Oracle Solaris 11.3 Cheat Sheet
General Administration

System Configuration

Common system configuration tasks have changed in Oracle Solaris 11 with the Service Mana gement Facility (SMF) configuration repository being used to store configuration data. With the addition of configuration layers, administrators now have better control and assurance that their configuration changes will be preserved across system updates.

Configure nodename

```
# hostname myhost
```

Configure nameserver via SMF

```
# svccfg -s dns/client \
 setprop config/nameserver = net_address: 192.168.1.1
# svccfg -s dns/client \
 setprop config/domain = astring: \"myhost.org\"
# svccfg -s name-service/switch \
 setprop config/host = astring: \"files dns\"
# svcadm restart name-service/switch
# svcadm restart dns/client
```

Configure nameserver via SMF when you can't remember the correct properties to edit.

```
# svccfg -s dns/client editprop
# svccfg -s name-service/switch editprop
```

Configure nameserver (alternate approach by editing /etc/resolv.conf and /etc/nsswitch.conf and then importing these modifications into SMF.)

```
# nscfg import -f svc:/system/name-service/switch:default
# nscfg import -f svc:/network/dns/client:default
```

`nscfg` is a transition tool and not meant for daily use, not matter how useful it is. Please learn the regular way with the methods described above.

Unconfigure a system and start an interactive configuration tool on reboot

```
# sysconfig configure -s
```

Create a system configuration profile

```
# sysconfig create-profile -o sc-profile.xml
```

Configure a system according to a system configuration profile

```
# sysconfig configure -c sc-profile.xml
```

Did You Know?

You can find out more information about Oracle Solaris 11 including full product documentation, how to guides, and other cheat sheets on Oracle Technology Network: http://www.oracle.com/technetwork/server-storage/solaris11/overview/index.html

Locales, Timezone, and Keyboard

Install nlsadm for easier management of national language properties (Solaris 11.2)

```
# pkg install nls-administration
```

Get current configuration

```
# nlsadm get-console-keymap
# nlsadm get-system-locale
# nlsadm get-timezone
```

List available timezones

```
# nlsadm list-timezone
```

List available console keymaps

```
# nlsadm list-console-keymap
```

List available locales

```
# nlsadm list-locale
```

Set timezone to Europe/Berlin

```
# nlsadm set-timezone Europe/Berlin
```

Set locale to de_DE.UTF-8

```
# nlsadm set-system-locale de_DE.UTF-8
```

Set console keymap to UK-English

```
# nlsadm set-console-keymap UK-English
```

Import Legacy Files into SMF (nscfg)

The nscfg commands allow you to import the content of legacy files into the SMF. While the cheat sheet showed already two examples, the nscfg command isn't limited to these scenarios. When you make changes to the legacy files mentioned in the following tables (instead of doing it the SMF way), you can import them by `nscfg -fq <FRMI>`.

However keep in mind this is is a tool for doing transitions and not for daily admin usage. So learning to do it via svccfg is a really useful and recommended.

FRMI	Legacy files
`svc:/system/name-service/switch:default`	`/etc/nsswitch.conf`
`svc:/system/name-service/cache:default`	`/etc/nscd.conf`
`svc:/network/dns/client:default`	`/etc/resolv.conf`
`svc:/network/nis/domain:default`	`/etc/defaultdomain` `/var/yp/binding/$DOMAIN/*`
`svc:/network/nis/client:default`	no legacy file import
`svc:/network/ldap/client:default`	`/var/ldap/*`

`svc:/network/nis/server:default`	`no legacy file import`
`svc:/network/nis/passwd:default`	`no legacy file import`
`svc:/network/nis/xfr:default`	`no legacy file import`
`svc:/network/nis/update:default`	`no legacy file import`
`svc:/system/name-service/upgrade:default`	`no legacy file import`

Did You Know?

You can find a list of features that might disappear in future Oracle Solaris versions, so you can prepare for this situation. It's available at : http://www.oracle.com/technetwork/systems/end-of-notices/eonsolaris11-392732.html

Users and Roles

The traditional root account has been changed to a 'root' role on all Oracle Solaris 11 installations as part of the Role Based Access Control (RBAC) feature set. This change gives improved auditability across the operating system, and the ability for administrators to delegate various system tasks to others in a safe way.

Revert root to a normal user account

```
# rolemod -K type=normal root
```

Configure root as a role (default)

```
# usermod -K type=role root
```

Configure root role to use the user password instead of root password

```
# rolemod -K roleauth=user root
```

Add a new user and delegate the System Adminstrator profile to the user

```
# useradd -m -P "System Administrator" joerg
```

Add a new user with a ZFS file system as the user's home directory and add an entry in auto_home

```
# useradd -m -d localhost:/export/home/joerg2 joerg2
# grep "joerg2" /etc/passwd
joerg2:x:101:10::/home/joerg2:/usr/bin/bash
# cat /etc/auto_home
[...]
joerg2 localhost:/export/home/joerg2
+auto_home
# zfs list | grep joerg2
rpool/export/home/joerg           35K   201G    35K  /export/home/joerg2
```

Add a new user on a second server using another NFS server for the user's home directory

```
# useradd  -d nfsserver:/export/home/joerg2 joerg2
# grep "joerg2" /etc/passwd
joerg2:x:103:10::/home/joerg2:/usr/bin/bash
# tail -2 /etc/auto_home
joerg2 nfsserver::/export/home/joerg2
+auto_home
```

Boot Environments

Boot Environments are individual bootable instances of the operating system that take advantage of the Oracle Solaris ZFS file system snapshot and clone capability. During a system update, new boot environments are created so that system software updates can be applied in a safe environment. Should anything go awry, administrators can boot back into an older boot environment. Boot environments have low overhead and can be quickly created, giving administrators an ideal best practice for any system maintenance work.

List available boot environment

```
# beadm list
```

Create a boot environment:

```
# beadm create solaris-05032012
```

Activate a boot environment

```
# beadm activate solaris-05032012
```

Delete an inactive boot environment

```
# beadm destroy solaris-05032012
```

Show boot environments from SPARC boot PROM

```
ok boot -L
```

Boot into a boot environment from SPARC boot PROM

```
ok boot -Z rpool/ROOT/solaris-05032012
```

Package Versioning in Oracle Solaris 11

Solaris 11 packages contain a version string. When you look at it that looks a little bit cryptic at first. Here is how you decode `0.175.3.7.0.5.0`:

	Description
`0.175`	Build number of the development gate
`3`	*Update version*. In this case: it's an Oracle Solaris 11.3
`7`	*SRU*. In this case: it's the SRU 7
`0`	*Reserved*. This isn't currently used.
`5`	*Build number of the SRU.*
`0`	*Nightly build number.*

Sometimes you may encounter a slightly longer version string . When you see it: This is a version string of an IDR package (Interim Diagnostic/Relief). This is an example for it: `0.175.1.6.0.4.2.824.4`

	Description
`0.175.1.6.0.4.2.`	Same as with a regular package
`824`	*IDR.* The name of the IDR
`4`	*IDR-ID.* The version of the IDR

Packaging

Oracle Solaris 11 includes IPS, a new network-centric package management framework with automatic dependency checking. IPS has integrated package and patching, and can seamlessly manage system updates to Oracle Solaris Zones environments.

Install a package called `diagnostic/wireshark`:

```
# pkg install diagnostic/wireshark
```

Install a group package to provide a desktop environment

```
# pkg install solaris-desktop
```

Install package as if you were installing it on a freshly installed system

Warning! All packages that are not a dependency of this package will be removed while installing it. This is what the command is meant for.

```
# pkg exact-install web/server/apache-22/module/apache-wsgi-26
```

Update all possible packages to the newest version, including any zones

```
# pkg update
```

List available packages

```
# pkg list -a
```

Do a dry run of a system update to identify what packages might change

```
# pkg update -nv
```

Uninstall a package called `diagnostic/wireshark`

```
# pkg uninstall wireshark
```

List all packages installed on a system:

```
# pkg list
```

Get more information about an installed package called `diagnostic/wireshark`

```
# pkg info wireshark
```

List the contents of an installed package called `diagnostic/wireshark`

```
# pkg contents wireshark
```

Search all packages in the configured repositories for a file called `math.h`

```
# pkg search math.h
```

Search for all packages installed on a system that have a dependency on `library/libxml2`:

```
# pkg search -l -o pkg.name 'depend::library/libxml2'
```

List currently associated package publishers

```
# pkg publisher
```

Connect to the Oracle support repository and update the system

```
# pkg set-publisher -g https://pkg.oracle.com/solaris/support \
 -G http://pkg.oracle.com/solaris/release -k /path/to/ssl_key \
 -c /path/to/ssl_cert solaris
# pkg update
```

Where can i get the nescessary certificates and keys to access the support repository to get fixes?

You can get them at https://pkg-register.oracle.com given you have a valid support contract.

Local Package Repository

Setting up your own package repository

Download all the repository files from http://www.oracle.com/technetwork/server-storage/solaris11/downloads/local-repository-2245081.html into `/export/home/<username>`.

```
# zfs create rpool/export/repo
# zfs create rpool/export/repo/solaris
# cd /export/home/<username>
# ./install-repo.ksh  -d /export/repo/solaris/
# zfs snapshot rpool/export/repo/solaris@initial
```

Setting up a depot server to enable other systems to update themself via HTTP

```
# svccfg -s application/pkg/server setprop
pkg/inst_root=/export/repo/solaris/
# svccfg -s application/pkg/server setprop pkg/readonly=true
# svccfg -s application/pkg/server setprop pkg/port=8081
# svcadm refresh application/pkg/server
# svcadm enable application/pkg/server
```

Updating the repository with a SRU

Get the most current SRU for your Oracle Solaris release. For 11.2 a list is available with Doc ID 1672221.1 . For Solaris 11.3 you find the list in MOS Note 2045311.1 .

For my example i've downloaded all three files in the "IPS repository" column for SRU9 (reachable via link to 20845979 and 20845983) and loaded into `/export/home/<username>/sru9` of my repository server.

```
# cd /export/home/<username>/sru9
# unzip p20845983_1100_SOLARIS64.zip
# ./install-repo.ksh -d /export/repo/solaris
# svcadm restart application/pkg/server:default
# zfs snapshot rpool/export/repo/solaris@sru9
```

File Systems - Basic ZFS Administration

Oracle Solaris ZFS is the default root file system on Oracle Solaris 11. ZFS has integrated volume management, preserves the highest levels of data integrity and includes a wide variety of data services such as data compression, RAID, and data encryption.

Create a ZFS pool with a single disk

```
# zpool create testpool c3t2d0
```

Create a ZFS pool with 3 disks in RAID0 configuration

```
# zpool create testpool c3t2d0 c3t3d0 c3t4d0
```

Create a ZFS pool with 3 disks in RAID1 configuration

```
# zpool create testpool mirror c3t2d0 c3t3d0 c3t4d0
```

Create a ZFS pool with 3 disks in a RAIDZ configuration (single parity)

```
# zpool create testpool raidz c2t2d0 c3t3d0 c3t4d0
```

Create a ZFS pool with 1 disk and 1 disk as seperate ZIL (ZFS Intent Log)

```
# zpool create testpool c3t2d0 log c3t3d0
```

Create a ZFS pool with 1 disk and 1 disk as L2ARC (Level 2 Storage Cache)

```
# zpool create testpool c3t2d0 cache c3t3d0
```

Create a ZFS file system and share it via NFS:

```
# zfs create rpool/export/nfstest
# zfs set share.nfs=on rpool/export/nfstest
```

Share a files system via CIFS

```
# svcadm enable -r smb/server
# echo "other password required pam_smb_passwd.so.1 nowarn" \
 >> /etc/pam.d/other
# passwd <username>
# zfs create -o nbmand=on rpool/export/smbservertest
# zfs share -o share.smb=on rpool/export/smbservertest%smb_st
# mkdir /export/smbservertest/archiv
# chown junior2 /export/smbservertest/archiv
```

Now you can access the share via `smb://junior2:password@192.168.1.200/smb_st` . You have to set a new password for each user that wants to access a SMB share **after** adding the PAM module

Use shadow migration

```
# pkg install shadow-migration
# svcadm enable shadowd
# zfs set readonly=on rpool/export/shadowmigtest
# zfs create \
 -o shadow=file:///export/shadowmigtest rpool/export/shadowmigtestnew
```

Create an encrypted zfs dataset

```
# zfs create -o encryption=on rpool/export/secretproject
```

Change the wrapping key

```
# zfs key -c rpool/export/secretproject
```

Change the encryption key of a dataset

Please keep in mind that you can't set the encryption key to a user-defined value, you just can initiate the generation of a new encryption key. The wrapping key encrypting the encrytion key can be set to a user definable value.

```
# zfs key -K rpool/export/secretproject
```

How to limit the ZFS ARC cache (up to Oracle Solaris 11.1)

```
# echo "set zfs:zfs_arc_max=0x40000000" >> /etc/system
# reboot
```

How to limit the ZFS ARC cache (since Oracle Solaris 11.2)

```
# echo "set user_reserve_hint_pct=80" > /etc/system.d/arclimit
# reboot
```

The `user_reserve_hint_pct` parameter works differently than the old `zfs_arc_max` parameter. This parameter doesn't set a hard limit for the arc cache.This parameter defines how much memory is reserved for application use. It therefore limits how much memory can be used by the ZFS ARC cache.

This value is dynamic. So you can change this parameter while the system is running. However it's a best practice to change it to it in small steps to it's intended final value. There is a script doing it this way available with the MOS note 1663862.1.

You can get the current value of the parameter by using:

```
# echo "user_reserve_hint_pct/D" | mdb -k
```

Manually you can do it by using this command line where percentage is a the current value or the last value you've set plus 1. Wait for 30 seconds. Repeat this until your target value is reached:

```
# echo "user_reserve_hint_pct/W0t(percentage)" | mdb –kw
```

Disk Devices

Show all disks on a system

```
# cfgadm -s "select=type(disk)"
```

Replace a faulty disk c1t1d0 from ZFS pool testpool

```
# zpool offline testpool c1t3d0
# cfgadm | grep c1t3d0
sata0/3::dsk/c1t3d0          disk         connected     configured    ok
# cfgadm -c unconfigure sata0/3
# echo "Now replace the failed disk"
# cfgadm -c configure sata0/3
# zpool replace tank c1t3d0
# zpool online tank c1t3d0
```

Replace a faulty disk of a ZFS rool pool (SPARC or x86/VTOC)

You have to ensure that your new disk contain a fdisk partition (on x86), a SMI label and a slice 0.

```
# zpool offline rpool c1t0d0s0
# cfgadm -c unconfigure c1::dsk/c1t0d0
# echo "now replace the disk"
# cfgadm -c configure c1::dsk/c1t0d0
# zpool replace rpool c1t0d0s0
# zpool online rpool c1t0d0s0
# zpool status rpool
# bootadm install-bootloader
```

Replace a faulty disk of a ZFS root pool (SPARC or x86/EFI (GPT))

```
# zpool offline rpool c1t0d0
# cfgadm -c unconfigure c1::dsk/c1t0d0
# echo "now replace the disk"
# cfgadm -c configure c1::dsk/c1t0d0
# zpool online rpool c1t0d0
# zpool replace rpool c1t0d0
# zpool status rpool
# bootadm install-bootloader
```

Checking the logical blocksize of a disk device

```
# devprop -vn /dev/dsk/c3t0d0 device-blksize
```

Checking the physical blocksize of a disk device

```
# devprop -vn /dev/dsk/c3t0d0 device-pblksize
```

iSCSI

Configure an iSCSI target

```
# pkg install group/feature/storage-server
# zfs create rpool/export/iscsiluns
# zfs create -V 16g rpool/export/iscsiluns/lun1
# svcadm enable stmf
# stmfadm create-lu /dev/zvol/rdsk/rpool/export/iscsiluns/lun1
Logical unit created: 600144F0BE1002000000553776B00001
# stmfadm add-view 600144F0BE1002000000553776B00001
# svcadm enable -r svc:/network/iscsi/target
# itadm create-target
Target iqn.1986-03.com.sun:02:e8e0aa2d-1011-4136-9c9a-ddebb6279801
successfully created
```

Use the iSCSI target just configured

```
# svcadm enable svc:/network/iscsi/initiator
# iscsiadm add discovery-address 192.168.1.200:3260
# iscsiadm modify discovery --sendtargets enable
# devfsadm -c iscsi
# iscsiadm list initiator-node
Initiator node name: iqn.1986-03.com.sun:01:e00000000000.55365ebd
[..]
```

Add bidirectional authentication between iSCSI target and initiator

From the last two examples, we know that the IQN of the target is `iqn.1986-03.com.sun:02:e8e0aa2d-1011-4136-9c9a-ddebb6279801` and from the initiator is `iqn.1986-03.com.sun:01:e00000000000.55365ebd`. The secret that authorizes the target to the initiator is `foobarfoobar`, the secret that authorizes the initiator to the target is `snafusnafusna`

```
target# itadm modify-target -s iqn.1986-03.com.sun:02:e8e0aa2d-1011-4136-
9c9a-ddebb6279801
Enter CHAP secret: snafusnafusna
Re-enter secret: snafusnafusna
target# itadm create-initiator -s iqn.1986-
03.com.sun:01:e00000000000.55365ebd
Enter CHAP secret: foobarfoobar
Re-enter secret: foobarfoobar
initiator# iscsiadm modify initiator-node --CHAP-secret
Enter secret: foobarfoobar
Re-enter secret: foobarfoobar
initiator# iscsiadm modify initiator-node --authentication CHAP
# iscsiadm modify target-param --authentication CHAP iqn.1986-
03.com.sun:02:e8e0aa2d-1011-4136-9c9a-ddebb6279801
# iscsiadm modify target-param --CHAP-secret iqn.1986-
03.com.sun:02:e8e0aa2d-1011-4136-9c9a-ddebb6279801
Enter secret: snafusnafusna
Re-enter secret: snafusnafusna
```

NFS

Mount a share via NFS

```
# mount 10.0.2.10:/export/nfsshare /mnt
```

Share a ZFS filesystem via NFS

```
# zfs set share.nfs=on rpool/export/nfstest
```

View currently active NFS mounts on a client with mount options

```
# nfsstat -m
```

Clear locks for a NFS client on a NFS server

```
# clear_lock <hostname>
```

Only use this command in the real rare case a NFS client crashed and failed to clear the locks on the server.

Enable multiple TCP connections for the NFS client to a server

This parameters controls the number of TCP connections that the NFS client uses when communicating with each NFS server. The default is 1 as the implementation is able to multplex the RPCs over a single connection. However, multiple connections can be used, if preferred for example to use more than one link in a link aggregation or to improve performance by having more than just one TCP connection.

```
echo "set rpcmod:clnt_max_conns=8“ >> /etc/system # up to 11.1
echo "set rpcmod:clnt_max_conns=8“ > /etc/system.d/nfstuning # since 11.2
```

Storage URI

In order to identify storage resources uniquely between nodes the concept of Storage URIs was introduced in Oracle Solaris 11. For example they are used for Zones on Shared Storage to identify the shared storage objects.

Looking up a Storage URI for a device

```
# suriadm lookup-uri /dev/dsk/c0t600144F00833C0000000573865760001d0
```

Looking up the mapping of an device to a Storage URI

```
# suriadm lookup-mapping \
iscsi://10.0.2.10/luname.naa.600144f00833c0000000573865760001
```

Basics of Oracle Solaris Zones

Oracle Solaris Zones provide isolated and secure virtual environments running on a single operating system instance, ideal for application deployment. When administrators create a zone, an application execution environment is produced in which processes are isolated from the rest of the system.

Create a zone

```
# zonecfg -z testzone
 testzone: No such zone configured
 Use 'create' to begin configuring a new zone.
 zonecfg:testzone> create
 zonecfg:testzone> set zonepath=/export/zones/testzone
 zonecfg:testzone> set autoboot=true
 zonecfg:testzone> verify
 zonecfg:testzone> commit
 zonecfg:testzone> exit
 root@test1:~# zoneadm -z testzone install
```

Create a zone on shared storage

```
# zonecfg -z zoss-zone
Use 'create' to begin configuring a new zone.
zonecfg:zoss-zone> create
create: Using system default template 'SYSdefault'
zonecfg:zoss-zone> set zonepath=/zones/zoss-zone
zonecfg:zoss-zone> add rootzpool
zonecfg:zoss-zone:rootzpool> add storage
iscsi://192.168.1.200/luname.naa.600144F0BE10020000000553776B00001
zonecfg:zoss-zone:rootzpool> end
zonecfg:zoss-zone> commit
zonecfg:zoss-zone> exit
# zoneadm -z zoss-zone install
```

Create a kernel zone

```
# echo "set zfs:zfs_arc_max=0x40000000" >> /etc/system  #up to 11.1
# echo "set user_reserve_hint_pct=80" > /etc/system.d/arclimit #since 11.2
# reboot
# zonecfg -z kernelz1 create -t SYSsolaris-kz
# zoneadm -z kernelz1 install
```

List all running zones verbosely

```
# zoneadm list -v
```

List all configured zones:

```
# zoneadm list -c
```

List all installed zones

```
# zoneadm list -i
```

Install a zone

```
# zoneadm -z testzone install
```

List configuration of a zone

```
# zonecfg -z testzone info
```

Login to the console of a zone

```
# zlogin -C testzone
```

Halt a zone

```
# zoneadm -z testzone halt
```

Shutdown a zone

```
# zoneadm -z testzone shutdown
```

Monitor a zone for CPU, memory and network utilization every 10 seconds:

```
# zonestat -z testzone 10
```

How can i have a different time in a non-global zone? (Solaris 11.3)

```
# zonecfg -z myzone
zonecfg:myzone> set limitpriv=default,sys_time
zonecfg:myzone> set global-time=false
zonecfg:myzone> exit
```

Please note: This behavior is now the default in Solaris 11.3.

Which parameters of a zone are enabled for live zone reconfiguration?

A number of parameters of the zone can be changed while the zone is running without requiring a reboot.

The parameters enabled for life reconfiguration for non-global zones are:

- `anet, except`
  - `anet:allowed-address`
  - `anet:configure-allowed-address`
  - `anet:defrouter`

- `fs`
- `capped-memory`
- `dedicated-cpu`
- `device`
- `net, except`
  - `net:allowed-address`
  - `net:configure-allowed-address`
  - `net:defrouter`
- `pool`
- `scheduling-class`
- `zone.* resource controls`

The parameters enabled for life reconfiguration for kernel zones are:

- `anet, except`
  - `anet:allowed-address`
  - `anet:configure-allowed-address`
  - `anet:defrouter`
- `device`
- `net, except`
  - `net:allowed-address`
  - `net:configure-allowed-address`
  - `net:defrouter`

How to make persistant live reconfiguration?

```
# zonecfg -z tbz1 "set cpu-shares=4"
# zoneadm -z tbz1 apply
```

How to make temporary live reconfiguration?

```
# zonecfg -z tbz1 -r  "set cpu-shares=8"
```

How to revert to the persistent configuration of the zone after a temporary live reconfigurations?

```
# zoneadm -z tbz1 apply
```

How to check the currently temporary configuration as configured by a live reconfiguration?

```
# zonecfg -z tbz1 -r info
```

How to start a live migration of a kernel zone?

```
target# svcadm enable -s svc:/system/rad:remote
target# svcadm enable –s svc:/network/kz-migr:stream
source# zoneadm -z kzone1 migrate target
```

Technically you just have to enable the services on the target. However quite often it's a good practice to enable it on both sides, given that you may want to migrate the service back tot he source server.

Immutable Oracle Solaris Zones

Making a non-global Zone immutable

```
# zonecfg -z zone1
zonecfg:zone1> set file-mac-profile=strict
```

Making a global zone immutable

```
# zonecfg -z global
zonecfg:global> set file-mac-profile=flexible-configuration
```

The following file-mac-profiles values are availble to configure a immutable zone:

Profile Name	Description
`none`	Standard, read-write zone. Is the default.
`strict`	Read-only file system, no exceptions. • IPS packages cannot be installed. • Persistently enabled SMF services are fixed. • SMF manifests cannot be added from the default locations. • Logging and auditing configuration files are fixed. Data can only be logged remotely.
`fixed-configuration`	Permits updates to /var/* directories, with the exception of directories that contain system configuration components. • IPS packages, including new packages, cannot be installed. • Persistently enabled SMF services are fixed. • SMF manifests cannot be added from the default locations. • Logging and auditing configuration files can be local. syslog and audit configuration are fixed.
`dynamic-zones`	This profile is eqal to `fixed-configuration`, but allows the creation and destruction of zones
`flexible-configuration`	Permits modification of files in /etc/* directories, changes to root's home directory, and updates to /var/* directories. Closest configuration to a Solaris 10 sparse zone. IPS packages, including new packages, cannot be installed. Persistently enabled SMF services are fixed. SMF manifests cannot be added from the default locations. Logging and auditing configuration files can be local. syslog and audit configuration can be

Log into the immutable zone to make changes via the Trusted Path

```
# zlogin -T <zonename>
```

Remove restriction rpool/dataset in an otherwise immutable global zone

```
zonecfg:global> add dataset
zonecfg:global:dataset> set name=rpool/dataset
zonecfg:global:dataset> end
```

Basic networking

Show physical network interfaces

```
# dladm show-phys
```

Show state information of physical ethernet ports

```
# dladm show-ether
```

Show datalinks

```
# dladm show-link
```

Show properties of the datalink net0

```
# dladm show-linkprop net0
```

Show IP interfaces

```
# ipadm show-if
```

Show properties of a IP interface

```
# ipadm show-ifprop net0
```

Show IP address objects

```
# ipadm show-addr
```

Show properties of a IP address objects

```
# ipadm show-addrprop
```

Putting an IP address down and up again

```
# ipadm down-addr net0/v4
# ipadm up-addr net0/v4
```

Create interface with static IPv4 configuration:

```
# ipadm create-ip net0
# ipadm create-addr -a 10.1.1.10/24 net0/addr1
```

Add an IP address to an existing IP interface:

```
# ipadm create-addr -a 10.1.2.10/24 net0/addr2
```

Create interface with DHCP configuration:

```
# ipadm create-ip net0
# ipadm create-addr -T dhcp  net0/addr1
```

Create interface with auto-generated IPv6 configuration:

```
# ipadm create-ip net0
# ipadm create-addr -T addrconf  net0/addrv6
```

Create a virtual network interface over existing physical interface net0 with address 192.168.0.80

```
# dladm create-vnic -l net0 vnic0
# ipadm create-ip vnic0
# ipadm create-addr -a 192.168.0.80 vnic0/v4
```

Set the default route

```
# route -p add default 192.168.1.1
```

Create two virtual network interfaces over a virtual switch (without a physical network interface)

```
# dladm create-etherstub stub0
# dladm create-vnic -l stub0 vnic0
# dladm create-vnic -l stub0 vnic1
```

Reduce the bandwidth of the virtual network interface vnic0 to 100Mbps

```
# dladm set-linkprop -p maxbw=100 vnic0
```

Restrict the bandwidth going to IP address 192.168.0.30 by creating a flow on virtual network interface vnic0, then restrict its bandwidth to 50Mbps:

```
# flowadm add-flow -l vnic0 -a remote_ip=192.168.0.30 flow0
# flowadm set-flowprop -p maxbw=50 flow0
```

Restrict network traffic to TCP for local port 443 for network interface net0 to 75 Mbps

```
# flowadm add-flow -l net0 -a transport=TCP,local_port=433 flow1
# flowadm set-flowprop -p maxbw=75 flow1
```

Activating Jumbo Frames (ethernet packets greater than 1500 bytes)

```
# dladm set-linkprop -p mtu=9000 net0
```

Configure Link Aggregation:

```
# dladm create-aggr -l net0 -l net1 aggr0
# ipadm create-ip aggr0
# ipadm create-addr -T static -a 10.1.1.2/24 aggr0/v4
```

Configure VLANS:

```
# dladm create-vlan -l net0 -v 100 administration1
# dladm create-vlan -l net0 -v 2 production1
# ipadm create-ip administration1
# ipadm create-ip production1
# ipadm create-addr -T static -a 192.168.2.2/24 administration1/v4static
# ipadm create-addr -T static -a 192.168.1.2/24 production1/v4static
```

Configure an IPMP group:

```
# ipadm create-ip net0
# ipadm create-ip net1
# ipadm create-ip net2
# ipadm create-ipmp ipmp0
# ipadm add-ipmp -i net0 -i net1 -i net2 ipmp0
# ipadm create-addr -T static -a 192.168.1.27/24 ipmp0/v4
# ipadm create-addr -T static -a 192.168.1.50/24 net0/test
# ipadm create-addr -T static -a 192.168.1.51/24 net1/test
# ipadm create-addr -T static -a 192.168.1.52/24 net2/test
```

Creating a LACP Trunk Aggregation

```
# dladm create-aggr -L active -l net0 -l net1 aggr1
```

Creating a Data Link Multi Pathing Aggregation

```
# dladm create-aggr -m dmlp -l net0 -l net1 aggr1
```

How to configure Probing for DLMP aggregates with a failure detection time of 15 sec

```
# dladm set-linkprop -p probe-ip=+ aggr1
# dladm set-linkprop -p probe-fdt=15 aggr1
```

Enable IPv4 forwarding between two interfaces

```
# routeadm -e ipv4-forwarding
# routeadm -u
```

Disable IPv4 forwarding between two interfaces

```
# routeadm -d ipv4-forwarding
# routeadm -u
```

How to configure an Virtual eXtensible LAN with the VNI 100 between two systems using the 10.254.1.0/24 network

```
node1# dladm create-vxlan -p vni=100,interface=net0 vxlan1
node1# dladm create-vnic -l vxlan1 vnic1
node1# ipadm create-ip vnic1
node1# ipadm create-addr -a 10.254.1.1/24 vnic1/vxlan1
node2# dladm create-vxlan -p vni=100,interface=net0 vxlan1
node2# dladm create-vnic -l vxlan1 vnic1
node2# ipadm create-ip vnic1
node2# ipadm create-addr -a 10.254.1.2/24 vnic1/vxlan1
```

Advanced Networking - highly available loadbalancer

In this example the Virtual Router Redundancy Protocol and the Integrated Loadbalancer features of Oracle Solaris are used to create an highly available loadbalancer. This longer example thus shows how to configure VRRP as well as the ILB feature, which could both used without the other.

Shorthand	System	Use	IP
ilb1	Loadbalancer 1	Outside interface	10.0.1.10/24
ilb2	Loadbalancer 2	Outside interface	10.0.1.20/24
ilb1/2	Loadbalancer	Virtual IP	10.0.1.100/24
ilb1	Loadbalancer 1	Inside interface	10.0.10.10/24
ilb2	Loadbalancer 2	Inside Interface	10.0.10.20/24
ilb1/2	Loadbalancer	Virtual Default Gateway	10.0.10.100/24
rs1	Real Server 1	Single Interface	10.0.10.200/24
rs2	Real Server 2	Single Interface	10.0.10.210/24

Preparing Webserver 1

```
ws1# ipadm create-ip net0
ws1# ipadm create-addr -T static -a 10.0.10.200 net0/v4
ws1# route -p add default 10.0.10.100
ws1# svcadm enable apache22
```

Preparing Webserver 2

```
ws2# ipadm create-ip net0
ws2# ipadm create-addr -T static -a 10.0.10.210 net0/v4
ws2# route -p add default 10.0.10.100
ws2# svcadm enable apache22
```

Installing prerequisites on both loadbalancers

```
ilb1/ilb2# pkg install vrrp
ilb1/ilb2# pkg install ilb
```

Configuring VRRP on the first loadbalancer

```
ilb1# dladm create-aggr -m dlmp -l net0 -l net2 outside0
ilb1# dladm create-aggr -m dlmp -l net1 -l net3 inside1
ilb1# ipadm create-ip outside0
ilb1# ipadm create-ip inside1
ilb1# ipadm create-addr -T static -a 10.0.1.10/24 outside0/v4
ilb1# ipadm create-addr -T static -a 10.0.10.10/24 inside1/v4
ilb1# dladm create-vnic -m vrrp -V 2 -A inet -l inside1 vnic2
ilb1# dladm create-vnic -m vrrp -V 1 -A inet -l outside0 vnic1
ilb1# ipadm create-ip vnic1
ilb1# ipadm create-addr -T static -d -a 10.0.1.100/24 vnic1/lb1
ilb1# vrrpadm create-router -V 1 -A inet -l outside0 -p 255 vrrp1
ilb1# ipadm create-ip vnic2
ilb1# ipadm create-addr -T static -d -a 10.0.10.100/24 vnic2/lb1
ilb1# vrrpadm create-router -V 2 -A inet -l inside1 -p 255 vrrp2
```

Configuring VRRP on the second loadbalancer

```
ilb2# dladm create-aggr -m dlmp -l net0 -l net2 outside0
ilb2# dladm create-aggr -m dlmp -l net1 -l net3 inside1
ilb2# ipadm create-ip outside0
ilb2# ipadm create-ip inside1
ilb2# ipadm create-addr -T static -a 10.0.1.20/24 outside0/v4
ilb2# ipadm create-addr -T static -a 10.0.10.20/24 inside1/v4
ilb2# dladm create-vnic -m vrrp -V 2 -A inet -l inside1 vnic2
ilb2# dladm create-vnic -m vrrp -V 1 -A inet -l outside0 vnic1
ilb2# ipadm create-ip vnic1
ilb2# ipadm create-addr -T static -d -a 10.0.1.100/24 vnic1/lb1
ilb2# vrrpadm create-router -V 1 -A inet -l outside0 -p 100 vrrp1
ilb2# ipadm create-ip vnic2
ilb2# ipadm create-addr -T static -d -a 10.0.10.100/24 vnic2/lb1
ilb2# vrrpadm create-router -V 2 -A inet -l inside1 -p 100 vrrp2
```

Configuring ILB on the first loadbalancer

```
ilb1# routeadm -u -e ipv4-forwarding
ilb1# svcadm enable ilb
ilb1# ilbadm create-servergroup -s server=10.0.10.200,10.0.10.210
servergroup1
ilb1# ilbadm create-rule -ep -i vip=10.0.1.100,port=80,protocol=tcp -m
lbalg=roundrobin,type=HALF-NAT,pmask=32 -o servergroup=servergroup1 rule1
```

Configuring ILB on the second loadbalancer

```
ilb2# routeadm -u -e ipv4-forwarding
ilb2# svcadm enable ilb
ilb2# ilbadm create-servergroup -s server=10.0.10.200,10.0.10.210
servergroup1
ilb2# ilbadm create-rule -ep -i vip=10.0.1.100,port=80,protocol=tcp -m
lbalg=roundrobin,type=HALF-NAT,pmask=32 -o servergroup=servergroup1 rule1
```

Compliance

List all compliance benchmarks available on the system

```
# compliance list -b -v
```

List all compliance benchmarks available on the system and their profiles

```
# compliance list -b -p -v
```

Run a compliance assessment with the PCI-DSS benchmark

```
# compliance  assess -b pci-dss
```

Run a compliance assessment with the "Oracle Solaris Security Policy" benchmark in the "Recommended" profile:

```
# compliance  assess -b solaris -p Recommended
```

Show all assessment results available on the system

```
# compliance list -a
```

Create a compliance report from an assessment

```
# compliance report -a pci-dss.Solaris_PCI-DSS.2015-04-03,11:01
```

How to taylor a compliance assessment to your needs?

```
root@solaris:~# compliance tailor -t c0t0d0s0basic
*** compliance tailor: Can't load tailoring 'c0t0d0s0basic': no existing
tailoring: 'c0t0d0s0basic', initializing
tailoring:c0t0d0s0basic> set benchmark=solaris
# Exclude all reports
tailoring:c0t0d0s0basic> exclude –a
# Either use interactive mode by using the pick command, to reenable some
checks
tailoring:c0t0d0s0basic> pick
# or include them one-by-one
tailoring:c0t0d0s0basic> include OSC-53005
tailoring:c0t0d0s0basic> include OSC-16005
tailoring:c0t0d0s0basic> include OSC-35000
tailoring:c0t0d0s0basic> include OSC-46014
tailoring:c0t0d0s0basic> include OSC-01511
tailoring:c0t0d0s0basic> include OSC-04511
tailoring:c0t0d0s0basic> include OSC-75511
# With both ways you have to commit the changes now
tailoring:c0t0d0s0basic> commit
tailoring:c0t0d0s0basic> exit
```

How to use the tailored compliance assessment?

```
# compliance assess -t c0t0d0s0basic
```

Security

Switch from SunSSH to OpenSSH and back

Since Solaris 11.3 it's possible to use OpenSSH instead of SunSSH.

```
# pkg install openssh
# pkg mediator -a ssh
MEDIATOR     VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
ssh          vendor            vendor     sunssh
ssh          system            system     openssh
# pkg set-mediator –I openssh ssh
and back
# pkg set-mediator –I openssh ssh
```

Per-file authorized edit of administrative files:

```
as root
# profiles -p "httpd.conf configure"
profiles: httpd configure> set
auths=solaris.admin.edit/etc/apache2/2.2/httpd.conf
profiles: httpd.conf configure> set desc="Edit http configuration"
profiles: httpd.conf configure> exit
# usermod –P +"httpd.conf configure" <username>
as normal user <username>:
# pfedit /etc/httpd.conf
```

Enabling logging of changes via pfedit in the audit log

```
profiles -p "httpd.conf configure"
profiles:httpd.conf configure> add always_audit=as
profiles:httpd.conf configure> exit
root@template:~#
```

Viewing the audit trail of the pfedit invocations

```
# auditreduce -c as | praudit
```

Check in which packages a given CVE-ID has been fixed

```
# pkg search :CVE-2015-0397:
```

Check what CVE-ID has been fixed in a Critical Patch Update (2015.4 in this example)

```
# pkg search -r info.cve: | grep "2015.4"|tr -s " " | cut -d " " -f 3
```

Check if a fix for a given CVE-ID has been installed

```
# pkg search -l CVE-2015-0397
```

How tomake a port above 1023 a privileged port?

```
# ipadm set-prop -p extra_priv_ports+=10025 tcp
```

How to lock down a service with Oracle Solaris Extended Policies?

Lets assume that you have a service listening to port 10025, you just made this port a privileged on to prevent a normal user to start a fake service on 10025. die FRMI of the service is `svc:/application/crcaserv`. It's started via `/lib/svc/method/creditcardservice`. You know that it just writes in `/var/CrCaServ/data` and `/var/CrCaServ/tmp`. It runs as user ccserv and group ccserv.

```
# ipadm set-prop -p extra_priv_ports+=10025 tcp
# profiles -p "CrCaServ Profile"
CrCaServ Profile> set desc="Jailing in creditcardservice"
CrCaServ Profile> add cmd=/lib/svc/method/creditcardservice
CrCaServ Profile:creditcardservice> set privs=basic
CrCaServ Profile:creditcardservice> add privs={net_privaddr}:10025/tcp
CrCaServ Profile:creditcardservice> add
privs={file_write}:/var/CrCaServ/data/*
CrCaServ Profile:creditcardservice> add
privs={file_write}:/var/CrCaServy/tmp/*
CrCaServ Profile:creditcardservice> end
CrCaServ Profile> set uid=ccserv
CrCaServ Profile> set gid=mysql
# svccfg -s svc:/application/crcaserv:default
svc:/application/crcaserv:default> setprop
method_context/profile="CrCaServ Profile"
svc:/application/crcaserv:default> setprop method_context/use_profile=true
svc:/application/crcaserv:default> refresh
svc:/application/crcaserv:default> exit
```

How to do packet filtering in Solaris 11?

Solaris has a packet filtering functionality for several versions now. For Solaris 10 and 11 you can use the Ipfilter (IPF) mechanism. In Solaris 11.3 however a new mechanism was introduced. It's based on the OpenBSD 5.5 Packet Filter (PF). In Solaris 11.3 you can use both mechanisms. As indicated by the „End-of-feature" list, IPF will not be available in future Solaris versions. Please keep in mind that IPF and PF are mutually exclusive. IPF will not start with PF enabled and vice versa. So disable one before you enable th other.

Differences between IPF and PF are documented at
https://docs.oracle.com/cd/E53394_01/html/E54829/pfovw-intr.html

How to enable packet filtering with ipfilter (IPF) ?

```
# svcadm enable network/ipfilter
```

How to configure IPF?

```
# svccfg -s ipfilter:default setprop firewall_config_default/policy =
astring: "custom"
# svccfg -s ipfilter:default setprop
firewall_config_default/custom_policy_file = astring:
"/etc/ipf/c0t0d0s0.ipf.conf"
# cat < EOT >> /etc/ipf/c0t0d0s0.ipf.conf
block in log all head 100
block out log all head 101
pass in quick on lo0
pass out quick on lo0
pass in quick on net0 proto tcp from any to 192.168.1.202 port = 22 keep
state group 100
pass in quick on net0 proto tcp from any to 192.168.1.202 port = 80 keep
state group 100
pass out quick proto tcp all flags S/SA keep state group 101
pass out quick proto udp all keep state group 101
pass out quick proto icmp all keep state group 101
EOT
# svcadm refresh ipfilter
```

How to disable packet filtering with IPF?

```
# svcadm disable network/ipfilter
```

How to enable packet filtering with PF in Solaris 11.3?

```
# pkg install network/firewall
# svcadm enable network/firewall #Do not use pfctl -e
```

How to configure Packet Filter (PF)?

```
# pfconf
# svcadm refresh svc:/network/firewall:default
```

How to disable PF?

```
# svcadm disable network/firewall #Do not use pfctl -d
```

How to view the current ruleset of PF loaded into the kernel?

```
# pfctl -s rules
```

How to enable IPsec?

This example assumes that server1 is 192.168.1.200 and server2 is 192.168.1.202. We will use IKEv2 in the shared-secret mode.

On server1 configure:

```
server1# cat << EOT > /etc/inet/ike/ikev2.config
ikesa_lifetime_secs 3600
ikesa_xform { encr_alg aes(256..256) auth_alg sha384 dh_group 20 }
ikesa_xform { encr_alg aes(128..128) auth_alg sha256 dh_group 19 }
{ label "server1-server2"
  auth_method preshared
  local_addr  192.168.1.200
  remote_addr 192.168.1.202
}
EOT
server1# /usr/lib/inet/in.ikev2d -c # to check your file is correct
server1# cat << EOT > /etc/inet/ike/ikev2.preshared
{ label "server1-server2"
   key "an obviously rather weak password. Choose wiser"
}
EOT
server1# svcadm enable ipsec/ike:ikev2
server1# /usr/sbin/ipsecconf -c /etc/inet/ipsecinit.conf
server1# svcadm refresh ipsec/policy:default
```

On server2 configure:

```
server2# echo "{laddr server2 raddr server1} ipsec {encr_algs aes
encr_auth_algs sha512 sa shared}" > /etc/inet/ipsecinit.conf
server2# cat << EOT > /etc/inet/ike/ikev2.config
ikesa_lifetime_secs 3600
ikesa_xform { encr_alg aes(256..256) auth_alg sha384 dh_group 20 }
ikesa_xform { encr_alg aes(128..128) auth_alg sha256 dh_group 19 }
{ label "server2-server1"
  auth_method preshared
  local_addr  192.168.1.202
  remote_addr 192.168.1.200
}
EOT
server2# /usr/lib/inet/in.ikev2d -c
server2# cat << EOT > /etc/inet/ike/ikev2.preshared
{ label "server2-server1"
   key "an obviously rather weak password. Choose wiser"
}
EOT
server2# svcadm enable ipsec/ike:ikev2
server2# /usr/sbin/ipsecconf -c /etc/inet/ipsecinit.conf
server2# svcadm refresh ipsec/policy:default
```

Managing Oracle Solaris 11 Security Extensions

Since Oracle Solaris 11 there is a command to manage security extensions in Solaris. The first to appear was the adress space layout randomization and it was the only one. Since Oracle Solaris 11.3 additional ones were introduced to Oracle Solaris.

Extension	Description
`aslr`	*Address Space Layout Randomization.* If enabled, the layout of the memory in the process is randomized. Thus attacks based on assumptions on the location of code in the address space can't work any longer because with each start the layout is different. However there are valid, non-malicious computer programs that rely on the fact, that the layout is always the same. Thus the default is that ASLR is only activated for files tagged accordingly. There is no logging available for this security extension as there is no event of violation.
`nxstack`	*Non-executable Stack.* If enabled, the stack of the process is labeled as non-executable. So an attacker can't try to inject code onto the stack in order to execute it there. As almost no non-malicious code needs this capability, the default is to have an non-executable stack for all programs.
`nxheap`	*Non-Executable Heap.* If enabled, the heap of the process is labeled as non-executable. So no code on heap can be executed by the system and an attacker couldn't use it as a attack vector. However there are classes of programs, that need this capability. For example interpreters execute code on the heap. Thus the default is that nxheap is only activated for files tagged accordingly.

The model defines which binaries will use a security extension. There are currently 3 possible selections:

Model	Description
`default`	The default as defined above.
`tagged-files`	*Only files that contain a certain tag. This tag is in the ELF-header of the binary.*
`all`	*As the name suggests, all binaries are subject to the mechanisms of the security extension.*

Some security extensions have the capability to log if a process tries something the security extension prevents. For example if it's tried to execute something on the stack, the nxstack security extension prevents this and logs it.

Check the current state of configuration of the security extensions

```
# sxadm get all
```

Instruct the system to use ASLR for all binaries.

```
# sxadm set model=all aslr
```

```
Revert setting for the ASLR security extension to the default behaviour
```

```
# sxadm set model=default aslr
```

Disable ASLR

```
# sxadm disable aslr
```

Enable ASLR

```
# sxadm enable aslr
```

Disable and enable logging for security extensions.

```
# sxadm set log=disable nxstack
# sxadm set log=enable nxstack
```

Check for security extension tags

```
# elfdump -d /usr/sbin/ping|grep "NXSTACK" #for the tag activating NXSTACK
# elfdump -d /usr/sbin/ping|grep "NXHEAP" #for the tag activating NXHEAP
# elfdump -d /usr/sbin/ping|grep "ASLR" #for the tag activating ASLR
```

Did You Know?

Just because a binary is setuid root in Oracle Solaris 11, it doesn't mean that it is run as root. Oracle Solaris 11 has a feature called Forced Privileges. Most of the setuid root binaries of Oracle Solaris just add the nescessary privileges when executed to allow the proper run of the application without switching to user id root at all. For for information read Darren Moffats blog about it at https://blogs.oracle.com/darren/entry/when_setuid_root_no_longer.

Tasks and Projects

Workloads seldomly consists just out of a single process, thus a convient way to lavel all processes of workloads is really useful. With such a label you could address all processes of workload in one step instead of repeating this step for each process. Tasks and projects are such facilities to label workloads. The predominant uses of Task and projects are accounting and (probably more important) a way to group processes for resource control

Name	Description
`task`	Collects a group of processes into a set of of processes, to give you an entity that you can manage or monitor as a whole. A new task is started, when you Login • cron • newtask • setproject • su
`project`	Projects are network-wide identifiers of workloads that are assigned by administrators. A user and group can be part of one or more projects. A user can start tasks and thus processes in any project he or she is member.

Show all projects on your system

```
# projects -l
```

Create a project with the name „`testproj`“ and the project id `4711` and assign the user `jmoekamp` to it

```
# projadd -U jmoekamp -p 4711 testproj
```

Create a project `user.oracle` that is the default project for the user `oracle`.

```
# projadd user.oracle
```

Delete a project

```
# projdel testproj
```

Check the project you are currently running in

```
# id -p
```

Start a task in a different project

```
# newtask -p testproj
```

Assign process 5431 to a different project

```
# newtask -p testproj -c 5431
```

What are the default projects of a user

The default project of a user is determined and assigned in the following order. First fullfilled condition exits the mechanism.

- The user has an project attribute in `/etc/user_attr`. The value of this attributed is used as the default project
- If there is a project with the name `user.<username of the user>` it's the default project of the user
- If theres is a project with the name `group.<groupname of the user >` it's the default project of the user
- If there is a project `default`, it's simply used as the default project.

Assign a project to a user (as in the first condition above)

```
# usermod -K project=testproj jmoekamp
```

Check the task id and the project of a running process

```
# ps -ef -o pid,user,zone,project,taskid,args
```

Processors

Show processors in the system

```
# psrinfo -v
```

Free processors from interrupt processing

```
# psradm -i 1-7
```

Put processors offline

```
# psradm -f 1-7
```

Put processors online again and reactivate interrupt processing

```
# psradm -n 1-7
```

Looking up processor groups

```
# pginfo
```

Looking up locality groups

Solaris organizes processors in locality groups in order to be able to schedule processes on processors as close as possible to resources needed by the process.

```
# lgroupinfo
```

Processor sets and pools

Create two processorpools and assign two zones to it

```
# pooladm –e
# cat << EOT >> pools.configfile
create pool testzone1pool
create pset testzone1pset (uint pset.min = 2 ; uint pset.max = 2 )
associate pool testzone1pool (pset testzone1pset)
transfer to pset testzone1pset ( cpu 4 ; cpu 5 )
create pool testzone2pool
create pset testzone2pset (uint pset.min = 2 ; uint pset.max = 2 )
associate pool testzone2pool (pset testzone2pset)
transfer to pset testzone2pset (  cpu 6 ; cpu 7)
EOT
# poolcfg -d -f pools.configfile
# zonecfg -z testzone1
zonecfg:testzone1> set pool=testzone1pool
# zonecfg -z testzone2
zonecfg:testzone2> set pool=testzone2pool
```

Binding processes to a CPU or a group of CPU

A new feature in Oracle Solaris 11.2 is the capability to bind a process not just to one processor, but to multiple ones. Before that you could only bind a process to a single CPU. However keep in mind that Oracle Solaris is usually doing a pretty good job on putting processes on the best CPUs.

Binding	Description
`strong`	If you use the strong binding, a process will only run on the processors you configure in the command. You use the `–b –s` options for this. It's the default.
`weak`	The scheduler will try to schedule in process on the configured processors, however if this is not possible it will run them on different ones. You configure this behaviour via the `–b –w` options.
`negative`	The processor will not run on the mentioned processors. The nescessary options are `–b –n` . This can combine this with either a strong or a weak binding.

	Technically you will see a strong or weak binding on all other processors except the ones mentioned in the command.

Bind a process to a single CPU
```
# pbind -b -c 3 -i pid 1605
```
Bind a single thread of a process to a CPU
```
# pbind -b -s -c 1 -i pid 48/2
```
Bind a process to multiple CPUs
```
# pbind -b -c 3,2 -i pid 1605
```
Bind a process to a processor group
```
# pbind -b -s -g 1 -i pid 48
```
Bind a process to a locality group
```
# pbind -b -s -l 1 -i pid 48
```
Bind a process weakly to multiple CPUs
```
# pbind -b -w -c 3,2 -i pid 1605
```
Configure a negative strong binding of a process to a group of CPUs
```
# pbind -b -n -w -c 0 -i pid 160
```
Bind all processes run by the user with the UID 100
```
# pbind -b -n -w -c 0 -i uid 100
```
Create a project with multi-cpu binding for all processes running in this project and start a shell in it.
```
# pooladm -e
# projadd -K project.mcb.cpus=1-3 -K project.mcb.flags=strong -K
project.pool=pool_default boundedproject
# newtask –p boundedproject
```
Check binding inforation for a process
```
# pbind -q -i pid 1605
```

Scheduling

List the currently configured scheduling classes
```
# dispadmin -l
```
Check which scheduling classes are in use by currently running processes
```
# ps -ef -o pid,class,pri,args
```
Move a process into the realtime scheduling class
```
# priocntl -c RT -s 1349
```
Move a process into the scheduling class FX with a priority of 10 and a user priority limit of 20
```
# priocntl -c FX -m 10 -p 20 -s 1349
```

Resource Management

Using the Fair Share Scheduler without processes.

In this example I want to ensure that one process is getting 75\% of the compute power and another one is getting 25% in case CPU resources are a contended resource. The FSS scheduler ist based on the concept of shares: Let's assume i cut the total compute power to 200 shares, i have to assign 150 shares to the first process and 50 to the second.
```
# dispadmin -d FSS
# reboot
# projmod -K "project.cpu-shares=(privileged,150,none)" importantproject
# projmod -K "project.cpu-shares=(privileged,50,none)" unimportantproject
# newtask -p importantproject /opt/bomb/cpuhog1.pl &
# newtask -p unimportantproject /opt/bomb/cpuhog1.pl &
```
How to create two zones using FSS to limit CPU consumption in case of resource contention?
```
# zonecfg -z tz1
zonecfg:tz1> create
create: Using system default template 'SYSdefault'
zonecfg:tz1> set zonepath=/export/zones/tz1
zonecfg:tz1> set autoboot=true
zonecfg:tz1> set cpu-shares=150
zonecfg:tz1> verify
zonecfg:tz1> commit
zonecfg:tz1> exit
root@aserver:~# zonecfg -z tz2
Use 'create' to begin configuring a new zone.
zonecfg:tz2> create
create: Using system default template 'SYSdefault'
zonecfg:tz2> set zonepath=/export/zones/tz1
zonecfg:tz2> set autoboot=true
zonecfg:tz2> set cpu-shares=50
zonecfg:tz2> verify
zonecfg:tz2> commit
zonecfg:tz2> exit
```
Setting a number of resource controls for the project user.oracle
```
# projmod -sK "project.max-shm-memory=(privileged,64G,deny)" user.oracle
# projmod -sK "process.max-sem-nsems=(priv,4096,deny)" user.oracle
# projmod -sK "project.max-shm-ids=(priv,1024,deny)" user.oracle
# projmod -sK "project.max-sem-ids=(priv,1024,deny)" user.oracle
```
Allow 10 processes per task in project class2005
```
# projmod -K "task.max-lwps=(privileged,10,deny)" class2005
```
What are the available resource controls?
```
# man resource_controls
```

Assign the processes of a SMF service to a project

```
# svccfg -s ssh setprop start/project = astring: testproj
```

Observability

How to install top on a Solaris system?

You don't need it. Really. Solaris has a `top` on steroids.

What should I use instead of top?

```
# prstat -mL
```

But I really want my top!

Okay, okay ... (in Solaris 11.2 it's installed by default anyway depending what „cluster" you are installing)

```
# pkg install diagnostic/top
```

What is the content of the environment variables of a process?

```
# pargs -e <pid>
```

Which arguments were used to start a process?

```
# pargs <pid>
```

Print the command that started a process ready to paste into a shell

```
# pargs -l <pid>
```

Get per-partition IO statistics

```
# iostat -mxPzn 1
```

Report TCP statistics

```
# tcpstat -T tcp -c 1
```

Report UDP statistics

```
# tcpstat -T udp -c 1
```

What user and process is using a port?

```
# netstat -aun
```

Unified Archives

Creating an unified archive

```
# archiveadm create /archivepool/aserver.uar
```

Creating an Unified Archive for recovery purposes

```
# archiveadm create -r /archivepool/aserver.uar
```

Creating an Unified Archive from a single zone of the source system

```
# archiveadm create -z tserver /archivepool/tserver.uar
```

Looking what's inside an Unified Archive (basic information)

```
# archiveadm info /archivepool/aserver.uar
```

Looking what's inside an Unified Archive (origin, deployable systems, size of deployable systems, etc)

```
# archiveadm info –v /archivepool/aserver.uar
```

Looking up the storage configuration of the origin system of an Unified Archive

```
# archiveadm info --target /export/aserver.uar
```

Installing a single zone from a Unified Archive

```
# zonecfg -z iserver create -a /archivepool/aserver.uar -z bserver
# zoneadm -z iserver install -a /archivepool/aserver.uar -z bserver
```

How to use an Unified archive in a AI manifest

```
# cat << EOT > /root/uar_ai.manifest
> <!DOCTYPE auto_install SYSTEM "file:///usr/share/install/ai.dtd.1">
> <auto_install>
>   <ai_instance name="archive0">
>     <target name="desired">
>       <logical>
>         <zpool name="rpool" is_root="true">
>         </zpool>
>       </logical>
>     </target>
>     <software type="ARCHIVE">
>       <source>
>         <file uri="http://example-ai.example.com/recovery.uar">
>         </file>
>       </source>
>       <software_data action="install">
>          <name>*</name>
>       </software_data>
>     </software>
>   </ai_instance>
> </auto_install>
> EOT
# installadm create-manifest -f /root/uar_ai.manifest -m uar_manifest -n
service_092910
```

Miscellaneous

How to switch to rsyslog?

```
# pkg install rsyslog
# svcadm disable svc:/system/system-log:default
# svcadm enable svc:/system/system-log:rsyslog
```

Did You Know?

There are a quite a lot free and open source software (FOSS) packages for evaluation purposes available in the „release“ repository at http://pkg.oracle.com/solaris/release.

These packages provide customers with evaluation copies of new and updated versions of FOSS ahead of officially supported Oracle Solaris product releases. Please go to https://community.oracle.com/docs/DOC-917308 in order to learn more about how you can use this packages. You will find news about updates and new additions to this FOSS packages at https://blogs.oracle.com/solarisfoss/ .

Oracle Solaris 11.3 Cheat Sheet
Service Management Facility

The Oracle Solaris Service Management Facility (SMF) is responsible for managing system and application services, replacing the legacy init scripting start-up mechanism common to other UNIX operating systems. SMF helps improves the availability of a system by ensuring that essential services run continuously even in the event of any software or hardware failures with an automatic restart capability. SMF is a part of the wider predictive self healing capability in Oracle Solaris. Another crucial component of this is the Fault Management Architecture (FMA), responsible for reporting and isolating failed hardware components.

Understanding the SMF Fault Managed Resource Indicator (FMRI)

Each SMF managed service instance is identified by an FMRI, that an administrator can use to enable or disable the service, find out information about, or modify configuration properties related to that service. For example, the file system automounter service identified by svc:/system/filesystem/autofs:default can be dissected as in the following table.

FMRI segment	Description
`svc:/`	FMRI scheme
`system/filesystem`	Service category
`autofs`	Service name
`default`	Service instance

Many SMF commands allow FMRI abbreviations by specifying the instance name, or any of the trailing portion of the service name, assuming it is unique on the system. For example, administrators could also refer to the above service as filesystem/autofs:default, autofs:default, and autofs. We will deliberately use multiple abbreviations in this cheat sheet.

Enabling, disabling and restarting services

Enable service svc:/network/smtp:sendmail

```
# svcadm enable smtp:sendmail
```

Disable service svc:/network/telnet:default

```
# svcadm disable telnet
```

Restart service svc:/network/httpd:apache22

```
# svcadm apache22 restart
```

List information about services

Show all enabled services (including temporarily disabled services)

```
# svcs
```

Show all enabled and disabled services:

```
# svcs -a
```

List detailed information about svc:/system/zones:default

```
# svcs -l zones:default
```

List processes associated with svc:/network/netcfg:default

```
# svcs -p network/netcfg
```

Show why services that are enabled but are not running (or preventing other services from running)

```
# svcs -xv
```

Display all services which depend on the svc:/network/ssh:default service

```
# svcs -D network/ssh
```

List all services svc:/network/ssh:default depends on

```
# svcs -d network/ssh
```

Show the location of the SMF logfile of network/ssh

```
# svcs -L network/ssh
```

Show the content of the SMF logfile of network/ssh

```
# svcs -Lv network/ssh
```

Configuration Layers in the SMF Repository

Service configuration is defined in a number of layers within the SMF configuration repository that helps preserve any local administrative customizations during system upgrade, particularly when the underlying vendor provided default configuration changes. A service property could have different values at different layers of the repository. A simple priority mechanism is used to determine which value is used by the service.

Configuration Layer	Description
`manifest`	Values provided as part of SMF manifests located in `/lib/svc/manifest/`
`system-profile`	Values provided as part of SMF profiles located in `/etc/svc/profile/generic.xml`
`site-profile`	Values provided as part of SMF profile located in `/etc/svc/profile/site/`
`admin`	Values provided by interactive use of SMF commands or libraries

List Service Property Configuration

Service configuration can be listed using two different commands, svcprop and svccfg, and can be used interchangeably.

List all properties (including inherited properties) of the service instance `svc:/network/ssh:default`

```
# svcprop ssh:default
```

List properties specific to the service instance `svc:/network/ssh:default`:

```
# svcprop -c ssh:default
```

List `firewall_context/ipf_method` property of the service instance

```
# svcprop -p firewall_context/ipf_method ssh:default
```

List all properties within the `firewall_context` property group of the service instance `svc:/network/ssh:default`

```
# svcprop -p firewall_context svc:/network/ssh:default
```

Interactively display the `general/enabled` **property for the service** `svc:/network/ssh:default`

```
# svccfg
svc:> select ssh:default
svc:/network/ssh:default> listprop general/enabled
svc:/network/ssh:default> exit
```

Set service property configuration

Configure the config/nodename property on the svc:/system/identity:node service instance:

```
# svccfg
svc:>select identity:node
svc:/system/identity:node> setprop config/nodename = "myhost"
svc:/system/identity:node> refresh
svc:/system/identity:node> exit
```

Configure the config/nameserver property on the svc:/network/dns/client service with two IP addresses:

```
# svccfg -s dns/client
svc:/network/dns/client> setprop config/nameserver = ("192.168.0.1"
"10.0.0.4")
svc:/network/dns/client> refresh
```

List all configuration changes (on all layers) to svc:/system/nameservice/switch:default:

```
# svccfg -s switch:default listcust -L
```

Delete an administrative customization to the config/nameserver property in the svc:/network/dns/client service:

```
# svccfg -s dns/client
```

```
svc:/network/dns/client> delcust config/nameserver
svc:/network/dns/client> refresh
```

Delete the config/nameserver property from the `svc:/network/dns/` client service (and thus masking it):

```
# svccfg -s dns/client
# svc:/network/dns/client> delprop config/nameserver
```

Extract an SMF system profile in order to apply configuration to other systems

```
# svccfg extract -a > system-profile.xml
```

Apply an SMF system profile to a system

```
# cp system-profile.xml /etc/svc/profile/site
# svcadm restart manifest-import
```

Notifications

Configure email notifications for all services that drop from online to maintenance state:

```
# svccfg setnotify -g from-online,to-maintenance mailto:junior
```

Show all service state notifications, that are configured on a system:

```
# svcs —n
```

Service Models

FRMI	Legacyfile
`contract`	The processes are monitored by the contract file system. As soon as a certain contract event is reported, the SMF restarts the service
`daemon`	Synonym for `contract`
`child`	As soon as the method script terminates, it's restarted by SMF.
`wait`	Synonym for `child`
`transient`	The method script is executed once. After this it's not longer monitored by SMF. Optimal for tuning scripts.

Using svcbundle

Creating and installing a service for a transient service

```
# svcbundle -i -s service-name=site/networktuning \
-s start-method=/lib/svc/method/networktuning
```

Creating and installing an manifest for a daemon service

```
# svcbundle -i -s service-name=site/ccprocessingdaemon \
-s start-method=/lib/svc/method/ccprocessingdaemon \
-s model=daemon
```

Using svcbundle to Create an SMF System Profile

```
svcbundle -o nameserver-config.xml -s service-name=network/dns/client \
 -s bundle-type=profile \
 -s service-property="config:nameserver:net_address:192.168.0.1"
```

Using an SMF manifest created by svcbundle

```
# cp ccproccessingdaemon.xml /etc/svc/profile/site
# svcadm restart manifest-import
```

Using the SMF System Profile

```
# cp nameserver-config.xml /etc/svc/profile/site
# svcadm restart manifest-import
```

Converting a rc-script running in run level 2 into an SMF script

```
# svcbundle -s service-name=narf -s rc-script=/etc/init.d/narf:2
```

SMF stencils

Basic configuration for an SMF stencil

```
# svccfg -s /network/http:apache22
svc:/network/http:apache22> addpg virtualhosts_stencil configfile
> setprop virtualhosts_stencil/path = astring:
"/etc/apache2/2.2/conf.d/vhost_smf.conf"
> setprop virtualhosts_stencil/stencil = astring: "vhost_smf.conf"
> setprop virtualhosts_stencil/mode = astring: "0444"
> setprop virtualhosts_stencil/user = astring: "root"
> setprop virtualhosts_stencil/group = astring: "sys"
> refresh
```

Basic configuration for an SMF stencil with static content

```
# cat << EOT > /lib/svc/stencils/vhost_smf.conf
> # Automatically generated ... do not edit
> EOT
# svcadm refresh svc:/network/http:apache22
```

SMF stencil with variables

```
# svccfg -s svc:/network/http:apache22 \
addpg vhost_config application
# svccfg -s svc:/network/http:apache22 \
setprop vhost_config/namevirtualhost = astring: "*:80"
# cat << EOT > /lib/svc/stencils/vhost_smf.conf
# Do not edit
NameVirtualHost $%{vhost_config/namevirtualhost}
# Do not edit
EOT
# svcadm refresh svc:/network/http:apache22
```

SMF stencils with repeating structures

```
root@master:~# cat << EOT > /lib/svc/stencils/vhost_smf.conf
```

```
# Do not edit
NameVirtualHost $%{vhost_config/namevirtualhost}

$%/vhosts_([0-9]*)/ {
<VirtualHost $%{vhost_config/namevirtualhost}>
ServerName $%{vhosts_$%1/servername}
ServerAlias $%{vhosts_$%1/serveralias}
DocumentRoot $%{vhosts_$%1/documentroot}
</VirtualHost>
}
# Do not edit
EOT
# svccfg -s apache22 addpg vhosts_1 application
# svccfg -s apache22 \
setprop vhosts_1/serveralias = astring: 'c0t0d0s0.org'
# svccfg -s apache22 \
setprop vhosts_1/servername = astring: 'www.c0t0d0s0.org'
# svccfg -s apache22 \
 setprop vhosts_1/documentroot = astring: '/var/www/c0t0d0s0.org'
# svccfg -s apache22 addpg vhosts_2 application
# svccfg -s apache22 \
setprop vhosts_2/serveralias = astring: 'moellenkamp.org'
# svccfg -s apache22 \
setprop vhosts_2/servername = astring: 'www.moellenkamp.org'
# svccfg -s apache22 \
setprop vhosts_2/documentroot = astring: '/var/www/moellenkamp.org'
# svcadm refresh svc:/network/http:apache22
```

Scheduled and periodic services in SMF

This feature was introduced in Oracle Solaris 11.3 and allows you to define services which are executed repeatedly either periodical or at a scheduled time.

Type	Example
`peridodic`	Execute this script to check the database every 10 minutes starting from now respectively from the boot of the system
`scheduled`	Execute this script to backup the database every Monday at 01:00

Start the script /root/scripts/check_db.sh every 60 seconds (conventional method)

```
# cat periodic.xml
<?xml version='1.0'?>
<!DOCTYPE service_bundle
  SYSTEM '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
<service_bundle type='manifest' name='periodic/checkdb'>
<service type='service' version='1' name='periodic/checkdb'>

<instance name='default' enabled='false'>
<periodic_method period='60' delay='5' jitter='5'
exec='/root/scripts/db_check.sh' timeout_seconds='0'>
<method_context><method_credential user='root' group='root'
/></method_context>
</periodic_method>
</instance>

<template>
<common_name><loctext xml:lang="C">Sample Periodic
Service</loctext></common_name>
<description><loctext xml:lang="C">What this service does
periodically.</loctext></description>
</template>
</service>
</service_bundle>
# cp periodic.xml /lib/svc/manifest/site/
# svcadm restart manifest-import
```

Please note that creation via svcbundle doesn’t work in the SRUs of Solaris 11.3 available at the publication of the document.

Start the script /root/scripts/backup_db.sh every day at 01:00 (via svcbundle)

```
# svcbundle -i -s service-name=periodic/dbbackup -s start-
method=/root/backup_db.sh -s interval=day -s hour=01 -s minute=00
```

Oracle Solaris 11.3 Cheat Sheet
Installation and Deployment

Automated Installer (AI) is the new network based multi-client provisioning system in Oracle Solaris 11. AI provides hands-free installation of both SPARC and x86 systems by using an installation service that installs systems by leveraging software package repositories on the network.

Automated Installation

Creating an AI zone on an existing server specifying an x86 based DHCP client starting at address 192.168.3.100 with a total count of 20 addresses

```
global# zonecfg -z instserv
zonecfg:instserv> create
zonecfg:instserver> set set zonepath=/export/zones/instserv
zonecfg:instserver> set zonepath=/export/zones/instserv
zonecfg:instserv> set autoboot=true
zonecfg:instserv> select anet linkname=net0
zonecfg:instserv:anet> set lower-link=net0
zonecfg:instserv:anet> end
zonecfg:instserv> add dataset
zonecfg:instserv:dataset> name=rpool/ai/install
zonecfg:instserv:dataset> set name=rpool/export/install
zonecfg:instserv:dataset> set alias=install
zonecfg:instserv:dataset> end
zonecfg:instserv> verify
zonecfg:instserv> commit
zonecfg:instserv> exit
global# zoneadm -z installserver install
global# zoneadm -z installserver boot
instserv# ipadm create-ip –a 192.168.3.200 net0/v4
instserv# pkg install install/installadm
instserv# mkdir /install/ai
instserv# installadm create-service -n s11-i386 -d /install/ai
instserv# installadm set-server -i 192.168.3.100 -c 20 -m
```

List all enabled services

```
# installadm list
```

List any installation manifests associated with the install services:

```
# installadm list -m
```

List any installation manifests associated with the install services:

```
# installadm export -n s11-i386 -m orig_default -o  manifest.xml
```

Import a manifest to be associated with the s11x86 service:

```
# installadm update-manifest -n s11-i386 -m orig_default -f ./manifest.xml
```

Apply a criteria that all clients must have 4096MB memory or greater to the manifest manimaxi of s11x86 service:

```
# installadm create-manifest -n s11-i386 -f ./bigmanifest.xml -m manimaxi
-c mem="4096-unbounded"
```

AI integration with ISC DHCP server configured via:

```
/etc/inet/dhcpd4.conf
```

Zones can be installed thru the AI manifest, when system is installed (method 1)

```
<configuration type="zone" name="zone1"
source="http://xyz/zone1/config.txt" />
```

Zones can be installed thru the AI manifest, when system is installed (method 2)

```
<configuration type="zone" name="zone1"
source="file:///net/server/zone2/config.txt" />
```

Specify an AI manifest for the Zone installation, to apply to either zone1 or zone2

```
# installadm create-manifest -n s11 -f /tmp/zmanifest.xml -c
zonename="zone1 zone2"
```

Define a system configuration profile for zone1

```
# installadm create-profile -n s11 -f /tmp/zprofile1.xml -c
zonename="zone1"
```

Install a Zone after system has been built, while leveraging AI manifest and profile

```
# zoneadm -z zone2 install -m /tmp/my_zone_AI_manifest -c
/tmp/my_zone_SC_profile
```

Installation Troubleshooting

For Open Boot Prom (OBP) on SPARC via install_debug boot argument

```
boot net:dhcp – install install_debug
```

For x86 via GRUB, to kernel line boot entry add the following

```
install_debug=enable
```

Default root password on AI clients during installation is

```
solaris
```

Installation log file during installation

```
/system/volatile/install_log
```

AI client manifest downloaded from the AI server during installation

```
/system/volatile/ai.xml
```

AI client derived manifest (if a derived manifest script is used)

```
/system/volatile/manifest.xml
```

System configuration profiles downloaded from the AI server during installation

```
/system/volatile/profile/*
```

List of AI services located

```
/system/volatile/service_list
```

AI client SMF service log for manifest/profile locator, during installation
`/var/svc/log/application-manifest-locator:default.log`
AI client SMF service log for Automated Installer installation service
`/var/svc/log/application-auto-installer:default.log`
AI server log file for access requests from AI clients
`/var/ai/image-server/logs/access_log`
AI server log file for errors encountered from AI clients
`/var/ai/image-server/logs/error_log`
AI server SMF service log
`/var/svc/log/system-install-server:default.log`
AI server boot configuration files
`/etc/netboot`
Specify location of AI imagepath, default is /export/auto_install/<service_name>
`# installadm create-service –d`
Boot without starting an installation on SPARC
`ok> boot net:dhcp`
Boot without starting an installation on x86
`From GRUB menu, select first entry (Text)`

System Configuration Profiles

System Configuration Profiles are used to provide system configuration information profiles, as used by Automated Installer.

Interactively create a system configuration profile and save it to a file, to be subsequently used for deployments
`# sysconfig create-profile -o sc-profile.xml`
Specify a system configuration profile to use when installing a system with a specific MAC criteria
`# installadm create-profile –n s11service –f sc_profile.xml –c MAC=00:11:22:33:44:55`
List what system configuration profiles are associated with a service, and for which criteria (if any)
`# installadm list -n s11service –p`
List all non-default system configuration profiles associated with any of the install services:
`# installadm list –p`
Validate a system configuration profile against the default x86 install service:
`# installadm validate -n default-i386 -P profile.xml`
Associate a system configuration profile with the default x86 install service and give it a name sc-profile:
`# installadm create-profile -n default-i386 -f profile.xml -p sc-profile`
Default system configuration profile and AI manifest used for zone installs are:
`/usr/share/auto_install/sc_profile/enable_sci.xml` `/usr/share/auto_install/manifest/zone_default.xml`

Migrating from Oracle Solaris 10 Jumpstart to Oracle Solaris 11 Automated Installer

Migration of Oracle Solaris 10 (and earlier) Jumpstart infrastructure can be aided with js2ai tool. It does a "Best-effort translation" and produces XML syntax for, and aids in conversion of:

Jumpstart	AI
Jumpstart rules	AI criteria
Jumpstart profiles	AI manifests
sysidcfg files	System configuration profiles

If there is a Jumpstart keyword that has no equivalent in AI, the user can manually edit the AI manifest to leverage AI.

Convert a sysidcfg file in the current directory to a system configuration profile named sc_profile.xml
`# js2ai -s`
Convert an entire Jumpstart directory under /export/jumpstart
`# js2ai -r -d /export/jumpstart`
Convert a rules file and associated profiles to AI criteria and AI manifests
`# js2ai -r`
Convert a profile to AI manifests and save in AI_<$profile> subdirectory
`# js2ai -p profile`

Please note, that the js2ai tool is on the End-Of-Feature list and thus will disappear in future versions of Solaris.

Suggestions, errors and comments

This cheatsheet is a living document, in case you have suggestions, spot any errors or you would like to comment feel free to send a mail directly to joerg.moellenkamp@oracle.com.

Contact Us

This Oracle Solaris 11 Administrator's Cheat Sheet was written by Joerg Moellenkamp, Principal Sales Consultant for Oracle. He is member of the Oracle Elite Engineering Exchange. Joerg writes a blog that can be found at http://www.c0t0d0s0.org/ .The SMF part was initially written by Glynn Foster. The AI part was initially written by Isaac Rozenfeld

Oracle Elite Engineering Exchange

Contributions over time ranging from corrections over substantial suggestions and ideas to full examples taken from the individuals blog or mails by (in alphabetical order):
Casper Dik, Rod Evans, Glenn Faden, Glynn Foster, Mike Gerts, Mary Jane Greenfield, Thomas Hildebrand, Artem Kachitchkine, Alfred Mayerhofer, Darren Moffat, Anup Sekhar, Jeff Taylor, Steffen Weiberle

For more information about Oracle Solaris 11, visit oracle.com/solaris or call +1.800.ORACLE1 to speak to an Oracle representative.

Last updated: 27.05.16 16:19

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together