Sun Enterprise™ 3000-6500 Dynamic Reconfiguration Best Practices

Technical White Paper

Joost Pronk Van Hoogeveen

Vanessa Heppolette

Sun microsystems

Contents

Foreword

This document intended to assist customers who are considering or implementing Dynamic Reconfiguration (DR).

This document covers the DR tools on the Sun™ Enterprise™ 3000-6500 (SunFire, SunFire+) servers only, and not DR on the Sun Enterprise 10000 (StarFire) platform. Because of the architectural difference between the passive backplane of the E3000-6500 servers and the active backplane of the E10000 server, DR is used differently in both cases. The active backplane allows the backplane itself to be logically divided into separate systems, which is not possible with a passive backplane. This document covers only the functionality of the Sun Enterprise 3000-6500 server range.

This document is based on systems running the Solaris™ 7 Operating Environment 5/99, although DR for SBus I/O boards was already supported with Solaris 2.6 5/98. The document covers the differences, but will mainly discuss Solaris 7 5/99 functionality. Information included in this document is strongly related to the Sun Enterprise 6x00, 5x00, 4x00, and 3x00 Systems Dynamic Reconfiguration User's Guide.

The intended audience for this document is people who are directly involved in the planning and configuration of DR on an E3000-6500 server. This document assumes that the reader is familiar with the Sun product line. Also, when the document goes into more details, it is assumed that the reader has good knowledge of the Solaris operating environment and Sun server architecture.

How to Read this Document:

This document is divided into three main sections, each with an intended audience and a recommended time to read it. That is why certain elements are discussed more than once, but from a different point of view or in a different style.

Chapter 2, “Architecture” discusses the architecture of DR and the Sun Enterprise 3000-6500 servers, and gives an indication of when DR can or should be used and when it cannot. It also describes the hardware and software components of DR and the mechanisms resulting from that. Finally it outlines the requirements needed for DR use. The purpose of this section is to give some insight and feeling to those who are thinking about using DR.

Chapter 3, “Procedures” talks about the procedural part of DR, including what should be considered and the influences of selected policies. It also shows typical configurations and highlights their strengths and weaknesses. In addition, this section discusses links with other applications. The purpose of this section is to provide an overview of how DR should be used and describe what steps should be taken to implement DR.

Chapter 4, “Help” is a bundling of easy to use charts, tables and checklists to help make the right decisions in different situations, as well as a list of common error messages. The purpose of this section is to have a place to look when searching for a quick answer.

All examples shown throughout this document are from Solaris 7 5/99 systems. All discussed functionality is with this version of Solaris in mind.

Introduction 1

In today's highly competitive marketplace, Information Technology managers struggle with the challenge of balancing expenses against the demands for improved service levels. Computing resources are increasingly distributed within the enterprise, adding to the complexity and cost of system management. Users, on the other hand, want systems and applications to be continuously accessible and available, and they need high levels of compute performance as well.

Without necessary data and applications, workers cannot be productive — orders cannot be processed, customers cannot be serviced, and products cannot be designed, manufactured, and delivered. Unplanned outages are extremely costly, both in lost productivity and, all too frequently, direct revenue.

Sun Enterprise X500 Servers — Unprecedented Availability

Sun Microsystems™, Inc., a world leader in enterprise system computing, is responding to customers' growing needs for higher system availability. In addition to excellent performance and scalability, Sun is now offering unprecedented availability with the Sun Enterprise X500 family of servers — the Sun Enterprise 3500, 4500, 5500, and 6500 systems.

Availability features such as Dynamic Reconfiguration and Alternate Pathing were previously exclusive to the Sun Enterprise 10000, a mainframe-class system designed for the data center. Sun is now incorporating these availability features in the full Sun Enterprise server product line, addressing availability requirements throughout the enterprise, from individual departments through the corporate data center. The availability offered by the Sun Enterprise X500 product family is unmatched by the current generation of competitive server products.

Availability

Availability is the time a particular resource, such as a system, application, or data, is accessible and usable. Obtaining higher levels of availability begins with core system design and extends to the overall software application architecture.

Availability is typically measured as a percentage of total uptime over the course of a year. System uptime requirements are almost always stated this way. For example, a 99.99 percent availability requirement translates to 52.8 minutes of downtime per year, while a 99.9 percent availability requirement means about eight hours of downtime per year. What's often unclear is whether or not the downtime factor includes planned maintenance or just unscheduled outages, and if the downtime represents a single outage or a total of outages through the year (e.g., 8 x 1 hour outages or 1 x 8 hour outage). Of course, such questions can only be answered on a case-by-case basis, with the customer assessing their business requirements and the cost of downtime for particular applications.

Achieving Availability Requirements

Meeting clearly defined availability requirements is principally a matter of configuring for availability. For example, base systems can be configured for availability by adding extra power supplies and I/O controllers, or by mirroring disks. Redundant system components minimize single points of failure in the hardware configuration. To take advantage of software enhancements such as Alternate Pathing and Dynamic Reconfiguration, configuring redundancy into the system becomes extremely critical.

Even higher levels of availability can be achieved through further redundancy — for example, adding an entire system as a hot spare backup — at an additional cost.

All of these approaches increase availability, but equally expand system cost and complexity. As with any other critical business issue, availability decisions must be made based on the cost of downtime versus the investment required to adequately safeguard against it.

Architecture 2

This chapter provide an overview of Dynamic Reconfiguration (DR), explaining in a broad sense what DR is and what functionality it delivers. It discusses when DR can or should be used and when it can't or shouldn't, provides hardware and software background information, and defines terms used in the remainder of the document. A list of requirements for making a system capable of DR is also included.

What is DR?

DR is an operating environment feature that provides the ability to replace and reconfigure system hardware while the system is running. This feature is optional and can be implemented at the discretion of the system administrator. The main benefit of DR is that a service provider can add or replace hardware resources (such as CPUs, memory, and I/O interfaces) with little interruption of normal system operations.

DR is available for Sun system architectures that contain multiple system boards and use board slots that support hot-plugging. The DR features described in this guide are specific to Sun Enterprise 6500, 6000, 5500, 5000, 4500, 4000, 3500, and 3000 systems. (DR is also possible on the Sun Enterprise 10000, however this uses a different framework that is not discussed in this document.) These features may not apply to other types of server systems. The specific hardware supported with DR depends on the version of Solaris running on the server. (See the "Requirements" section on page 25 for more details.)

In short, DR is a tool that allows the administrator to insert and remove system boards and bring them on- and off-line in a live system.

The key benefits that DR provides are:

- *Resource Management* — The ability to dynamically change the system hardware setup to meet the changes in the system load, thereby eliminating the need to reboot to reconfigure the Solaris device-tree.
- *Availability* — The ability to minimize system service disruption improves the system uptime. WIth DR, the system freezes briefly when adding or removing hardware, but it does not require the system to be halted or rebooted.

Planning plays a major part in successfully using DR, and knowing the system setup is essential. This includes being aware of which systems there are, what their functions are, what applications they run and which components they contain (i.e. system boards, expansion cards and software). It should be clear before starting which systems can utilize DR, and which components can be dynamically added or removed, and under which conditions. This knowledge also makes it possible to have qualified personnel available for DR operations.

One common misconception is that DR provides fault tolerance. DR by itself does *not* provide fault tolerance. In order to achieve fault tolerance, the system must have a way to recover information in the case of a failure and DR does not provide that. DR is a tool to avoid or shorten the downtime of a system, not to guarantee uptime.

When to Use DR

There are various scenarios in which DR helps keep the system running and available when you would normally have to take the system down. The different scenarios and simple examples are listed below.

- Adding hardware, if the system has room to grow
  - *Adding CPU/Memory Boards* — Assume an E4500 application server has six CPUs but is being overloaded because of insufficient CPU power. With DR you can add and configure a new CPU/Memory Board with two extra CPUs while the system is running. This can help meet the performance demands, assuming the application can dynamically incorporate the added CPU power.

    - *Adding SBus I/O Boards* — In a similar fashion, an SBus I/O Board occupied with the necessary SBus cards can be added to the E4500. For example, an X1049A SBus Quad Fast Ethernet Card can be inserted to add more networks to the server, or an X1065A SBus Differential Ultra Wide SCSI card can be added to allow a StorEdge D1000 to be connected. (DR is not supported with PCI I/O boards, however.)

- Removing hardware, if the system either doesn't need a part anymore or can donate it to another system
    - *Removing unneeded hardware* — Assume an E3500 application server has eight CPUs but is actually over-configured, while an E4500 database server next to it needs two extra CPUs. DR enables you to remove the CPU/Memory Board with the CPUs on it from the E3500 and put it in the E4500.
- Replacing hardware, often a combination of removing and adding
    - *Replacing Memory or CPUs* — Assume an E5500 is giving correctable memory faults. The responsible memory bank is traced, and the corresponding CPU/Memory Board is taken off-line. The board can now be taken out of the system and have its memory replaced. The board is inserted back into the system and configured. This can also be done with CPUs in a similar fashion.
    - *Replacing NICs or Storage Cards* — Assume an E4500 has an SBus I/O Board whose X1059A SBus Fast Ethernet Card is giving problems. The network traffic is Alternate Pathed to an other Ethernet connection, redirecting the information stream through another Ethernet connection on another system board. Then the SBus I/O Board with the failing Ethernet card is unconfigured and unplugged. The card is replaced and after the board is added to system again the network traffic is re-routed back to this NIC. The same can be done with the Fiber Channel storage. In the case of SCSI storage, there has to be a mirror on an alternate I/O board.
- Upgrading hardware, where the system board is removed and new hardware is added
    - *Upgrading Memory* — Assume an E3500 has three CPU/Memory Boards, two of which are totally populated with two GB of memory while the third has only one bank filled with 128 MB DIMMs supplying only one GB of memory. This CPU/Memory Board can be taken off-line and

unplugged, its second bank filled with the X7023A 1 GB memory option, and then re-inserted into the system and reconfigured. This can, of course, also be done when a 256 MB bank is replaced by a one GB bank.

  - *Upgrading the CPUs (the Ecache)* — Assume an E5500 has eight CPUs, two of which are the X2570A 400 MHz with four MB Ecache and six are the X2580A 400 MHz with eight MB Ecache. This is supported as long as the different Ecache sizes are on separate boards. The CPU/Memory Board with the four MB Ecache modules can be taken off-line and pulled out of the system. The X2570As can then be removed and upgraded to the X2580As. The board is then placed back into the system and reconfigured.
  - *Upgrading Systems Boards* — Assume an E3000 has four CPUs, the X2550A 250 MHz four MB Ecache modules. However two of them are on an X2600A that supports up to two MB Ecache, so the modules on this CPU/Memory Board can only use two MB of their four MB Ecache. By taking this board off-line and out of the system the CPUs and the memory can be placed on an X2602A CPU/Memory Board. When this board is placed in the system and reconfigured the same CPUs have doubled their Ecache. In a similar way other types of boards can be upgraded or exchanged.
  - *Upgrading Storage Capacity* — Assume an E4500 has an StorEdge A3500 connected to two X1065A Differential Ultra Wide SCSI Cards on separate I/O boards. To add an A3500 to this configuration on the same I/O boards, first one of the SBus I/O Boards is taken off-line after routing all storage traffic through the second board. Then the board is taken out of the system and an X1065A is added. Next the I/O board can be put back into the system and brought back on-line. Both A3500s can now be connected to this board. Now there is one I/O board with two X1065As and another with only one. The whole sequence can now be repeated on the second I/O board, resulting in both A3500s connected to both I/O boards.

- *Creating a HA environment* — Assume an E5500 has four CPU/Memory Boards, each with two CPUs and some memory. Memory interleaving is set to minimum, a required setting if the boards are to be Dynamically Reconfigured. There are two SBus I/O boards, each with the onboard Ethernet connection used. Finally two A5200s connected to the I/O boards, either directly or via hubs, are each other's mirror. Every time something is added in such a redundant way the system can use DR capabilities to become more highly available.

When Not to Use DR

There are certain cases where DR is not an option, or is simply not a very good idea. It is also possible that only parts of the system can be used with DR.

The following are unsupported:

- *Clustering* — Currently DR is unsupported in combination with Sun Cluster software. During certain DR steps the system can appear to be dead to the outside world, and this can cause the other side of the cluster to fail over (undesired behavior in this case).
- *Upgrading CPUs (the MHz)* — There is a scenario where CPUs in a running system are replaced by faster (in MHz) CPUs in such a way the down time is minimized. In this case only the first two CPUs are replaced while the system is down, then a reboot allows the system to boot at the faster speed. However mixing CPU speeds in *not* supported in a running system.
- *Vital hardware* — It is not possible to Dynamically Reconfigure vital hardware. The vital hardware components include:
  - The *primary CPU*, which is the first CPU on the first CPU/Memory Board in the system.
  - The memory space where the Open Boot PROM (OBP) and the Kernel reside, also referred to as the *kernel cage*. This is located in the highest memory addresses, and therefore isn't necessarily on the same system board as the primary CPU.
  - The *primary network interface*, which is the interface that caries the same name as the system itself. This is vital only if the system is networked and this interface is not Alternate Pathed.
  - The *Boot Disk*, if it is not Alternate Pathed or mirrored.

These are partially supported:

- *Unsupported hardware* — There are two levels of unsupported devices:
  - The device supports the needed system freeze called quiesce, but sits on a system board that does not support the hot plug and/or it does not support the attach/detach needed for the hot plug;
  - The device does not support the freeze and the attach/detach.

  In the first case, DR is supported but just not with these parts of the system. In the second case, DR is not possible on this system.

The first case occurs, for example, when a system has a PCI I/O Board with PCI cards on it that support the quiesce, however the PCI I/O Board itself cannot be Dynamically Reconfigured. The second case occurs, for example, when the system has an SBus card that does not support DR at all. More information on attach/detach is included in the “Definitions” section on page 17.

- *Using interleaved memory* — Boards that have memory interleaved can *not* be removed with DR. However, adding a system board with new memory to a system with interleaved memory is supported. In this case the new memory is *not* included in the interleave. Interleaving memory across boards makes them interdependent, thereby making it impossible to take one out without affecting the other. Interleaving within a board, however, is not a problem. Turning memory interleaving off between boards can have negative performance implications that should be considered.

It is not wise to do the following, although they are not unsupported:

- *Running real time processes* — It is undesirable to have the system freeze when running real time processes. During DR operations the system does freeze briefly, endangering the reliability of these types of applications.
- *Using VxVM with DMP* — There are interoperability issues with DR and Veritas Volume Manager™ (VxVM) with Dynamic Multi Pathing (DMP) turned on. An alternative to DMP is Alternate Pathing, which is supported but is not as dynamic and does not perform load balancing.
- *Unplanned DR with unqualified personnel on production systems* — It should be clear that DR is a tool for changing a system, when needed, and should be used selectively and only on a planned basis by trained personnel. Using DR can minimize the down time on a system but does not guarantee zero down time. If something goes wrong with the hardware, like a bent pin, the system can panic. Therefore, changing a production system should always be performed in as quiet a moment as possible.

Hardware Background

DR requires both hardware and software support, and design choices must be made concerning how much is implemented in each layer. There is even an intermediate zone between the two that is the firmware, software embedded in the hardware components. Having more elements embedded in the hardware generally makes the implementation faster although less flexible for change or error correction.

In the case of DR for the E3000-6500 servers, the emphasis is in the firmware and software. The minimal DR hardware requirements were included in the design of the first versions of these servers, however the Solaris 2.5.1 software did not yet have these capabilities. Having the hardware foundation present made it possible for software engineers to add DR capability at a later stage.

There are four elements in the hardware design that enable DR capability:

- The general design of the connectors on the back of the card make it hot pluggable. Part of this design includes certain pins that are longer than others, meaning that these will connect earlier than the others. This also helps to ground the board for static electricity.
- Some of the pins that are longer are for the power supply to the board. This, in combination with a feature called pre-charge, makes it possible to insert the board without disturbing the voltage levels of the rest of the system.
- The boards are also able to electronically synchronize with the bus on the backplane, making it possible for the board to adapt to the different clock speeds that the Gigaplane™ system interconnect can have.
- Every board has a JTAG (Joint Test Action Group) connection to the backplane. JTAG, an IEEE standard for testing electronic circuits, allows the system to test the boards for functionality and to diagnose it. This is done a boot time but can also be performed when a new board is inserted.

The way this all comes into play when inserting a board is further discussed in the “Mechanisms” section on page 18.

Software Background

The software model of a modern UNIX server contains many layers (see Figure 1). One of the benefits of this model is that it makes a very complex system manageable. It also ensures a form of security in that different parts of the system can only access and control what they need to. Thirdly, it also allows platform independence, when the same interfaces are used between the layers.

In this way the higher the application is in the layers, the more platform independent it can be. In this section both firmware and the software running on top of that, the Operating System, are discussed.

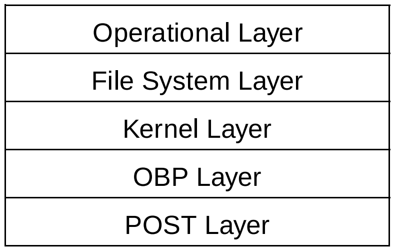


Figure 1 A model of the different layers that keep a representation of the system.

The following sequence of steps describes a simplified version of the software boot model, when the system is booted and told to search for new devices.

1. At boot time the first software run is POST (Power-On Self Test), which is run by the boot PROM using the JTAG interface. This tests and diagnoses all hardware components. It then loads the Open Boot PROM (OBP) software into memory and activates it.
2. The OBP software builds a device tree from the information the POST acquired and then starts the boot by finding the boot device. It then loads the Solaris kernel into memory and activates it.
3. The kernel uses the device tree created by the OBP and creates a system structure and set of environment variables with which Solaris can further boot up.
4. The other components in the Solaris operating environment, like `drvconfig(1M)` and `disks(1M)`, are then run to create the `/devices` and `/dev` directory trees and map the devices so they can be used by applications. When a system is booted normally this step skipped because the entries in those directories are still there from the “reconfigure boot”.
5. Finally, these devices are used to further setup the system so it is operational. This is done by commands like `mount(1M)` and `ifconfig(1M)`.

Now it's clear that at the very beginning of the boot process the system investigates itself and then uses this information as a foundation to build the higher layers. During this process the system keeps several representations of itself, each different and constructed in such a way that the higher layer can use it.

Before DR, the only important goal was to correctly create (boot) the system in such a way that all elements were represented and were usable by applications running on the operational layer. The only modification to this structure available without rebooting was possible in the operational layer, performing functions such as mounting file systems or reconfiguring network devices.

This is in contrast to adding or removing resources, which implies that the representation of the system itself has to be altered. Because the system has built several representations, all must be altered in the right sequence to preserve the integrity of the system.

Another new element is the ability to disable and detach hardware from the system. This means that the system must have a way of taking a hardware device out of operation and halting it. This has to be done before hardware can be removed from the different representations. For example, a system board with a SCSI controller may need to be removed, but the SCSI controller might still be in use because a disk connected to it is still mounted and still active. In this situation, the DR software refuses to unconfigure the system board; the disk must first be unmounted. And even then the SCSI controller must also be able to be detached. A discussion of what is detachable will follow later in this chapter.

To implement DR, new software has been added to Solaris at various levels. At the user level, an administrator uses the shell command `cfgadm` to transition a system device from one state to another.

The `cfgadm` command, the top-level user interface command, uses a library called `libcfgadm.so` to push commands to or to pull status information from system boards. This library is also a generic, which means that other applications have the opportunity to use these as handles for DR use. This library turns to one or more platform specific libraries to operate the platform specific commands. In the case of the E3000-6500 servers, there are two platform specific libraries:

- `sysctrl.so.1` — This library talks to the CPU and ndi (nexus device interface) controls. The ndi controls the system boards.

- `ac.so.1` — This library talks with the address controller (ac), which in turn talks with kphysm which controls the memory.

These libraries lie on top of the OS which can get information from OBP. The OBP in turn received its information from POST. This is all portrayed in Figure 2.

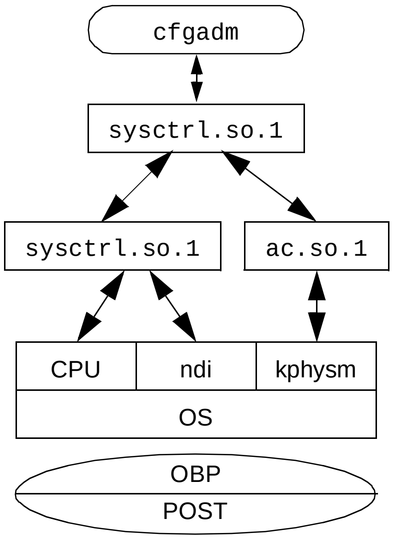


Figure 2 An overview of the DR interfaces.

There is one final item that needs to be discussed in the context of the software background — the *kernel cage*, which is the part of memory that holds the OBP and the kernel memory. These two elements in the memory are vital to the operation of the system and cannot be simply moved; the operating system depends on it. The parts of memory where these two are located, and are therefore marked as permanent memory, cannot be touched by DR. If these two pieces were located on two different system boards, both would be prevented from participating in DR. This is why the kernel cage was implemented for CPU / Memory Boards. This cage holds these two vital parts of memory together on a certain spot in memory. This spot is currently a static place in the highest memory addresses of the system, generally on the last CPU / Memory Board.

Definitions

When discussing DR procedures and mechanisms it is important to have a clearly defined set of terms that represent devices and their possible states. Listed below are definitions that are used in the rest of the document. These definitions are greatly focussed on the E3000-6500 server implementation of DR. In the future, these definitions may change as new system capabilities increase or improve.

- *Receptacle* — A slot. Is this document a receptacle can be a system board slot or a memory bank. The system board slots vary in number for different models (e.g., an E3500 has 5 slots). There are always two memory banks per CPU/Memory board.
- *Occupant* — A reconfigurable component that can go into a receptacle. In this document, occupants are the system boards and the memory DIMMs. There are different types of system boards, like CPU/Memory boards and SBus I/O Boards. Memory DIMMs vary only in capacity.
- *Attachment point* — Each attachment point is represented by a UNIX device node in the `/devices` directory. An example of an attachment point device path is:
  `/devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot0`

  There is one attachment point per receptacle-occupant pair or device. An attachment point is named after the slot that it controls.
- *Logical attachment point* — A shorthand version of the physical attachment point mentioned above. An example of a logical attachment point is: `sysctrl0:slot0`.
- *Hot Plug* — The ability to physically add or remove a component on a powered-up system without disrupting system operation.
- *Memory Interleaving* — The ability to bundle several memory banks in such a way that they are accessed much like disk striping. Memory interleaving can improve performance by reducing the average latency when reading from or writing to memory. Two options are possible: bundling the memory banks on one board, or combining memory banks of different boards. The latter option links the two (multiple) boards and makes them interdependent, and currently this disables them from participating in DR.

- *Quiesce* — When adding or removing a board, the system is "frozen" for a few seconds, or sometimes up to a minute. This is known as operating environment *quiescence*, or "to *quiesce* the operating environment".
- *Suspend/Resume* — To get a system safely in quiescence, all devices must be *suspended* so they don't create bus traffic while in quiescence; afterwards they need to *resume*. This capability is dependent on the device driver. It is important that all devices drivers on systems using DR can do this and thus are *suspend-safe*. In this case the device supports `DDI_SUSPEND` and `DDI_RESUME`.
- *Detach/Attach* — To add or remove a device to a running system, the device driver must support `DDI_DETACH` and `DDI_ATTACH`. Devices with this capability are called *detachable*. In order to add or remove a system board, all devices must have detachable drivers.

Mechanisms

This section is closely related to the "Preparing the System for DR" section on page 48 and the "Checklists" section on page 79.

To understand what is needed for DR, it is important to first comprehend the sequence of events that take place in the system. Every board in the system is in a certain state, normally up and running. Finding out the current state of a board in the system helps determine which DR phase a system board is in and where it can go. There are also different perspectives that can be looked at, like hardware or software. Plus, there are different zoom levels — that is, looking at the process in a high level way or going into fine details such as the exact commands to use.

This section starts by providing an overview of the DR phases. It then goes through the steps, first concentrating on the hardware and then discussing the software steps.

DR Phases

DR on the Sun Enterprise 3000-6500 has distinct phases, resulting in a fairly disjoint flow from start to finish. This is in contrast to the semi-automatic PCMCIA process of adding or removing of a card. Having a phased implementation has one drawback in that the administrator has to go through multiple steps. However, it does provide the administrator with greater control. In addition, DR should be a rare and well-planned operation, and in those cases it is good to have insight and control. Although it is currently possible to combine steps or to create a script, this automation will become more important in later DR implementations when reconfiguration is more common.

Since this is the first implementation of DR on this range of Sun servers, the model used is fairly simple. System boards — the only hot-pluggable component — are the center of this model. Other devices can be added, but only if they are located on a system board that is itself being added. Furthermore, the devices on the board can only be accessed when the board is configured. This results in a model with a straight line going from an empty system board slot to a system board with all its devices up and running.

Each individual board can be in six distinct phases, as shown in Figure 3. These phases are similar to those that occur when the system in booting up, because a new board must be added and configured in the same way as boards already in the system. Currently all movement between these phases is initiated by the administrator. Earlier (see the “Software Background” section on page 13), this paper described the different representations the system builds of itself in the different layers. The DR phases duplicate the same Power-on through system boot up phases.

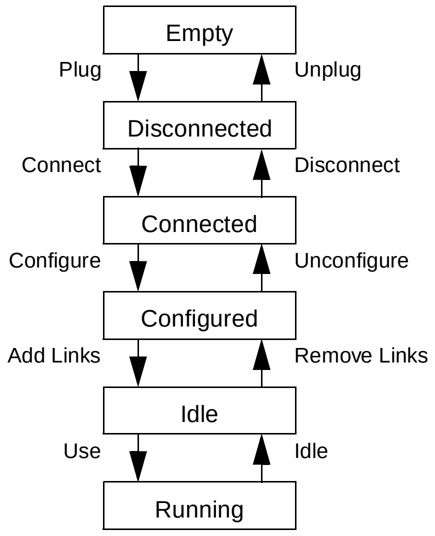


Figure 3 A model of movement between DR phases.

The following steps are taken when moving from an empty slot in a server to having a board ready and running in the system:

1. The sequence starts in the *Empty* phase, with no system board in the slot. The board is plugged into the system. When the lights on the board go on, the board is now *Disconnected*. The system is interrupted very briefly while inserting the board.

2. When moving from the *Disconnected* to the *Connected* phase, the system goes from the POST layer to the OBP layer. The system builds the POST device table by testing the devices to see if they are compatible and usable. Then the OBP uses this information to add devices to the system device tree. When all goes well, the board enters the *Connected* phase. The system quiesces while connecting.

3. When transitioning from the *Connected* to the *Configured* phase, the information the OPB has on the new board devices is added to the kernel device tree. At this time, only the kernel knows about the new devices. The board is now *Configured*.

4. When moving to the *Idle* phase, the system makes the devices known to the whole environment and adds their references to the file system. To accomplish this, the kernel starts the device drivers for the new devices. These drivers in turn create new links in the file system. At this point the devices on the board are visible but not in use. This state is *Idle*.

5. Finally, when the devices on the board are taken by applications, mounted, or otherwise put to use the board is *Running*.

The new board is now up and running and is an integrated part of the system. When removing a board, the system goes though the same steps but in reverse order:

1. Idle the components on the board by unlinking or unmounting the I/O and clearing memory and CPU of any information or processes. This is necessary so that no processes are counting on this board's resources when it is removed. The board is made *Idle*.

2. Optionally, remove any links to the board in the file system, so there are no references to the board that other resources or processes can acquire. The board is now simply *Configured*.

3. Removing any links to the board in the kernel unconfigures the board and it is only *Connected*. In this state, only the OBP layer has any knowledge of the board. POST only runs when the board is initially connected, its knowledge goes into the OBP and the POST layer halts.

4. After disconnecting the board with a command, the board is still physically present but is ready to be taken out of the system. It is now in the *Disconnected* state.

5. If the board is now physically removed, the slot is *Empty*.

Information on how to check if movement from one state to another is possible and details on the specific commands are covered in later sections of this document.

The beginning phases are dominated by configuring the system board, while later phases are more dominated by configuring the devices on the board. In the first phase there is no board — only an empty slot or receptacle. In the next few phases the board is the occupant in this receptacle. But at the end of the DR process, the system board itself has become a receptacle and its devices are the occupants on the board. In this way it becomes clear that there are actually different layers, like puzzle pieces, each clicking into the other.

Currently, later puzzle pieces can be laid only if the first are already there. That is, a system board can only be configured if it is plugged in and then connected. This single dimensional model is sufficient today. However, if machines become more complex and capable, the model may need to be changed to reflect this.

Hardware/Firmware Steps

As can be expected, the lowest levels of the DR phases are accommodated by hardware. However this is a relatively short part of the DR operations and the firmware on the boards quickly take up the next steps.The steps taken by the lower layers are as follows:

1. The board is inserted and the system senses the longest pins.

2. The system halts all traffic on the Gigaplane bus.

3. All pins are now connected.

4. The system pauses briefly to let everything settle down and then resumes the traffic on the bus. This typically takes 0.7 seconds.
5. The JTAG interface is used to check the board type and get the board compatibility status.
6. The JTAG interface is used to set the board to “power off” state. This prevents the devices on the board from interacting with the rest of the system.

The board (or slot) has now gone from the *empty* phase to the *disconnected* phase. Everything on the board is now ready to undergo tests and be added to the system logically, or removed again.

The next step is going to the *connected* phase. This is done by powering up the devices on the board, syncing the new board to the bus clock, and running POST. To not compromise the other boards during this operation, the system goes into quiescence during the POST run. The POST run-time depends mostly on the board type and the number of devices on the board. A CPU/Memory Board takes much longer to run through POST than an SBus I/O Board.

Being connected doesn't mean that Solaris knows what is on the board. The board is now ready for the OBP to be run. OBP maps the devices and then checks if all are compatible and usable. The disconnected to connected phases are executed by the firmware.

Software Steps

From this point forward, Solaris uses the foundation laid by POST and OBP to add the appropriate links so that applications can use the new devices. In a certain way, it is as if Solaris will “boot” the new board.

When the system boots, it runs through a sequence of automated steps to configure the devices. These steps can be modified by changing settings in certain files. With the current DR, these steps are not automated. The administrator must go through the sequence of steps needed to get these devices added and usable, and so it is important to know what type of board is being added and what type of devices it contains.

First the kernel uses the device tree created by the OBP and maps its own internal representation of this tree. This enables it to load drivers for these devices and use them. Once it sets up this kernel layer representation, the system board is now in the *configured* phase.

In order for processes outside the kernel to use the new devices, these devices must be linked into the file system. This new link is also called an attachment point. This is done by the system when the drivers for the new devices are loaded. The process of loading drivers and creating links by the kernel is started on request of a user command. Now the devices are *idle*, but ready to be used by other processes. This is the case for all devices, except for the CPUs, which are powered-off but ready to use the moment the board is configured.

In the case of memory, it must be added to the file system before it can be tested by the DR software and made ready for use. This can be done by the DR software because memory is also seen as a receptacle-occupant pair in the current DR implementation.

The devices on the SBus I/O Board must also added to the file system by running specific commands to identify disks, tapes, serial ports and other devices. Once this is done, all devices on the board are *idle* and ready to use.

Finally the devices can be put to use. Memory is tested and set to use, disks can be mounted, network links can be set up, and other devices can be used. This is an administrator-intensive part of the DR operation, mainly because every machine has its own setup and use. The CPUs and memory, however, generally have no extra complications and are set to *running*/configured to be used.

At this point, other applications — like the Alternate Pathing software, Solstice™ DiskSuite™, Veritas Volume Manager and Sun Resource Manager™ — can take over and use the newly acquired resources. Sun Management Center can also be used with the new resources.

Having scripts perform many of the commands is possible for many steps, and is especially trivial for the CPU/Memory Boards. However, the last steps for I/O boards can be more challenging.

Removing the Board

When a board needs to be removed, the administrator must go through the same steps in reverse. Most of these steps are a simple command, and it is even possible to do more than one step at once, making removing a board for the most part easier than adding one. This is the case for going down from the *idle* phase to the *empty* phase. However getting a board with devices in use to the *idle* state needs the most attention. Many times applications running on the system hold resources that need to be Dynamically Reconfigured out.

Freeing a CPU or memory of processes and information and making it idle can generally be accomplished with a single command. However, if there are processes bound to this CPU (e.g., a process is bound to this particular CPU with the `pbind` command), some extra work is required. In this case, the process must free the CPU before it can be made idle.

I/O devices require more work. The administrator needs to find out if any processes are using the resources on the particular board. If they are, they either need to be re-routed to an other board (with Alternate Pathing software, for example), or if this is not possible the process needs to be removed. A mounted file system needs to be unmounted and removed from Volume Manager or DiskSuite. A network connection needs to be unplumbed. If there are any resources that use the board, the unconfiguration will fail and the board will not go into the *connected* phase.

Secondary processes should not be over looked. For example, if a file system is unmounted and not alternate pathed to another disk, and a user is expecting it to still be there, this can create problems for the user.

If a Step Fails

There is always a danger that movement from one state to another will fail. This can be due to several reasons, most related to incompatible or broken hardware. When adding a board, a failed step indicates there is something wrong with the board. The only solution is removing the board, replacing the failed part, and starting over. When removing a board, failure only occurs if the board's resources are still in use. In that case the board was not actually in the *idle* state but still in the *running* state. Only from the *idle* state can the board move on down. Failing is discussed later further in the “Failing, Failed, Unusable” section on page 76.

Requirements

Making a good assessment of whether a system can use DR for its benefit starts with checking if the system and its surroundings meet all the requirements. Some of these requirements are hardware and software based, while others are environmentally based. The three most important requirements are having the right personnel, good planning and hardware and software support.

The first requirement that should be met is the need for capable and certified personnel. Being familiar with the system setup and the DR process and commands is crucial. One of the main reasons for using DR is to prolong uptime. If the administrator is not familiar with the circumstances, there is a risk of unscheduled down-time that was the goal to avoid. In addition, the physical handling of the boards should be done only by Sun certified personnel.

Good planning is the second requirement. This includes determining which parts of the system support DR. Also needed is a clear understanding of how the system is setup, so that there will be minimal impact on users when a board is removed. Furthermore, when DR is used — planned or unplanned — the right personnel must be there to perform the necessary steps. A smooth operation minimizes the impact on the users.

The third requirement is the compliance of all the parts of the system with the list of Sun supported hardware and software. If a part is going to be dynamically configured, it must be supported. Other parts that are simply in the system are permitted, but cannot be Dynamically Reconfigured.

Hardware

It important to make a distinction between system boards that contain devices that support *only* quiescence (suspend/resume) and those that support quiescence *and* detach/attach. Devices that support only quiescence (suspend/resume) are referred to as *suspend-safe*. Devices that support both quiescence and detach/attach are called *detachable* devices or *Hot Pluggable*.

Similarly, system boards that support quiescence *and* detach/attach are called *detachable* devices or *Hot pluggable* boards. This capability is required if the board will physically move in or out the system. An up-to-date list of supported boards and devices is found at:
http://sunsolve5.sun.com/sunsolve/Enterprise_dr/

The currently supported systems, boards and devices are detailed in the following sections.

Systems

Sun Enterprise 3000, 3500, 4000, 4500, 5000, 5500, 6000 and 6500 servers are all supported, including all types of chassis. The HPC versions of these machines are not supported.

System Boards that Support Hot Plug

Type of Board	Market Part #	Sun Part #
CPU/Memory (<2MB, 83 MHz)	X2600A	501-2976
CPU/Memory (83 MHz)	X2601A	501-4312
CPU/Memory (83/90/100 MHz)	X2602A	501-4882
SBus I/O (SOC, 83 MHz)	X2610A	501-2977, 501-4287
SBus I/O (SOC+, 83 MHz)	X2611A	501-4266
SBus I/O (SOC+,83/90/100MHz)	X2612A	501-4883

Table 1 System boards that support Hot Plug in Solaris 7 5/99.

Only the CPU/Memory Boards and the SBus I/O Boards are Hot Pluggable. These boards do not require any patches in Solaris 7 5/99. However, some boards are not supported at all in earlier versions of Solaris, or may require one or more patches. Please see the “Supported Devices” section on page 91 or check the Web site mentioned at the beginning of this section for more details.

System Boards Not Supported for Hot Plug

Graphics I/O Boards and PCI I/O Boards

Type of Board	Market Part #	Sun Part #
Graphics I/O (SOC, 83 MHz)	X2620A	501-2749, 501-4288
Graphics I/O (SOC+, 83/90/100 MHz)	X2622A	501-4884
PCI I/O (83 MHz)	X2630A	501-3023
PCI I/O (83/90/100 MHz)	X2632A	501-4881

Table 2 System boards that support only quiescence (and not Hot Plug) in Solaris 7 5/99.

The Graphics I/O Boards and PCI I/O Boards currently don't support Hot Plug. If the devices on these boards support quiescence, the board as a whole will support quiescence. These system boards do not require any patches in Solaris 7 5/99. However, some boards are not supported at all for earlier versions of Solaris, or may require one or more patches. Please see the "Supported Devices" section on page 91 or check the Web site mentioned at the beginning of this section for more details.

DR is not needed for any of the Disk Boards for these systems. The Disk Boards only use power from the backplane and they have a JTAG interface. Other than that there is no logic involved, because they are connected externally via the external SCSI connections on an I/O board. The JTAG interface is only for optional probing of the board, but this does not bind into the system.

The system boards can also be classified in the following way:

Type	Name and Identifying Characteristics
CPU/Mem	CPU/Memory board
Disk Board	System board containing two disk drives
Type 1	Dual SBus I/O board with 3 SBus slots
Type 2	Graphics, SBus-UPA I/O board with 2 SBus slots and 1 frame buffer slot
Type 3	Dual PCI I/O board with 2 PCI card adapter slots
Type 4	SOC+ SBus I/O board with 3 SBus slots
Type 5	SOC+ Graphics UPA I/O board with 2 SBus slots and 1 frame buffer slot

Table 3 Board types.

All other non-system boards and modules, like power-cooling modules, do not involve DR. They are either hot pluggable on their own, like the power-cooling modules, or they are not removable at all, like the clock board. In any case, this is outside the scope of DR.

Expansion Cards

Expansion cards can be classified into the following categories:

- SBus cards that support detach/attach and quiescence
- SBus cards that support only quiescence
- PCI cards that support only quiescence
- UPA cards that support only quiescence

Supported cards for each of these categories are listed in the following tables.

SBus Cards that Support Detach/Attach and Quiescence

Type of Card	Market Part #	Driver
SunSwift, SE Fast/Wide SCSI, Fast Ethernet 10/100BaseT	X1018A	fas/sd, hme
Gigabit Ethernet 1.0 1000BaseFX	X1045A	ge
Quad Fast Ethernet 2.0 10/100BaseT	X1049A	qfe
Differential Fast/Wide SCSI, Buffered Ethernet 10BaseT	X1052A	esp/sd, le
Single-End Fast/Narrow SCSI, Buffered Ethernet 10BaseT	X1053A	esp/sd, le
Fiber Channel, FC-25	X1057A	soc/pln/ssd
Quad Ethernet 10BaseT	X105L8A	le
Fast Ethernet 2.0 10/100BaseT, MII	X1059A	hme
Differential Fast/Wide SCSI	X1062A	isp/sd
Single-Ended Fast/Wide SCSI	X1063A	isp/sd
Differential Ultra/Wide SCSI	X1065A	isp/sd
Gigabit Ethernet 2.0 1000BaseFX	X1140A	ge
FDDI 6.0 Single Attach	X1142A	nf
FDDI 6.0 Dual Attach	X1143A	nf
High Speed Serial Interface	X1145A	hsi
ATM 4.0 155/MFiber	X1147A	ba
ATM 4.0 155/UTP5	X1148A	ba
ATM 4.0 622/MFiber	X1149A	ba
TurboGX 8-bit color graphics	X3655A	cgsix
Fiber Channel-Arbitrated Loop, FC-100	X6730A	socal/sf/ssd
TurboGX Plus 8-bit color graphics	X7110A	cgsix

Table 4 SBus cards that fully support DR in Solaris 7 5/99.

The cards listed in Table 4 fully support DR in Solaris 7 5/99. For the list of earlier versions of Solaris please see the “Supported Devices” section on page 91, or check the Web site mentioned at the beginning of this section.

SBus Cards that Support Only Quiescence

Type of Card	Market Part #	Driver
Token Ring Interface 4.0	X1144A	tr
Serial Parallel Controller	X1146A	spif

Table 5 SBus cards that support only quiescence in Solaris 7 5/99.

These cards can be in the system while performing DR but they cannot be on a system board that is being added or removed. This list is for Solaris 7 5/99. Some boards are not supported in earlier versions of Solaris, or may require one or more patches. Please see the “Supported Devices” section on page 91 or check the Web site mentioned at the beginning of this section for more details.

PCI Cards that Support Only Quiescence

Type of Card	Market Part #	Driver
SunSwift, SE Ultra/Wide SCSI, Fast Ethernet 10/100BaseT	X1032A	fas/sd, hme
Fast Ethernet 2.0 10/100BaseT, MII	X1033A	hme
Quad Fast Ethernet 2.0 10/100BaseT	X1034A	qfe
Gigabit Ethernet 1.0 1000BaseFX	X1044A	ge
Gigabit Ethernet 2.0 1000BaseFX	X1141A	ge
FDDI Single Attach	X1152A	pf
FDDI Dual Attach	X1153A	pf
Token Ring Interface	X1154A	tr
High Speed Serial Interface	X1155A	hsi
Serial Asynchronous Interface	X1156A	sai
ATM 6.0 155/MFiber	X1157A	ba
ATM 6.0 155/UTP5	X1158A	ba
ATM 6.0 622/MFiber	X1159A	ba
PGX8 Graphics Card	X3660A	m64
PGX32 Graphics Card	X3668A	gfxp

Table 6 PCI cards that support quiescence in Solaris 7 5/99.

The PCI cards listed in Table 6 are suspend-safe and support quiescence. This is the only possible classification of PCI cards, as the system boards they go on do not support Hot Plug in this version of DR. This list is for Solaris 7 5/99. For a list of earlier versions of Solaris please see the "Supported Devices" section on page 91, or check the Web site mentioned at the beginning of this section.

UPA Cards That Support Only Quiescence

Type of Card	Market Part#	Driver
Creator Graphics	X3653A	ffb
Creator 3D Graphics	X3669A, X3675A, X3671A	ffb

Table 7 UPA cards that support only quiescence in Solaris 7 5/99.

The UPA cards listed in Table 7 are suspend-safe. This is the only possible classification of UPA cards, as the system boards they go on do not support Hot Plug. This list is for Solaris 7 5/99. For the list of earlier versions of Solaris please see the "Supported Devices" section on page 91, or check the Web site mentioned at the beginning of this section.

Third party boards

Third party boards — either on a board being detached/attached or only suspended — are not covered in this document. Please refer to the vendor of the card for details. If the vendor can not assure support, then it is unwise to use this card. The cards in the lists above have been intensively tested to ensure there are no issues.

If, however, the card is imperative and the vendor can't help, testing the card in a development system is an alternative. This testing consists of first doing several suspend/resume operations, then several disconnect (detach/attach) operations and finally several unconfigure/disconnect/unplug operations. This has to be done at least a dozen times, because certain problems only appear after a series of DR operations.

All Sun cards not listed above should be treated as unsupported. If there is no option, then self-testing like the third-party cards is a possibility, although it is considered very unwise.

Software

The software requirements for DR are less complex than for hardware because there are fewer variations. Having the right version of the software and in some cases the appropriate patches is all that is needed for software. However, checking the latest patch level on an on-going basis and ensuring that the system is on this level is always advisable.

Solaris

The Solaris version defines what types of boards are supported with DR on the E3000-6500 server. Before Solaris 2.6 5/98, DR is not supported on any E3000-6500 server. With Solaris 2.6 5/98 DR is supported on the SBus I/O Boards, and support for the CPU/Memory Boards is added in Solaris 7 5/99 (see Table 8).

OS Version	Firmware	DR Support
Solaris 7 8/99	Ver. 3.2.22	CPU, SBus I/O, SBus I/O+
Solaris 7 5/99	Ver. 3.2.22	CPU, SBus I/O, SBus I/O+
Solaris 7 3/99	Ver. 3.2.21	SBus I/O, SBus I/O+
Solaris 7 FCS	Ver. 3.2.21	SBus I/O, SBus I/O+
Solaris 2.6 5/98	Ver. 3.2.21	SBus I/O, SBus I/O+

Table 8 DR support provided by various operating system/firmware combinations.

The kernel must also have certain settings to enable the DR support. Specifically, the following entries must be added to the `/etc/system` file:

```
set soc:soc_enable_detach_suspend=1
set pln:pln_enable_detach_suspend=1
```

To enable the memory kernel cage, needed when using DR with CPU/Memory Boards, the `/etc/system` file should also contain:

```
set kernel_cage_enable=1
```

Firmware

The system boards themselves contain software in the form of firmware, which must also be on the right version level on every board. The required firmware revision levels are listed in Table 8. The versions shown here are for the OBP on the CPU/Memory Boards. The I/O Boards contain version of FCODE, which corresponds to the OBP firmware versions. In all cases these FCODE versions should at least be 1.8.3. Patches to the most recent versions of the firmware for the boards can be found on the Web page mentioned earlier in this section. This Web page also describes how to install these latest revisions.

CPU Boards that have interleaving memory cannot be Hot Plugged. To disable cross-board memory interleaving, the firmware (OBP in this case) memory-interleave environment must be set to min:

```
ok setenv memory-interleave min
```

Adding a board to systems configured with memory interleaving *is* supported. All non-CPU/Memory Boards have no memory, and therefore memory interleaving is irrelevant. CPU/Memory Boards can be added to a running system when interleaving is turned on, but they are not added to the memory interleave (this is not possible). However, when the system reboots at a later time, the new board will be included in the interleave. And, when included in an interleave, the CPU/Memory Board *cannot* be removed from the system.

Other Issues

There are a few other issues that need to be checked when considering DR, including system fault lights on the system, the vital resources of the system and tape devices.

A group of three lights on the front and the back of the system indicate the general condition of the system. The middle of the three is an orange light, which is the system fault light. This light is on if there is a hardware fault on any of the hardware components (e.g., a broken power supply). It is wise to check what is causing this light to be on and solve this problem before starting a DR operation. For example, if the peripheral power supply is faulty, the pre-charge needed for the new board will fail. If the system has a second redundant peripheral power supply, this is not a problem.

All boards with vital resources are considered permanent and not Hot Pluggable. There are four vital resources:

- *The Primary CPU* — This is the first CPU on the lowest numbered board. In the current version of DR, the board with this CPU on it is non-detachable and therefore not Hot Pluggable.

- *The Kernel Cage* — This is the piece of memory that contains the OBP and the kernel in it. This cage is placed in the highest memory addresses at boot time. With the current version of DR, the board containing the kernel cage is not Hot Pluggable. In practice this cage is located on the board with the highest number, making two boards— the first and the last — not Hot Pluggable. A way around this is to boot with only the memory banks on the first CPU/Memory board enabled, place the cage on the same board as the primary CPU, and then bring up the remaining memory banks after the boot. This entails more administrative work, but it is easy to put this in a script.
- *The Primary Network interface* — This is the network interface that has the same name as the system itself. There are several higher level parts of the OS, like NIS and DHCP, that use this interface. If this is the case, the device cannot be idled. One solution is to use the Sun Alternate Pathing software, and have one if its devices be the primary network interface. In this way the AP software can switch over to another device on another board, thereby idling the device on the board that needs to be taken out.
- *Boot Disk* — This is the disk partition that the system boots from. It contains important directories for the Solaris operating environment, and also includes the swap partition. The boot disk needs to be available at all times, and therefore the I/O board that this connects to cannot be Hot Plugged. The way to avoid this is to use the Sun Alternate Pathing software to provide the possibility of re-routing the path to this disk via another board, and thus free the board that needs to be Hot Plugged. Another option is to use an application, like DiskSuite, to mirror the file system. In this case the mirror will have to be broken before performing DR operations. The `swap` command can also be used to spread swap over more than one disk, thereby enabling the movement of the swap.

The sequential nature of tape devices prevents them from being reliably suspended in the middle of an operation and then resumed. Therefore, all tape drivers are suspend-unsafe. Before executing an operation that activates operating environment quiescence, make sure all tape devices are closed or not in use. All tape drives connected to the system must be at the beginning of tape (BOT).

A final but crucial element is having the DR software installed. DR for the E3000-6500 servers is part of the core SunOS and therefore on the Solaris CD. If only the core Solaris packages are installed on the server, DR will be installed. This is in contrast to DR for the E10000 where there are two separate DR packages that need to be installed on the system.

Procedures 3

As stated in the previous chapter, planning is important, and understanding the way DR functions is essential for making a good plan. In the previous chapter the focus was on explaining how DR works and when it is and isn't a good idea to use DR. However, this chapter didn't discuss procedural details.

This chapter therefore goes into the details, discussing how to create policies and outlining issues to consider. It describes some typical configurations to illustrate how to meet certain needs. This chapter also discusses additional software and explains whether they can be used with DR. Finally, this chapter provides an in-depth view of the procedures administrators follow while performing DR, including the specific commands used.

Policies

This section is closely related to the following sections:

- "When to Use DR" section on page 8
- "When Not to Use DR" section on page 11
- "Checklists" section on page 79
- "System Templates" section on page 84

The first two sections discuss what can and cannot be done with DR. The other sections contain checklists and templates to help plan for DR.

Planning is critical, and having policies regarding the system in question is therefore important. Many things need to be considered when setting up a policy for DR use on a system. Every system will be in a different situation when system configuration change is considered. This section tries to help identify elements that can help establish a policy.

Specifically, three areas should be looked at:

- ***Goal***

  First and foremost, a goal must be set. The question that has to be answered is: "*What do we want to achieve?*" The answer to this question can contain more than one part, but should be a reference for the rest of the policy. It should be determined if DR can help achieve this goal, if additional hardware or software can help, and if the system has all necessary elements.

  In most cases the main part of the goal will be to achieve greater availability. This can be the only goal or be part of a larger statement like: "The goal is to keep the system running when adding more CPUs to the system."

  Money is often a major factor in setting goals. Availability is in a sense money, but so is leveraging existing hardware. Assume, for example, that it is possible to move a CPU/Memory Board from one system which has extra capacity to another that has a shortage of CPU power. If this transfer can occur while both systems are kept running, this saves money in terms of both availability and hardware expense.

- ***Types***

  Different situations mandate different plans of action. In certain cases DR is an option, while in others it is not. Identifying these different types of situations and setting out rules of engagement is an integral part of defining policies. Different types of things to consider include the following:

  - *Scheduled versus unscheduled DR* — With scheduled DR there is time to plan the operation, get the needed resources, and ensure that it happens at a quiet time, so that the impact is minimized if anything goes wrong. Although there is less time with unscheduled DR, there can still be a prepared plan for those types of occasions. When considering an unscheduled DR operation it is critical to know the risks of a failure and to try to minimize them by warning users or by backing up the most critical parts of the system. In other words, having a minimal version of a scheduled plan. Dismissing unscheduled DR altogether is also an option.

- *Planned versus unplanned DR* — Although unexpected situations are always hard to handle, some can be planned for in advance. However it is not advisable to use DR to solve these situations without having a plan. In most cases DR is not an operation that administrators perform on a daily basis, so real fluency is hardly ever achieved.
- *Who should touch what* — Having the right level administrator do the operations is important. There should always be someone present who knows the system configuration and the dependencies of the elements in the system. Existing service contracts should be considered when adding or removing a board, because in many cases only Sun certified personnel are allowed to add or remove the system boards, or even the SBus cards, memory and CPUs.
- *Different levels of emergency* — Determine if there are different levels of emergency that can be defined. For example, does a part need to be added now, or can it wait until a quieter period at night or on the weekend? By waiting, administrators gain time to verify that everything needed for the DR operation is available.
- *What needs to be in stock* — Determine which extra hardware components, if any, need to be in stock, and for which occasions. This also results in a policy to check if the needed components are really in stock.
- Checklists — Use the checklists supplied in this document (see “Checklists” section on page 79) as a starting point, adding elements as needed according to the situation. These checklists can be referred to during DR operations, to help ensure consistent and successful configuration changes.

- ***System***

  The system should always be checked before any DR operation is considered. This can be done when the system is initially set up, but can also be repeated just before a DR operation. The things that should be considered are:

  - Is *the system an E3000-6500 server?* — If the system is not one of these servers, DR is either not possible or the rules are different. An E10000, for example, uses different commands and capabilities.

- *Firmware levels of the system boards* — Every board in the system must have the correct firmware levels installed. Table 8 lists the minimal firmware levels, however the newest firmware level is always best. When adding a new board to a mission critical system, checking the firmware levels is strongly advised.
- *Solaris version* — The version of Solaris running on a server determines which boards, if any, can be used in a DR operation. Table 8 lists the capabilities provided by the various Solaris releases. If the current version of Solaris does not support DR, consider an upgrade of Solaris (after also determining what effect this would have on the applications running on the system).
- *On which parts of the system can DR be conducted*? — Use the machine templates, or something similar, to find out and register which boards can be removed from the system and where there is room to add new boards.
- *Install Patches* — Certain hardware configurations require patches to support DR. These should be localized and the correct patches installed.
- *How will the applications running on the system react*? — The system will briefly freeze when going through a DR operation. Determine in advance how will this affect the applications and users.
- *Test to establish if the system can quiesce* — Even if all lists indicate that the system should be able to quiesce, nothing beats the real thing.
- *Check for dependencies inside the system* — There are certain thing to look for, including vital resources (primary CPU, memory cage, and boot disk) and processes bound to a fixed CPU.
- *Check for dependencies outside the system* — Everything outside the system itself can be affected by the DR operation. For example, traffic over a network connection will need to be re-routed to an other board, otherwise every user or process behind this connection will be impacted. This re-routing can be done using Sun Alternate Pathing software. Dependencies can be any form of I/O.
- *No memory interleaving* — The CPU/Memory Boards that are to be removed cannot be configured for memory interleaving with other boards. This can be turned off in the OBP, but can have a negative impact on system performance.

- *Use DR on development system first* — It is very advisable to first test DR on a development system. This gives the administrators experience, can reveal issues that were overlooked, and provides an indication of how long the whole operation will take. Plus, administrators can test if turning off the memory interleaving between boards degrades performance.
- *Run backup* — Running a tape backup of the system before a DR operation helps to minimize the risk of information loss.
- *Check if a reboot is needed* — The system will probably need to reboot at least once for many settings (like turning the memory interleaving to minimal or installing kernel patches) to take effect. Rebooting now makes the configuration operational and enables DR operations to be performed at any chosen time.
- *Know the risks* — There is always the chance that something could go wrong when inserting a system board into a live system. A pin might get bent or a board might be broken, and these things can take the system down. These risks have been minimized in the E3000-6500 server architecture, but not completely eliminated. Therefore, all the points listed above are necessary, as they help provide an estimation of the risks if something does go wrong.

Links to Other Applications

DR can be used on its own to help increase the availability of a system. But DR is more of a team player — it is a building block that can be used together with other programs, with the sum adding up to more than the separate parts. This section discusses some additional software packages that can be used in conjunction with DR.

Alternate Pathing

Alternate Pathing (AP) supports high availability of I/O controllers — the hardware components that reside on system boards and enable the Sun Enterprise server to communicate with I/O devices such as disks and networks. With AP, each I/O device connects to two I/O controllers. The I/O controllers are part of two separate electrical pathways to the I/O device, known as alternate paths.

There are two purposes for AP. The first purpose is to help protect against I/O controller failures. With AP, if one I/O controller fails, you can switch to the alternate controller. For disk controllers, this switch occurs automatically whenever a path failure is detected during normal operation. For network controllers, you must manually switch paths by using a single AP command, `apconfig(1M)`.

The second purpose of AP is to support DR. If you want to detach a board that is connected to an I/O device and that I/O device is alternately pathed, you can use AP to redirect the I/O flow to a controller on a different board. You can then use DR to detach the system board without interrupting the I/O flow.

Currently, supported AP versions are linked to specific Solaris versions:

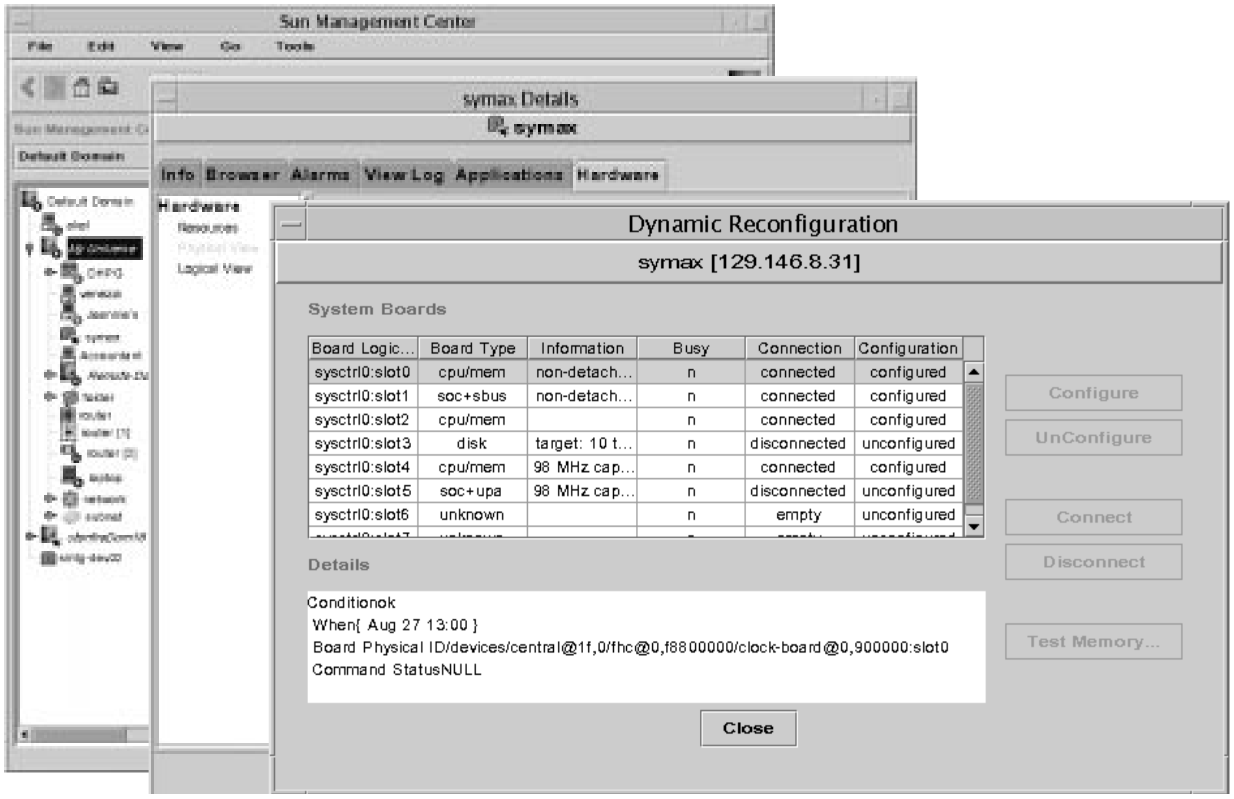
- ***AP 2.1*** — AP 2.1 is only supported with Solaris 2.6 and was co-shipped with the Solaris 2.6 5/98 package on the supplemental CD.
- ***AP 2.2*** — AP 2.2 is only supported with Solaris 7 and was co-shipped with Solaris 7 5/99 package on the supplemental CD. It works with earlier versions of Solaris 7 if they have all the patches installed to bring them up to at least Solaris 7 5/99 version.

AP is supported and recommended with DR. However AP cannot be used with SCSI devices, only with Fiber Channel. This is because AP needs two paths to reach the disk.

Sun Management Center

Sun Management Center (previously called SyMON) is a graphical system management tool that helps administrators manage their systems remotely from a single console.

It is possible to add a module to Sun Management Center to support DR actions on an E3000-6500 server. This module allows the administrator to run through all the steps of DR remotely with a graphical user interface. All the steps discussed further in this document can also be done via this GUI, enabling the user to administer a DR system from anywhere in the Sun Management Center framework.



Board Logic...	Board Type	Information	Busy	Connection	Configuration
sysctrl0:slot0	cpu/mem	non-detach...	n	connected	configured
sysctrl0:slot1	soc+sbus	non-detach...	n	connected	configured
sysctrl0:slot2	cpu/mem		n	connected	configured
sysctrl0:slot3	disk	target: 10 t...	n	disconnected	unconfigured
sysctrl0:slot4	cpu/mem	98 MHz cap...	n	connected	configured
sysctrl0:slot5	soc+upa	98 MHz cap...	n	disconnected	unconfigured
sysctrl0:slot6	unknown		n	empty	unconfigured

```
Conditionok
When{ Aug 27 13:00 }
Board Physical ID/devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot0
Command StatusNULL
```

Figure 4 Sun Management Center screen shots.

Figure 4 contains snapshots of Sun Management Center's relevant windows, as seen when administering DR on a system.

Sun Management Center is supported in combination with DR. If the administrator is familiar with Sun Management Center, using this tool can make managing DR easier.

DiskSuite

Solstice DiskSuite is a disk management program the enables the administrator to perform RAID 0, 1, 1+0 and 5 configurations with the disks connected to the server. Media and license are shipped with the Solaris server packages.

When DiskSuite is used in combination with DR, it is possible to set up a mirror over two different disk chains hung off two different system boards. Administrators can take down a chain of disks located on a system board that is being removed with DR, and still access the information. To do this, however, the mirror has to be temporarily broken, because otherwise the board is still busy and can't be released.

AP is an alternative, but only for Fiber Channel devices like the StorEdge A5000.

DiskSuite is supported in combination with DR and AP.

Veritas Volume Manager

Veritas Volume Manager is similar to Solstice DiskSuite in functionality. It also manages disks and can implement the different flavors of RAID. Every StorEdge A5x00 is shipped with a Volume Manager license for use on that system. A separate license must be purchased to run Volume Manager on other systems.

Volume Manager can be used in much the same way as the DiskSuite example. Volume Manager has an extra feature called Dynamic Multi Pathing (DMP), in which the Volume Manager can, if possible, dynamically fail over the current route to a disk to a different route if the current route breaks. This feature creates problems when used in combination with DR, because DMP keeps the driver attached to the board and therefore causes DR to refuse to unconfigure the board.

Using AP and Veritas Volume Manager can solve this problem, but only for Fiber Channel devices, like the StorEdge A5000. If AP is installed, the DMP will automatically be switched off.

Veritas Volume Manager is supported with DR and AP, currently without the DMP.

Raid Manager for A3000/A3500

Raid Manager (RM) is the software that runs on the hardware RAID controllers of the StorEdge A3000 and A3500. The RM allows the administrator to configure different flavors of RAID, all of which take place on the A3x00. The A3x00 has two Differential Ultra / wide SCSI connections with the host and it can fail over from one connection to the other.

The A3x00 is *not* supported with the AP software. It *is* supported with DR, however there are some issues with the driver releasing the board that has to be Dynamically Reconfigured. It is best to check with the service organization to get the latest status on how this should be done.

SunCluster

The SunCluster software enables the clustering of several Sun servers to create a High Availability solution.

SunCluster is currently *not* supported with DR and AP, because the system has to quiesce when adding boards and this can take up to a minute or more. The other half of the cluster will think the system has gone down and will start to fail over. This can create very strange and unwanted behavior.

SunTrunking

SunTrunking is software that enables an administrator to bundle several Ethernet connections going from the server to the same point into a single virtual fat pipe. These can be Ethernet connections on the same system board or on separate system boards. SunTrunking provides two advantages. First, it increases the possible throughput of the connection. Second, it can eliminate the risk of being affected by the loss off a single system board.

SunTrunking could be used in combination with DR. However this is currently *not* supported, and therefore shouldn't be used.

Typical Configurations

A good way of getting ideas on how to implement DR is to look at example configurations, which can then be molded into a solution that fits the needs of the situation. As discussed in the previous section, it is important to set a goal and then create a solution to fit this goal.

This section describes configurations focused on a single goal. A combination of these simple solutions can give solutions for more complex goals.

CPU

When configuring a system, there are several CPU-related issues to keep in mind:

- Try to get all CPU/Memory Boards on one side of the system, either the front or the back. This provides a better overview of what is in the system and helps the administrator establish a default place to look.
- Try to have the memory distributed over different boards. This is not so much for performance as for spreading the risk of failure over multiple boards.
- Spread boards over power/cooling modules to decrease the impact if a power/cooling module fails.
- Have the memory interleaving set to min in OBP. Boards that are interleaving with other boards cannot be removed.
- Disable memory on all boards that contain it except for the first board. This will cause the memory cage to be placed on the same board as the primary CPU. Just after the boot the remaining memory can be configured. This can be done by a script if necessary.

One possible configuration is:

- Sun Enterprise 4500 server
- Four CPU boards, located in slots 0, 2, 4 and 6
- Eight CPUs, two on every board
- Four GB memory (1 GB per bank, one bank on every board)
- Three I/O boards, located in slots 1, 3 and 7

I/O

When concentrating on the I/O portion of the configuration, it is important to determine if the intended I/O device will work with DR. If an I/O card will not detach/attach (e.g., a token ring card), then the system board this card is on should be considered permanent.

General I/O Issues

- Find out which cards are detachable and which are merely suspendable.
- If a card is neither detachable or suspendable, then a replacement must be found. If no replacement is available, DR is not possible on the system.
- Try to concentrate all the cards that are only suspendable on one board.
- Try to concentrate all other cards on others boards.
- Try to have at least two boards that are detachable.
- Try to have all I/O boards on one side of the system, on the back side for example. This helps the administrator have a better overview of the system and creates a default place to look.
- Try to have the detachable boards on different power/cooling modules.

Disk Issues

- When possible, use AP to create two paths over two separate I/O boards. This only applies to storage hardware that is supported with AP.
- In all cases it is important to have the boot disk(s) and swap partitions spread over two I/O boards. This will remove these dependencies.
- When impossible to dual path (AP or mirror), try to hang this resource on the suspend-only board. This concentrates the non-DR participants on one board.
- When using a Disk Board, plug it into the slot next to the I/O board it's connected to. This creates overview.

Networking

- When possible use AP to create two paths over two separate I/O boards. This only applies for networking that is supported with AP.

- Try to AP the primary network interface, thereby removing this dependency.
- When it's impossible to dual path (AP), try to hang this resource on the suspend-only board. This concentrates the non-DR participants on one board.

Same example configuration:

- Sun Enterprise 4500 server
- Four CPU/Memory Boards in slots 0, 2, 4, and 6
- Three I/O Boards located in slots 1, 3, 7
  - Slot 1 contains: Token Ring interface, ATM 155 interface
  - Slot 3 contains: AP Ethernet connection with primary interface, AP FDDI connection, AP GBIC connection to A5000.
  - Slot 7 contains: AP Ethernet connection with primary interface, AP FDDI connection, AP GBIC connection to A5000.
- AP & DiskSuite installed

Preparing the System for DR

Before planning and starting any DR activity, a system must be DR-ready. In most cases, making the system DR-ready involves just checking if the system is up to specifications and determining if it contains elements that prevent any DR operations. This section is closely related to the "Requirements" section on page 25 and the "Checklists" section on page 79.

First, determine the goal of the DR operation to enable a focused search. If, for example, the goal is to be able to add new boards into the system, then only the basic things need to be checked. If, on the other hand, boards are to be removed, then the detachability of the system devices is also important.

Certain things always must be checked. The first is the OBP version on the system boards, including any boards that will be added to the system. This can be done by going into the `ok` prompt (OBP) mode and then entering the `.version` command. Going into the ok prompt mode can be accomplished by either halting the system and booting it again or by suspending it with the STOP-A key combination and then typing `go` to have the system resume.

```
ok .version
Slot  1 - I/O Type 5 FCODE 1.8.22 1999/05/12 15:33  iPOST 3.4.22 1999/05/12 15:37
Slot  3 - I/O Type 4 FCODE 1.8.22 1999/05/12 15:33  iPOST 3.4.22 1999/05/12 15:37
Slot  5 - CPU/Memory OBP   3.2.22 1999/05/12 15:34  POST  3.9.22 1999/05/12 15:37
Slot  7 - CPU/Memory OBP   3.2.22 1999/05/12 15:34  POST  3.9.22 1999/05/12 15:37
Slot  9 - CPU/Memory OBP   3.2.22 1999/05/12 15:34  POST  3.9.22 1999/05/12 15:37
```

The example output listed above shows the OBP and FCODE versions for all system boards. See the “Software” section on page 33 for details on which versions are minimal. Also double-check this information by running the `banner` command:

```
ok banner
5-slot Sun Enterprise E3500, No Keyboard
OpenBoot 3.2.22, 1024 MB memory installed, Serial #10805690.
Ethernet address 8:0:xx:xx:xx:xx, Host ID: xxxxxxx.
```

This, of course, can only be done if the system can be halted or suspended; this is not advisable if the system is currently depended upon.

When not all boards in the system have the correct or up-to-date version, they need to be updated. The new OBP versions and details on how to install them can be found on:
http://sunsolve5.sun.com/sunsolve/Enterprise-dr/PROMDownload.html

It is important to realize that if something goes wrong while installing (flashing) a new OBP version on a system board, it is very hard to correct this. So first update one board and verify that this was successful, then flash the others. In this way the old situation can always be recreated.

Next, check the Solaris version installed on the system, which can be seen in the `/etc/release` file:

```
# cat /etc/release
                    Solaris 7 5/99 s998s_u2SunServer_09 SPARC
           Copyright 1999 Sun Microsystems, Inc.  All Rights Reserved.
                              Assembled 19 April 1999
```

If the Solaris version supports the DR operations needed to achieve the goal, check on the Web site mentioned above to see the recommended patches list and verify that they are installed.

The next step is determining what boards and devices are in the system. It is handy to write the setup down in a table template (see the “System Templates” section on page 84). When all the elements are filled in, the table provides a good idea of which boards can be removed and where empty slots are located.

There are several commands that can be used to get the information to fill up the table. The `.version` command returns a list of board types in the first column of the output. In the example shown above, the system has five system boards: Board 1 is a type 5 or Graphics I/O Board; board 3 is a type 4 or SBus I/O Board; and boards 5, 7, and 9 are CPU/Memory Boards.

The `prtdiag` command displays general information about the system. Here is a full `prtdiag` output:

```
# /usr/platform/sun4u/sbin/prtdiag
System Configuration:  Sun Microsystems  sun4u 5-slot Sun Enterprise E3500
System clock frequency: 100 MHz
Memory size: 2048Mb

========================= CPUs =========================

Run   Ecache   CPU    CPU
Brd   CPU   Module   MHz     MB    Impl.   Mask
---   ---  -------  -----  ------  ------  ----
5     10      0      400     4.0   US-II    10.0
5     11      1      400     4.0   US-II    10.0
7     14      0      400     4.0   US-II    10.0
7     15      1      400     4.0   US-II    10.0
9     18      0      400     4.0   US-II    10.0
9     19      1      400     4.0   US-II    10.0

========================= Memory =========================

Intrlv.  Intrlv.
Brd   Bank   MB     Status    Condition   Speed   Factor   With

---   -----  ----  -------   ----------  -----   -------  -------
5      0     1024   Active      OK        60ns    1-way
7      0     1024   Active      OK        60ns    1-way

========================= IO Cards =========================

Bus   Freq
Brd   Type  MHz   Slot  Name                              Model
---   ----  ----  ----  --------------------------------  ----------------------
1     SBus   25     3   SUNW,hme
1    SBus   25     3   SUNW,fas/sd (block)
1    SBus   25    13   SUNW,socal/sf (scsi-3)           501-3060
1    UPA   100     2   FFB, Single Buffered             SUNW,501-2634
3    SBus   25     3   SUNW,hme
3     SBus   25     3   SUNW,fas/sd (block)
3     SBus   25    13   SUNW,socal/sf (scsi-3)           501-3060
No failures found in System
===========================
No System Faults found
======================
```

This output indicates that the system is a five slot E3500, its Gigaplane is running at 100 Mhz, and it has a total internal memory of two GB. There are six CPUs located on boards 1, 5 and 7. Boards 5 and 7 both have one GB of internal memory each, and this memory is not cross-board interleaved. Board 1 has four devices on it, of which the first three are the on-board devices, and the fourth is an added Creator graphics ffb card. Board 3 contains only the three on-board devices.

Now check with the physical system and see if this output gives a good representation of the system. If there are more boards or cards physically in the system, then there may be something wrong that is worth finding out. Also check the invoice or the parts list to find out which devices correspond with which part numbers in the supported parts list. If it is not possible to find the part numbers, contact Sun to find out.

The `cfgadm` command provides more information on the composition of the system hardware, like system boards and memory banks. (See "The cgfadm Command" section on page 100 for more details.) For example:

```
# cfgadm -s cols=ap_id:type:info
Ap_Id                Type         Information
ac0:bank0            memory       slot5 1Gb base 0x0 permanent
ac0:bank1            memory       slot5 empty
ac1:bank0            memory       slot7 1Gb base 0x100000000
ac1:bank1            memory       slot7 empty
ac2:bank0            memory       slot9 empty
ac2:bank1            memory       slot9 empty
sysctrl0:slot1       soc+upa      100 MHz capable
sysctrl0:slot3       soc+sbus     100 MHz capable
sysctrl0:slot5       cpu/mem      non-detachable   100 MHz capable
sysctrl0:slot7       cpu/mem      disabled at boot   100 MHz capable
sysctrl0:slot9       cpu/mem      100 MHz capable
```

This output shows that two of the six memory banks in the system are used, and both contain one GB of memory. The first bank of memory is permanent, meaning that the board it sits on (the board in slot 5) cannot participate in any DR operation in this boot setup. If the system is rebooted with the same settings, this permanent memory probably will sit in the same memory bank. However, if the configuration changes, it could move to another location.

Another thing that becomes clear from this output is that the first CPU/Memory Board in the system (the board in slot 5) is non-detachable because it holds the primary CPU. The fact that the permanent memory and the primary CPU are on the same board is coincidence; they can be located on separate boards. They are on the same board in this example because the board in slot 7 was disabled at boot by an OBP setting, causing the CPUs and memory on that board to not participate in the boot. Because of this, the only place the system could place the permanent memory was in the memory bank on the board in slot 5. The board in slot 7 is configured later.

With this information, a table showing which boards are currently able to participate in a DR operation can be completed and double checked. This table, combined with the information given in the “Requirements” section on page 25, provides a total picture of the system capabilities.

If a system does not support DR for any reason, the `cfgadm` command will return the following status message.

```
# cfgadm
Hot Plug not supported in this system
```

This message is also displayed in the console at system boot time.

The next thing to check is that the `/etc/systems` file contains the required entries (see the “Requirements” section on page 25 for more details).

Checking the OBP setting for memory interleaving is also important. This can be done while the system is running with the `eeprom` command:

```
# eeprom memory-interleave
memory-interleave=min
```

If memory-interleave is not set to `min`, go to the ok prompt and set it correctly to make the system ready for DR.

Finally test the system capability to quiesce with the `cfgadm` command:

```
# cfgadm -x quiesce-test sysctrl0:slot1
```

The test only has to be run on one of the valid occupants because the quiesce effects the whole system. This must, of course, be an occupied slot. On a large system this test may take about a minute or so. If no messages appear during this test, then there are no incompatible device drivers.

If all goes well, the system now supports DR. In some cases the system has to reboot to let the new settings, such as memory interleaving, take effect. Introducing Alternate Pathing also occurs at this stage, however that topic is beyond the scope of this document.

There is still one thing missing from a complete picture of the system, and that is a mapping of the peripherals connected to this system. Peripherals can introduce dependencies to system boards. For example, if the boot disk and the swap partition are not Alternate Pathed, this creates dependencies. The applications running on the system may also create these types of dependencies.

```
# df -k
Filesystem            kbytes    used   avail capacity  Mounted on
/proc                      0       0       0     0%    /proc
/dev/dsk/c0t0d0s0
                     8176566  632777 7462024     8%    /
fd                         0       0       0     0%    /dev/fd
swap                 2233664       8 2233656     1%    /tmp

# swap -l
swapfile             dev  swaplo blocks   free
/dev/dsk/c0t0d0s1 150,41     16 1052144 1021568

# ls -l /dev/dsk/c0t0d0s0
lrwxrwxrwx   1 root     root          74 Aug 20 15:40 /dev/dsk/c0t0d0s0 ->
../../devices/sbus@3,0/SUNW,socal@d,10000/sf@1,0/ssd@w210000203709c10b,0:a
```

The first two commands in the example above provide good insight on where boot disk and swap file are located. In this case, they are both located on disk `c0t0d0`. Some further investigation leads to the conclusion that this is linked to the `sbus@3,0`. The general rule is that the SBus addresses are S*2 and (S*2)+1, where S is the system board slot (a SBus I/O Board has two SBuses). So `sbus@3,0` turns out to be on the I/O board in slot 1. (This last conclusion was made by looking at what is physically connected and what is in the `/dev/dsk` directory.)

Network connections can be checked with the `ifconfig` command:

```
# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
        inet 127.0.0.1 netmask ff000000
hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
        inet x.x.x.x netmask ffffff00 broadcast x.x.x.255
        ether 8:0:xx:xx:xx:xx
```

This output shows that there is only one link, and physical inspection shows that it is linked to the first I/O board, slot 1.

Also note that the `prtdiag` output in the earlier example showed that the system is running with a 100 MHz clock on the bus. Be careful that only 100 MHz-capable system boards are added to this system.

Flowchart

This section refers frequently to the `cfgadm` command, whose general use is explained in the DR User's Guide and in “The cgfadm Command” section on page 100. The “Mechanisms” section on page 18 explains the underlying mechanisms. The “Phases - States - Conditions Figure” section on page 90 also provides information related to this section.

Up until now, everything has been mostly theoretical. This section goes into the different commands that the administrator uses to utilize DR. All outputs shown actually came from a live system. This section also describes the different phases that a board can be in (Figure 5), and shows in which way a board can be moved from one phase to another.

States

Earlier, this paper discussed the phases a system board can be in and what happened when moving from one phase to another (see “Mechanisms” section on page 18). This model offers six distinct phases: Empty, Disconnected, Connected, Configured, Idle and Running. When a system board is in a phase, there is actually a receptacle-occupant pair that has a certain condition. Consider, for example, a slot-system board in the Connected phase. The POST and OBP have run and so the slot can be considered connected, but the system board is still unconfigured. This condition is called a state, and every receptacle and occupant are in a state. The states that the receptacle-occupant pair are in dictate the phase the board-subsystem as a whole is in.

This can lead to some confusion, not in the least because many names are the same. This section compares the two and explains how everything fits into place.

The model first has to be split into two parts (see Figure 5). The first part consists of the phases *Empty*, *Disconnected*, *Connected* and *Configured*, while the second consists of the phases *Configured*, *Idle* and *Running*. The state *Configured* is on the boundary between the two parts, and therefore is in both parts. This boundary is located here because it is at this point the receptacle-occupant pair move from slot-system board to system board-device. This device can, for example, be memory.

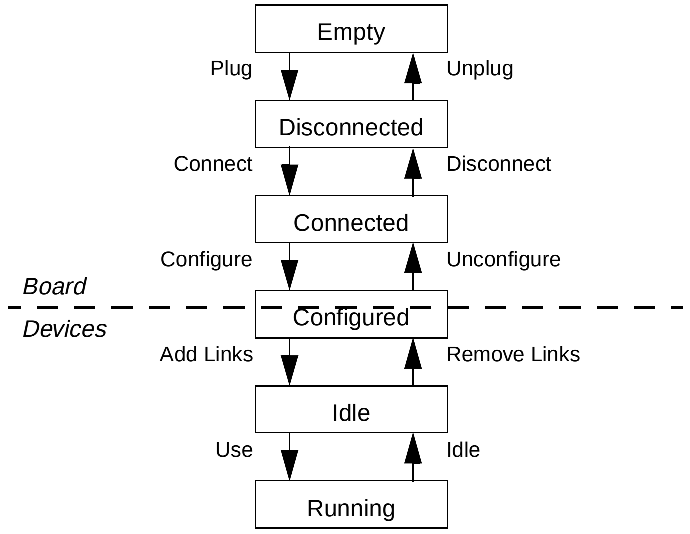


Figure 5 A model of the system board phases, split into two parts.

Empty to Configured Phases

The first half of the model reflects the phases a system board goes through, and is comparable to the process that occurs when the system starts up (i.e., before Solaris boots up and until the device is mapped in the kernel). When performing DR through these phases, it is irrelevant what type of system board is being added and what the devices are on it. The commands used are identical for all types of system boards.

The main command used in DR is `cfgadm`. The `-l` option can be used with this command to list the current status of the system concerning system boards and devices, as shown below:

```
# cfgadm -l
Ap_Id                 Receptacle   Occupant     Condition
ac0:bank0             empty        configured   ok
ac0:bank1             empty        unconfigured unknown
ac2:bank0             empty        unconfigured unknown
ac2:bank1             empty        unconfigured unknown
sysctrl0:slot1        connected    configured   ok
sysctrl0:slot3        connected    configured   ok
sysctrl0:slot5        connected    configured   ok
sysctrl0:slot7        disconnected unconfigured unknown
sysctrl0:slot9        connected    configured   ok
```

When looking at the output of `cfgadm` in this example, notice that there are two groups — the lines that start with `ac` and those that start with `sysctrl`. The first group reflects the status of the memory modules, which is not relevant in this part of the model. The second group reflects the status of the boards in the system, classified by their connection to the backplane. In these servers there is only one system controller, and so this argument is always `sysctrl0` (where the last character is a zero). The connection to the backplane is the receptacle, and the system board itself is the occupant.

This section concentrates on the system boards, which are connected to the system controller slots. Listed below are the relevant lines from the above example:

```
# cfgadm -l
Ap_Id                 Receptacle   Occupant     Condition
...
sysctrl0:slot1        connected    configured   ok
sysctrl0:slot3        connected    configured   ok
sysctrl0:slot5        connected    configured   ok
sysctrl0:slot7        empty        unconfigured unknown
sysctrl0:slot9        connected    configured   ok
```

This is the output from an E3500 with 5 slots (receptacles). The `sysctrl0:` reflects that these connections are on the first system controller. (There is only one system controller in a E3000-6500 type server, so this will always be the case in this document.) All slots are taken, except Receptacle sysctrl0:slot7 is empty. The board (the Occupant) is unconfigured and its Condition is unknown. This section looks at this slot and system board in more detail. Note that the slot Condition is unknown. If it is unusable, then don't attempt to insert the board. This condition may be due to insufficient power or a faulty power/cooling module. There is more about unusable later in this chapter.

The slot is in the phase *empty*. To go to the next phase, *disconnected*, the system board must be inserted. Be sure to always check the physical condition of the connector: there should be no damage to the pins or sockets in any way. If there is, this greatly increases the risks of a system crash when inserting the board.

After inserting the board, the console displays the entries:

```
Sep 30 17:16:19 sgkona-3 unix: NOTICE: cpu board has been inserted into slot 7
Sep 30 17:16:19 sgkona-3 unix: NOTICE: board 7 can be removed
```

A cfgadm command (using no arguments is the same as using the -l option) now gives:

```
# cfgadm
Ap_Id                  Receptacle   Occupant     Condition
...
sysctrl0:slot7         disconnected unconfigured unknown
...
```

The board has now moved into the *disconnected* state. The Receptacle is now disconnected and the Occupant and Condition are unchanged. The LEDs also reflect this state. They normally have the following colors: green, yellow, green. In this state they are: Off, On, Off.

The next step is going to the connected phase. This is done by using the `cfgadm -c connect` statement. The system will ask if the system may be frozen (quiesce), and this is answered by a `y`:

```
# cfgadm -c connect sysctrl0:slot7
system will be temporarily suspended to connect a board: proceed (yes/no)? y
Sep 30 17:32:57 sgkona-3 unix: NOTICE: connecting cpu board in slot 7
Sep 30 17:33:53 sgkona-3 unix: NOTICE: cpu board in slot 7 is connected
# cfgadm
Ap_Id                Receptacle   Occupant     Condition
...
sysctrl0:slot7       connected    unconfigured ok
...
```

The board is now *connected*. Up until now, the `Condition` of the board was `unknown` as there where no tests run on it yet. After going from *disconnected* to *connected*, POST has been run and so the system can now know it is `ok`. The LEDs on the board now show: On, Off, Off.

The board is now ready to move to the *configured* phase. The previous `cfgadm` report shows that the `Occupant` is `unconfigured` yet the `Receptacle` is `connected`. This is the way the `cfgadm` reflects the *connected* phase. Going to the *configured* phase is done with the `cfgadm -c configure` statement:

```
# cfgadm -c configure sysctrl0:slot7
Sep 30 17:47:27 sgkona-3 unix: NOTICE: configuring cpu board in slot 7
Sep 30 17:47:27 sgkona-3 unix: NOTICE: cpu board in slot 7 is configured
# cfgadm
Ap_Id                Receptacle   Occupant     Condition
...
sysctrl0:slot7       connected    configured   ok
...
```

The board is now in the *configured* phase. The `cfgadm` output reflects this by showing `configured` under the `Occupant` heading, and the LEDs on the board show: On, Off, Flash.

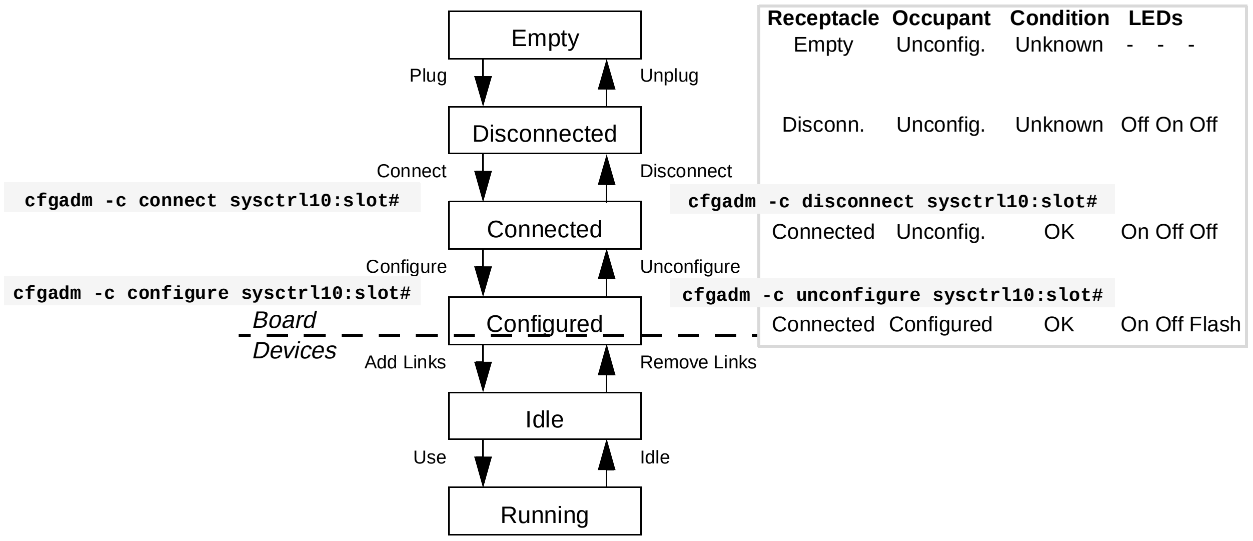
The board is now ready to go into the second part of the model. However, this section first shows how to go back down to the *empty* phase:

```
# cfgadm -c unconfigure sysctrl0:slot7
Sep 30 17:58:20 sgkona-3 unix: NOTICE: unconfiguring cpu board in slot 7
Sep 30 17:58:21 sgkona-3 unix: NOTICE: cpu board in slot 7 is unconfigured
# cfgadm
Ap_Id                 Receptacle   Occupant     Condition
...
sysctrl0:slot7        connected    unconfigured ok
...

# cfgadm -c disconnect sysctrl0:slot7
Sep 30 18:00:12 sgkona-3 unix: NOTICE: disconnecting cpu board in slot 7
Sep 30 18:00:13 sgkona-3 unix: NOTICE: cpu board in slot 7 is disconnected
Sep 30 18:00:13 sgkona-3 unix: NOTICE: board 7 is ready to remove
# cfgadm
Ap_Id                 Receptacle   Occupant     Condition
...
sysctrl0:slot7        disconnected unconfigured unknown
...

# Sep 30 18:03:09 sgkona-3 unix: NOTICE: board 7 has been removed
# cfgadm
Ap_Id                 Receptacle   Occupant     Condition
...
sysctrl0:slot7        empty        unconfigured unknown
...
```

The `cfgadm -c unconfigure` command is used to move from *configured* to *connected*. The LEDs change into: On, Off, Off. Then, the `disconnect` argument is used with the `cfgadm` command to move from *connected* to *disconnected*. At this point the LEDs show: Off, On, Off.

Finally the board is unplugged, leaving the receptacle empty.

This now leads to a more complete picture of the DR model, as shown in Figure 6.


Empty
Plug
Unplug
Disconnected
Connect
Disconnect
cfgadm -c connect sysctrl10:slot#
Connected
cfgadm -c disconnect sysctrl10:slot#
Configure
Unconfigure
cfgadm -c configure sysctrl10:slot#
cfgadm -c unconfigure sysctrl10:slot#
Configured
Board
Devices
Add Links
Remove Links
Idle
Use
Idle
Running

Receptacle	Occupant	Condition	LEDs
Empty	Unconfig.	Unknown	- - -
Disconn.	Unconfig.	Unknown	Off On Off
Connected	Unconfig.	OK	On Off Off
Connected	Configured	OK	On Off Flash

Figure 6 A model of system board phases during DR.

When a system boots normally, all system boards are started and are taken all the way into the running phase automatically. However, it is possible to set an OBP parameter that disables a board. For example, setting the memory as disabled is done by setting the OBP `disabled-board-list` parameter:

```
ok setenv disabled-board-list 7
disabled-board-list =  7
```

When the system now boots, the board in slot 7 is disabled. In this case there are two other CPU / Memory Boards, located in slot 5 and 9:

```
# cfgadm -s cols=ap_id:r_state:o_state:condition:info
Ap_Id                Receptacle   Occupant     Condition  Information
ac0:bank0            connected    configured   ok         slot5 1Gb base 0x0 permanent
ac0:bank1            empty        unconfigured unknown    slot5 empty
ac2:bank0            empty        unconfigured unknown    slot9 empty
ac2:bank1            empty        unconfigured unknown    slot9 empty
sysctrl0:slot1       connected    configured   ok         100 MHz capable
sysctrl0:slot3       connected    configured   ok         100 MHz capable
sysctrl0:slot5       connected    configured   ok         non-detachable   100 MHz capable
sysctrl0:slot7       disconnected unconfigured unknown    disabled at boot 100 MHz capable
sysctrl0:slot9       connected    configured   ok         100 MHz capable
```

This output shows that the board in slot 7 is disabled and disconnected. All other boards are normally started and running. Because it is disconnected, the board in slot 7 doesn't show the memory banks that are located on it. To get everything on this board to the running phase, the administrator must go through all the same steps as when adding a just-inserted board. There is only one small difference — when connecting this board, the administrator must use the `-f` parameter to the `cfgadm` command.

```
# cfgadm -f -c connect sysctrl0:slot7
system will be temporarily suspended to connect a board: proceed (yes/no)? y
Oct 13 09:52:44 sgkona-3 unix: NOTICE: connecting cpu board in slot 7
Oct 13 09:53:44 sgkona-3 unix: NOTICE: cpu board in slot 7 is connected
```

Configured to Running Phases

The second half of the model compares with the booting of the Solaris. Here the type of board and what types of devices are on it *is* important. The OS treats every type of device differently, and it must hang a different device driver on it. This is why there is now a distinction between the different types of devices, and configuration and unconfiguration becomes more complex. This is also where having an experienced administrator becomes important.

This document makes the distinction in devices between CPUs, memory, disks, tapes, serial ports and other devices. This classification is mainly due to the commands that test and configure the devices. Some are relatively easy to do, while an in-depth discussion of others goes beyond the scope of this document.

CPUs

Once the board is in the configured phase, the CPUs are powered off but ready to be turned on and used. This example starts with the same system with every board in the configured phase. The system board in slot 7 has just been added and configured. A `prtdiag` shows:

```
# /usr/platform/sun4u/sbin/prtdiag
System Configuration:  Sun Microsystems  sun4u 5-slot Sun Enterprise E3500
System clock frequency: 100 MHz
Memory size: 1024Mb

========================= CPUs =========================

Run   Ecache   CPU    CPU
Brd  CPU   Module   MHz     MB    Impl.   Mask
---  ---  -------  -----  ------  ------  ----
 5    10     0      400     4.0   US-II   10.0
 5    11     1      400     4.0   US-II   10.0
 7    14     0      400     4.0   US-II   10.0
 7    15     1      400     4.0   US-II   10.0
 9    18     0      400     4.0   US-II   10.0
 9    19     1      400     4.0   US-II   10.0
...
```

This shows that the board in slot 7 has two CPUs, both 400 Mhz, and that their CPU numbers are 14 and 15. The `psrinfo` command gives:

```
# psrinfo
10      on-line   since 09/22/99 12:46:27
11      on-line   since 09/22/99 12:46:31
14      powered-off  since 09/30/99 19:14:49
15      powered-off  since 09/30/99 19:14:49
18      on-line   since 09/22/99 12:46:31
19      on-line   since 09/22/99 12:46:31
```

This output shows that CPUs number 14 and 15, corresponding with board 7, are powered-off.

The `psradm -n` *`numbers`* command is used to turn on the CPUs, where `numbers` are the CPU numbers.

```
# psradm -n 14 15

# psrinfo
...
14      on-line   since 09/30/99 19:31:45
15      on-line   since 09/30/99 19:31:45
...
```

Now both CPUs have been started up. The moment a CPU becomes *idle* it automatically goes into the *running* phase, and so the administrator doesn't need to do anything extra.

To bring the CPU back off-line so the board can go into the *configured* phase, the `psradm -f` *`numbers`* command is used (again, `numbers` are the CPU numbers):

```
# psradm -f 14 15

# psrinfo
...
14      off-line  since 09/30/99 19:40:22
15      off-line  since 09/30/99 19:40:22
...
```

The board is now in the *configured* phase concerning the CPUs. Only if all the devices on the board are idle will the board really be in the *configured* phase.

This is all summarized in Figure 7.

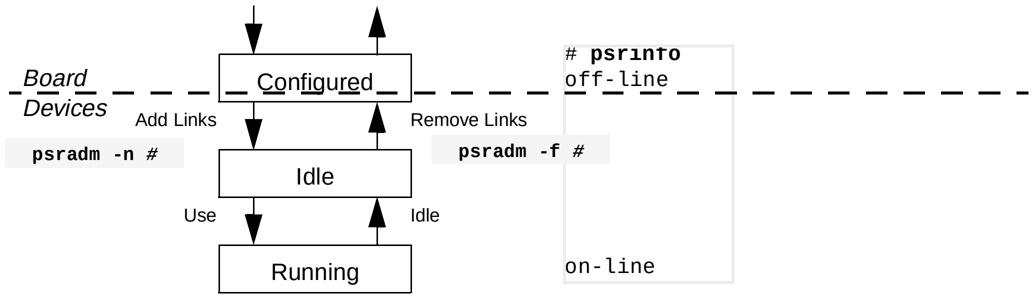


Figure 7 Partial model of the DR states, concentrating on CPUs.

Memory

Memory is slightly more complex than CPUs. The memory first has to be brought in the *idle* phase by adding links in the file system to access the memory. The kernel must be told to load the device drivers, and the links must be made. The command to do this is `drvconfig`, which checks all devices in the kernel, sees if the drivers are loaded for these devices, checks if the representations in `/devices` are all there, and creates them if they are missing. This can take a long time on large systems. However, this can also be made more specific by using `drvconfig -i ac`. This checks all devices reporting to the Address Controller (AC), including the memory banks. Once this is done the `cfgadm -v` command can also see the memory.

First the `prtdiag` command shows the current status of the memory:

```
# /usr/platform/sun4u/sbin/prtdiag
System Configuration:  Sun Microsystems  sun4u 5-slot Sun Enterprise E3500
System clock frequency: 100 MHz
Memory size: 1024Mb
...
========================= Memory =========================

                                              Intrlv.  Intrlv.
Brd   Bank   MB    Status   Condition  Speed   Factor   With
---  -----  ----  -------  ----------  -----  -------  -------
 5     0    1024  Active      OK        60ns    1-way
 7     0    1024  Spare    Unknown      60ns    1-way
...

# cfgadm -v
Ap_Id                Receptacle   Occupant     Condition  Information
When         Type         Busy     Phys_Id
ac0:bank0            connected    configured   ok         slot5 1Gb base 0x0 permanent
Sep 22 12:46 memory       n        /devices/fhc@a,f8800000/ac@0,1000000:bank0
ac0:bank1            empty        unconfigured unknown    slot5 empty
Sep 22 12:46 memory       n        /devices/fhc@a,f8800000/ac@0,1000000:bank1
ac2:bank0            empty        unconfigured unknown    slot9 empty
Sep 22 12:46 memory       n        /devices/fhc@12,f8800000/ac@0,1000000:bank0
ac2:bank1            empty        unconfigured unknown    slot9 empty
Sep 22 12:46 memory       n        /devices/fhc@12,f8800000/ac@0,1000000:bank1
sysctrl0:slot1       connected    configured   ok         100 MHz capable
Sep 22 12:46 soc+upa      n        /devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot1
...
```

The `prtdiag` output shows that the board in slot 7 has `spare` memory with an `unknown` condition. The `cfgadm -v` shows that the only memory that is `connected` is on the board in slot 5.

Now the `drvconfig -i ac` command:

```
# drvconfig -i ac
# cfgadm -v
Ap_Id                     Receptacle   Occupant     Condition  Information
When          Type         Busy      Phys_Id
ac0:bank0                 connected    configured   ok         slot5 1Gb base 0x0 permanent
Sep 22 12:46 memory        n          /devices/fhc@a,f8800000/ac@0,1000000:bank0
ac0:bank1                 empty        unconfigured unknown    slot5 empty
Sep 22 12:46 memory        n          /devices/fhc@a,f8800000/ac@0,1000000:bank1
ac1:bank0                 connected    unconfigured unknown    slot7 1Gb base 0x100000000
Sep 30 19:14 memory        n          /devices/fhc@e,f8800000/ac@0,1000000:bank0
ac1:bank1                 empty        unconfigured unknown    slot7 empty
Sep 30 19:14 memory        n          /devices/fhc@e,f8800000/ac@0,1000000:bank1
ac2:bank0                 empty        unconfigured unknown    slot9 empty
Sep 22 12:46 memory        n          /devices/fhc@12,f8800000/ac@0,1000000:bank0
ac2:bank1                 empty        unconfigured unknown    slot9 empty
Sep 22 12:46 memory        n          /devices/fhc@12,f8800000/ac@0,1000000:bank1
sysctrl0:slot1            connected    configured   ok         100 MHz capable
Sep 22 12:46 soc+upa       n          /devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot1
...
```

This now shows that `ac1` has two banks, of which only one is filled with 1 GB of memory.

The memory is now in the *idle* phase, ready to be set to the *running* phase. This is done with two commands. First the memory is tested with `cfgadm -o` *`test_type`* `-t ac`*`number`*`:bank`*`number`*, where *test_type* is either quick, normal or extended, and the number reflects the location of the bank:

```
# cfgadm -o normal -t ac1:bank0

# /usr/platform/sun4u/sbin/prtdiag
System Configuration:  Sun Microsystems  sun4u 5-slot Sun Enterprise E3500
System clock frequency: 100 MHz
Memory size: 1024Mb
...
========================= Memory =========================

                                              Intrlv.  Intrlv.
Brd   Bank   MB    Status   Condition  Speed   Factor   With
---  -----  ----  -------  ----------  -----  -------  -------
 5     0    1024   Active      OK       60ns    1-way
 7     0    1024   Spare       OK       60ns    1-way
...

# cfgadm
Ap_Id                Receptacle   Occupant     Condition
ac0:bank0            connected    configured   ok
ac0:bank1            empty        unconfigured unknown
ac1:bank0            connected    unconfigured ok
ac1:bank1            empty        unconfigured unknown
ac2:bank0            empty        unconfigured unknown
ac2:bank1            empty        unconfigured unknown
sysctrl0:slot1       connected    configured   ok
...
```

The testing can take short while. The `Status` shows `Spare` and `Condition`: `OK`.

Then the bank is configured using `cfgadm -c configure ac`*`number`*`:bank`*`number`*, as shown below:

```
# cfgadm -c configure ac1:bank0
Sep 30 20:48:43 sgkona-3 unix: NOTICE: configuring memory bank 0 in slot 7
Sep 30 20:48:49 sgkona-3 unix: NOTICE: memory bank 0 in slot 7 is configured

# /usr/platform/sun4u/sbin/prtdiag
System Configuration:  Sun Microsystems  sun4u 5-slot Sun Enterprise E3500
System clock frequency: 100 MHz
Memory size: 2048Mb
...
========================= Memory =========================

                                              Intrlv.  Intrlv.
Brd   Bank   MB    Status    Condition  Speed   Factor   With
---  -----  ----  -------  ----------  -----  -------  -------
 5     0    1024   Active      OK       60ns    1-way
 7     0    1024   Active      OK       60ns    1-way
...

# cfgadm
Ap_Id                 Receptacle   Occupant     Condition
ac0:bank0             connected    configured   ok
ac0:bank1             empty        unconfigured unknown
ac1:bank0             connected    configured   ok
ac1:bank1             empty        unconfigured unknown
ac2:bank0             empty        unconfigured unknown
ac2:bank1             empty        unconfigured unknown
sysctrl0:slot1        connected    configured   ok
...
```

The memory is now in the *running* phase. This is reflected in the `prtdiag` output by now showing a total of `2048 MB` and a memory Status of Active. In addition, `cfgadm` output shows the bank `configured`. Also notice that the interleaving factor is `1-way` on both boards, indicating that they are not interleaving outside the board.

Bringing the memory to a state where the board can be unconfigured is very much the same, only backwards. First the memory is made *idle* with `cfgadm -c unconfigure ac`*`number`*`:bank`*`number`* command:

```
# cfgadm -c unconfigure ac1:bank0
Oct  1 08:57:02 sgkona-3 unix: NOTICE: unconfiguring memory bank 0 in slot 7
Oct  1 08:57:03 sgkona-3 unix: NOTICE: memory bank 0 in slot 7 is unconfigured

# cfgadm
Ap_Id                 Receptacle   Occupant     Condition
ac0:bank0             connected    configured   ok
ac0:bank1             empty        unconfigured unknown
ac1:bank0             connected    unconfigured ok
ac1:bank1             empty        unconfigured unknown
ac2:bank0             empty        unconfigured unknown
ac2:bank1             empty        unconfigured unknown
sysctrl0:slot1        connected    configured   ok
...
```

The memory is now unconfigured and this part of the board is in the *idle* phase. Bringing the memory further down into the board *configured* phase is not really necessary. The links that `drvconfig` makes *can* to be removed, but *be careful not to remove the wrong links!* The `cfgadm` output no longer shows the memory banks, and so this part of the system board is in the *configured* phase.

Figure 8 summarizes this section.

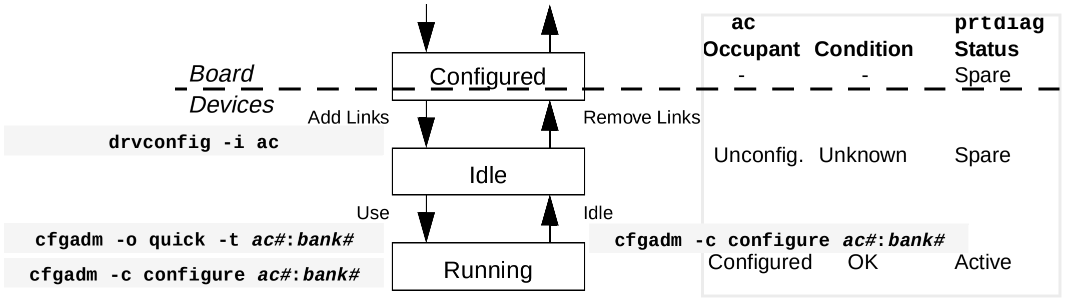


Figure 8 Partial model of DR states, concentrating on Memory boards.

The path memory takes to the *running* phase strongly resembles the path a single system board takes. The *idle* phase is called connected, the *running* phase is called configured, and the memory moves in the same way. But empty is always empty. This is because the system board and memory can also be seen as an receptacle-occupant pair. However memory cannot be hot plugged, and so the receptacle system board it is on can never move to or from the empty state.

When a system boots, all memory is be automatically configured and used. It is, however, possible to set an OBP setting that disables all memory on a certain system board at boot time. In this case, the memory is connected but stays unconfigured. To configure the memory, the administrator only has to test and configure the memory. The advantage of this configuration is that the administrators can control which memory is turned on and which is turned off. This is handy if they want to influence where the memory cage is located (see “Tips” section on page 97).

The OBP `disabled-memory-list` parameter is used to set the memory as disabled:

```
ok setenv disabled-memory-list 7 9
disabled-memory-list =  7 9
```

When the system now boots, the memory on boards 7 and 9 is disabled. In this example, there is only one other CPU/Memory Board, which is located in slot 5:

```
# cfgadm -s cols=ap_id:r_state:o_state:condition:info
Ap_Id                Receptacle   Occupant     Condition  Information
ac0:bank0            connected    configured   ok         slot5 1Gb base 0x0 permanent
ac0:bank1            empty        unconfigured unknown    slot5 empty
ac1:bank0            connected    unconfigured unknown    slot7 1Gb base 0x80000000 dis-
abled at boot
ac1:bank1            empty        unconfigured unknown    slot7 empty
ac2:bank0            empty        unconfigured unknown    slot9 empty
ac2:bank1            empty        unconfigured unknown    slot9 empty
sysctrl0:slot1       connected    configured   ok         100 MHz capable
sysctrl0:slot3       connected    configured   ok         100 MHz capable
sysctrl0:slot5       connected    configured   ok         non-detachable   100 MHz capable
sysctrl0:slot7       connected    configured   ok         100 MHz capable
sysctrl0:slot9       connected    configured   ok         100 MHz capable
```

Only the 1 GB bank on board 5 is configured and running, which implies that it has the permanent memory cage. Further down the output shows that the primary CPU is also located on the board in slot 5.

Starting up the other memory bank on the board in slot 7 is shown earlier in this section. However, an extra option, `-f`, needs to be used here with the `cfgadm` command to force the configuration:

```
# cfgadm -f -c configure ac1:bank0
Oct 13 09:07:42 sgkona-3 unix: NOTICE: configuring memory bank 0 in slot 7
Oct 13 09:07:48 sgkona-3 unix: NOTICE: memory bank 0 in slot 7 is configured
```

Disks

Disks can be connected to all kinds of different devices and come in all types of subsystems. When a new disk connected to a newly added system board is added to a system, the process of finding it and putting it to use is hardly ever the same. The first steps, however, are very similar.

First, the device drivers are loaded (if they are not already running) and are made aware that there could be new devices. This is done by the kernel, but can be triggered by the `drvconfig` command. This takes some time, but can be shortened if the device driver is known by using the `-i device_driver_name` option. However there are quite a number of SCSI drivers alone. Unless the driver is known, use the `drvconfig` command, which also activates all the drivers for the other new devices on the board. Although it takes more time when using just the `drvconfig` command, the result is guaranteed.

Now the `disks` command is run. This checks all the entries in the `/devices` directory tree to see if they refer to a disk and creates links in the `/dev` directory tree as needed. This step is needed because the entries in the `/devices` directory are very machine specific and scattered, while the entries in the `/dev` directory are all grouped (e.g., in the `/dev/dsk` directory). Try to keep a record of which board was added and the resulting new entries in the `/dev/dsk` directory. This information makes it easier to find the disks if they need to be removed (e.g. if the system board concerned is Dynamically Reconfigured again).

The disks are now in the *idle* phase ready to be used.

The next steps involve general system administration. If it is a new disk, the slices are setup to the liking of the administrator with `format`, receive a file system with `newfs`, and finally are mounted with `mount`. Some of these tasks can be complemented with or taken over by volume manager software, like Solstice DiskSuite or Veritas Volume Manager. (Use of these software packages is outside the scope of this document.) These new disks can also be used in the Alternate Pathing framework that is possibly setup on the system.

Once everything is setup, the disks are considered to be in the *running* phase.

Getting the disks *idle* again is a bit trickier. First, the id of the disks affected by the DR operation needs to be found. This can be difficult if the administrator doesn't know which `/dev/dsk` directory entries are associated with each system board. To confirm the findings, check where the entries in the `/dev/dsk` directory are pointing to:

```
# ls -l c2t0d0s0
lrwxrwxrwx   1 root     root          74 Aug 20 15:40 c2t0d0s0 ->
../../devices/sbus@6,0/SUNW,socal@d,10000/sf@1,0/ssd@w220000203709c10b,0:a
```

In this case `c2t0d0s0` is linked to `sbus@6,0`, which is on the board in slot 3, the second I/O board. If this is the board that needs to be removed, all disks on controller 2 (`c2`) need to be checked to see if they are mounted in any way. This is done with `mount`:

```
# mount
/proc on /proc read/write/setuid on Wed Oct  6 18:03:42 1999
/ on /dev/dsk/c0t0d0s0 read/write/setuid/largefiles on Wed Oct  6 18:03:42 1999
/dev/fd on fd read/write/setuid on Wed Oct  6 18:03:42 1999
/tmp on swap read/write/setuid on Wed Oct  6 18:03:43 1999
/mnt on /dev/dsk/c2t0d0s0 read/write/setuid/largefiles on Mon Oct 11 08:35:23 1999
```

In this example, only the previously mentioned `c2t0d0s0` is mounted on `/mnt`. Now check which processes are using `/mnt` with `fuser`:

```
# fuser /mnt
/mnt:      938c     251c
# ps -eaf | grep 251
    root   941   251  0 08:42:11 console  0:00 grep 251
    root   251     1  0   Oct 06 console  0:00 -sh
# pwd
/mnt
```

This case is fairly straightforward. The current console shell is in `/mnt`, but it can be freed by moving up to root, or any place else other than in this directory. Now the `unmount` is simple:

```
# cd /
# fuser /mnt
/mnt:
# umount /mnt
```

Although this was an overly simplified example, it shows what can be used to accomplish the *idle* phase. What has not been taken into account here is the case when the directory is not currently in use but users are depending on it later. It is up to the administrator to communicate this to the users.

Figure 9 summarizes this section.

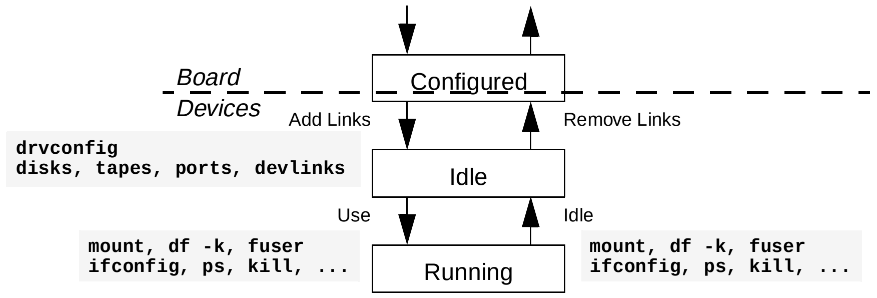


Figure 9 Partial model of DR phases, concentrating on disks, tapes, serial ports and other devices.

Tapes

The first part of adding new tapes is very much the same as with disks. First, `drvconfig` is used, which ultimately creates the links in `/devices`. Then instead of running `disks`, `tapes` is run. This creates the appropriate links in the `/dev` directory that point to the newly created links in the `/devices` directory.

Now the tapes are in the *idle* phase. But they only go to the *running* phase when they are in use. They don't get mounted for long periods of time like disks. This means that a system board with tapes connected to it is always ready to be unconfigured as long as no application is using them.

There is one final issue regarding tape devices during a quiesce: The sequential nature of tape devices prevents them from being reliably suspended in the middle of an operation, and then resumed. Therefore, all tape drivers are suspend-*un*safe. Before executing an operation that activates operating environment quiescence, make sure all tape devices are closed or not in use.

Figure 9 summarizes this section.

Serial Ports and Other Devices

There are a number of other SBus expansion cards that can be added to a system that do not fall under the “storage connection device” category. These include additional serial ports or an additional frame buffer (graphics) card. After running `drvconfig`, the administrator either runs `ports` or `devlinks`. The first creates new `/dev/term` and `/dev/cua` links where needed and creates entries in the `/etc/inittab` for non-system ports found. The `devlinks` command uses the specifications set in the `/etc/devlink.tab` file and creates pretty much everything else that all the other commands don't.

In all cases, once the links in `/devices` and `/dev` have been created they are considered to be in the *idle* phase, ready to be used. Once they are really in use by an application or a daemon, they are considered in the *running* phase.

Getting these devices back to the *idle* phase is very dependent on the device in question. The most common command that can be used to locate an application or daemon that is possibly using the device is `ps`. Then, ultimately the `kill` command is used to stop the process that is causing the device to be busy.

Failing, Failed, Unusable

The last element that hasn't been discussed yet is the condition an occupant (i.e., system board or memory) can have. This condition is listed in the right-most column of the output `cfgadm` returns:

```
# cfgadm
Ap_Id                   Receptacle   Occupant     Condition
ac0:bank0               empty        configured   ok
ac0:bank1               empty        unconfigured unknown
ac2:bank0               empty        unconfigured unknown
ac2:bank1               empty        unconfigured unknown
sysctrl0:slot1          connected    configured   ok
sysctrl0:slot3          connected    configured   ok
sysctrl0:slot5          connected    configured   ok
sysctrl0:slot7          disconnected unconfigured unknown
sysctrl0:slot9          connected    configured   ok
```

This condition reflects the physical condition of the occupant has, and can be `unknown`, `ok`, `unusable`, `failing` or `failed`. The first two are the most common. An occupant condition is `unknown` when it is either not there or it hasn't been tested. In the case of a system board, the tests are run when it goes from the *disconnected* to the *connected* phase by POST. If all goes well, the state changes to `ok` and stays there throughout the other phases.

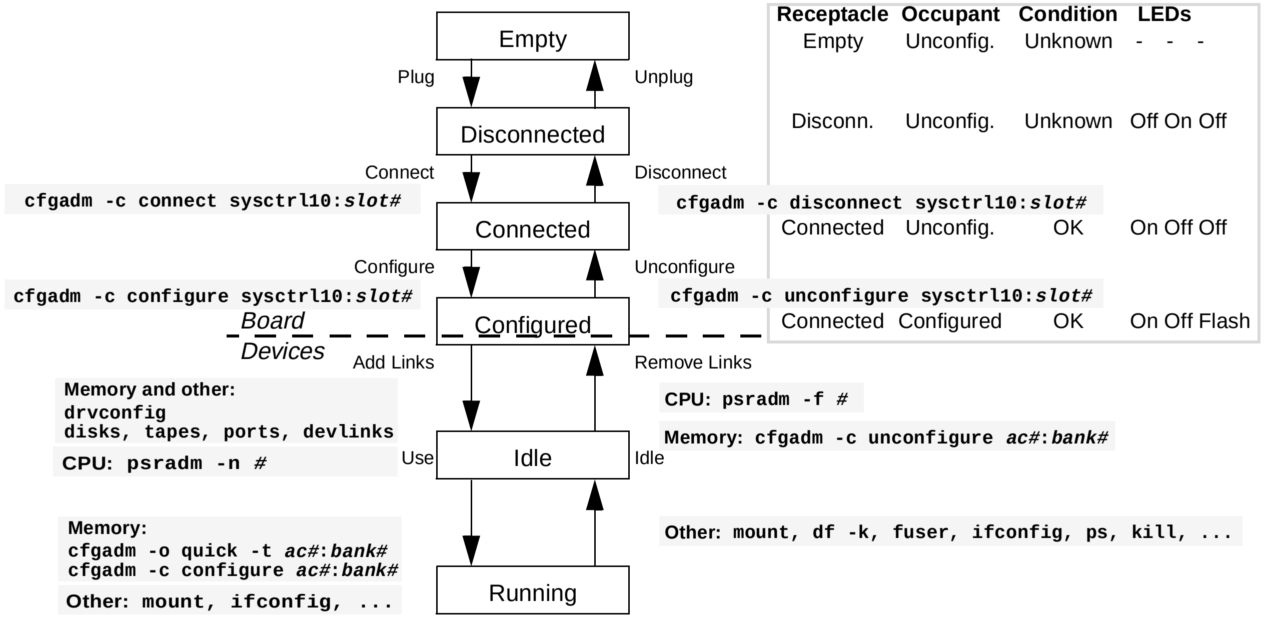
If the tests fail (e.g., there is not enough power to feed the new system board), then the condition doesn't go to `ok` but rather changes to unusable. This also occurs if the memory fails the test needed to connect (`cfgadm -o normal -t ac1:bank0`). An `unusable` occupant cannot go to any other phase and can only be unplugged. A slot will also be `unusable` before attempting a DR operation if there is insufficient power or a faulty power/cooling module. Therefore, always check the slot condition with `cfgadm` before starting a DR operation. If it states `unknown,` everything is ok.

If something goes wrong after the initial tests are run and the occupant condition is `ok`, the condition will change to `failing`. This can occur with a system board if it is overheating, possibly because a fan has failed, but hasn't been shut off yet. If this were a CPU/Memory Board, the CPUs would be close to critical temperatures, and Sun Management Center (if installed) would start warning the administrator. If the failing power/cooling module is replaced before the CPUs shut off the board, the condition will revert to `ok`. The condition will go to `failed` if the board overheats and the CPUs are turned off automatically. The only way to recover is to disconnect the board (logically, not physically) and then reconnect, reconfigure and restart the board and devices.

If many correctable errors are occurring in memory, its condition will go to failing. This is a sign that the memory has to be replaced before uncorrectable errors start occurring. The condition will move to `failed` if unrecoverable errors start to occur. This can soon result in a system panic.

Final Figure

This leads to a final figure containing all elements in it (see Figure 10). If this figure is too confusing, please refer back to the earlier figures that reflect the partial picture.



Receptacle	Occupant	Condition	LEDs
Empty	Unconfig.	Unknown	- - -
Disconn.	Unconfig.	Unknown	Off On Off
Connected	Unconfig.	OK	On Off Off
Connected	Configured	OK	On Off Flash

```
cfgadm -c connect sysctrl10:slot#
cfgadm -c disconnect sysctrl10:slot#
cfgadm -c configure sysctrl10:slot#
cfgadm -c unconfigure sysctrl10:slot#
```

Memory and other:
```
drvconfig
disks, tapes, ports, devlinks
```
CPU: `psradm -n #`

Memory:
```
cfgadm -o quick -t ac#:bank#
cfgadm -c configure ac#:bank#
```
Other: `mount, ifconfig, ...`

CPU: `psradm -f #`

Memory: `cfgadm -c unconfigure ac#:bank#`

Other: `mount, df -k, fuser, ifconfig, ps, kill, ...`

Figure 10 Complete model of all DR phases.

Help 4

This chapter contains lists and tables that make it easy to look up "what you always wanted to know but couldn't find." Many of these lists and tables are a single page, making them easy to copy.

Like an appendix, this chapter is an incoherent grouping of different sections. There are checklists to help think of everything, machine templates to get an overview of what is in a system, copies of tables used earlier in this document, a section containing useful tips, and a listing of common error messages. The `cfgadm` command is also explained in more detail.

All can be copied, expanded, and altered to suit a specific situation.

Checklists

This section contains two checklists designed to help sort out things to think about and do. The first checklist itemizes things that should be checked before implementing DR on a system (see Table 9). The second list contains things to think of when starting DR or while in the midst of a DR operation (see Table 10).

Before:

What to look at:	References:
What is the **goal** of the DR operation?	“Policies” section on page 37, “What do I want to achieve?”
Check the Web site:	*http://sunsolve5.sun.com/sunsolve/Enterprise_dr/*
Make an overview of what hardware is in the system.	“Example Template” section on page 84, use `prtdiag` and `cfgadm -v` to locate hardware, “Preparing the System for DR” section on page 48
Solaris version	`/etc/release` and the “Requirements” section on page 25
OBP Firmware versions	`banner` and `.version` in ok prompt mode, install patches, the “Requirements” section on page 25 & “Preparing the System for DR” section on page 48
No interleaving. (OBP setting)	`setenv memory_interleave min`, in the ok prompt.
Are `/etc/system` settings set correctly?	The “Requirements” section on page 25 and “Preparing the System for DR” section on page 48
Are all the devices in the system suspend-safe?	The “Requirements” section on page 25 and Web site.
Can I quiesce the system?	`cfgadm -x quiesce-test sysctrl0:slot#`, “Preparing the System for DR” section on page 48
Where is the vital hardware located in the system?	The “Requirements” section on page 25, “Preparing the System for DR” section on page 48, and `cfgadm -v`, `df -k`, `swap -l`
Do all the devices that will be added or removed support detach/attach?	The “Requirements” section on page 25 and Web site.
Is there hardware that requires patches for DR (e.g., FC-AL connections)?	Web site.
Policies and Procedures:	
Do I want prepare for unscheduled DR?	“Policies” section on page 37
What to do when doing scheduled and unscheduled DR.	“Policies” section on page 37 and this checklist, or setup own checklist. Different levels of emergency.
Do I want to use spares?	Should there be spares in stock in case they are needed.
Try on a development system first.	Get familiar with DR in controlled environment first.
Where are the DR plans kept?	Make it clear were the know-how is stored.
Is there any point in the day when the system is less stressed, reducing the risk in the event of a failure?	

Table 9 Checklist of issues to consider before implementing DR

Applications:	
What type of applications are running on this system?	“Policies” section on page 37
How will these applications react to DR?	
Are there any “real-time” applications?	
Do I want to use the AP software?	“Links to Other Applications” section on page 41
Do I want to use the Sun Management Center software?	“Links to Other Applications” section on page 41
Is this system using DiskSuite?	“Links to Other Applications” section on page 41
Is this system using Veritas Volume Manager with DMP or AP?	“Links to Other Applications” section on page 41
Is there anything else on the system that can impact DR?	

Table 9 Checklist of issues to consider before implementing DR

During:

What to look at:	References:
What is the goal of the DR operation?	"Policies" section on page 37, "what I want to achieve?"
Check the Web site:	*http://sunsolve5.sun.com/sunsolve/Enterprise_dr/*
Is all the needed personnel present?	Different levels: administrator familiar with the setup, certified personnel to take out boards, etc.
Where are the DR plans kept?	The overview of the hardware and software setup of the system
Check overview the hard- and software in the system.	Use prtdiag and cfgadm -v to locate hardware, and df -k, fuser and ps to locate software, "Preparing the System for DR" section on page 48
Solaris version	`/etc/release` and the "Requirements" section on page 25
Try on a development system first.	
OBP Firmware versions	Check new boards in development system: banner and .version in ok prompt mode, the "Requirements" section on page 25 and "Flowchart" section on page 55
No interleaving. (OBP setting)	`eeprom memory_interleave`, in a shell.
Are /etc./system settings set correctly?	The "Requirements" section on page 25 and "Preparing the System for DR" section on page 48
Are all the devices in the system suspend-safe?	The "Requirements" section on page 25 and Web site.
Can I quiesce the system?	`cfgadm -x quiesce-test sysctrl0:`*`slot#`* "Preparing the System for DR" section on page 48
Where is the vital hardware located in the system?	The "Requirements" section on page 25, "Preparing the System for DR" section on page 48, and `cfgadm -v`, `df -k`, `swap -l`
Do all the devices that will be added or removed support detach/attach?	The "Requirements" section on page 25 and Web site.
Is there hardware that requires patches for DR (FC-AL connections)?	Web site: *http://sunsolve5.sun.com/sunsolve/Enterprise_dr/*
Do I want have spares?	

Table 10 Checklist of issues to consider during installation.

What type of applications are running on this system?	“Policies” section on page 37
Is there a point in the day that the system is less stressed, reducing the risk in case of a failure?	
How will these applications react to DR?	Are there any “real-time” applications?
Does the system have the AP software?	“Links to Other Applications” section on page 41
Do the system have the Sun Management Center software?	“Links to Other Applications” section on page 41
Is this system using DiskSuite?	“Links to Other Applications” section on page 41
Is this system using Veritas Volume Manager with DMP or AP?	“Links to Other Applications” section on page 41
Is there anything else on the system that can impact DR?	
Is the fault light on?	If the fault light is on, DR should not be started before it is clear what caused the fault.
What state are the system boards in? `cfgadm -v` and `prtdiag`	
When inserting a board, be sure it is done straight and swiftly.	
When taking an I/O board from the running phase to the idle phase use: `ps`, `fuser`, `mount`, `lockfs`, `prtdiag` and cto help.	“Configured to Running Phases” section on page 62

Table 10 Checklist of issues to consider during installation.

System Templates

It is useful to gain an overview of what is in a system and which boards can participate in a DR operation — if the system supports DR at all. This section contains a few system templates intended to help create a major part of this overview.

There is a separate template for every system type because each contains a different number of receptacles (slots). Each template is on a separate page, so that it can be easily copied and then filled in for DR installations.[1]

Example Template

Board Information				Device Information[1]						Total	
Slot #	**Board Type**	**Sun Part #**	**DR**							**DR**	**Sus-pend**
0	CPU/Mem	501-4946	Y	X2570A	Pr	X2570A				N	Y
1	Graphic I/O	501-4884	N	X1065A	S/D	X3651A	S	X1145A	S/D	N	Y
2	CPU/Mem	501-4946	Y	X2570A		x2570A		1 GB		Y	Y
3	SBus I/O	501-4883	Y					X1057A	S/D	Y	Y
4	CPU/Mem	501-4946	Y	X2570A		X2570A				Y	Y
5	CPU/Mem	501-4946	Y	X2570A		X2570A		2 GB	Pm	N	Y
6											
7	CPU/Mem	501-4946	Y	X2570A		X2570A		1 GB		Y	Y

1. Comments: Pm = Permanent Memory; Pr = Primary CPU; S = Suspend -safe; S / D = Suspend-safe / Detachable; NS = Not Suspend Safe

Solaris Version:	7 5/99
OBP Version:	3.2.22
System Name:	Iowa
Host ID:	

Table 11 Example template for a Sun Enterprise 4000, 4500, 5000, or 5500 server.

1. If the table entries are too small, use your copier's enlarge feature and copy in landscape format..

Table 11 contains an example template that has been filled out for a Sun Enterprise 4500 server. The first columns contain information about the board itself: the slot number, board type, Sun part number, and whether this board type is Hot Pluggable. The next columns contain information about the devices that are present on the board and if they support suspend/resume and detach/attach. The final columns contain information about the sum of the board and its devices, indicating which can participate in DR operations.

Sun Enterprise 3000 Template

Board Information				Device Information[1]			Total	
Slot #	Board Type	Sun Part #	DR				DR	Sus-pend
0								
1								
2								
3								

1. Comments: Pm = Permanent Memory; Pr = Primary CPU; S = Suspend -safe; S / D = Suspend-safe / Detachable; NS = Not Suspend Safe

Solaris Version:	
OBP Version:	
System Name:	
Host ID:	

Table 12 Example template for Sun Enterprise 3000 servers.

Sun Enterprise 3500 Template

Board Information				Device Information[1]			Total	
Slot #	Board Type	Sun Part #	DR				DR	Sus-pend
0								
1								
2								
3								
4								

1. Comments: Pm = Permanent Memory; Pr = Primary CPU; S = Suspend -safe; S / D = Suspend-safe / Detachable; NS = Not Suspend Safe

Solaris Version:

OBP Version:

System Name:

Host ID:

Table 13 Example template for Sun Enterprise 3500 servers.

Enterprise 4000/4500/5000/5500 Template

Board Information				Device Information[1]			Total	
Slot #	Board Type	Sun Part #	DR				DR	Sus-pend
0								
1								
2								
3								
4								
5								
6								
7								

1. Comments: Pm = Permanent Memory; Pr = Primary CPU; S = Suspend -safe; S / D = Suspend-safe / Detachable; NS = Not Suspend Safe

Solaris Version:

OBP Version:

System Name:

Host ID:

Table 14 Example template for Sun Enterprise 4000, 4500, 5000, and 5500 server.

Enterprise 6000/6500 Template

Board Information				Device Information[1]			Total	
Slot #	Board Type	Sun Part #	DR				DR	Sus-pend
0								
1								
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								

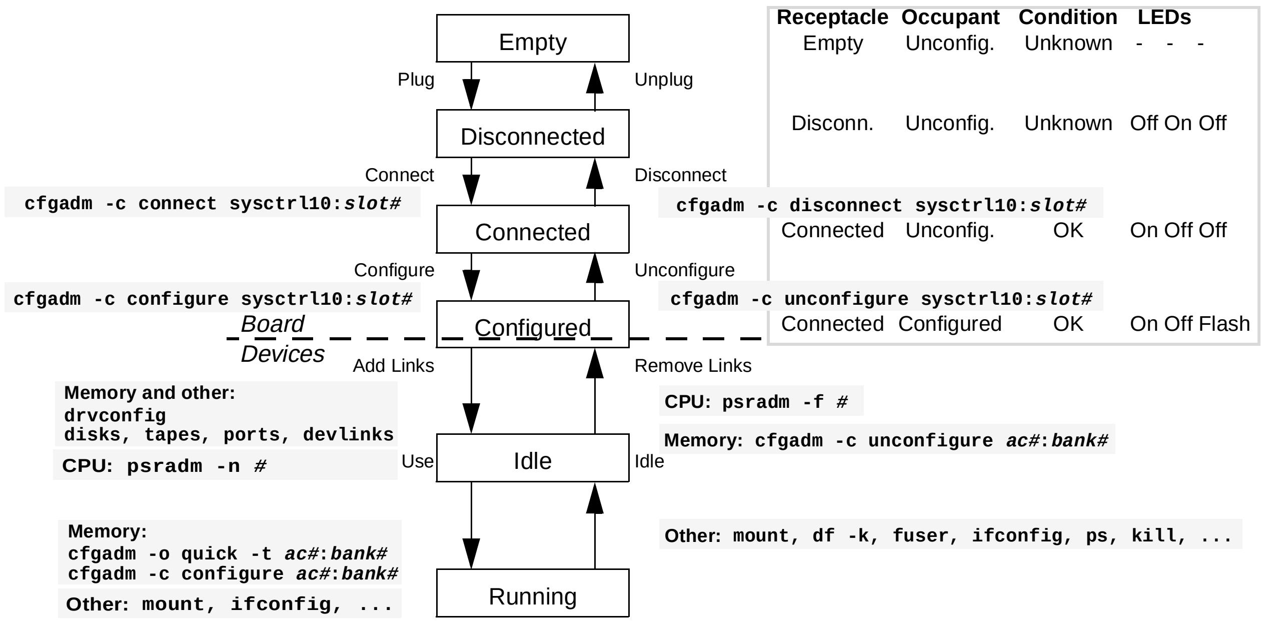
1. Comments: Pm = Permanent Memory; Pr = Primary CPU; S = Suspend -safe; S / D = Suspend-safe / Detachable; NS = Not Suspend Safe

Solaris Version:	
OBP Version:	
System Name:	
Host ID:	

Table 15 Example template for Sun Enterprise 6000 and 6500 servers.

Phases - States - Conditions Figure

Figure 11 is a repeat of the final figure from Chapter 3. This figure shows the different states a board can have and what kind of conditions correspond to this.



```
cfgadm -c connect sysctrl10:slot#
cfgadm -c disconnect sysctrl10:slot#
cfgadm -c configure sysctrl10:slot#
cfgadm -c unconfigure sysctrl10:slot#
```

Memory and other:
```
drvconfig
disks, tapes, ports, devlinks
```
CPU: `psradm -n #`

CPU: `psradm -f #`

Memory: `cfgadm -c unconfigure ac#:bank#`

Memory:
```
cfgadm -o quick -t ac#:bank#
cfgadm -c configure ac#:bank#
```
Other: `mount, ifconfig, ...`

Other: `mount, df -k, fuser, ifconfig, ps, kill, ...`

Receptacle	Occupant	Condition	LEDs
Empty	Unconfig.	Unknown	- - -
Disconn.	Unconfig.	Unknown	Off On Off
Connected	Unconfig.	OK	On Off Off
Connected	Configured	OK	On Off Flash

Figure 11 Complete model of all DR phases

Supported Devices

This section contains a list of the system boards and devices that support DR. This section is closely related to the “Requirements” section on page 25, and many of the tables found here are identical. This section is intended as an overview. For in-depth information, please refer to the earlier section.

The up-to-date list of supported boards and devices is at: *http://sunsolve5.sun.com/sunsolve/Enterprise_dr/*

The following section include lists of systems, boards and devices and details when they are supported with DR.

Systems

Sun Enterprise 3000, 3500, 4000, 4500, 5000, 5500, 6000 and 6500 servers are all supported, including all types of chassis. The HPC versions of these machines are not supported.

System Boards Supported for Hot Plug and Quiescence

The following CPU/Memory Boards and SBus I/O boards are supported for Hot Plug and Quiescence in Solaris 7 5/99:

Type of Board	**Market Part #**	**Sun Part #**
CPU/Memory (<2MB, 83 MHz)	X2600A	501-2976
CPU/Memory (83 MHz)	X2601A	501-4312
CPU/Memory (83/90/100 MHz)	X2602A	501-4882
SBus I/O (SOC, 83 MHz)	X2610A	501-2977, 501-4287
SBus I/O (SOC+, 83 MHz)	X2611A	501-4266
SBus I/O (SOC+,83/90/100MHz)	X2612A	501-4883

Table 16 System boards that support Hot Plug with Solaris 7 5/99.

System Boards Supported for Quiescence Only

The following system boards are supported for quiescence only and do not provide Hot Plug support in Solaris 7 5/99:

Graphics I/O Boards and PCI I/O Boards

Type of Board	Market Part #	Sun Part #
Graphics I/O (SOC, 83 MHz)	X2620A	501-2749, 501-4288
Graphics I/O (SOC+, 83/90/100 MHz)	X2622A	501-4884
PCI I/O (83 MHz)	X2630A	501-3023
PCI I/O (83/90/100 MHz)	X2632A	501-4881

Table 17 System boards that support only quiescence with Solaris 7 5/99.

In Solaris 2.6 all system boards that have SOC+ connections need multiple patches. Check with the Web site mentioned above for the most recent patches.

Note – Disk Boards connect only for power, and therefore are always supported with DR.

Expansion Cards

SBus Cards that Support Detach/Attach and Quiescence

Type of Card	Market Part #	Driver
SunSwift, SE Fast/Wide SCSI, Fast Ethernet 10/100BaseT	X1018A*	fas/sd, hme
Gigabit Ethernet 1.0 1000BaseFX	X1045A	ge
Quad Fast Ethernet 2.0 10/100BaseT	X1049A*	qfe
Differential Fast/Wide SCSI, Buffered Ethernet 10BaseT	X1052A*	esp/sd, le
Single-End Fast/Narrow SCSI, Buffered Ethernet 10BaseT	X1053A	esp/sd, le
Fiber Channel, FC-25	X1057A*	soc/pln/ssd
Quad Ethernet 10BaseT	X105L8A	le
Fast Ethernet 2.0 10/100BaseT, MII	X1059A*	hme
Differential Fast/Wide SCSI	X1062A*	isp/sd
Single-Ended Fast/Wide SCSI	X1063A*	isp/sd
Differential Ultra/Wide SCSI	X1065A*	isp/sd
Gigabit Ethernet 2.0 1000BaseFX	X1140A	ge
FDDI 6.0 Single Attach	X1142A*	nf
FDDI 6.0 Dual Attach	X1143A*	nf
High Speed Serial Interface	X1145A	hsi
ATM 4.0 155/MFiber	X1147A	ba
ATM 4.0 155/UTP5	X1148A	ba
ATM 4.0 622/MFiber	X1149A	ba
TurboGX 8-bit color graphics	X3655A*	cgsix
Fiber Channel-Arbitrated Loop, FC-100	X6730A*	socal/sf/ssd
TurboGX Plus 8-bit color graphics	X7110A*	cgsix

Table 18 SBus cards that fully support DR.

All boards (see Table 18) that have a “*” next to their part number are supported with Solaris 2.6. Be aware that the FC-AL Card has a SOC+ controller which needs multiple patches with Solaris 2.6.

SBus Cards that Support only Quiescence in Solaris 7 5/99

Type of Card	Market Part #	Driver
Token Ring Interface 4.0	X1144A	tr
Serial Parallel Controller	X1146A	spif

Table 19 SBus cards that support only quiescence.

Neither card is currently supported in Solaris 2.6.

PCI Cards that Support only Quiescence in Solaris 7 5/99:

Type of Card	Market Part #	Driver
SunSwift, SE Ultra/Wide SCSI, Fast Ethernet 10/100BaseT	X1032A*	fas/sd, hme
Fast Ethernet 2.0 10/100BaseT, MII	X1033A*	hme
Quad Fast Ethernet 2.0 10/100BaseT	X1034A*	qfe
Gigabit Ethernet 1.0 1000BaseFX	X1044A	ge
Gigabit Ethernet 2.0 1000BaseFX	X1141A	ge
FDDI Single Attach	X1152A	pf
FDDI Dual Attach	X1153A	pf
Token Ring Interface	X1154A	tr
High Speed Serial Interface	X1155A	hsi
Serial Asynchronous Interface	X1156A	sai
ATM 6.0 155/MFiber	X1157A	ba
ATM 6.0 155/UTP5	X1158A	ba
ATM 6.0 622/MFiber	X1159A	ba
PGX8 Graphics Card	X3660A	m64
PGX32 Graphics Card	X3668A	gfxp

Table 20 PCI cards that support only quiescence.

All boards that have a “*” next to their part number are supported with Solaris 2.6.

UPA Cards that Support only Quiescence in Solaris 7 5/99

Type of Card	Market part#	Driver
Creator Graphics	X3653A*	ffb
Creator 3D Graphics	X3669A*, X3675A*, X3671A*	ffb

Table 21 UPA cards that support only quiescence.

All boards that have a "*" next to their part number are supported with Solaris 2.6.

Operating system/Firmware versions

OS Version	Firmware	DR Support
Solaris 7 8/99	Ver. 3.2.22	CPU, SBus I/O, SBus I/O+
Solaris 7 5/99	Ver. 3.2.22	CPU, SBus I/O, SBus I/O+
Solaris 7 3/99	Ver. 3.2.21	SBus I/O, SBus I/O+
Solaris 7 FCS	Ver. 3.2.21	SBus I/O, SBus I/O+
Solaris 2.6 5/98	Ver. 3.2.21	SBus I/O, SBus I/O+

Table 22 DR support provided by various operating system / firmware combinations.

Tips

This section contains tips and tricks for using DR.

- ***Use a development system to pre-test new boards.***

  To minimize the risk of putting faulty parts into a production system, pre-test the board on a development system. This will help trace incorrectly assembled CPUs, memory or other devices. Then unplug the board and plug it into the production system.

- ***Check the connector on the system board that is going to be added.***

  Check if the connector shows any faults or if it is dirty. This will greatly diminish the risk of getting bent pins.

- ***Don’t force new boards when plugging them in.***

  If the board shows great resistance when attempting to plug it in, this is an indication something is wrong. If the board is then forced in, chances are very high that a pin will be bent.

- ***Make swap on more than one chain to avoid “vitalness.”***

  Make two or more swap partitions on different disks on different I/O boards to avoid dependency on one I/O board.

- ***Mirror the boot disk to avoid “vitalness.”***

  Mirroring or alternate pathing the boot disk eliminates its dependency on one I/O board.

- ***The network interface carrying the system name can't be dynamically reconfigured.***

  If the network interface has the same name as the system (the name in `/etc/nodename`), it generally can't be Dynamically Reconfigured. This interface is considered the primary interface by certain network services, like NIS, and is where they poll for external information. Taking this interface off-line will cause these services to freeze and wait for its return. One solution to this problem is to use Alternate Pathing and have this primary interface alternate pathed over two interfaces on two separate boards.

- ***Check the fault lights on the system.***

  There are two sets of three system lights, one set on the back and another on the front. These lights indicate the general system status. If there are any hardware faults, like a broken power supply, the middle orange light will be lit. Never start a DR operation if this light is on and it isn't clear why.

- ***Get the Memory-Cage on the same board as the primary CPU.***

  It is possible to place the memory-cage on the same board as the primary CPU when booting. This is handy because it reduces the number of permanent boards to one, but it does require some additional configuration. Memory banks can be disabled at boot time by putting them in the `disabled-memory-list` in the OBP. The numbers in that list represent the boards that have their memory banks disconnected at boot. *Be very sure that one bank is still enabled, or it is impossible to boot or get into the OBP!* Disable all boards except the first CPU/Memory Board. After the system boots, connect and configure the remaining memory on the other boards (as if it's newly added memory) with `cfgadm`. Then start all the applications. The memory-cage and the primary CPU are on the same board.

- ***Do not perform your first DR on a critical live system.***

  Even later be very careful with DR on critical system. While DR can help reduce the need for downtime, there are no guarantees of zero downtime.

- ***Use diag-switch? for more output.***

  By setting the `diag-switch?` to true the administrator will see more output when connecting a system board. This is not as extensive as at boot up, but more than if the switch is set to false. The switch can be set to true by using: `eeprom diag-switch? true`.

- ***SBus address numbers.***

  There is an easy translation from the SBus address (i.e. `/devices/sbus@3,0/SUNW...`) to the slot this SBus I/O board is located on. Every SBus I/O board has two SBuses. The system board slot numbers start with 0 and can go up to 15. So for slot 0 the corresponding SBus numbers are 0 and 1. In slot 15 they are 30 and 31. The general equation is S*2 and (S*2)+1, where S is the slot number.

- ***Find out which devices on a board can be made idle.***

  Find out which devices can be made idle with the following commands: `fuser`, `ps`, `prtdiag`, `mount` and `ifconfig`.

- ***Why can't I get the I/O board in board slot 1 to disconnect?***

  A common cause of I/O board disconnect failure is when the system detects devices in use on the board. A typical error message is:

```
cfgadm: Hardware specific failure: unconfigure failed: device did not detach:
/sbus@3,0/SUNW,fas@3,8800000/sd@6,0
```

Many times the process using the device can be found using the `fuser` program:

```
# fuser /devices/sbus@3,0/SUNW,fas@3,8800000/sd@6,0*
/devices/sbus@3,0/SUNW,fas@3,8800000/sd@6,0:a,raw: 315o
```

The process(es) may then be listed using the `ps` command:

```
# ps -f -p 315
 UID   PID  PPID  C    STIME TTY      TIME CMD
root   315     1  0 22:43:51 ?        0:00 /usr/sbin/vold
```

The above example is typical of disconnect problems with the board in slot 1, where `vold` may be holding open devices that are connected internally to the SCSI bus on an I/O board in this location. The `vold` daemon is commonly used for factory-installed SCSI devices in the SCSI cage. Note that slot 1 is unique among the board slots in that it is the only slot with a cable connector on the center-plane. In the factory configuration, a cable links this connector to the SCSI cage at the front of the system.

The following procedure applies to disconnect problems due to device monitoring by the `vold` daemon. If you want to disconnect an I/O board that has a SCSI device that is controlled by the `vold` daemon, you can temporarily disable `vold` before disconnecting the board:

a. As root, halt the `vold` daemon:

```
# /etc/init.d/volmgt stop
```

b. You can now dynamically disconnect I/O board.

c. Restart `vold`:

```
# /etc/init.d/volmgt start
```

The cgfadm Command

The cfgadm command is the primary tool used to perform all DR-related tasks. This commandline command is used in a terminal window, and can also be incorporated into shell scripts.

There are two sides to the cfgadm command. One the one hand, it can be used as a reporting tool to find out the status of the system and its elements. On the other hand, it's a configuration tool used to manipulate the states of the boards and devices in the system.

The basic way to see what is in the system is with cfgadm -l (omitting the -l gives the same result):

```
# cfgadm -l
Ap_Id                    Receptacle   Occupant     Condition
ac0:bank0                empty        configured   ok
ac0:bank1                empty        unconfigured unknown
ac2:bank0                empty        unconfigured unknown
ac2:bank1                empty        unconfigured unknown
sysctrl0:slot1           connected    configured   ok
sysctrl0:slot3           connected    configured   ok
sysctrl0:slot5           connected    configured   ok
sysctrl0:slot7           disconnected unconfigured unknown
sysctrl0:slot9           connected    configured   ok
```

Use the `cfgadm -v` command to see more detailed information:

```
# cfgadm -v

Ap_Id                    Receptacle   Occupant     Condition  Information
When            Type          Busy     Phys_Id
ac0:bank0                connected    configured   ok         slot5 1Gb base 0x0
Oct  5 15:08 memory       n        /devices/fhc@a,f8800000/ac@0,1000000:bank0
ac0:bank1                empty        unconfigured unknown    slot5 empty
Oct  5 15:08 memory       n        /devices/fhc@a,f8800000/ac@0,1000000:bank1
ac1:bank0                connected    configured   ok         slot7 1Gb base 0x40000000 permanent
Oct  5 15:08 memory       n        /devices/fhc@e,f8800000/ac@0,1000000:bank0
ac1:bank1                empty        unconfigured unknown    slot7 empty
Oct  5 15:08 memory       n        /devices/fhc@e,f8800000/ac@0,1000000:bank1
ac2:bank0                empty        unconfigured unknown    slot9 empty
Oct  5 15:08 memory       n        /devices/fhc@12,f8800000/ac@0,1000000:bank0
ac2:bank1                empty        unconfigured unknown    slot9 empty
Oct  5 15:08 memory       n        /devices/fhc@12,f8800000/ac@0,1000000:bank1
sysctrl0:slot1           connected    configured   ok         100 MHz capable
Oct  5 15:08 soc+upa      n        /devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot1
sysctrl0:slot3           connected    configured   ok         100 MHz capable
Oct  5 15:08 soc+sbus     n        /devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot3
sysctrl0:slot5           connected    configured   ok         non-detachable   100 MHz capable
Oct  5 15:08 cpu/mem      n        /devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot5
sysctrl0:slot7           connected    configured   ok         100 MHz capable
Oct  5 15:08 cpu/mem      n        /devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot7
sysctrl0:slot9           connected    configured   ok         100 MHz capable
Oct  5 15:08 cpu/mem      n        /devices/central@1f,0/fhc@0,f8800000/clock-board@0,900000:slot9
```

This gives much more information, however the output is pretty crowded. The `cfgadm` command also has options to just display certain entries from the `cfgadm -v` output.

For example, if only the Information entries are desired, then use: `cfgadm -s cols=ap_id:info` (The ap_id is included to see the corresponding receptacle):

```
# cfgadm -s cols=ap_id:info
Ap_Id                  Information
ac0:bank0              slot5 1Gb base 0x0
ac0:bank1              slot5 empty
ac1:bank0              slot7 1Gb base 0x40000000 permanent
ac1:bank1              slot7 empty
ac2:bank0              slot9 empty
ac2:bank1              slot9 empty
sysctrl0:slot1         100 MHz capable
sysctrl0:slot3         100 MHz capable
sysctrl0:slot5         non-detachable   100 MHz capable
sysctrl0:slot7         100 MHz capable
sysctrl0:slot9         100 MHz capable
```

All kinds of combinations can be used to output exactly what is needed for a good assessment of the system state. Use the blocks `ap_id`, `physid`, `r_state`, `o_state`, `condition`, `type`, `busy`, `status_time`, `status_time_p` and `info`, with the symbol “:” as a field separator.

The `cfgadm` command also can be used to change the state of a system board. This is done with the `cfgadm -c` command. For example to change the state of the system board in slot 7 (`sysctrl0:slot7`) from *connected* to *configured*, use: `cfgadm -c configure sysctrl0:slot7`. The output will look like this:

```
# cfgadm -c configure sysctrl0:slot7
Oct  5 18:06:01 sgkona-3 unix: NOTICE: configuring cpu board in slot 7
Oct  5 18:06:01 sgkona-3 unix: NOTICE: cpu board in slot 7 is configured
```

The `-c` options all require a *desired state* and *receptacle* as arguments. The *desired state* can be: `connect`, `configure`, `disconnect` and `unconfigure`. The *receptacle* either starts with `sysctrl` or `ac`.

Before newly added memory can be configured, it first must be tested. This is done with the `-t` option, in combination with the `-o` option specifying the level of testing (`quick`, `normal` and `extended`). An example is: `cfgadm -o quick -t ac1:bank0`. If all goes well, there is no output from this command.

Finally, the `cfgadm` command also allows the administrator to test if a certain system board supports quiescence. This done by entering `cfgadm -x quiesce-test sysctrl0:slot3`, which either results in an error message or just the return of the prompt.

For further in-depth information please check the DR User's Guide and the `man` pages on `cfgadm`.

Error Messages

This section discusses typical error conditions and their possible meaning.

- ***Unable to perform an Unconfigure operation.***

  Unconfigure may fail because of an attempt to unconfigure a board with a busy or open device. In such a situation the I/O board is now only partially unconfigured. The operation has stopped at the busy device. In order to regain access to the devices which did get unconfigured, the board must be completely unconfigured and then reconfigured. In such a case, the system will log messages similar to the following:

```
NOTICE: unconfiguring dual-sbus-soc+ board in slot 7

NOTICE: dual-sbus-soc+ board in slot 7 partially unconfigured reason:
sysc iohelp unconfigure: Device busy
output from sysctrl unconfigure is:
detach failed: /sbus@f,4000/SUNW,foo@3/bar@2,0 is busy
```

  To continue the unconfigure operation, the remedy is to unmount the device and retry the unconfigure operation. Note that an attempt to configure this board again without first successfully transitioning to the unconfigured state is not allowed.

- ***Unable to perform a Configure operation.***

  Configure may fail because of an attempt to configure a board with a device that does not currently support hot plugging. In such a situation the I/O board is now only partially configured. The operation has stopped at the unsupported hot plug device. In this situation, the board must be brought back to the unconfigured state before another configure attempt is made. In such a case, the system will log messages similar to the following:

```
NOTICE: configuring dual-sbus-soc+ board in slot 4
NOTICE: dual-sbus-soc+ board in slot 4 partially configured reason:
sysc iohelp configure: Bad address
output from sysctrl configure is:
attach failed: /sbus@8,0/SUNW,foo@d,10000/bar
```

To fix this problem and continue the configure operation, either remove the unsupported device's driver or replace it with a new version of the driver that will support hot plugging.

References A

For additional information on Dynamic Reconfiguration and related subjects, please see the following Web sites and other resources:

- Dynamic Reconfiguration information:
  - *http://sunsolve5.sun.com/sunsolve/Enterprise_dr/*
  - The Dynamic Reconfiguration users guide.
  - The answerbook on DR on the documentation CD of Solaris.
  - *http://docs.sun.com*
- Alternate Pathing information:
  - *http://sunsolve5.sun.com/sunsolve/Enterprise-dr/APIndex.html*
  - The Alternate Pathing users guide.
  - The answerbook on the supplemental CD shipped with Solaris that contains AP
  - *http://docs.sun.com*
- Sun Management Center information:
  - *http://www.sun.com/sunmanagementcenter*
  - *http://docs.sun.com*
- General information:
  - *http://www.sun.com/servers/white-papers/dc-planning-guide.pdf*
  - *http://docs.sun.com*
  - The answerbook on the documentation CD of Solaris.
  - *http://www.sun.com*
  - *http://www.sun.com/blueprints*
  - *http://www.sun.com/servers/midrange/*

Glossary B

AP

See Alternate Pathing.

ac

Address controller. The `cfgadm` status report lists memory banks in the order of the board address controller numbers (ac0, ac1, ac2, and so forth). Note that the ac numbers are not listed in the order of their physical board slot numbers, but in the chronological order in which the CPU / memory boards were inserted into the system. Thus, if the second CPU / memory board is already in slot 7, and you now install a third CPU / memory board in slot 4, a cfgadm status report would list the third CPU / memory board (ac2) after the second CPU / memory board, even though the third CPU / memory board is in a lower-numbered physical slot.

ap_id

Attachment point identifier. An ap_id specifies the type and location of the attachment point in the system and is unambiguous. There are two types of identifiers: physical and logical. A physical identifier contains a fully specified pathname, while a logical identifier contains a shorthand notation.

Alternate Pathing

Alternate Pathing (AP) is software package that allows the use of multiple paths between a server and a disk array or a network. If one path fails, AP can ensure that the disk array or network is still available through the alternate path. For example, the alternate path can be a second port on an interface board, or an entirely separate interface board. See also Dynamic Reconfiguration.

Attachment point

A collective term for a board and its card cage slot. A physical attachment point describes the software driver and location of the card cage slot. A logical attachment point is an abbreviated name created by the system to refer to the physical attachment point.

cfgadm command

The `cfgadm` is the primary command for dynamic reconfiguration on the Sun Enterprise 6x00, 5x00, 4x00, and 3x00 systems. For information about the command and its options, refer to the `cfgadm(1M)`, `cfgadm_sysctrl(1M)`, and `cfgadm_ac(1M)` man pages. For any late-breaking news about this and related commands, refer to the Solaris 7 5/99 section at the DR web site.

Condition

The operational status of an attachment point.

Configuration (system)

The collection of attached devices known to the system. The system cannot use a physical device until the configuration is updated.

Configuration (board)

The operating system assigns functional roles to a board and loads device drivers for the board and for devices attached to the board.

Connection

A board is present in a slot and is electrically connected. The temperature of the slot is monitored by the system.

Detachability

The device driver supports `DDI_DETACH` and the device (such as an I/O board or a SCSI chain) is physically arranged so that it can be detached.

Disconnection

The system stops monitoring the board and power to the slot is turned off. A board in this state can be unplugged.

DR

See Dynamic Reconfiguration.

Dynamic Reconfiguration

Dynamic Reconfiguration (DR) is software that allows the administrator to (1) view a system configuration; (2) suspend or restart operations involving a port, storage device, or board; and (3) reconfigure the system (detach or attach hot-swappable devices such as disk drives or interface boards) without the need to power down the system. When DR is used with Alternate Pathing or Solstice DiskSuite software (and redundant hardware), the server can continue to communicate with disk drives and networks without interruption while a service provider replaces an existing device or installs a new device. DR supports replacement of a CPU/Memory board, provided the memory on the board is not interleaved with memory on other boards in the system.

Hot-plug

Hot-plug boards and modules have special connectors that supply electrical power to the board or module before the data pins make contact. Boards and devices that do not have hot-plug connectors cannot be inserted or removed while the system is running.

Hot swap

A hot swap device has special DC power connectors and logic circuitry that allow the device to be inserted without the necessity of turning off the system.

Logical DR

A DR operation in which hardware is not physically added or removed. An example is the deactivation of a failed board that is then left in the slot (to avoid changing the flow of cooling air) until a replacement is available.

Occupant

Hardware resource such as a system board or a disk drive that occupies a DR receptacle or slot.

Physical DR

A DR operation that involves the physical addition or removal of a board. See also Logical DR.

Quiescence

A brief pause in the operating environment to allow an unconfigure and disconnect operation on a system board with non-pageable OpenBoot PROM (OBP) or kernel memory. All operating environment and device activity on the backplane must cease for a few seconds during a critical phase of the operation.

Receptacle

A receiver such as a board slot or SCSI chain.

State

The operational status of either a receptacle (slot) or an occupant (board).

Sun Management Center

Sun Management Center is a graphical user interface for monitoring and managing systems. The interface includes dynamic reconfiguration capability.

Suspendability

To be suitable for DR, a device driver must have the ability to stop user threads, execute the DDI_SUSPEND call, stop the clock, and stop the CPUs.

Suspend-safe

A suspend-safe device is one that does not access memory or interrupt the system while the operating system is in quiescence. A driver is considered suspend-safe if it supports operating system quiescence (suspend/resume). It also guarantees that when a suspend request is successfully completed, the device that the driver manages will not attempt to access memory, even if the device is open when the suspend request is made.

Suspend-unsafe

A suspend-unsafe device is one that allows a memory access or a system interruption while the operating system is in quiescence.

Unconfiguration

The system detaches a board logically from the operating system and takes the associated device drivers off-line. Environmental monitoring continues, but any devices on the board are not available for system use.