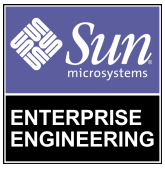
SCSI-Intitiator ID

By David Deeths - Enterprise Engineering

Sun BluePrints™ OnLine - August 2000



http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-6199-10
Revision 01, August 2000

SCSI-Initiator ID

Overview

Setting up a cluster is a complex task involving detailed configuration changes. Many of the changes involve the operating system, the volume manager, and sometimes high availability data services. These types of configuration changes are routinely made by system administrators and are commonplace—however, when a clustered system employs a shared SCSI chain there are additional configuration requirements which are performed using OpenBoot PROM (OBP) commands.

Because OBP commands are unfamiliar to many system administrators, implementing changes can be complicated. This article provides all necessary information for using OBP commands to change a host's SCSI-initiator ID.

Changing the SCSI-initiator ID is necessary for configurations using shared SCSI devices to function correctly when connected to multiple hosts—each SCSI ID on the chain must be unique (including the SCSI-initiator ID). Although there are numerous ways of configuring shared SCSI chains, the procedure described in this article can eliminate many of the potential problems faced when using other methods.

SCSI issues in clusters

When connecting multiple SCSI devices in a chain, it is important to ensure the SCSI IDs do not conflict. Each SCSI device requires a unique ID because the ID identifies a physical address on the SCSI bus.

To prevent conflict problems, the IDs for SCSI devices needs to be set. However, the methods used will vary depending on whether the device is a disk or host. For example, with disks, the ID can be set on the hardware, (although, with a Sun

storage enclosure, the ID is generally hardwired to a drive position within the enclosure—this method ensures that each position has a unique SCSI ID associated with the position).

Setting the SCSI ID for a host that is a SCSI-initiator is more complex.

The SCSI protocol supports two classes of devices:

Initiators

An initiator is generally a host device that has the ability to initiate a SCSI operation. The default SCSI-initiator ID is 7.

Targets

A target is a device (generally a storage device, although it could be a printer, scanner, or any other device on the SCSI chain), that will respond to SCSI operations.

Because targets generally require an initiator to function, they should not use ID 7. Available IDs fall within the ranges of 0-6 and 8-15. Narrow SCSI targets have IDs available in the 0-6 range, while wide SCSI have IDs available in the range of 0-15. If only one system is connected to a SCSI chain, the SCSI-initiator ID (the SCSI ID of the system) will not require changing.

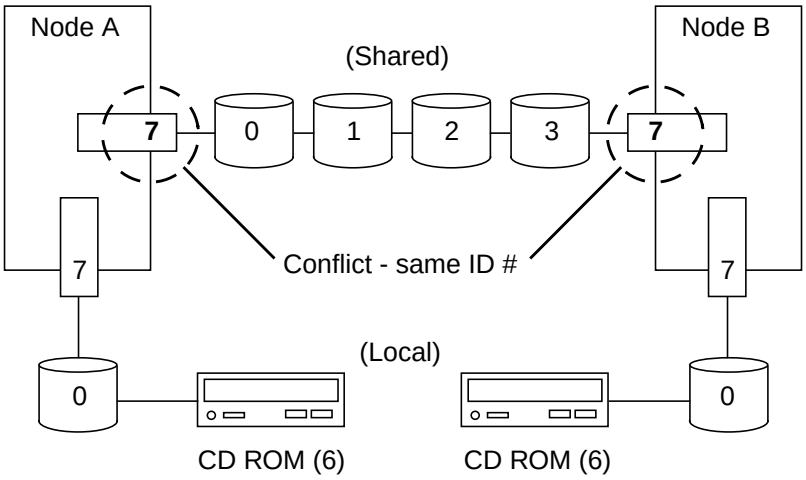
Each SCSI chain attached to a system is associated with a single SCSI controller. Each controller is the connection point of the system to a specific SCSI chain. The host can have a different address on each SCSI chain, therefore it can be useful to think of the SCSI address as belonging to the controller (the host's connection to a specific chain) itself, rather than to the host. The SCSI-initiator ID represents the system's address on a specific chain and can be set environment-wide for all controllers on the system, or individually for each SCSI controller. An environment-wide ID sets the default value for all controllers not having an ID explicitly set.

Clustered configurations are more complex to configure than a single machine because there could be multiple initiators on the SCSI chain. However, if only one initiator is on a chain, all targets will be local to the initiator—which means they can only receive requests from that initiator. In a single non clustered system configuration there can be several SCSI chains—each with its own set of SCSI IDs, however, each chain will still be local to the single initiator.

A clustered configuration can comprise a combination of local and global chains. The local chains will have connections from a specific host to target devices only, while the global chains will contain connections to target devices and to other initiators.

On local chains, each host must retain the default SCSI ID of 7 to prevent local chain conflicts—however, on global chains, the default ID on one of the initiators requires changing. If the ID is not changed, both initiators will share the same ID (and therefore the same address) as shown in FIGURE 1 which will result in a SCSI address conflict.

FIGURE 1 Conflicting SCSI-initiator IDs



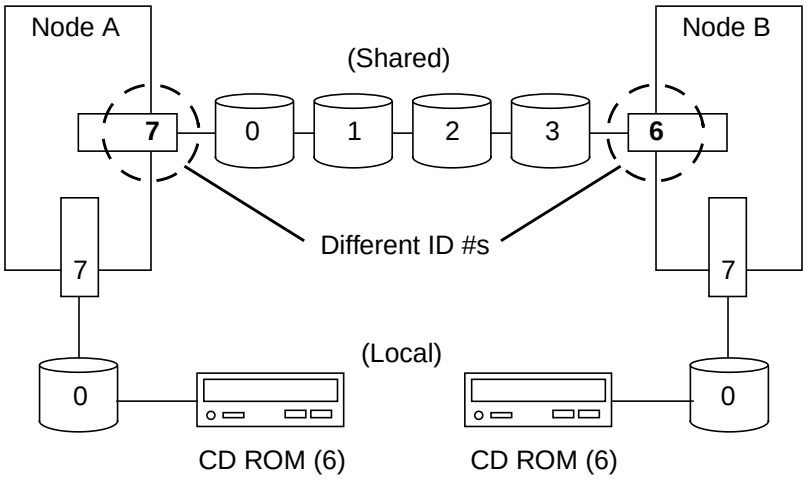
For local chains, the SCSI-initiator ID should be kept at the default value of 7—thereby preventing SCSI ID conflicts with devices such as the boot disk or CD-ROM (and any internal devices to be installed in the future).

For shared chains, one of the hosts should have its SCSI ID set to 6—this will prevent conflict problems (see FIGURE 2). The other host SCSI ID should be left at the default value.

Note – Setting the SCSI ID to 6 will prevent drives with a higher priority from hijacking the bus. Although changing the global SCSI-initiator ID of a device to 8 (or another unique ID on the chain) could solve SCSI initiator conflicts (and avoids the necessity of writing an nvram script), there could be a precedence problem because ID 8 has a precedence below that of other targets. A SCSI ID of 7 has the highest precedence, followed by 6, decreasing down to zero. IDs 8 to 15 have a precedence below ID zero.

When changing the SCSI-initiator ID, ensure none of the drives or other devices in the chain have the same ID. This holds true for global and local chains—on any given chain ensure each SCSI address is only used once. For example, in some Sun™ servers, the internal drives exist at IDs 0 and 1, and the CD-ROM has an ID of 6. Therefore, these IDs must be avoided by the SCSI-initiator ID.

FIGURE 2 Non-conflicting SCSI-initiator IDs



Changing SCSI initiator ID Overview

Any changes made to the SCSI-initiator ID should be made prior to connecting the host to the SCSI chain affected.

The procedure for changing the SCSI initiator ID requires modifying some OBP variables and setting up a script that will run as part of the system initialization. This can be performed using shell commands, or by working directly at the OBP level (the ok prompt). Changes made to an OBP environment variable are nonvolatile—all changes will be retained (even when the OS is reloaded).

The procedure outlined here should only be performed on one node of a 2 node cluster—a similar procedure could be followed for clusters with more than two nodes, however, the SCSI chains have to be mapped out, and a decision made as to which controllers on specific chains are to be changed. In most cases, it is best to have changes made on as few machines as possible. For example, in a 4-node Ring-Topology cluster, 2 nodes should remain unchanged, while the other two nodes should have the SCSI ID of the external controllers ID set to 6. The nodes with their IDs set to 7 would only share disks with the nodes that have their IDs set to 6. Remember, all IDs on a chain must be unique.

If you administer multiple clusters, it is a good idea to have a standard procedure for determining which device node will have the SCSI ID changed. For example, changing the initiator ID of the node with the highest ethernet address, or the node with the highest suffix on the host name (if the node names are the cluster name affixed with a number) will make it easier to remember which node has a non-default SCSI-initiator ID.

All changes made to the SCSI- initiator ID, and at the OBP level should be documented to assist with future troubleshooting.

Changing an environment-wide SCSI ID on one node of a two node cluster, and explicitly setting the local chains back to 7 eliminates conflicts on the shared chains and is transparent to devices on the local chains. A cluster environment usually has only one local SCSI chain, but could have several shared chains. This method reduces the number of controllers that need the SCSI IDs to be explicitly changed. Additionally, most cluster configurations will not require extra work to add a new shared chain—this is because any new controllers will have their SCSI IDs set to the default.

Note – It is important to ensure the host SCSI IDs do not interfere with each other, and also that target IDs do not interfere with the host IDs. Each SCSI ID on the chain must be unique—regardless of whether used by a target or initiator device—this could mean leaving an empty slot in a storage enclosure if the SCSI ID for the slot conflicts with the modified SCSI-initiator ID.

Note – Setting the SCSI ID for individual controllers is more complex than setting an environment-wide default. Both require using the OBP prompt or the Solaris Operating Environment `eeprom` command. Veritas Volume Manager (and other programs) store information in the NVRAM, therefore, it is important to check the NVRAM contents prior to making any changes. This is done using the `printenv nvramrc` OBP command, or the `eeprom nvramrc` command at the Solaris™ Operating Environment shell prompt. In most cases the existing NVRAM contents can be appended to the NVRAM script that sets the initiator ID. Some firmware upgrades will modify the OBP variables, see the section 'Maintaining Changes to NVRAM'.

The NVRAM script consists of OBP commands that are run in sequence. The script is called after system initialization (but before the device tree is built). Because a device tree must already exist for some OBP commands to work correctly, the NVRAM script must first run the probe-all command (which creates the OBP device tree). However because the probe-all command will also be run independently during the startup procedure, two OBP device trees would be created—only one of which will be seen by the operating system. The nvramrc changes will be effected in the tree which is not seen by the operating system. To prevent this happening, a procedure to keep the second `probe-all` operation from occurring needs to be invoked. The system start-up procedure has a work-around to prevent two sets of trees being created. This work-around will allow the running of the `probe-all` command in the NVRAM script, and prevents the `probe-all` command from running during a standard system start-up procedure (as listed in TABLE 1).

TABLE 1 System Start-up Process

Order	Procedure	Description
1	POST	Power on self-test
2	system initialization	
3	evaluate `nvramrc`	Evaluates the `nvramrc` script (if the `use-nvramrc?` OpenBoot environment variable is true.)
4	Run `probe-all` command*	Builds the device tree and initializes the devices
5	Run `install-console` command*	Sets up console (keyboard or terminal port)
6	Run `banner` command*	Prints system information to console
7	secondary diagnostics	
8	boot	Boots the system

* Only if the banner command is not called during the nvramrc script evaluation.

The work-around script uses the `banner` command (which displays system configuration information) in the NVRAM script to signal that the part of the standard system start up that runs the commands `probe-all`, `install-console`, and `banner` commands is being executed by the NVRAM script. If the `banner` command appears anywhere in the NVRAM script, the `banner` command will not run outside of the script (neither will the `install-console` or `probe-all` commands).

Note – Instead of using the banner command to signal the `probe-all` and `install-console` commands are to be run within the nvram script, the `suppress banner` command can be used (anywhere in the nvramrc script). The suppress-banner command does not display system information.

After the `probe-all` command has created the device node tree, it is then possible to change the device node properties (including the SCSI ID). The OpenBoot directory structure lists devices in a manner similar to the UNIX®device tree—each device node represents a level in the hardware hierarchy.

The peripheral bus (sbus or pci) is one of the main nodes under the root. All peripheral devices branch off this node—each device has a unique name which comprises several components:

- Device Identifier

The first part of the name identifies the device, however, the naming convention may differ for each type of device node. The device identifier is followed by the @ symbol.

Note – Device identifiers for peripheral cards are generally identified by the manufacturers symbol, followed by a comma, then the device type abbreviation (for example, SUNW,qfe@2,8c00000 identifies a quad-fast ethernet device manufactured by Sun Microsystems).

- Device Address and Offset

Following the @ symbol, the device address and offset identifier are listed—the address is listed first, followed by a comma, then the offset is listed, these identifiers generally reflect a connector or slot location. For example, the `sbus@f,0` and `sbus@e,0` entries in FIGURE 3 indicate two different sbus hardware slots.

The identifier-address-offset combination allows for multiple devices (of the same type) to be uniquely identified using the address and offset. Figures 3 and 4 show examples of OBP device trees. Some devices in the OBP device tree do not represent actual devices—these are termed pseudodevices. Other OBP devices represent hardware components unrelated to the peripheral cards, for example, in Figure 4, the CPU is represented by the name
“SUNW, UltraSPARC-II”.

Although an understanding the OBP device tree is beneficial for comprehending the architecture of the machine, it is not mandatory for changing the SCSI-initiator ID.

FIGURE 3 Example of `sbus` OBP Device Tree (incomplete)

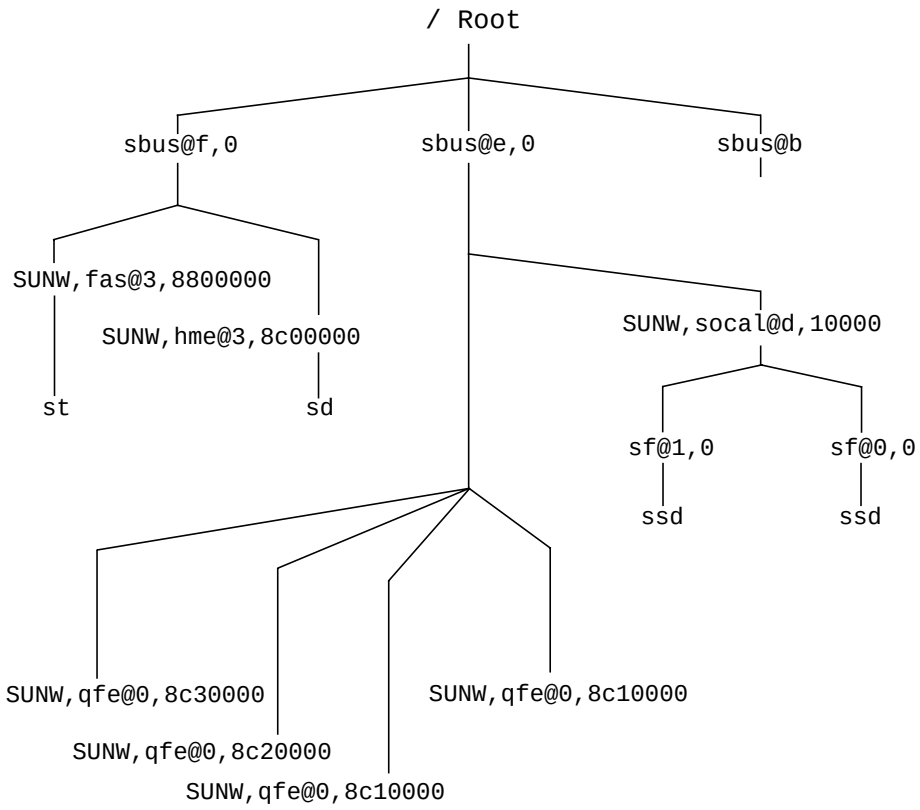
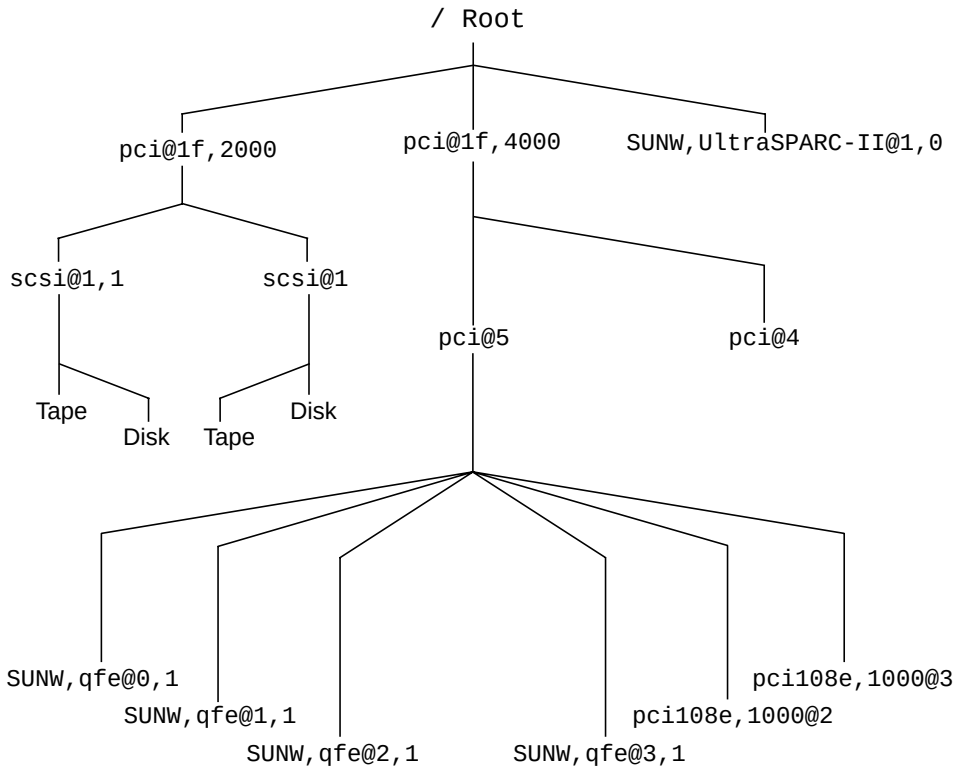


FIGURE 4 Example of pci OBP Device Tree (Sun Enterprise™ 250) (incomplete)



Mapping the OBP address to a physical card location can be difficult. To map the hardware location to an OBP node, use the following tables (2, 3, 4, and 5). These tables display the mapping between physical device slots and OBP device nodes for the current generation of Sun systems across several product families. Additional information can be found in the appropriate service manuals. The variable *devname* represents the name of the device in a given slot. Figure 5 displays the physical device slots referred to in TABLE 2.

FIGURE 5 I/O Board Interfaces in the E XX00 Series

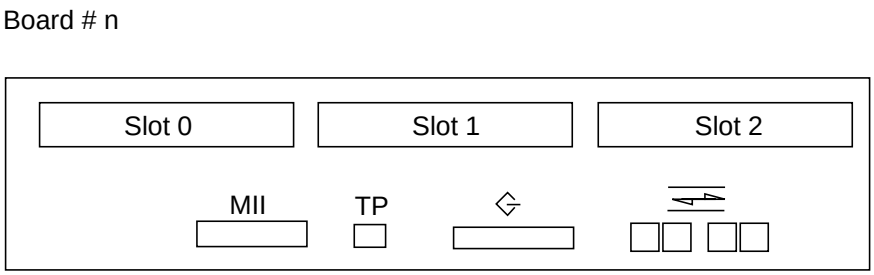


TABLE 2 E XX00 Series: Relationships of Physical Locations to OBP Nodes

Hardware Slot on Board # n	OBP Device Tree Node
0	`/sbus@(2n+1)/`*devname*`@0`
1	`/sbus@(2n)/`*devname*`@1`
2	`/sbus@(2n)/`*devname*`@2`
Ethernet (TP)	`/sbus@(2n+1)/hme@3`
SCSI	`/sbus@(2n+1)/fas@3`
Fiber	`/sbus@(2n)/socal@d`

TABLE 3 Sun Enterprise 450: Relationships of Physical Locations to OBP Nodes

PCI Slot Number (labeled)	OBP Device Tree Node
10	pci@1f,4000/*devname*@4
9	pci@4,4000/*devname*@2
8	pci@4,4000/*devname*@3
7	pci@4,4000/*devname*@4
6	pci@4,2000/*devname*@1
5	pci@1f,2000/*devname*@1
4	pci@6,2000/*devname*@1
3	pci@6,4000/*devname*@2
2	pci@6,4000/*devname*@3
1	pci@6,4000/*devname*@4

TABLE 4 Sun Enterprise 420R or Netra™ t1400/1405: Relationships of Physical Locations to OBP Nodes

PCI Slot Number (labeled)	OBP Device Tree Node
4	pci@1f,4000/*devname*@5
3	pci@1f,4000/*devname*@2
2	pci@1f,4000/*devname*@4
1	pci@1f,2000/*devname*@1

TABLE 5 Sun Enterprise 250, Sun Enterprise 220R, or Netra t1400/1405: Relationships of Physical locations to OBP Nodes

PCI Slot Number (labeled)	OBP Device Tree Node
3	pci@1f,2000/*devname*@1
2	pci@1f,4000/*devname*@2
1	pci@1f,4000/*devname*@4
0	pci@1f,4000/*devname*@5

Navigating through the device tree is accomplished in a similar manner as in the Solaris Operating Environment—by using the `cd` and `ls` commands. However, when working at the OBP prompt, you begin outside of the device tree. To gain access to the device tree, the `dev` command or an absolute path (one beginning with the root slash) must be used. To exit the device tree, use the `device-end` command.

The `.properties` command lists the properties of the current device node and can be useful for debugging problems when setting the SCSI-initiator ID. The following codebox is an example of an output from the `.properties` command run from a device node that represents a SCSI controller:

```
ok pwd

/sbus@1f,0/SUNW,fas@e,8800000

ok .properties

scsi-initiator-id        00000007

hm-rev                   00 00 00 22

device_type              scsi

clock-frequency          02625a00

intr                     00000020 00000000

interrupts               00000020

reg                      0000000e 08800000 00000010

                         0000000e 08810000 00000040

name                     SUNW,fas
```

In the above example, the current SCSI-initiator ID for this controller is listed in the `scsi-initiator-ID` field. Unless the value of the `scsi-initiator-id` field of the controller is set explicitly, it will default to the value of the `scsi-initiator-id` OBP environment variable.

The method for changing the ID is explained in the following sections.

Changing the SCSI-initiator ID

The NVRAM script can be edited from either a Solaris Operating Environment shell, or from the OBP prompt. The shell method is easier to use if the OS is operational. However, if the OS is down, it will be easier to use the OBP method. In either case, the machine will have to be rebooted for the changes to take effect.

This procedure is intended for a two node cluster. The issues involved in using a similar process for clusters with more than two nodes have been discussed earlier in the article.

Using the Shell Command Prompt Method

Set the environment-wide SCSI-initiator ID to 6 by using the `eeprom` command. This command must be run as the root user.

```
# eeprom scsi-initiator-id=6
```

Because the above change was made at the OBP level, the change becomes permanent (even if the OS is reloaded). To ensure the change does not interfere with local SCSI chains (including the CD-ROM and boot devices), you need to write a script that resets the local chain controller ID(s) back to 7.

The easiest way to create an NVRAM script at the shell prompt is to make a file that will contain a script of the OBP commands (*not* shell commands). The script is then stored in NVRAM using the `eeprom` command. Create this file using a text editor—store the file where it can be retrieved at a later date. The following is a step-by-step description of the commands needed in the script.

The first command required is the `probe-all` command. This command will be the first action taken by the `nvram` script—it will probe the devices and create the device tree. The file should now look as follows:

```
probe-all
```

Next, change the SCSI-initiator ID back to 7 on the local chains (which are not connected to other initiators). For each controller connected to a local chain, the script should enter the device node that represents the controller (using the `cd` command) and change the SCSI-initiator ID value. For example, if the internal / local controller is represented by the device node `SUNW,fas@e,8800000` connected to the sbus card at address `1f` (as with Ultra™ 1 and Ultra 2 systems), the following command would be added to the file:

```
cd /sbus@1f,0/SUNW,fas@e,8800000
```

Note – The device node path we use is only an example. In a situation where there are multiple local chains, the script will need to execute commands to enter each device node representing a local controller and explicitly change its initiator ID.

After entering the device node, change the SCSI-initiator ID to 7 by inserting the following line to the file:

```
7 " scsi-initiator-id" integer-property
```

Note – There should be a space before and after the first quotation mark. The quotation mark is a special OBP command that tells the tokenizer to treat the next word (which must be immediately followed by a quotation mark) as a string of characters. Therefore, the quote is separated from the string by a space because it is a command that operates on the string (not a syntactic marker as would be the case at the shell prompt).

The file should now look similar to the following:

```
probe-all
cd /sbus@1f,0/SUNW,fas@e,8800000
7 " scsi-initiator-id" integer-property
```

Change the SCSI-initiator ID for any other local chains to 7 (by using the method described previously).

After the SCSI-initiator IDs have been changed for all local chains, insert the `device-end` command into the file. Inserting this command will allow the script to exit the device tree. The file should now look similar to the following:

```
probe-all
cd /sbus@1f,0/SUNW,fas@e,8800000
7 " scsi-initiator-id" integer-property
device-end
```

As discussed in the beginning of this article, when the `probe-all` command is used in an NVRAM script, you also need to include the `banner` command in the file as a flag to keep the `probe-all` command from running again. Because the banner command also prevents the `install-console` command from running automatically, you must include `install-console` in the script as well. At the end of the file, include the `install-console` and banner commands. The file should now look similar to the following:

```
probe-all
cd /sbus@1f,0/SUNW,fas@e,8800000
7 " scsi-initiator-id" integer-property
device-end
install-console
banner
```

To store the file contents in NVRAM, run the `eeprom` command. The actual command to be run depends on the name of your file. For example, if your file is named `/nvram_file`, the command to use would be as follows:

```
# eeprom nvramrc="‘cat /nvram_file‘"
```

In the previous command, the *backtics* (‘) will interpret the `cat` command to output the file. If the quotation marks are not included, only the first line will make it into the nvramrc variable. If the backtics are not included, the nvramrc variable will contain the unevaluated string `cat /nvarm_file`. Because different shells may treat backtics and quotes differently, check the variables when finished. The above syntax is suitable for the Bourne and Korn shells.

The `nvramrc` environment variable should now be correctly configured, however, the NVRAM script will not be run at initialization unless the `use-nvramrc?` environment variable is set to `true` using the `eeprom` command.

```
# eeprom use-nvramrc?=true
```

Confirm all set values by using the `eeprom` command, The `eeprom` command will display a value if you run it without specifying a new value with the equal sign. Enter the following commands to check the appropriate OBP variables:

```
# eeprom scsi-initiator-id
scsi-initiator-id=6
# eeprom nvramrc
nvramrc=probe-all
cd /sbus@1f,0/SUNW,fas@e,8800000
7 " scsi-initiator-id" integer-property
device-end
install-console
banner
# eeprom use-nvramrc?
use-nvramrc?=true
```

Using the OpenBoot PROM (OBP) Prompt Method

All steps performed in the previous section can also be implemented at the OBP prompt, however, the process differs slightly.

Note – The steps performed in this section are for an OBP prompt only and are not suitable for running at the shell command prompt.

Set the environment-wide SCSI-initiator ID to 6 by using the `setenv` command.

Enter the following command into the OBP prompt.

```
ok setenv scsi-initiator-id 6
```

Because the preceding command sets an OBP environment variable (stored in NVRAM), the change becomes permanent automatically. Therefore, a script should be created to reset the local chain controller ID(s) back to 7. This script can be created using the `nvedit` command—this command starts a basic editor that operates in insert mode only and uses keystrokes similar to EMACS (see Table 6). The `nvedit` editor displays a line number at the beginning of each line (which is the only way to know which line you are on).

The editor begins with line zero—the line you are currently editing is always displayed at the bottom of the screen. Pressing return at the end of any line will create a new line. If there are additional lines after a newly added line, they will have their line numbers adjusted to account for the new line.

Note – It is easy to unintentionally press return and move existing lines down without realizing it, therefore, it is a good idea to review the finished file using the Ctrl-p or Ctrl-n keystrokes.

To start the nvedit editor, enter the following command at the OBP prompt.

```
ok nvedit
   0:
```

Note – In the preceding codebox, the `'0'` is the first line number.

TABLE 6 The `nvedit` Commands

Keystroke	Action
Ctrl-c	Exits the `nvramrc` editor and returns to the OBP command interpreter. The temporary buffer is preserved, but is not written back to the `nvramrc` variable. (Use `nvstore` to write back.)
Ctrl-l	List all lines
Ctrl-p	Moves to previous line
Ctrl-n	Moves to next line
Ctrl-b	Moves back one character
Ctrl-f	Moves forward one character
Delete	Deletes a character
Enter / Return or Ctrl-o	Starts a new line
Ctrl-k	Delete (kill) all characters from the cursor to the end of the line (including the next carriage return). If used at the end of a line, it will join the current line with the next line. If used at the beginning of a line, it will delete the entire line.

On line zero, enter the `probe-all` command (followed by the return key). This will cause the `NVRAM` script to probe the devices and create the device tree. The `NVRAM` script should look as follows:

```
0:  probe-all
```

After pressing the return key, line zero will move up the page and the cursor will be on line 1.

For each controller connected to a local chain, use the `cd` command to enter the device node that represents the controller—reset the SCSI-initiator ID value back to 7. For example, if the internal / local controller is represented by the device node `SUNW,fas@e` (connected to the `sbus` card at address `1f`),enter the device node by inserting the following command:

```
cd /sbus@1f,0/SUNW,fas@e,8800000
```

The SCSI-initiator ID value is changed by entering the following line:

```
7 " scsi-initiator-id" integer-property
```

Note – There should be a space before and after the first quotation mark.

In this example, the SCSI-initiator ID for the controller can be set in NVRAM when the previous two commands are added to lines 1 and 2 of the `NVRAM` script—as shown:

```
1: cd /sbus@1f,0/SUNW,fas@e,8800000
2: 7 " scsi-initiator-id" integer-property
```

The previous lines should be repeated for each local chain. After the IDs have been changed for all local devices, add the following commands to the end of the script (the line numbers used in the following codebox are example only):

```
3: device-end
4: install-console
5: banner
6:
```

The return keystroke following the banner command (line 5) will create a new line (6). Because the script is now completed, press Ctrl-c to exit the `nvedit` editor. Although the script is completed, it is not yet committed to NVRAM. To save the changes, use the `nvstore` command to move the `nvedit` buffer into NVRAM. Enter the following command into the OBP prompt:

```
ok nvstore
```

The nvramrc script should now be correctly configured, however, it will not be run unless the `use-nvramrc?` environment variable is set to `true` by using the `setenv` command. Enter the following command into the OBP prompt, note the output:

```
ok setenv use-nvramrc? true
use-nvramrc? =        true
```

Verify the contents of the `nvramrc` variable using the `printenv nvramrc` command. This command will display the contents of `nvramrc` variable. The display should look as follows:

```
ok printenv nvramrc
nvramrc =   probe-all
   cd /sbus@1f,0/SUNW,fas@e,8800000
   7 " scsi-initiator-id" integer-property
   device-end
   install-console
   banner
```

If there are any errors use the nvedit editor again to make the required changes. Move to the line(s) to be edited by using the Ctrl-p, or Ctrl-n keystrokes—edit the line, or delete it using the Ctrl-k keystroke.

After making changes, run the `nvstore` command again. When the `nvramrc` script is correct—restart the machine for changes to take effect. Enter the following command:

```
ok reset-all
```

Maintaining Changes to NVRAM

Another consideration is that some patches or system upgrades may alter the NVRAM, or disable the `use-nvramrc?` environment variable, and could even erase the NVRAM script. For example, the Solaris Operating Environment version 7 software includes a firmware patch to allow some older Ultra Enterprise™ 1 and Ultra Enterprise 2 systems to use 64 bit addressing. Although using this patch is optional, it will reset the `use-nvramrc?` environment variable and prevent the NVRAM script from running. After any changes are made involving firmware or EEPROM, check the values of any OBP variables that may have changed when using the `eeprom` command.

Run the following unix commands:

```
eeprom use-nvramrc?
eeprom nvramrc
eeprom scsi-initiator-id
```

Summary

Changing the SCSI-initiator ID on a cluster node allows two nodes to share SCSI storage devices on a single SCSI chain. This is essential for correct operation of a shared SCSI chain. Changing IDs requires either direct or indirect modification of the `nvramrc` script (which is a set of OpenBoot commands). Changes are permanent, even after reinstallation of the operating system.

Because some firmware upgrades can modify OBP variables, it is critical to check these after any firmware changes.

Author's Bio: David Deeths

David Deeths is a member of the technical staff of the Enterprise Engineering group, a part of Sun Microsystems' Computer Systems division. He has been working at Sun for 3 years. His current focus is clusters, including doing some of the research, development, and writing for the upcoming Sun Blueprints(tm) "Introduction to Clusters" book. David has degrees in Electrical Engineering and Cognitive Science from the University of California, San Diego.