



SOLARIS CONTAINERS

Virtualization Techniques

White Paper

March 2006

Table of Contents

Executive summary	3
About the authors	3
Introduction	4
Solaris Zones	5
Provisioning	5
Configuration	5
Installation	7
Initial boot.....	7
Advanced Solaris Zone design and implementation	8
Backup and recovery considerations	10
Product and patch management	10
Package management	10
Patch management.....	11
Security	12
Service consolidation	13
Resource management	15
Key elements of resource management	16
Dynamic Resource Pools and Fair Share Scheduler	17
Projects	18
Memory capping	19
IP quality of service	20
Extended accounting	20
Conclusion	21
More information	22
Glossary (Appendix)	23

Chapter 1

Executive summary

With the release of the Solaris™ 10 Operating System (OS), support was introduced for the virtualization of application environments. This ability, described as Solaris Containers, permits the system administrator to more effectively manage the workloads of both dissimilar and similar applications on a single host.

Containers are groupings of application processes and tasks with resource controls applied. Given the granularity of this resource management technique, the data center can benefit from server segmentation and isolation with Solaris Zones, a special form of resource control, while making maximum utilization of system resources.

This document is designed to help system administrators get started using best practices when deploying Containers, and is meant to be used in addition to existing documentation. The information in this document was determined through a joint effort between Sun and Electronic Data Systems Corporation (EDS). Best practices described within this document are based on existing Sun support as of the time of this writing. This includes core administrative concepts such as provisioning, maintenance, patch management, and security considerations. Also discussed are best practices relating to resource management in Solaris Containers, such as using projects, applying resource limits, and using accounting.

Solaris Containers can have a significant impact on total cost of ownership for IT systems. They enable businesses to better manage large-scale systems through workload consolidation, without traditional concerns about impacts caused by resources, security, or faults. EDS invested significant resources in evaluating Solaris Containers, and is applying this knowledge to an initiative that will assist their customers in reducing IT costs while improving responsiveness and availability—this paper shows some of the intellectual property that has been gained through their efforts.

About the authors

Ronald DeMena III serves EDS as an Infrastructure Associate in the Service Delivery organization. DeMena holds more than 10 years of experience in the IT industry. His tenure with EDS has consisted of technical leadership roles focusing on datacenter practices and platform interoperability.

Keith Bates is a Senior Infrastructure Specialist in the Service Delivery organization within EDS. Bates has more than 20 years of experience in the IT industry. During most of his tenure with EDS, he has held technical leadership roles supporting client delivery, with a focus on operating systems and infrastructure platforms.

The authors gratefully acknowledge the contributions of Joost Pronk Van Hoogeveen, Product Line Manager for virtualization technologies in the Solaris group at Sun Microsystems.

Chapter 2

Introduction

Hardware and software advances have led to the concept of server virtualization—the partitioning of a networked server into several independent execution environments. Server virtualization enables a data center to be viewed and managed as a set of compute resources rather than a room of individual systems, allowing businesses to reduce costs through improved efficiency and better resource utilization while improving service levels.

Solaris Containers establish boundaries for resources, such as CPUs, and can be expanded to adapt to changing processing requirements of the application or applications running in the container. A container is a virtualized operating system environment created within a single instance of the Solaris OS. Applications within Containers are isolated, preventing processes in one container from monitoring or affecting processes that are running in another container.

A container also provides an abstract layer that separates applications from the physical attributes of the system where they are deployed. An example of these attributes is physical device paths. Containers also enable more efficient system utilization. Dynamic resource reallocation permits unused resources to be shifted to other Containers as needed. Fault and security isolation means that poorly behaved applications do not require dedicated and under-utilized systems. With Containers, these dissimilar applications can be consolidated onto the same physical system. Containers also allow the system administrator to delegate some administrative functions to localized container accounts, while maintaining overall system security and integrity.

Solaris Containers are designed to provide fine-grained control over resources that applications use, allowing multiple applications to operate on a single server while maintaining specified Quality of Service (QoS) levels. Fixed resources such as processors and memory can be partitioned into pools on multiprocessor systems, with different pools shared by different projects (a specified collection of processes) and isolated application environments.

Dynamic resource sharing allows different projects to be assigned varied ratios of system resources. Solaris IP Quality of Service (IP QoS) can be employed to manage network bandwidth required by multiple, competing network applications. When resources such as CPUs and memory are dynamically allocated, resource capping controls can set limits on the amount of resources used by a project. With all of these resource management capabilities, organizations can consolidate many applications onto a single server, helping reduce operational and administrative costs while increasing availability.

Solaris Containers can be used to isolate and manage multiple applications on the same server to lower costs and complexity. The technology enables IT organizations to deploy multiple, competing applications on the same platform, and multiple departments or organizations can employ a single server, each with its own secure environment. For example, Containers can effectively deploy hosting facilities, consolidate multiple database instances, consolidate development and test environments, stage applications, and consolidate Web services systems.

Solaris Containers consist of two distinct, yet complementary, parts—Solaris Zones and Solaris Resource Manager. Solaris Zones create independent environments where applications can run. Solaris Resource Manager allocates resources to each of these environments. A zone with or without resource management is a Solaris Container, and vice versa. However, an optimized Solaris Container combines both Solaris Zones and Solaris Resource Manager.

Chapter 3

Solaris Zones

Solaris Zones, a new form of resource control described and referenced as a virtual platform, enables the administrator of a Solaris host to consolidate applications and isolate them from intrusion and faults by provisioning independent virtual environments. This isolation essentially provides improved security by allowing a cleaner, more simplified management of multiple customers within a single Solaris host, also referenced in this document as the *global zone*. Within Solaris 10, any zone that is not the global zone is described as *non-global zone*, because of its locality to a single Solaris 10 host.

This section provides a brief overview of Solaris Zones. Additional information can be obtained from documents cited in the More Information section.

Provisioning

Once a Solaris 10 host is identified, an administrator may begin to provision zones to provide a virtual environment to future application occupants. When deciding on whether to use a sparse or *whole root* zone, and what system components to bind to a zone, it is important to carefully evaluate which services will be provided. Consider the following before implementing core functionality:

1. Is the application sensitive to the versioning of the Solaris shared libraries?
2. How much disk space is available for consumption?
3. What network configurations are needed?

Non-global zones can consist of two types: *sparse root* or *whole root*. The sparse root zone model optimizes sharing of objects. The whole root zone model provides maximum configurability. The benefits of using either of these two types should be carefully weighed against the needs of the application or service owners' requirements:

A *sparse root zone* is a partial replica of the global zone using a loopback file system (LOFS) to access its shared libraries. It inherits packages and patches distributed through the global zone. When implemented, this zone type can initially consume as little as 188 MB of disk space (or less in some cases).

A *whole root zone* is a whole replica of the global zone using its own physical copy of the system's shared libraries. It also inherits the entire package and patch database, and maintains its own physical copy of product contents. When implemented, the disk space costs are significantly higher, starting with approximately three GB.

Through the use of these two zone types, system operations can be segregated and further isolated for both various customers and applications, depending on the architectural design. In addition to zone types, each zone can be found in one of several states. These states—*configured*, *incomplete*, *installed*, *ready*, *running*, *shutting down*, and *down*—are used to describe the current operational state of the non-global zone.

Configuration

Every zone isolates resources at several layers within the operating system. This isolation and interactivity with the Solaris host is achieved using the zone administration daemon (zoneadmd), which is responsible for all communication between

the various administrative components of the zone and the operating system. These components break down as follows:

- Console (`zcons`)
- Network interfaces (`ifconfig`)
- File systems (LOFI, LOFS, UFS, NFS, and others)
- Devices (`/dev`)
- Package and patch registry (`pkg(3m)` and `patch(3m)`)

New Solaris Zones are designed and assembled using `zonecfg` (a zone-specific XML validator). It is recommended that the design take into account the aforementioned provisioning questions and the physical resources available for the system, including capacities for disk, CPU, memory, and network interfaces.

A zone name must be defined at the execution of `zonecfg` (`zonecfg -z <zonename>`). Zone naming practices should consider the following:

- Using a common string of characters for all non-global zones within a global zone.
- Using standard hostname conventions, which should not exceed 15 characters.

After creating the specified zone with its defined name (using `create`), the design phase of any given zone should include the following specifications:

- Zone storage path location (`set zonepath`)
- Autoboot functionality (`set autoboot`)
- Inherit package directories (`add inherit-pkg-dir`, `remove inherit-pkg-dir`)
- Network configuration (`add net`)

Although a zone does not require a network interface and IP address, most will be configured with these capabilities. By default the design uses inheritance of the `/usr`, `/lib`, `/platform`, and `/sbin` directories through a loopback file system to create a *sparse root zone*. If needed, administrators can also include the `/opt` directory for package inheritance (`add inherit-pkg-dir dir=/opt`). When all of these inheritances are removed, the zone is changed from sparse to whole root; however, it is advised that unless there is a significant need for local copies of the system's shared libraries, inheritance should be used. To remove inheritance, employ the following entries using `zonecfg`:

```
zonecfg:zonename> remove inherit-pkg-dir dir=/usr
zonecfg:zonename> remove inherit-pkg-dir dir=/lib
zonecfg:zonename> remove inherit-pkg-dir dir=/platform
zonecfg:zonename> remove inherit-pkg-dir dir=/sbin
```

Zone functionality can be greatly improved using the `zonecfg` command and the following parameters:

- Device configuration (`add device`)
- File systems configuration (`add fs`)
- Resource management (`add rctl`)

Once an administrator completes the design and verifies that all required information is present, the zone is committed. As a global administrator in the global zone, you can use the following commands:

```
zonecfg:zonename> verify verifies the configuration of a non-global zone
zonecfg:zonename> commit commits the zone configuration for the zone
zonecfg:zonename> info displays the configuration of a named non-global zone
```

Installation

The second phase to zone provisioning involves the actual construction of the physical zone on the system. When constructing the physical zone, an administrator must consider disk utilization, and determine how the location defined in the configuration is to be provided. By establishing a separate branch from the root file systems, such as `/zones`, any administrator may clearly identify the zone path of each zone as `/zones/<zonenumber>`.

When establishing the physical zone path, an administrator should perform the following functions:

1. Create a directory in the `/zones` tree named after the `zonename` itself.
2. Provision a separate file system to eliminate disk space contention; this can be done using Solaris Volume Manager or other volume managers such as Veritas.
3. Mount the provisioned disk to the defined `zonepath`.
4. Change the permissions of the physical directory to `0700`.

Once these steps are performed, the system administrator can then install the zone by executing the `zoneadm` command (`zoneadm -z <zonenumber> install`). At this time, the global zone issues a series of commands executing Live Upgrade to create the initial zones. Full replication of the current package and patch database occurs. Upon completion, the zone's root directory is constructed (`/zones/<zonenumber>/root`) in the installed state. If the zone is not found in the installed state, or is found as uninstalled (`zoneadm list -cv`), then the administrator must uninstall the zone, and then reinstall.

Initial boot

An administrator can boot a zone, which places it in the running state. Use the `zoneadm` command with the `-z` option, the name of the zone, which is `my-zone`, and the `boot` subcommand to boot the zone.

```
global# zoneadm -z zonename boot
```

When the installation completes, use the `list` subcommand with the `-v` option to verify the status.

```
global# zoneadm list -v
```

Something similar to the following should be displayed:

ID	NAME	STATUS	PATH
0	global	running	/
1	zonename	running	/export/home/zonename

When booting, if the following message is displayed:

```
# zoneadm -z zonename boot
zoneadm: zone 'zonename': WARNING: hme0:1: no matching subnet
found in netmasks(4) for 192.168.0.1; using default of
255.255.255.0.
```

The command has succeeded, and the message is only a warning. The message indicates that the system was unable to find the netmask to be used for the IP address specified in the zone's configuration. To stop the warning from displaying on subsequent reboots, ensure that the correct netmasks databases are listed in the `/etc/nsswitch.conf` file in the global zone and that at least one of these databases contains the subnet and netmasks to be used for the zone `zonename`.

When the zone is first booted, the system builds the slave device tree (`/zones/<zonename>/dev`) used by `zoneadmd` to manage interaction with global zone devices. These devices are representative of the zone console (`zcons`), network interfaces, and any other additional devices bound to the zone. After creation of these devices, an administrator must login to the zone through its console (`zlogin -C <zonename>`) and answer the following prompts:

1. Zonename
2. Kerberos Security Setting
3. Name Service Details
4. Locality and Time zone
5. Root Password
6. NFS Settings

After the prompts are completed, the zone reboots to instantiate the new settings and begins standard operations. To avoid these prompts, a system administrator may create a `sysidcfg` file following standard JumpStart™ rules, which can be placed into the `/zones/<zonename>/root/etc` directory. It is extremely important that the same user account name service, such as NIS or LDAP, is utilized in both the global and non-global zone to avoid conflicts or compromise of user identities and or data, and to ensure a clear understanding of data ownership from within the global zone.

Advanced Solaris Zone design and implementation

Additional changes may be required in a zone's configuration, either during the initial design of the zone or after it has been installed. For example, administrators may need to further provision hard disk devices, file systems, networks, or other devices.

When establishing additional storage space for an up and running zone, the best way to protect data and provision the space among one or more file systems is to mount the specific storage device to a branch extending from the `zonepath` itself (`/zones/<zonename>/fs/<storage device>`). By creating a location for mounting these devices behind the `zonepath`, you can protect the device from compromise by unprivileged users within the global zone, as well as ensure simple comprehension of the device within the zone. By using the name of the storage device as its mount point within the file system tree, any administrator can quickly comprehend and manage the storage by traditional means, and `zonecfg` from within the global zone.

Consider the following code and `zonecfg` commands. See Figure 1.

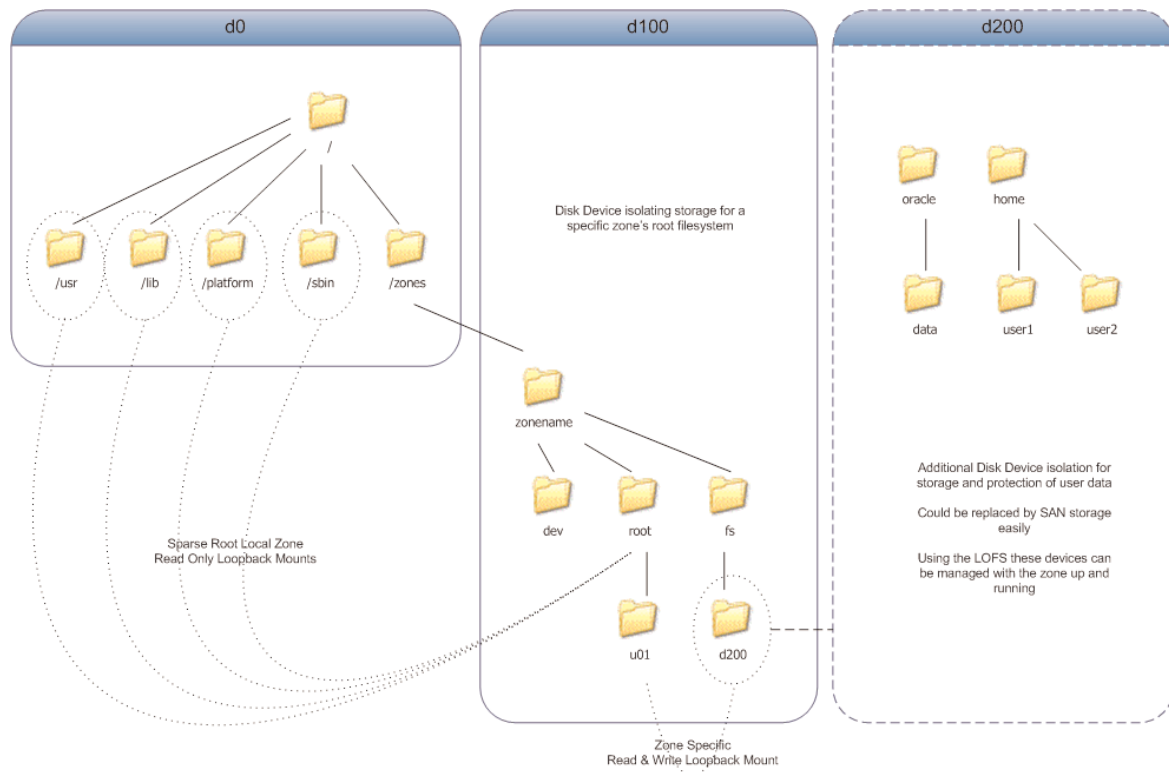
```

/zones/<zonename>/fs/c1t0d0s0      # Physical Device
/zones/<zonename>/fs/d200         # Meta Device
/zones/<zonename>/fs/vxvol01     # Veritas Volume
/zones/<zonename>/fs/lofi-file   # Loopback File Interface

zonecfg:zonename> add fs
zonecfg:zonename:fs> set dir=/u01
zonecfg:zonename:fs> set special=/zones/zonename/fs/d200
zonecfg:zonename:fs> set type=lofs
zonecfg:zonename:fs> add options [rw,nodevices]
zonecfg:zonename:fs> end

```

Figure 1: File system view



Any need for a physical device to be used within a zone must also be carefully evaluated. Upon assignment, that device becomes unavailable to the remaining system processes. It is best to provision mounts using `zonecfg` from the non-global zone itself. Assignment of devices such as disks, tapes, libraries, serial interfaces, and more are best left in the global zone and shared appropriately through `zoneadm`. If a loopback file interface (LOFI) is used it must be recreated after every system reboot. For additional reference, the following mounting abilities exist within Solaris 10:

Table 1: File system capabilities

File System	Mount via <code>zonecfg</code>	Mount from the global into the non-global zone	Mount from within the non-global zone
AutoFS	No	No	Yes
CacheFS	N/A	N/A	N/A
FDFS	Yes	Yes	Yes
HSFS	Yes	Yes	Yes
LOFS	Yes	Yes (safe but some performance issues)	Yes
MNTFS	No	No	Yes
NFS	No	No	Yes (V2, V3, and V4)
PCFS	Yes	Yes	Yes
PROCFS	No	No	Yes
TMPFS	Yes	Yes	Yes
UDFS	Yes	Yes	Yes
UFS	Yes	Yes	Yes
XMEMFS	Yes	Yes	Yes

Backup and recovery considerations

With the introduction of Solaris Zones, significant changes in backup and recovery techniques need to be made. Basic technologies such as `tar`, `ufsdump`, `fssnap`, and `ufsrecover` enable backup of user information within a non-global zone while operating from the global zone as part of the Solaris 10 OS. Independent software vendors (ISVs) employing these technologies for the foundations of their products should have little difficulty in this adoption.

Solaris Zone technology enables administrators to back up an entire virtual data center from a single backup manager, where previously one backup manager would be required for each server. However, zone technology can create complications for non-global zone users who in the traditional single-host arrangement would have been able to instantiate self recovery of their data. Now, due to the security isolation of the non-global zone from both the global zone and neighboring zones, these users are unable to access the information needed to perform such backup and recovery themselves.

Users must work through appropriate backup and recovery teams with an individual who has access to the global zone from which they are hosted, and they need to know the `zonename`. Without the `zonename`, the administrator is unable to determine the `zonename` where their root file system is extended, which for security reasons is unavailable to the end user from within the zone. The easiest way for users to identify if they are running from within a zone is to execute the `zonename` command. If it returns a name different than `global`, this information needs to be provided along with the `hostname` to the system administrator responsible for data protection efforts.

Product and patch management

To address package and patch management on a Solaris 10 host, the system administrator must consider all zone environments. When applying a patch, the use of the `inherit-pkg-dir` has significant importance. If a package has been installed into any of the inherited directories, as defined from with `zonecfg (zonecfg -z <zonename> info)`, the system administrator must understand the propagation rules for its registry information. In addition, when applying patches to a whole root zone, administrators should consider the impact of additional writes across I/O channels.

When a zone is initially created, the entire package and patch registry information is replicated into it. Additionally, a subset of packages with the `SUNW_PKGTYPE=root` parameter is also replicated into the zone, and the remainder of the files provisioned through a read-only LOFS (loopback file system). In the event the zone is a whole root zone, all packages and patches referenced in the global zones registry are also replicated to a second location, occupying up to an additional three GB of disk space initially, compared to the minimal occupied by the creation of a sparse root zone.

To ensure work against the Solaris Zones on any given host is accounted for, it is critical that all zones be in a running state before any activity is performed. If a zone is not in a running state, the potential that a package or patch will fail is significant, as the OS will attempt to online in single user mode all installed zones.

Package management

To ensure packages are accounted for regardless of their installation into either the global zone only, the non-global zone only, or all zones, the features of the new package and patch info files are as follows:

- The `SUNW_PKG_ALLZONES` parameter can control propagation throughout the system when set to either `true` (to enforce), or `false` (to deny). When set to `true`, this feature forces the default installation of the package into the global zone, followed by distribution into each of the non-global zones on the Solaris 10 host—the package must be identical across all zones. When set to `false` the package installs only into the specific zone where it is executed unless requested with the `pkgadd -z`.

- The `SUNW_PKG_HOLLOW` parameter is used to indicate placeholder needs within the non-global zones. When installed with this parameter set to `true`, the global zone need only replicate its package database into both current and future non-global zones. When set to `false`, it forces the package to be replicated into the non-global zone, regardless of its type.
- The `SUNW_PKG_THISZONE` parameter is used to force restriction of installation to the zone where the package install is executed. Similar to running `pkgadd -G` in the global zone, the package does not propagate to non-global zones. When it is true and executed within the non-global zone, the package installs only within that non-global zone.

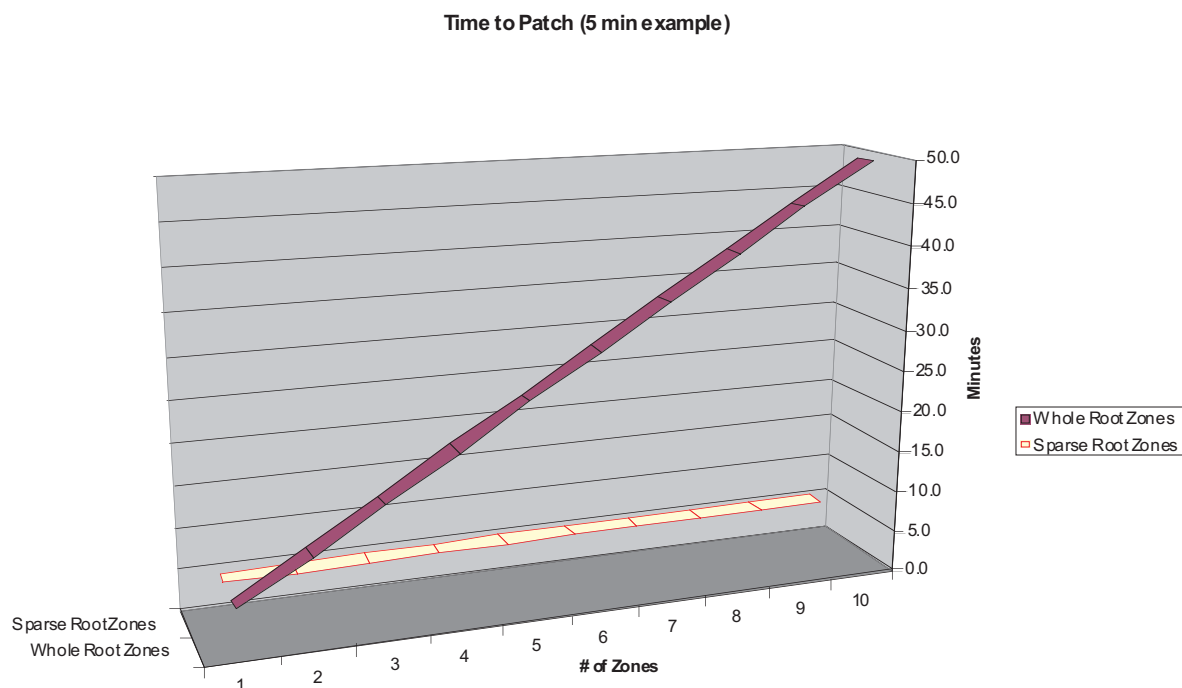
Although specific combinations of these parameters are invalid, it is important to understand the underlying concepts. From a best practices perspective, all OS-dependent software that would be patched by traditional means should be installed using `pkgadd` from the global zone. This technique forces adequate propagation of package details into each non-global zone, and maintains an accurate master copy of the package registry within the global zone for patch management.

If a package is installed and dispensed to all non-global zones, there is a possibility that it could be removed from one or more of them (using `pkgrm`), which would leave it in a partially installed state within those zones. This effect is caused by the inheritance of the global zone's directory structures via the LOFS, and the lack of permissions to affect the specified file systems.

Patch management

When addressing patch application within a Solaris 10 host with zones, administration is best performed from within the global zone.

Figure 2: Predicted time difference to patch sparse root zones compared to whole root zones.



Similar to traditional patch application using the `patchadd` command, using this command in the global zone by default applies the patch to the global zone. The default action is to propagate those patches into all non-global zones, as well. This affects both the storage space occupied by the patch and its entries into the registries of each zone. Although execution of `patchadd` within the non-global zone can be performed under specific conditions, it is best managed only through the global zone, to ensure appropriate distribution to each non-global zone.

Situations where removal of a patch (`patchrm`) from within a non-global zone may occur, will affect only local files that are not shared between the Solaris Zones. When patch removal occurs from the global zone, the patch is removed from all non-global zones, as well. This is also an example of protection through the use of the LOFS with a sparse root. Additionally, system instability can occur when the global zone has a shared library mismatch, such as `libc`, due to such a removal or an addition to a specific whole root zone only.

When choosing zone types, you should also account for the time to administer patches, as they are sequential in execution.

As shown in Figure 2, a patch that takes five minutes to apply is deployed across a series of zones. As the number of Solaris Zones running on the system increases, the time to patch sparse root zones increases slightly; whereas, the time to patch whole root zones increases significantly because of the duplicate writes to I/O. If combinations of non-global zones are chosen, the time could vary.

Security

Within the Solaris 10 Zone, several security features have been introduced. Through the use of read-only LOFS, administrators can minimize the threat of data destruction at both disk partitioning and file system levels, while limiting performance degradation to the impact of a symbolic link traversal. Commands such as `dd`, `fmthard`, `format`, `fdisk` (x86 and x64 systems), `mkfs`, and `newfs` are by default not present for use in the zone. Additional commands such as `mount`, which use system privileges that could have potential environmental impact, can also be disabled.

Due to these security features, if NFS mounts are used, they must be connected manually. To use NFS mounts for more permanent integration, employ the `vfstab` file with `soft` and `background` options, or more appropriately, the auto mounter.

Now root can be distributed within the non-global zone, without concern for possible system-wide impact. If preferred, root can be completely eliminated and privileges assigned through role-based access control (RBAC) profiles, although this will require extensive work to support. Privileges and associated availability within the non-global zone can be seen in Table 2.

Table 2: Solaris 10 privileges and their non-global zone availability

Privilege	Zone Privilege	Privilege	Zone Privilege	Privilege	Zone Privilege
Contract_event	Yes	ipc_dac_write	Yes	proc_taskid	Yes
Contract_observer	Yes	ipc_owner	Yes	proc_zone	No
cpc_cpu	No	net_icmpaccess	Yes	sys_acct	Yes
Dtrace_kernel	No	net_privaddr	Yes	sys_admin	Yes
Dtrace_proc	No	net_rawaccess	No	sys_audit	Yes
Dtrace_user	No	proc_audit	Yes	sys_config	No
file_chown	Yes	proc_chroot	Yes	sys_devices	No
file_chown_self	Yes	proc_clock_highres	No	sys_ipc_config	No
file_dac_execute	Yes	proc_exec	Yes	sys_linkdir	No
file_dac_read	Yes	proc_fork	Yes	sys_mount	Yes
file_dac_search	Yes	proc_info	Yes	sys_net_config	No
file_dac_write	Yes	proc_lock_memory	No	sys_nfs	Yes
file_link_any	Yes	proc_owner	Yes	sys_res_config	No
file_owner	Yes	proc_priocntl	No	sys_resource	Yes
file_setid	Yes	proc_session	Yes	sys_suser_compat	No
ipc_dac_read	Yes	proc_setid	Yes	sys_time	No

Because each Solaris Zone maintains its own protected `proc` and `dev` tree, no other non-global zone can access, administer, create, destroy, or modify its processes. Through this mechanism, one zone cannot access the resources of another zone, nor can the processes or executed tasks be snooped or traced. Due to this design, none of the non-global zones are aware of the others' presence on the system. However, the global zone still has one-way oversight and control of all processes contained within the non-global zones.

Within the IP stack, non-global zone traffic is clearly parsed through the global zone. When traffic occurs, the global zone uses IP filters and multiple default routes as defined in the `/etc/defaultrouter` file in the global zone. A non-global zone cannot view the traffic of any other zone. Only the interfaces initially assigned to the zone through `zonecfg` are available for use by that zone.

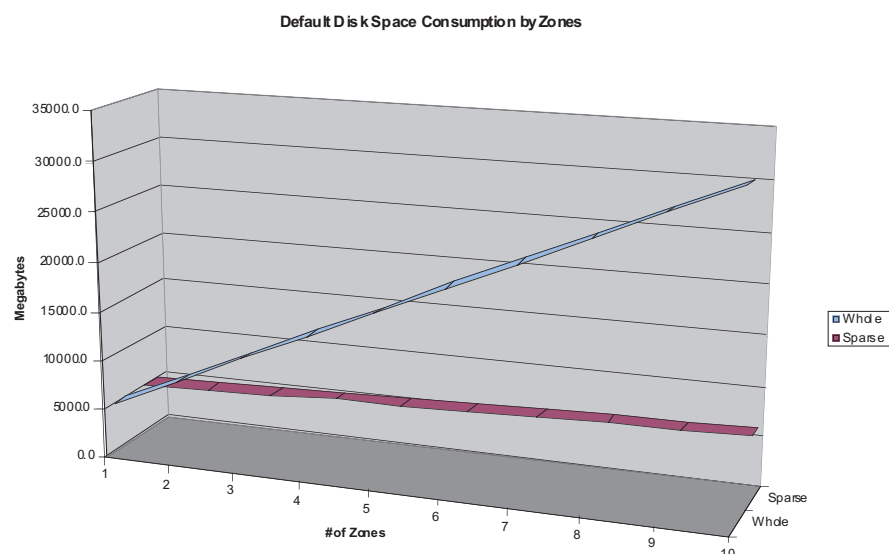
When considering IP multipathing, administrators should understand that it is configured only from within the global zone. If the zone-assigned interface is down and there is no access to its assigned network interface, the zone cannot bring up network connectivity and boot properly. Additionally, to further protect the global zone, an administrator can configure only the core interface within the global zone. Once the administrator completes the configuration, IP filters prevent external access to the global zone.

Service consolidation

Initial service consolidation can be best performed when limiting the number of Solaris Zones to 10 or less. In an effort to minimize the complexity of the system, and minimize potential customer impact, limiting the number of zones to four or less when using whole root zones is recommended.

When consolidating applications from across many systems, Solaris Zones enable significant benefits, which can be easily exploited by carefully designing each zone. In many cases, templates and scripting for zone creation can significantly accelerate zone provisioning and initial usage.

Figure 3: Default disk space consumption by zones.



When implementing more complex environments or even whole root zones, the administrative costs associated with both time and disk consumption must be accepted. Network complexities should also be considered. Consolidation of services into a zoned environment should be carefully evaluated and engineered to those situations, including required performance characteristics.

Chapter 4

Resource management

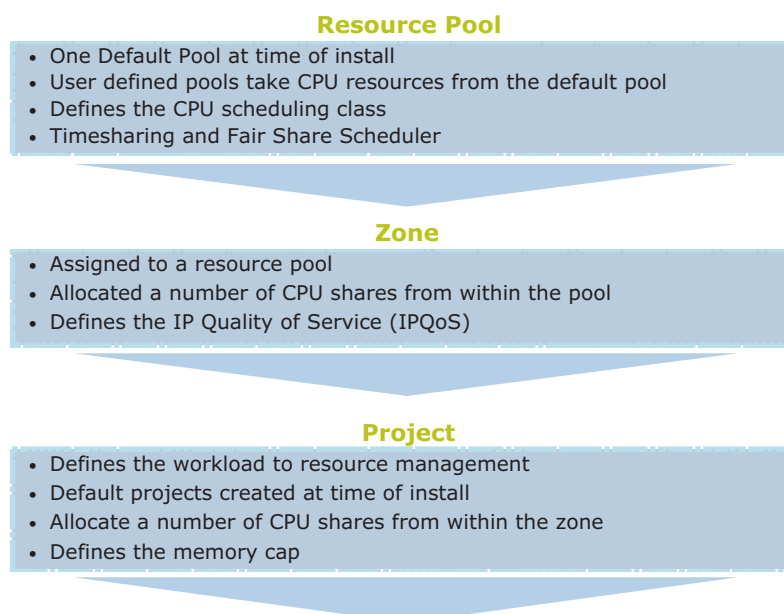
Resource management is a mechanism that defines resource utilization for application workload within Solaris. It is the means of allocating and controlling resources such as CPU, physical memory, and network I/O bandwidth. Applications, users, groups, and *virtual platforms* (zones) can be controlled through resource definitions. The resources available to each application can be prioritized based on business requirements, while still maintaining the flexibility to redirect unused and underutilized resources where they are needed within the UNIX® host. Resource limits can be set to prevent one user or application from impacting others on the system, allowing multiple workload environments to coexist on the same operating system platform.

Application collocation is best achieved by first understanding the resource requirements of workloads. Implementing Solaris Container technology can be considered as a three-step process:

1. Classify the workload into a container (zone or project).
2. Measure the resource requirements of the container.
3. Place resource controls on the container.

The number of applications that can coexist on the same Solaris platform is determined by the resource requirements of the workloads as a whole. Solaris Containers technology should be approached as a tiered hierarchy of resource management. Container resources can be managed at the host level (resource pools), zone level (zone CPU), or the project/task level within a zone. Resource management can be implemented independently, or integrated together at any level for a container solution. Complexity, required skills, and administrative overhead required for managing these resources all increase as you move down the hierarchy to the project level. From a strategic perspective, resource management should be maintained at the highest level of the hierarchy possible. Administrative overhead and complexity is reduced with this approach.

Figure 4: Applying resource management within the zone hierarchy



At the top level of the hierarchy are resource pools. CPU resources are initially contained in the default resource pool that is shared by all zones. CPUs may also be grouped in user-defined pools and then allocated to specific workloads as a means of resource isolation.

The middle tier of resource management is the allocation of CPU resources to the zone. The Fair Share Scheduler (FSS) can be enabled at the system level and CPU shares allocated to several Solaris Zones from a single resource pool, while a user-defined pool may be created with a subset of CPUs designated to a specific zone or zones. User-defined pools are optional and should be created only when there is a technical or business rationale for doing so.

At the lowest level of the hierarchy are projects within zones. Projects can be implemented within the global or non-global zone. Workloads or applications are logically grouped in project definitions within the zone as needed. Default project definitions are created when the zone is created. Project priority between workloads must be determined, and resource allocation defined on an as-needed basis, depending on the processing and business requirements of those workloads. In order to guarantee service levels, it is critical to understand the applications' resource needs, so resources can be allocated in a way that enables competing workloads to coexist in the same resource environment.

A resource plan provides a systematic approach to resource management. An initial workload can be placed on a system with reasonable resource controls applied. Thresholds can be defined to issue warnings if resource usage is out of bounds without denying the request for resources. Adjustments can be made until the appropriate resource requirements are identified. Finally, thresholds can be set to deny requests outside the allocated shares. A second workload can now be targeted at the environment without impacting the service level expectations of the initial workload.

Key elements of resource management

Resource Pools

- Partitions processor sets and CPUs into groups
- Provides a mechanism to isolate workload resources
- Defines the CPU scheduling class

Zones

- Provides the virtual environment
- Namespace, security, and fault isolation
- Prioritizes workload based on resource shares within the pool

Projects

- Identifies the user, group, or application to Solaris Resource Manager
- Logically groups related workloads into an administrative tag name
- Defines resource controls for a workload
- Prioritizes workload based on resource shares allocated within the zone

By default, all user workloads on a system share resources equally with all other user processes through the Time-sharing Scheduler (TS). This type of configuration may be acceptable in some situations, but the risk of one application workload taking control of the CPU resources does not inspire confidence about achieving reliable service levels. Invoking resource controls provides predictable service.

Dynamic Resource Pools and Fair Share Scheduler

A main feature of Solaris is the ability to control CPU allocations to applications through the use of Resource Pools and the Fair-Share Scheduler (FSS) scheduling class. Resource pools provide a means to separate workloads into pools of CPUs so that workload requirements do not directly compete for resources. Applications that are licensed by CPU may be another avenue to implement resource pools. Although CPU guarantees are possible through the use of the FSS and CPU shares, workloads compete for the same resources. Resource pools eliminate this competition when workloads are directed to a set of CPUs reserved for a specific application.

A new feature in Solaris 10 is Dynamic Resource Pools. This feature allows the system administrator to set guarantees around the utilization and number of CPUs available within the pool. When the resources defined to the pool are not fully utilized, CPUs are reallocated temporarily via the `poold` daemon to other pools as needed to take full advantage of the resources. This means that resources are not wasted. Resource changes performed by `poold` are logged for future analysis. This feature also reduces the administrative overhead required to manage pools since guarantees can be set, but resources are still fluid across pools.

From a best practices perspective, simpler is usually better. In an environment where all workloads or zones should be granted equal service and resources, user-defined resource pools may not be required. By default, all resources are initially allocated to the default resource pool. The FSS can be enabled and each zone granted equal CPU shares. Again, if one zone does not utilize the resources allocated, the resources are made available to other zones based on current resource requests of active zones that are competing for resources.

When a zone is granted shares of CPU resources at the host level, allocations of those shares can be further refined within the zone. Projects defined within the zone access only the CPU resources allocated to the zone from the pool. These allocations can be subdivided to a specific zone workload (if required) by the zone administrator, through the implementation of projects within the zone. A zone may be assigned to only one resource pool, although several zones may be assigned to a single pool.

Several options to the `zonecfg` utility are available for zone resource controls.

- `zone.cpu-shares=` specifies the number of FSS CPU shares available to the zone from the resource pool.
- `pool=` specifies the name of the resource pool the zone should be bound to when booted.
- `zone.max-lwps` is a resource control that enhances resource isolation by preventing too many lightweight processes (LWPs) in one zone from affecting other zones. The `prctl` command can be used to adjust these values in real time.

Resource pools may be useful in a situation where a different CPU scheduling class—such as time share, interactive, or real time—would benefit a particular workload or application. CPUs could be allocated to the user-defined pool and scheduling class set for that pool. An application or zone could then be directed to use that pool and scheduling class.

Another scenario to consider is the implementation of pools for server consolidation. Applications can be moved onto a new host server in phases. When the initial application is turned on, by default it has all resources available—possibly even more resources than will eventually be granted to the application when the consolidation of other applications is completed. This could have a misleading effect on the users of initial phase applications. Pools provide a means to set a hard cap on the number of CPUs available to the first phases of a consolidation, and validate initial sizing estimates before moving to subsequent phases of the consolidation.

The configuration option `CPU local` should be considered when creating resource pools. In general, the locality objective should be set to `tight`. However, to maximize memory bandwidth or minimize the impact of Dynamic Reconfiguration (DR) operations on a resource set, configure this objective to `loose`, or keep it at the default setting of `none` for applications that do not benefit from CPU proximity. DR operations do not proceed without a warning that the objectives of the pool cannot be maintained. Note that to proceed with the DR operation, the `force` option to DR must be specified.

As with projects, resource pools should always be administered through the facilities provided: `poolcfg`, `pooladm`, and `poolstat` are useful commands for pool administration. Editing files directly is discouraged, as it could lead to unanticipated system behavior. Dynamic changes invoked on the pool configuration are temporary and if desired, should be captured to the static file to survive reboots. To create a new static file that matches the dynamic configuration for subsequent loading of the configuration, the command `pooladm -s` is preferred over the previous function `poolcfg -c`.

Lastly, appropriate logging levels for `poold` should be enabled to capture warning and error messages in `syslog`, enabling operator intervention as needed. To specify the logging parameter, use the property name `system.poold.log-level`.

Projects

Projects define resource limits for the workload. All applications and users not associated with a specific, user-defined project run in the system-generated default project. Default projects and resource pools are initially created by the Solaris OS when resource management is activated. System processes run within the default project and are guaranteed resources as needed. As a best practice, it is recommended not to make changes to the default project, although it is possible. If the default project is not defined, a request from a user to login or start a process is denied.

A user's initial project is determined sequentially at login by the following methods:

- User has a project attribute in the `/etc/user_attr` file
- A project is defined as the `user.username` for this user's `user-id`
- A project is defined as the `group.groupname` for this user's `group-id`
- If none of the above produce a match, the system default project is used

Projects typically are defined in the local `/etc/project` database. Alternatively, a naming service such as NIS, or LDAP can supply additional entries. The `/etc/project` or `/etc/user_attr` files should not be edited directly. Commands such as `projadd`, `projmod`, `usermod`, and the Solaris Management Console should be used to ensure that syntax is correct in these files. If an invalid entry is encountered, such as might be the case if this file is edited directly, the system does not read remaining entries, possibly causing a catastrophic effect on service. Changes to the `/etc/project` file are not recognized until reboot, so changes that should persist across reboots should also be made to the `/etc/project` file with the `projmod` command. The `prctl` command can be used to make runtime changes to project resource controls. It is important to remember these changes are only in effect until the system reboots. All available resource controls are listed in the man page `resource_controls`.

The default project file contains the following entries at install time:

```
system:0:System:::
user.root:1:Super-User:::
noproject:2:No Project:::
default:3:::
group.staff:10:::
```

If the default configurations do not meet the processing requirements, user-defined projects should be created as a runtime environment to match the workloads. A user can be a member of several projects and invoke applications or processes within those projects where they are defined. User-defined projects may be created as application identifiers (such as `emp_benefits` or `prod_db`), with a list of specified users who are allowed to start processes within the project. The recommended implementation for user-defined projects is to create a `user.username` or `group.groupname` project for a user. This is assigned by default at login, but provides flexibility for that user to start applications through additional project definitions and memberships, as needed.

A user can start an application within a project in the following ways:

- As the user's default or current project membership, with the `newtask` command while specifying the `-p` option
- At system startup through the Solaris Service Management Facility (SMF). As a best practice, the SMF should be used for production applications to ensure consistency across system restarts.

The `prstat` command is used to examine real-time resource usage. The `prstat` utility iteratively examines all active processes on the system and reports statistics based on the selected output mode and sort order. `prstat` provides options to examine only processes matching specified PIDs, UIDs, zone IDs, processor sets, or projects. The `-J` option reports information about projects. In this mode, `prstat` displays separate reports by project ID. The `-Z` option can be added to report information per zone.

Projects are administered the same way within global zones and non-global zones. Each zone maintains an independent copy of the `/etc/project` file and has its own users, groups, and projects for resource management within it. To aid in reporting, and to ease user and application migrations, it is best to use consistent project names and numbers across the enterprise.

Memory capping

Memory capping is a function of the resource capping daemon `rcapd`. Restrictions can be placed on a project to limit the amount of physical memory consumed by processes running within it. Resource caps are defined as attributes of project entries in the `/etc/project` database. Within each zone environment, a separate `rcapd` daemon must be configured individually to control zone resources.

In order to use the facility, administrators must enable the daemon and configure a project with a resource cap applied. The `rcapadm` command allows a privileged administrator to configure various attributes of `rcapd`. `rcapadm` includes threshold options for cap enforcement. An individual workload can use physical memory up to and beyond its cap, until the resource capping threshold is reached, at which time physical memory is paged out to meet the resource threshold for the project and system. Resource caps are not enforced until a defined threshold is reached.

- `rcapadm` with no options displays current resource capping settings for the zone.
- `rcapadm -E` is used to enable `rcapd` at boot time.
- Specifying a system resource capping threshold of zero (the default value) always enforces memory caps defined on projects.

It is important to set memory caps at reasonable levels, otherwise the system physical memory is consumed before resource caps are exceeded, basically negating the purpose of the resource capping configuration. Projects that are set too low have excessive paging under normal processing load, negatively impacting performance. Care should be taken in the scope of implementation. CPU utilization of the `rcapd` daemon is in direct proportion to the amount of virtual memory and size of workload address space. A shorter scan interval defined through `rcapadm` can also increase CPU usage. `rcapstat` can be used to monitor the effectiveness of the resource capping configuration.

To safely disable resource capping, an administrator must always use the `rcapadm -D` option. If the `rcapd` daemon is killed outside of `rcapadm`, processes could be left in a stopped status and require a manual restart with the `prun` command.

IP quality of service

The IP QoS feature provides a means to define consistent levels of services to network users and manage network traffic at the zone level. It also allows administrators to rank and control network-based traffic, as well as gather network statistics. Before implementing IP QoS on the host, verify the compatibility of other network devices, such as firewalls and routers.

This feature is capable of controlling inbound and outbound traffic of a Solaris Zone by specifying the upper limit of the input/output network bandwidth. The packet is dropped if the limit is exceeded. Because IP QoS has a fair amount of CPU overhead, this is an optional feature and should be implemented with care.

Extended accounting

If projects are used to identify workloads, extended accounting can capture associated resource usage statistics. Accounting is useful to identify resource demands of the workloads on a historical basis and plan for future capacity. When run in the global zone, accounting information is gathered for both the global and non-global zones, and accounting records are tagged with the zone name to which they apply.

The `/etc/acctadm.conf` file contains the current configuration for extended accounting. This file should not be edited directly. The `acctadm` command is provided to specify configuration information. The default location for storing extended accounting data is `/var/adm/exacct`. A Perl interface, `libexacct`, is available to allow for the creation of custom reporting tools.

If the environment does not categorize workloads into identifiable projects, extended accounting may be of limited value, since task and projects accounting data are reported under the default project IDs.

Conclusion

Solaris Containers are created by combining two complementary, but independent technologies: Solaris Zones and Solaris Resource Manager software. These components address different qualities delivered by the container, and work together to create a complete container. Understanding the basic concepts and how to use them when installing and maintaining a Solaris Container can help administrators maintain a highly manageable and responsive IT environment.

Unlike systems that use high-overhead, virtual machine techniques to implement virtualization, Solaris Containers support mainframe-level partitioning capabilities with almost zero overhead. Sun's approach is unique in the industry for its ability to accomplish sub-CPU partitioning on systems as small as a single CPU. Solaris Containers are supported by the Solaris 10 OS, and customers can deploy them on systems with UltraSPARC™ or x86 architecture processors, including the AMD Opteron family. These capabilities are being explored by EDS for potential inclusion in its Service Delivery mode as part of an Agile Infrastructure. Sun is an EDS Agility Alliance partner.

More information

“Solaris Containers — What They Are and How to Use Them,” Menno Lageman

<http://www.sun.com/blueprints/0505/819-2679.pdf>

“System Administrators Guide: Resource Management and Containers,”

<http://docs.sun.com/app/docs/doc/817-1592>

“Bringing Your Application Into the Zone,”

http://developers.sun.com/solaris/articles/application_in_zone.html

EDS Agility Alliance

<http://www.eds.com/services/alliances/agility/>

EDS Hosting Services

<http://www.eds.com/services/hosting/>

System Administration Guide: Solaris Containers—Resource Management and Solaris Zones

<http://docs.sun.com/app/docs/doc/817-1592>

“Understanding the Basics About Solaris Containers in the Solaris 10”

http://www.sun.com/bigadmin/features/articles/basics_containers.html

Slicing and Dicing Servers: A Guide to Virtualization and Containment Technologies - Harry J. Foxwell

<http://www.sun.com/blueprints/1005/819-3734.pdf>

OpenSolaris Zones Community

<http://www.opensolaris.org/os/community/zones/>

“Creating Self-Balancing Solutions with Solaris Containers,” David Collier-Brown

<http://www.sun.com/blueprints/0605/819-2888.pdf>

Glossary (Appendix)

NFS	Network File System
UFS	UNIX File System
SCM	Solaris Container Management
LOFS	Loopback File system
LOFI	Loopback File Interface
SAN	Storage Area Network
FSS	Fair Share Scheduler

© 2006 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054 USA

All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California.

Sun, Sun Microsystems, the Sun logo, Java, JumpStart, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a). DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS HELD TO BE LEGALLY INVALID.



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



© 2006 Sun Microsystems, Inc. All rights reserved. Sun, Sun Microsystems, the Sun logo, Java, JumpStart, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd. Information subject to change without notice. EDS and the EDS logo are registered trademarks of Electronic Data Systems Corporation. 04.06