on x86 platforms.

If you need to manually update the Oracle Solaris boot archive on an x86 based system that does not support the Fast Reboot feature, see "How to Clear a Failed Automatic Boot Archive Update on a System That Does Not Support Fast Reboot" on page 69.

x86: Debugging Early Panics That Might Occur

Because the `boot-config` service has dependencies on the multiuser milestone, users who need to debug early panics can patch a global variable, `fastreboot_onpanic` in the `/etc/system` file, as shown in the following example, which includes a confirmation of the command you issued.

```
$ echo "set fastreboot_onpanic=1" >> /etc/system
$ echo "fastreboot_onpanic/W 1" | mdb -kw

$ mdb -kw
> fastreboot_onpanic/W 1
fastreboot_onpanic:             2               =       0x1
```

x86: Conditions Under Which Fast Reboot Might Not Work

The following are possible conditions under which the Fast Reboot feature might not work:

- GRUB configuration cannot be processed.
- The driver does not implement the `quiesce` function.

  If you attempt a fast reboot of a system with an unsupported driver, a message similar to the following is displayed:

  ```
  Sep 18 13:19:12 too-cool genunix: WARNING: nvidia has no quiesce()
  reboot: not all drivers have implemented quiesce(9E)
  ```

  If the driver for the network interface card (NIC) does not implement the `quiesce` function, you can attempt to unplumb the interface first, then retry a fast reboot of the system.
- There is insufficient memory.

If there is not enough memory on the system, or not enough free memory to load the new kernel and the boot archive, the fast reboot attempt fails with the following messages, then falls back to a regular reboot:

```
Fastboot: Couldn't allocate size below PA 1G to do fast reboot
Fastboot: Couldn't allocate size below PA 64G to do fast reboot
```

- The environment is unsupported.

  Fast reboot functionality is not supported in the following environments:

  - An Oracle Solaris release that is running as a paravirtualized (PV) guest domain
  - Non-global zones

For more information, see the following man pages:

- `reboot`(8)
- `init`(8)
- `quiesce`(9E)
- `uadmin`(2)
- `dev_ops`(9S)

Troubleshooting Issues With Booting and the Service Management Facility

The following are issues you might encounter when booting a system:

- Services do not start at boot time.

  A system might hang during boot time, if there are problems starting any Service Management Facility (SMF) services. To troubleshoot this type of issue, you can boot the system without starting any services. For more information, see "How to Investigate Problems Starting Services at System Boot" in *Managing System Services in Oracle Solaris 11.4*

- `system/filesystem/local:default` SMF service fails during boot.

  Local file systems that are not required to boot the system are mounted by the `svc:/system/filesystem/local:default` service. When any of those file systems cannot be mounted, the service enters a maintenance state. System startup continues, and any services that do not depend on `filesystem/local` are started. Subsequently, any services that require `filesystem/local` to be online before starting through dependencies are not started. The workaround for this issue is to change the configuration of your system so that a `sulogin` prompt is displayed immediately after the service fails instead of allowing the

system startup to continue. For more information, see "How to Force Single-User Login if the Local File System Service Fails During Boot" in *Managing System Services in Oracle Solaris 11.4*.

Problems Booting After an Installation

After installing onto a GPT labeled disk that had previously used MBR partitioning, sometimes the x86 firmware can not boot from the disk. This happens if your BIOS implementation requires that at least one hard disk have at least one MBR partition that is marked as bootable to boot in BIOS mode. The problem exists because UEFI specification forbids the EFI protective MBR partition from being marked as active. To fix this situation, use the `fdisk` command to set the partition's status to be active before attempting to boot.

Index

E

F

G

H

I

K

U

V

W

X