```
**********************************************************
   46017 Sat Jun 13 11:58:36 2015
new/usr/src/man/man1m/boot.1m
2056 update boot(1m) for right location of the GRUB menu
**********************************************************
'\" te
.\" Copyright 2015 Nexenta Systems Inc.
.\" Copyright (c) 2008 Sun Microsystems, Inc. All Rights Reserved
.\" Copyright 1989 AT&T
.\" The contents of this file are subject to the terms of the Common Development
.\" See the License for the specific language governing permissions and limitati
.\" fields enclosed by brackets "[]" replaced with your own identifying informat
.TH BOOT 1M "Jun 7, 2015"
.TH BOOT 1M "Jan 14, 2015"
.SH NAME
boot \- start the system kernel or a standalone program
.SH SYNOPSIS
.SS "SPARC"
.LP
.nf
\fBboot\fR [\fIOBP\fR \fInames\fR] [\fIfile\fR] [\fB-aLV\fR] [\fB-F\fR \fIobject
     [\fB-Z\fR \fIdataset\fR] [\fIboot-flags\fR] [\fB\(mi\(mi\fR] [\fIclient-pro
.fi

.SS "x86"
.LP
.nf
\fBkernel$\fR \fB/platform/i86pc/kernel/$ISADIR/unix\fR [\fIboot-args\fR]
     [\fB-B\fR \fIprop\fR=\fIval\fR [,\fIval\fR...]]
.fi

.SH DESCRIPTION
.LP
Bootstrapping is the process of loading and executing a standalone program. For
the purpose of this discussion, bootstrapping means the process of loading and
executing the bootable operating system. Typically, the standalone program is
the operating system kernel (see \fBkernel\fR(1M)), but any standalone program
can be booted instead. On a SPARC-based system, the diagnostic monitor for a
machine is a good example of a standalone program other than the operating
system that can be booted.
.sp
.LP
If the standalone is identified as a dynamically-linked executable, \fBboot\fR
will load the interpreter (linker/loader) as indicated by the executable format
and then transfer control to the interpreter. If the standalone is
statically-linked, it will jump directly to the standalone.
.sp
.LP
Once the kernel is loaded, it starts the UNIX system, mounts the necessary file
systems (see \fBvfstab\fR(4)), and runs \fB/sbin/init\fR to bring the system to
the "initdefault" state specified in \fB/etc/inittab\fR. See \fBinittab\fR(4).
.SS "SPARC Bootstrap Procedure"
.LP
On SPARC based systems, the bootstrap procedure on most machines consists of
the following basic phases.
.sp
.LP
After the machine is turned on, the system firmware (in PROM) executes power-on
self-test (POST). The form and scope of these tests depends on the version of
the firmware in your system.
.sp
.LP
After the tests have been completed successfully, the firmware attempts to
autoboot if the appropriate flag has been set in the non-volatile storage area
used by the firmware. The name of the file to load, and the device to load it
from can also be manipulated.
```

```
.sp
.LP
These flags and names can be set using the \fBeeprom\fR(1M) command from the
shell, or by using \fBPROM\fR commands from the \fBok\fR prompt after the
system has been halted.
.sp
.LP
The second level program is either a fileystem-specific boot block (when
booting from a disk), or \fBinetboot\fR or \fBwanboot\fR (when booting across
the network).
.sp
.LP
Network Booting
.sp
.LP
Network booting occurs in two steps: the client first obtains an IP address and
any other parameters necessary to permit it to load the second-stage booter.
The second-stage booter in turn loads the boot archive from the boot device.
.sp
.LP
An IP address can be obtained in one of three ways: RARP, DHCP, or manual
configuration, depending on the functions available in and configuration of the
PROM. Machines of the \fBsun4u\fR and \fBsun4v\fR kernel architectures have
DHCP-capable PROMs.
.sp
.LP
The boot command syntax for specifying the two methods of network booting are:
.sp
.in +2
.nf
boot net:rarp
boot net:dhcp
.fi
.in -2
.sp

.sp
.LP
The command:
.sp
.in +2
.nf
boot net
.fi
.in -2
.sp

.sp
.LP
without a \fBrarp\fR or \fBdhcp\fR specifier, invokes the default method for
network booting over the network interface for which \fBnet\fR is an alias.
.sp
.LP
The sequence of events for network booting using RARP/\fBbootparams\fR is
described in the following paragraphs. The sequence for DHCP follows the
RARP/\fBbootparams\fR description.
.sp
.LP
When booting over the network using RARP/\fBbootparams\fR, the PROM begins by
broadcasting a reverse ARP request until it receives a reply. When a reply is
received, the PROM then broadcasts a TFTP request to fetch the first block of
\fBinetboot\fR. Subsequent requests will be sent to the server that initially
answered the first block request. After loading, \fBinetboot\fR will also use
reverse ARP to fetch its IP address, then broadcast \fBbootparams\fR RPC calls
(see \fBbootparams\fR(4)) to locate configuration information and its root file
system. \fBinetboot\fR then loads the boot archive by means of NFS and
```

```
transfers control to that archive.
.sp
.LP
When booting over the network using DHCP, the PROM broadcasts the hardware
address and kernel architecture and requests an IP address, boot parameters,
and network configuration information. After a DHCP server responds and is
selected (from among potentially multiple servers), that server sends to the
client an IP address and all other information needed to boot the client. After
receipt of this information, the client PROM examines the name of the file to
be loaded, and will behave in one of two ways, depending on whether the file’s
name appears to be an HTTP URL. If it does not, the PROM downloads
\fBinetboot\fR, loads that file into memory, and executes it. \fBinetboot\fR
loads the boot archive, which takes over the machine and releases
\fBinetboot\fR. Startup scripts then initiate the DHCP agent (see
\fBdhcpagent\fR(1M)), which implements further DHCP activities.
.sp
.LP
If the file to be loaded is an HTTP URL, the PROM will use HTTP to load the
referenced file. If the client has been configured with an HMAC SHA-1 key, it
will check the integrity of the loaded file before proceeding to execute it.
The file is expected to be the \fBwanboot\fR binary. The WAN boot process can
be configured to use either DHCP or NVRAM properties to discover the install
server and router and the proxies needed to connect to it. When \fBwanboot\fR
begins executing, it determines whether sufficient information is available to
it to allow it to proceed. If any necessary information is missing, it will
either exit with an appropriate error or bring up a command interpreter and
prompt for further configuration information. Once \fBwanboot\fR has obtained
the necessary information, it loads the boot loader into memory by means of
HTTP. If an encryption key has been installed on the client, \fBwanboot\fR will
verify the boot loader’s signature and its accompanying hash. Presence of an
encryption key but no hashing key is an error.
.sp
.LP
The \fBwanboot\fR boot loader can communicate with the client using either HTTP
or secure HTTP. If the former, and if the client has been configured with an
HMAC SHA-1 key, the boot loader will perform an integrity check of the root
file system. Once the root file system has been loaded into memory (and
possibly had an integrity check performed), the boot archive is transferred
from the server. If provided with a \fBboot_logger\fR URL by means of the
\fBwanboot.conf\fR(4) file, \fBwanboot\fR will periodically log its progress.
.sp
.LP
Not all PROMs are capable of consuming URLs. You can determine whether a client
is so capable using the \fBlist-security-keys\fR OBP command (see
\fBmonitor\fR(1M)).
.sp
.LP
WAN booting is not currently available on the x86 platform.
.sp
.LP
The \fBwanboot\fR Command Line
.sp
.LP
When the client program is \fBwanboot\fR, it accepts \fBclient-program-args\fR
of the form:
.sp
.in +2
.nf
boot ... -o \fIopt1\fR[,\fIopt2\fR[,...]]
.fi
.in -2
.sp

.sp
.LP
where each option may be an action:
```

```
.sp
.ne 2
.na
\fB\fBdhcp\fR\fR
.ad
.sp .6
.RS 4n
Require \fBwanboot\fR to obtain configuration parameters by means of DHCP.
.RE

.sp
.ne 2
.na
\fB\fBprompt\fR\fR
.ad
.sp .6
.RS 4n
Cause \fBwanboot\fR to enter its command interpreter.
.RE

.sp
.ne 2
.na
\fB\fI<cmd>\fR\fR
.ad
.sp .6
.RS 4n
One of the interpreter commands listed below.
.RE

.sp
.LP
\&...or an assignment, using the interpreter’s parameter names listed below.
.sp
.LP
The \fBwanboot\fR Command Interpreter
.sp
.LP
The \fBwanboot\fR command interpreter is invoked by supplying a
\fBclient-program-args\fR of "\fB-o prompt\fR" when booting. Input consists of
single commands or assignments, or a comma-separated list of commands or
assignments. The configuration parameters are:
.sp
.ne 2
.na
\fB\fBhost-ip\fR\fR
.ad
.sp .6
.RS 4n
IP address of the client (in dotted-decimal notation)
.RE

.sp
.ne 2
.na
\fB\fBrouter-ip\fR\fR
.ad
.sp .6
.RS 4n
IP address of the default router (in dotted-decimal notation)
.RE

.sp
.ne 2
.na
\fB\fBsubnet-mask\fR\fR
```

```
.ad
.sp .6
.RS 4n
subnet mask (in dotted-decimal notation)
.RE

.sp
.ne 2
.na
\fB\fBclient-id\fR\fR
.ad
.sp .6
.RS 4n
DHCP client identifier (a quoted ASCII string or hex ASCII)
.RE

.sp
.ne 2
.na
\fB\fBhostname\fR\fR
.ad
.sp .6
.RS 4n
hostname to request in DHCP transactions (ASCII)
.RE

.sp
.ne 2
.na
\fB\fBhttp-proxy\fR\fR
.ad
.sp .6
.RS 4n
HTTP proxy server specification (IPADDR[:PORT])
.RE

.sp
.LP
The key names are:
.sp
.ne 2
.na
\fB\fB3des\fR\fR
.ad
.sp .6
.RS 4n
the triple DES encryption key (48 hex ASCII characters)
.RE

.sp
.ne 2
.na
\fB\fBaes\fR\fR
.ad
.sp .6
.RS 4n
the AES encryption key (32 hex ASCII characters)
.RE

.sp
.ne 2
.na
\fB\fBsha1\fR\fR
.ad
.sp .6
.RS 4n
```

```
the HMAC SHA-1 signature key (40 hex ASCII characters)
.RE

.sp
.LP
Finally, the URL or the WAN boot CGI is referred to by means of:
.sp
.ne 2
.na
\fB\fBbootserver\fR\fR
.ad
.sp .6
.RS 4n
URL of WAN boot’s CGI (the equivalent of OBP’s \fBfile\fR parameter)
.RE

.sp
.LP
The interpreter accepts the following commands:
.sp
.ne 2
.na
\fB\fBhelp\fR\fR
.ad
.sp .6
.RS 4n
Print a brief description of the available commands
.RE

.sp
.ne 2
.na
\fB\fB\fIvar\fR=\fIval\fR\fR\fR
.ad
.sp .6
.RS 4n
Assign \fIval\fR to \fIvar\fR, where \fIvar\fR is one of the configuration
parameter names, the key names, or \fBbootserver\fR.
.RE

.sp
.ne 2
.na
\fB\fB\fIvar\fR=\fR\fR
.ad
.sp .6
.RS 4n
Unset parameter \fIvar\fR.
.RE

.sp
.ne 2
.na
\fB\fBlist\fR\fR
.ad
.sp .6
.RS 4n
List all parameters and their values (key values retrieved by means of OBP are
never shown).
.RE

.sp
.ne 2
.na
\fB\fBprompt\fR\fR
.ad
```

```
.sp .6
.RS 4n
Prompt for values for unset parameters. The name of each parameter and its
current value (if any) is printed, and the user can accept this value (press
Return) or enter a new value.
.RE

.sp
.ne 2
.na
\fB\fBgo\fR\fR
.ad
.sp .6
.RS 4n
Once the user is satisfied that all values have been entered, leave the
interpreter and continue booting.
.RE

.sp
.ne 2
.na
\fB\fBexit\fR\fR
.ad
.sp .6
.RS 4n
Quit the boot interpreter and return to OBP’s \fBok\fR prompt.
.RE

.sp
.LP
Any of these assignments or commands can be passed on the command line as part
of the \fB-o\fR options, subject to the OBP limit of 128 bytes for boot
arguments. For example, \fB-o\fR \fBlist,go\fR would simply list current
(default) values of the parameters and then continue booting.
.SS "iSCSI Boot"
.LP
iSCSI boot is currently supported only on x86. The host being booted must be
equipped with NIC(s) capable of iBFT (iSCSI Boot Firmware Table) or have the
mainboard’s BIOS be iBFT-capable. iBFT, defined in the Advanced Configuration
and Power Interface (ACPI) 3.0b specification, specifies a block of information
that contains various parameters that are useful to the iSCSI Boot process.
.sp
.LP
Firmware implementing iBFT presents an iSCSI disk in the BIOS during startup as
a bootable device by establishing the connection to the iSCSI target. The rest
of the process of iSCSI booting is the same as booting from a local disk.
.sp
.LP
To configure the iBFT properly, users need to refer to the documentation from
their hardware vendors.
.SS "Booting from Disk"
.LP
When booting from disk, the OpenBoot PROM firmware reads the boot blocks from
blocks 1 to 15 of the partition specified as the boot device. This standalone
booter usually contains a file system-specific reader capable of reading the
boot archive.
.sp
.LP
If the pathname to the standalone is relative (does not begin with a slash),
the second level boot will look for the standalone in a platform-dependent
search path. This path is guaranteed to contain
\fB/platform/\fR\fIplatform-name\fR. Many SPARC platforms next search the
platform-specific path entry \fB/platform/\fR\fIhardware-class-name\fR. See
\fBfilesystem\fR(5). If the pathname is absolute, \fBboot\fR will use the
specified path. The \fBboot\fR program then loads the standalone at the
appropriate address, and then transfers control.
```

```
.sp
.LP
Once the boot archive has been transferred from the boot device, Solaris can
initialize and take over control of the machine. This process is further
described in the "Boot Archive Phase," below, and is identical on all
platforms.
.sp
.LP
If the filename is not given on the command line or otherwise specified, for
example, by the \fBboot-file\fR NVRAM variable, \fBboot\fR chooses an
appropriate default file to load based on what software is installed on the
system and the capabilities of the hardware and firmware.
.sp
.LP
The path to the kernel must not contain any whitespace.
.SS "Booting from ZFS"
.LP
Booting from ZFS differs from booting from UFS in that, with ZFS, a device
specifier identifies a storage pool, not a single root file system. A storage
pool can contain multiple bootable datasets (that is, root file systems).
Therefore, when booting from ZFS, it is not sufficient to specify a boot
device. One must also identify a root file system within the pool that was
identified by the boot device. By default, the dataset selected for booting is
the one identified by the pool’s \fBbootfs\fR property. This default selection
can be overridden by specifying an alternate bootable dataset with the \fB-Z\fR
option.
.SS "Boot Archive Phase"
.LP
The boot archive contains a file system image that is mounted using an
in-memory disk. The image is self-describing, specifically containing a file
system reader in the boot block. This file system reader mounts and opens the
RAM disk image, then reads and executes the kernel contained within it. By
default, this kernel is in:
.sp
.in +2
.nf
/platform/‘uname -i‘/kernel/unix
.fi
.in -2
.sp

.sp
.LP
If booting from ZFS, the pathnames of both the archive and the kernel file are
resolved in the root file system (that is, dataset) selected for booting as
described in the previous section.
.sp
.LP
The initialization of the kernel continues by loading necessary drivers and
modules from the in-memory filesystem until I/O can be turned on and the root
filesystem mounted. Once the root filesystem is mounted, the in-memory
filesystem is no longer needed and is discarded.
.SS "OpenBoot PROM \fBboot\fR Command Behavior"
.LP
The OpenBoot \fBboot\fR command takes arguments of the following form:
.sp
.in +2
.nf
ok boot [\fIdevice-specifier\fR] [\fIarguments\fR]
.fi
.in -2
.sp

.sp
.LP
The default \fBboot\fR command has no arguments:
```

```
.sp
.in +2
.nf
ok boot
.fi
.in -2
.sp

.sp
.LP
If no \fIdevice-specifier\fR is given on the \fBboot\fR command line, OpenBoot
typically uses the \fIboot-device\fR or \fIdiag-device\fR \fBNVRAM\fR variable.
If no optional \fIarguments\fR are given on the command line, OpenBoot
typically uses the \fIboot-file\fR or \fIdiag-file\fR \fBNVRAM\fR variable as
default \fBboot\fR arguments. (If the system is in diagnostics mode,
\fIdiag-device\fR and \fIdiag-file\fR are used instead of \fIboot-device\fR and
\fIboot-file\fR).
.sp
.LP
\fIarguments\fR may include more than one string. All \fIargument\fR strings
are passed to the secondary booter; they are not interpreted by OpenBoot.
.sp
.LP
If any \fIarguments\fR are specified on the \fBboot\fR command line, then
neither the \fIboot-file\fR nor the \fIdiag-file\fR \fBNVRAM\fR variable is
used. The contents of the \fBNVRAM\fR variables are not merged with command
line arguments. For example, the command:
.sp
.in +2
.nf
ok \fBboot\fR \fB-s\fR
.fi
.in -2
.sp

.sp
.LP
ignores the settings in both \fIboot-file\fR and \fIdiag-file\fR; it interprets
the string \fB"-s"\fR as \fIarguments\fR. \fBboot\fR will not use the contents
of \fIboot-file\fR or \fIdiag-file\fR.
.sp
.LP
With older PROMs, the command:
.sp
.in +2
.nf
ok \fBboot net\fR
.fi
.in -2
.sp

.sp
.LP
took no arguments, using instead the settings in \fIboot-file\fR or
\fIdiag-file\fR (if set) as the default file name and arguments to pass to
boot. In most cases, it is best to allow the \fBboot\fR command to choose an
appropriate default based upon the system type, system hardware and firmware,
and upon what is installed on the root file system. Changing \fIboot-file\fR or
\fIdiag-file\fR can generate unexpected results in certain circumstances.
.sp
.LP
This behavior is found on most OpenBoot 2.x and 3.x based systems. Note that
differences may occur on some platforms.
.sp
.LP
The command:
```

```
.sp
.LP
ok \fBboot cdrom\fR
.sp
.LP
\&...also normally takes no arguments. Accordingly, if \fIboot-file\fR is set
to the 64-bit kernel filename and you attempt to boot the installation CD or
DVD with \fBboot cdrom\fR, boot will fail if the installation media contains
only a 32-bit kernel.
.sp
.LP
Because the contents of \fIboot-file\fR or \fIdiag-file\fR can be ignored
depending on the form of the \fBboot\fR command used, reliance upon
\fIboot-file\fR should be discouraged for most production systems.
.sp
.LP
When executing a WAN boot from a local (CD or DVD) copy of wanboot, one must
use:
.sp
.LP
ok \fBboot cdrom -F wanboot - install\fR
.sp
.LP
Modern PROMs have enhanced the network boot support package to support the
following syntax for arguments to be processed by the package:
.sp
.LP
[\fIprotocol\fR,] [\fIkey\fR=\fIvalue\fR,]*
.sp
.LP
All arguments are optional and can appear in any order. Commas are required
unless the argument is at the end of the list. If specified, an argument takes
precedence over any default values, or, if booting using DHCP, over
configuration information provided by a DHCP server for those parameters.
.sp
.LP
\fIprotocol\fR, above, specifies the address discovery protocol to be used.
.sp
.LP
Configuration parameters, listed below, are specified as \fIkey\fR=\fIvalue\fR
attribute pairs.
.sp
.ne 2
.na
\fB\fBtftp-server\fR\fR
.ad
.sp .6
.RS 4n
IP address of the TFTP server
.RE

.sp
.ne 2
.na
\fB\fBfile\fR\fR
.ad
.sp .6
.RS 4n
file to download using TFTP or URL for WAN boot
.RE

.sp
.ne 2
.na
\fB\fBhost-ip\fR\fR
.ad
```

```
.sp .6
.RS 4n
IP address of the client (in dotted-decimal notation)
.RE

.sp
.ne 2
.na
\fB\fBrouter-ip\fR\fR
.ad
.sp .6
.RS 4n
IP address of the default router
.RE

.sp
.ne 2
.na
\fB\fBsubnet-mask\fR\fR
.ad
.sp .6
.RS 4n
subnet mask (in dotted-decimal notation)
.RE

.sp
.ne 2
.na
\fB\fBclient-id\fR\fR
.ad
.sp .6
.RS 4n
DHCP client identifier
.RE

.sp
.ne 2
.na
\fB\fBhostname\fR\fR
.ad
.sp .6
.RS 4n
hostname to use in DHCP transactions
.RE

.sp
.ne 2
.na
\fB\fBhttp-proxy\fR\fR
.ad
.sp .6
.RS 4n
HTTP proxy server specification (IPADDR[:PORT])
.RE

.sp
.ne 2
.na
\fB\fBtftp-retries\fR\fR
.ad
.sp .6
.RS 4n
maximum number of TFTP retries
.RE

.sp
```

```
.ne 2
.na
\fB\fBdhcp-retries\fR\fR
.ad
.sp .6
.RS 4n
maximum number of DHCP retries
.RE

.sp
.LP
The list of arguments to be processed by the network boot support package is
specified in one of two ways:
.RS +4
.TP
.ie t \(bu
.el o
As arguments passed to the package’s \fBopen\fR method, or
.RE
.RS +4
.TP
.ie t \(bu
.el o
arguments listed in the NVRAM variable \fBnetwork-boot-arguments\fR.
.RE
.sp
.LP
Arguments specified in \fBnetwork-boot-arguments\fR will be processed only if
there are no arguments passed to the package’s \fBopen\fR method.
.sp
.LP
Argument Values
.sp
.LP
\fIprotocol\fR specifies the address discovery protocol to be used. If present,
the possible values are \fBrarp\fR or \fBdhcp\fR.
.sp
.LP
If other configuration parameters are specified in the new syntax and style
specified by this document, absence of the \fIprotocol\fR parameter implies
manual configuration.
.sp
.LP
If no other configuration parameters are specified, or if those arguments are
specified in the positional parameter syntax currently supported, the absence
of the \fIprotocol\fR parameter causes the network boot support package to use
the platform-specific default address discovery protocol.
.sp
.LP
Manual configuration requires that the client be provided its IP address, the
name of the boot file, and the address of the server providing the boot file
image. Depending on the network configuration, it might be required that
\fBsubnet-mask\fR and \fBrouter-ip\fR also be specified.
.sp
.LP
If the \fIprotocol\fR argument is not specified, the network boot support
package uses the platform-specific default address discovery protocol.
.sp
.LP
\fBtftp-server\fR is the IP address (in standard IPv4 dotted-decimal notation)
of the TFTP server that provides the file to download if using TFTP.
.sp
.LP
When using DHCP, the value, if specified, overrides the value of the TFTP
server specified in the DHCP response.
.sp
```

```
.LP
The TFTP RRQ is unicast to the server if one is specified as an argument or in
the DHCP response. Otherwise, the TFTP RRQ is broadcast.
.sp
.LP
\fIfile\fR specifies the file to be loaded by TFTP from the TFTP server, or the
URL if using HTTP. The use of HTTP is triggered if the file name is a URL, that
is, the file name starts with \fBhttp:\fR (case-insensitive).
.sp
.LP
When using RARP and TFTP, the default file name is the ASCII hexadecimal
representation of the IP address of the client, as documented in a preceding
section of this document.
.sp
.LP
When using DHCP, this argument, if specified, overrides the name of the boot
file specified in the DHCP response.
.sp
.LP
When using DHCP and TFTP, the default file name is constructed from the root
node’s \fBname\fR property, with commas (,) replaced by periods (.).
.sp
.LP
When specified on the command line, the filename must not contain slashes
(\fB/\fR).
.sp
.LP
The format of URLs is described in RFC 2396. The HTTP server must be specified
as an IP address (in standard IPv4 dotted-decimal notation). The optional port
number is specified in decimal. If a port is not specified, port 80 (decimal)
is implied.
.sp
.LP
The URL presented must be "safe-encoded", that is, the package does not apply
escape encodings to the URL presented. URLs containing commas must be presented
as a quoted string. Quoting URLs is optional otherwise.
.sp
.LP
\fBhost-ip\fR specifies the IP address (in standard IPv4 dotted-decimal
notation) of the client, the system being booted. If using RARP as the address
discovery protocol, specifying this argument makes use of RARP unnecessary.
.sp
.LP
If DHCP is used, specifying the \fBhost-ip\fR argument causes the client to
follow the steps required of a client with an "Externally Configured Network
Address", as specified in RFC 2131.
.sp
.LP
\fBrouter-ip\fR is the IP address (in standard IPv4 dotted-decimal notation) of
a router on a directly connected network. The router will be used as the first
hop for communications spanning networks. If this argument is supplied, the
router specified here takes precedence over the preferred router specified in
the DHCP response.
.sp
.LP
\fBsubnet-mask\fR (specified in standard IPv4 dotted-decimal notation) is the
subnet mask on the client’s network. If the subnet mask is not provided (either
by means of this argument or in the DHCP response), the default mask
appropriate to the network class (Class A, B, or C) of the address assigned to
the booting client will be assumed.
.sp
.LP
\fBclient-id\fR specifies the unique identifier for the client. The DHCP client
identifier is derived from this value. Client identifiers can be specified as:
.RS +4
.TP
```

```
.ie t \(bu
.el o
The ASCII hexadecimal representation of the identifier, or
.RE
.RS +4
.TP
.ie t \(bu
.el o
a quoted string
.RE
.sp
.LP
Thus, \fBclient-id="openboot"\fR and \fBclient-id=6f70656e626f6f74\fR both
represent a DHCP client identifier of 6F70656E626F6F74.
.sp
.LP
Identifiers specified on the command line must must not include slash (\fB/\fR)
or spaces.
.sp
.LP
The maximum length of the DHCP client identifier is 32 bytes, or 64 characters
representing 32 bytes if using the ASCII hexadecimal form. If the latter form
is used, the number of characters in the identifier must be an even number.
Valid characters are 0-9, a-f, and A-F.
.sp
.LP
For correct identification of clients, the client identifier must be unique
among the client identifiers used on the subnet to which the client is
attached. System administrators are responsible for choosing identifiers that
meet this requirement.
.sp
.LP
Specifying a client identifier on a command line takes precedence over any
other DHCP mechanism of specifying identifiers.
.sp
.LP
\fBhostname\fR (specified as a string) specifies the hostname to be used in
DHCP transactions. The name might or might not be qualified with the local
domain name. The maximum length of the hostname is 255 characters.
.LP
Note -
.sp
.RS 2
The \fBhostname\fR parameter can be used in service environments that require
that the client provide the desired hostname to the DHCP server. Clients
provide the desired hostname to the DHCP server, which can then register the
hostname and IP address assigned to the client with DNS.
.RE
.sp
.LP
\fBhttp-proxy\fR is specified in the following standard notation for a host:
.sp
.in +2
.nf
\fIhost\fR [":"" \fIport\fR]
.fi
.in -2
.sp

.sp
.LP
\&...where \fIhost\fR is specified as an IP ddress (in standard IPv4
dotted-decimal notation) and the optional \fIport\fR is specified in decimal.
If a port is not specified, port 8080 (decimal) is implied.
.sp
.LP
```

```
\fBtftp-retries\fR is the maximum number of retries (specified in decimal)
attempted before the TFTP process is determined to have failed. Defaults to
using infinite retries.
.sp
.LP
\fBdhcp-retries\fR is the maximum number of retries (specified in decimal)
attempted before the DHCP process is determined to have failed. Defaults to of
using infinite retries.
.SS "x86 Bootstrap Procedure"
.LP
On x86 based systems, the bootstrapping process consists of two conceptually
distinct phases, kernel loading and kernel initialization. Kernel loading is
implemented in GRUB (GRand Unified Bootloader) using the BIOS ROM on the system
board, and BIOS extensions in ROMs on peripheral boards. The BIOS loads GRUB,
starting with the first physical sector from a hard disk, DVD, or CD. If
supported by the ROM on the network adapter, the BIOS can also download the
\fBpxegrub\fR binary from a network boot server. Once GRUB is located, it
executes a command in a menu to load the \fBunix\fR kernel and a
pre-constructed boot archive containing kernel modules and data.
.sp
.LP
If the device identified by GRUB as the boot device contains a ZFS storage
pool, the \fBmenu.lst\fR file used to create the GRUB menu will be found in the
dataset at the root of the pool’s dataset hierarchy. This is the dataset with
the same name as the pool itself. There is always exactly one such dataset in a
pool, and so this dataset is well-suited for pool-wide data such as the
\fBmenu.lst\fR file. After the system is booted, this dataset is mounted at
/\fIpoolname\fR in the root file system.
.sp
.LP
There can be multiple bootable datasets (that is, root file systems) within a
pool. By default, the file system in which file name entries in a
\fBmenu.lst\fR file are resolved is the one identified by the pool’s
\fBbootfs\fR property (see \fBzpool\fR(1M)). However, a \fBmenu.lst\fR entry
can contain a \fBbootfs\fR command, which specifies an alternate dataset in the
pool. In this way, the \fBmenu.lst\fR file can contain entries for multiple
root file systems within the pool.
.sp
.LP
Kernel initialization starts when GRUB finishes loading the boot archive and
hands control over to the \fBunix\fR binary. At this point, GRUB becomes
inactive and no more I/O occurs with the boot device. The Unix operating system
initializes, links in the necessary modules from the boot archive and mounts
the root file system on the real root device. At this point, the kernel regains
storage I/O, mounts additional file systems (see \fBvfstab\fR(4)), and starts
various operating system services (see \fBsmf\fR(5)).
.SS "Failsafe Mode"
.LP
A requirement of booting from a root filesystem image built into a boot archive
then remounting root onto the actual root device is that the contents of the
boot archive and the root filesystem must be consistent. Otherwise, the proper
operation and integrity of the machine cannot be guaranteed.
.sp
.LP
The term "consistent" means that all files and modules in the root filesystem
are also present in the boot archive and have identical contents. Since the
boot strategy requires first reading and mounting the boot archive as the
first-stage root image, all unloadable kernel modules and initialization
derived from the contents of the boot archive are required to match the real
root filesystem. Without such consistency, it is possible that the system could
be running with a kernel module or parameter setting applied to the root device
before reboot, but not yet updated in the root archive. This inconsistency
could result in system instability or data loss.
.sp
.LP
Once the root filesystem is mounted, and before relinquishing the in-memory
```

```
filesystem, Solaris performs a consistency verification against the two file
systems. If an inconsistency is detected, Solaris suspends the normal boot
sequence and falls back to failsafe mode. Correcting this state requires the
administrator take one of two steps. The recommended procedure is to reboot to
the failsafe archive and rebuild the boot archive. This ensures that a known
kernel is booted and functioning for the archive rebuild process.
Alternatively, the administrator can elect to clear the inconsistent boot
archive service state and continue system bring-up if the inconsistency is such
that correct system operation will not be impaired. See \fBsvcadm\fR(1M).
.sp
.LP
If the boot archive service is cleared and system bring-up is continued (the
second alternative above), the system may be running with unloadable kernel
drivers or other modules that are out-of-date with respect to the root
filesystem. As such, correct system operation may be compromised.
.sp
.LP
To ensure that the boot archive is consistent, the normal system shutdown
process, as initiated by \fBreboot\fR(1M) and \fBshutdown\fR(1M), checks for
and applies updates to the boot archive at the conclusion of the
\fBumountall\fR(1M) milestone.
.sp
.LP
An update to any kernel file, driver, module or driver configuration file that
needs to be included in the boot archive after the \fBumountall\fR service is
complete will result in a failed boot archive consistency check during the next
boot. To avoid this, it is recommended to always shut down a machine cleanly.
.sp
.LP
If an update is required to the kernel after completion of the \fBumountall\fR
service, the administrator may elect to rebuild the archive by invoking:
.sp
.in +2
.nf
# \fBbootadm update-archive\fR
.fi
.in -2
.sp

.SS "Failsafe Boot Archive"
.LP
The failsafe archive can be used to boot the machine at any time for
maintenance or troubleshooting. The failsafe boot archive is installed on the
machine, sourced from the miniroot archive. Booting the failsafe archive causes
the machine to boot using the in-memory filesystem as the root device.
.SS "SPARC"
.LP
The SPARC failsafe archive is:
.sp
.in +2
.nf
/platform/‘uname -i‘/failsafe
.fi
.in -2
.sp

.sp
.LP
\&...and can be booted as follows:
.sp
.in +2
.nf
ok \fBboot [\fIdevice-specifier\fR] -F failsafe\fR
.fi
.in -2
.sp
```

```
.sp
.LP
If a user wishes to boot a failsafe archive from a particular ZFS bootable
dataset, this can be done as follows:
.sp
.in +2
.nf
ok \fBboot [\fIdevice-specifier\fR] -Z \fIdataset\fR -F failsafe\fR
.fi
.in -2
.sp

.SS "x86"
.LP
The x86 failsafe archive is:
.sp
.in +2
.nf
/boot/x86.miniroot-safe
.fi
.in -2
.sp

.sp
.LP
\&...and can be booted by selecting the \fBSolaris failsafe\fR item from the
GRUB menu.
.SH OPTIONS
.SS "SPARC"
.LP
The following SPARC options are supported:
.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.sp .6
.RS 4n
The boot program interprets this flag to mean \fBask me\fR, and so it prompts
for the name of the standalone. The \fB\&'\fR\fB-a\fR\fB\&'\fR flag is then
passed to the standalone program.
.RE

.sp
.ne 2
.na
\fB\fB-D\fR \fIdefault-file\fR\fR
.ad
.sp .6
.RS 4n
Explicitly specify the \fIdefault-file\fR. On some systems, \fBboot\fR chooses
a dynamic default file, used when none is otherwise specified. This option
allows the \fIdefault-file\fR to be explicitly set and can be useful when
booting \fBkmdb\fR(1) since, by default, \fBkmdb\fR loads the default-file as
exported by the \fBboot\fR program.
.RE

.sp
.ne 2
.na
\fB\fB-F\fR \fIobject\fR\fR
.ad
.sp .6
.RS 4n
Boot using the named object. The object must be either an ELF executable or
```

```
bootable object containing a boot block. The primary use is to boot the
failsafe or \fBwanboot\fR boot archive.
.RE

.sp
.ne 2
.na
\fB\fB-L\fR\fR
.ad
.sp .6
.RS 4n
List the bootable datasets within a ZFS pool. You can select one of the
bootable datasets in the list, after which detailed instructions for booting
that dataset are displayed. Boot the selected dataset by following the
instructions. This option is supported only when the boot device contains a ZFS
storage pool.
.RE

.sp
.ne 2
.na
\fB\fB-V\fR\fR
.ad
.sp .6
.RS 4n
Display verbose debugging information.
.RE

.sp
.ne 2
.na
\fB\fIboot-flags\fR\fR
.ad
.sp .6
.RS 4n
The boot program passes all \fIboot-flags\fR to \fBfile\fR. They are not
interpreted by \fBboot\fR. See the \fBkernel\fR(1M) and \fBkmdb\fR(1) manual
pages for information about the options available with the default standalone
program.
.RE

.sp
.ne 2
.na
\fB\fIclient-program-args\fR\fR
.ad
.sp .6
.RS 4n
The \fBboot\fR program passes all \fIclient-program-args\fR to \fIfile\fR. They
are not interpreted by \fBboot\fR.
.RE

.sp
.ne 2
.na
\fB\fIfile\fR\fR
.ad
.sp .6
.RS 4n
Name of a standalone program to \fBboot\fR. If a filename is not explicitly
specified, either on the \fBboot\fR command line or in the \fIboot-file\fR
NVRAM variable, \fBboot\fR chooses an appropriate default filename.
.RE

.sp
.ne 2
```

```
.na
\fB\fIOBP\fR \fInames\fR\fR
.ad
.sp .6
.RS 4n
Specify the open boot prom designations. For example, on Desktop SPARC based
systems, the designation \fB/sbus/esp@0,800000/sd@3,0:a\fR indicates a
\fBSCSI\fR disk (sd) at target 3, lun0 on the \fBSCSI\fR bus, with the esp host
adapter plugged into slot 0.
.RE

.sp
.ne 2
.na
\fB\fB-Z\fR \fIdataset\fR\fR
.ad
.sp .6
.RS 4n
Boot from the root file system in the specified ZFS dataset.
.RE

.SS "x86"
.LP
The following x86 options are supported:
.sp
.ne 2
.na
\fB\fB-B\fR \fIprop\fR=\fIval\fR...\fR
.ad
.sp .6
.RS 4n
One or more property-value pairs to be passed to the kernel. Multiple
property-value pairs must be separated by a comma. Use of this option is the
equivalent of the command: \fBeeprom\fR \fIprop\fR=\fIval\fR. See
\fBeeprom\fR(1M) for available properties and valid values.
.sp
If the root file system corresponding to this menu entry is a ZFS dataset, the
menu entry needs the following option added:
.sp
.in +2
.nf
-B $ZFS-BOOTFS
.fi
.in -2
.sp

.RE

.sp
.ne 2
.na
\fB\fIboot-args\fR\fR
.ad
.sp .6
.RS 4n
The boot program passes all \fIboot-args\fR to \fBfile\fR. They are not
interpreted by \fBboot\fR. See \fBkernel\fR(1M) and \fBkmdb\fR(1) for
information about the options available with the kernel.
.RE

.sp
.ne 2
.na
\fB\fB/platform/i86pc/kernel/$ISADIR/unix\fR\fR
.ad
.sp .6
```

```
.RS 4n
Name of the kernel to boot. When using the \fBkernel$\fR token, \fB$ISADIR\fR
expands to \fBamd64\fR on 64-bit machines, and a null string on other machines.
As a result of this dereferencing, this path expands to the proper kernel for
the machine.
.RE

.SH X86 BOOT SEQUENCE DETAILS
.LP
After a PC-compatible machine is turned on, the system firmware in the \fBBIOS
ROM\fR executes a power-on self test (POST), runs \fBBIOS\fR extensions in
peripheral board \fBROMs,\fR and invokes software interrupt INT 19h, Bootstrap.
The INT 19h handler typically performs the standard PC-compatible boot, which
consists of trying to read the first physical sector from the first diskette
drive, or, if that fails, from the first hard disk. The processor then jumps to
the first byte of the sector image in memory.
.SH X86 PRIMARY BOOT
.LP
The first sector on a floppy disk contains the master boot block (GRUB
\fBstage1\fR). The stage 1 is responsible for loading GRUB \fBstage2\fR. Now
GRUB is fully functional. It reads and executes the menu file
\fB/boot/grub/menu.lst\fR. A similar sequence occurs for DVD or CD boot, but
the master boot block location and contents are dictated by the El Torito
specification. The El Torito boot also leads to \fBstrap.com\fR, which in turn
loads \fBboot.bin\fR.
.sp
.LP
The first sector on a hard disk contains the master boot block, which contains
the master boot program and the \fBFDISK\fR table, named for the \fBPC\fR
program that maintains it. The master boot finds the active partition in the
\fBFDISK\fR table, loads its first sector (GRUB \fBstage1\fR), and jumps to its
first byte in memory. This completes the standard PC-compatible hard disk boot
sequence. If GRUB \fBstage1\fR is installed on the master boot block (see the
\fB-m\fR option of \fBinstallgrub\fR(1M)), then \fBstage2\fR is loaded directly
from the Solaris partition regardless of the active partition.
.sp
.LP
A similar sequence occurs for DVD or CD boot, but the master boot block location
and contents are dictated by the El Torito specification. The El Torito boot
will then continue in the same way as with the hard disk.
.sp
.LP
Floppy booting is not longer supported. Booting from USB devices follows the
same procedure as with hard disks.
.sp
.LP
An x86 \fBFDISK\fR partition for the Solaris software begins with a
one-cylinder boot slice, which contains GRUB \fBstage1\fR in the first sector,
the standard Solaris disk label and volume table of contents (VTOC) in the
second and third sectors, and GRUB \fBstage2\fR in the fiftieth and subsequent
sectors. The area from sector 4 to 49 might contain boot blocks for older
versions of Solaris. This makes it possible for multiple Solaris releases on
the same FDISK to coexist. When the \fBFDISK\fR partition for the Solaris
software is the active partition, the master boot program (\fBmboot\fR) reads
the partition boot program in the first sector into memory and jumps to it. It
in turn reads GRUB \fBstage2\fR program into memory and jumps to it. Once the
GRUB menu is displayed, the user can choose to boot an operating system on a
different partition, a different disk, or possibly from the network.
.sp
.LP
The behavior is slightly different when a disk is using \fBEFI\fR
partitioning. In that case the GRUB \fBstage1\fR is always installed
in the first sector of the disk, and it always loads \fBstage2\fR from
the partition specified at GRUB installation time. This only works on
partitions containing a ZFS root pool.
.sp
```

```
.LP
For network booting, the supported method is Intel’s Preboot eXecution
Environment (PXE) standard. When booting from the network using PXE, the system
or network adapter BIOS uses DHCP to locate a network bootstrap program
(\fBpxegrub\fR) on a boot server and reads it using Trivial File Transfer
Protocol (TFTP). The BIOS executes the \fBpxegrub\fR by jumping to its first
byte in memory. The \fBpxegrub\fR program downloads a menu file and presents
the entries to user.
.SH X86 KERNEL STARTUP
.LP
The kernel startup process is independent of the kernel loading process. During
kernel startup, console I/O goes to the device specified by the \fBconsole\fR
property.
.sp
.LP
When booting from UFS, the root device is specified by the \fBbootpath\fR
property, and the root file system type is specified by the \fBfstype\fR
property. These properties should be setup by the Solaris Install/Upgrade
process in \fB/boot/solaris/bootenv.rc\fR and can be overridden with the
\fB-B\fR option, described above (see the \fBeeprom\fR(1M) man page).
.sp
.LP
When booting from ZFS, the root device is specified by a boot parameter
specified by the \fB-B\fR \fB$ZFS-BOOTFS\fR parameter on either the
\fBkernel\fR or \fBmodule\fR line in the GRUB menu entry. This value (as with
all parameters specified by the \fB-B\fR option) is passed by GRUB to the
kernel.
.sp
.LP
If the console properties are not present, console I/O defaults to \fBscreen\fR
and \fBkeyboard\fR. The root device defaults to \fBramdisk\fR and the file
system defaults to \fBufs\fR.
.SH EXAMPLES
.SS "SPARC"
.LP
\fBExample 1 \fRTo Boot the Default Kernel In Single-User Interactive Mode
.sp
.LP
To boot the default kernel in single-user interactive mode, respond to the
\fBok\fR prompt with one of the following:

.sp
.in +2
.nf
\fBboot\fR \fB\fR\fB-as\fR

\fBboot\fR \fBdisk3\fR \fB-as\fR
.fi
.in -2
.sp

.LP
\fBExample 2 \fRNetwork Booting with WAN Boot-Capable PROMs
.sp
.LP
To illustrate some of the subtle repercussions of various boot command line
invocations, assume that the \fBnetwork-boot-arguments\fR are set and that
\fBnet\fR is devaliased as shown in the commands below.

.sp
.LP
In the following command, device arguments in the device alias are processed by
the device driver. The network boot support package processes arguments in
\fBnetwork-boot-arguments\fR.

.sp
```

```
.in +2
.nf
\fBboot net\fR
.fi
.in -2
.sp

.sp
.LP
The command below results in no device arguments. The network boot support
package processes arguments in \fBnetwork-boot-arguments\fR.

.sp
.in +2
.nf
\fBboot net:\fR
.fi
.in -2
.sp

.sp
.LP
The command below results in no device arguments. \fBrarp\fR is the only
network boot support package argument. \fBnetwork-boot-arguments\fR is ignored.

.sp
.in +2
.nf
\fBboot net:rarp\fR
.fi
.in -2
.sp

.sp
.LP
In the command below, the specified device arguments are honored. The network
boot support package processes arguments in \fBnetwork-boot-arguments\fR.

.sp
.in +2
.nf
\fBboot net:speed=100,duplex=full\fR
.fi
.in -2
.sp

.LP
\fBExample 3 \fRUsing \fBwanboot\fR with Older PROMs
.sp
.LP
The command below results in the \fBwanboot\fR binary being loaded from DVD or
CD, at which time \fBwanboot\fR will perform DHCP and then drop into its
command interpreter to allow the user to enter keys and any other necessary
configuration.

.sp
.in +2
.nf
\fBboot cdrom -F wanboot -o dhcp,prompt\fR
.fi
.in -2
.sp

.SS "x86 (32-bit)"
.LP
\fBExample 4 \fRTo Boot the Default Kernel In 32-bit Single-User Interactive
```

```
Mode
.sp
.LP
To boot the default kernel in single-user interactive mode, edit the GRUB
kernel command line to read:

.sp
.in +2
.nf
kernel /platform/i86pc/kernel/unix -as
.fi
.in -2
.sp

.SS "x86 (64-bit Only)"
.LP
\fBExample 5 \fRTo Boot the Default Kernel In 64-bit Single-User Interactive
Mode
.sp
.LP
To boot the default kernel in single-user interactive mode, edit the GRUB
kernel command line to read:

.sp
.in +2
.nf
kernel /platform/i86pc/kernel/amd64/unix -as
.fi
.in -2
.sp

.LP
\fBExample 6 \fRSwitching Between 32-bit and 64-bit Kernels on 64-bit x86
Platform
.sp
.LP
To be able to boot both 32-bit and 64-bit kernels, add entries for both kernels
to \fB/boot/grub/menu.lst\fR, and use the \fBset-menu\fR subcommand of
\fBbootadm\fR(1M) to switch. See \fBbootadm\fR(1M) for an example of the
\fBbootadm set-menu\fR.

.SH FILES
.ne 2
.na
\fB\fB/platform/\fR\fIplatform-name\fR\fB/ufsboot\fR\fR
.ad
.sp .6
.RS 4n
Second-level program to boot from a disk, DVD, or CD
.RE

.sp
.ne 2
.na
\fB\fB/etc/inittab\fR\fR
.ad
.sp .6
.RS 4n
Table in which the \fBinitdefault\fR state is specified
.RE

.sp
.ne 2
.na
\fB\fB/sbin/init\fR\fR
.ad
```

```
.sp .6
.RS 4n
Program that brings the system to the \fBinitdefault\fR state
.RE

.SS "64-bit SPARC Only"
.ne 2
.na
\fB\fB/platform/\fR\fIplatform-name\fR\fB/kernel/sparcv9/unix\fR\fR
.ad
.sp .6
.RS 4n
Default program to boot system.
.RE

.SS "x86 Only"
.ne 2
.na
\fB\fB/boot\fR\fR
.ad
.sp .6
.RS 4n
Directory containing boot-related files.
.RE

.sp
.ne 2
.na
\fB\fB/rpool/boot/grub/menu.lst\fR\fR
\fB\fB/boot/grub/menu.lst\fR\fR
.ad
.sp .6
.RS 4n
Menu of bootable operating systems displayed by GRUB.
.sp
\fBNote:\fR this file is located on the root ZFS pool. While many installs
often name their root zpool 'rpool', this is not required and the
/rpool in the path above should be substituted with the name of
the root pool of your current system.
.RE

.sp
.ne 2
.na
\fB\fB/platform/i86pc/kernel/unix\fR\fR
.ad
.sp .6
.RS 4n
32-bit kernel.
.RE

.SS "64-bit x86 Only"
.ne 2
.na
\fB\fB/platform/i86pc/kernel/amd64/unix\fR\fR
.ad
.sp .6
.RS 4n
64-bit kernel.
.RE

.SH SEE ALSO
.LP
\fBkmdb\fR(1), \fBuname\fR(1), \fBbootadm\fR(1M), \fBeeprom\fR(1M),
\fBinit\fR(1M), \fBinstallboot\fR(1M), \fBkernel\fR(1M), \fBmonitor\fR(1M),
\fBshutdown\fR(1M), \fBsvcadm\fR(1M), \fBumountall\fR(1M), \fBzpool\fR(1M),
\fBuadmin\fR(2), \fBbootparams\fR(4), \fBinittab\fR(4), \fBvfstab\fR(4),
\fBwanboot.conf\fR(4), \fBfilesystem\fR(5)
.sp
.LP
RFC 903, \fIA Reverse Address Resolution Protocol\fR,
\fBhttp://www.ietf.org/rfc/rfc903.txt\fR
.sp
.LP
RFC 2131, \fIDynamic Host Configuration Protocol\fR,
\fBhttp://www.ietf.org/rfc/rfc2131.txt\fR
.sp
.LP
RFC 2132, \fIDHCP Options and BOOTP Vendor Extensions\fR,
\fBhttp://www.ietf.org/rfc/rfc2132.txt\fR
.sp
.LP
RFC 2396, \fIUniform Resource Identifiers (URI): Generic Syntax\fR,
\fBhttp://www.ietf.org/rfc/rfc2396.txt\fR
.sp
.LP
\fI\fR
.sp
.LP
\fISun Hardware Platform Guide\fR
.sp
.LP
\fIOpenBoot Command Reference Manual\fR
.SH WARNINGS
.LP
The \fBboot\fR utility is unable to determine which files can be used as
bootable programs. If the booting of a file that is not bootable is requested,
the \fBboot\fR utility loads it and branches to it. What happens after that is
unpredictable.
.SH NOTES
.LP
\fIplatform-name\fR can be found using the \fB-i\fR option of \fBuname\fR(1).
\fIhardware-class-name\fR can be found using the \fB-m\fR option of
\fBuname\fR(1).
.sp
.LP
The current release of the Solaris operating system does not support machines
running an UltraSPARC-I CPU.
```