

NEW!

**CramSession**Comprehensive **Study Guides**

A+  
Adobe  
C++  
Cisco CCNA

**Your Trusted  
Study Resource  
for  
Technical  
Certifications**

Written by experts.  
The most popular  
study guides  
on the web.

In Versatile  
PDF file format

Check out these great features  
at [www.cramsession.com](http://www.cramsession.com)

> **Discussion Boards**

<http://boards.cramsession.com>

> **Info Center**

<http://infocenter.cramsession.com>

> **SkillDrill**

<http://www.skilldrill.com>

> **Newsletters**

<http://newsletters.cramsession.com/default.asp>

> **CramChallenge Questions**

<http://newsletters.cramsession.com/signup/default.asp#cramchallenge>

> **Discounts & Freebies**

<http://newsletters.cramsession.com/signup/ProdInfo.asp>

INFORMATION TECHNOLOGY

# Sun Solaris 8 Certified Systems Administration II

Version 3.0.0

Microsoft Office  
Microsoft Windows 2000  
Microsoft Windows XP  
Network Security  
Network+  
Networking  
Nortel Networks  
Novell  
Oracle  
Proxy Server  
Red Hat Linux  
SAIR Linux  
SANS  
SCO  
Server+  
SQL  
Sun Solaris  
Unix  
Visual Basic  
Web Design

**Notice:** While every precaution has been taken in the preparation of this material, neither the author nor Cramsession.com assumes any liability in the event of loss or damage directly or indirectly caused by any inaccuracies or incompleteness of the material contained in this document. The information in this document is provided and distributed "as-is", without any expressed or implied warranty. Your use of the information in this document is solely at your own risk, and Cramsession.com cannot be held liable for any damages incurred through the use of this material. The use of product names in this work is for information purposes only, and does not constitute an endorsement by, or affiliation with Cramsession.com. Product names used in this work may be registered trademarks of their manufacturers. This document is protected under US and international copyright laws and is intended for individual, personal use only.

For more details, visit our [legal page](#).



**CramSession**  
Prepare for Success!



# Sun Solaris 8 Certified Systems Administration II

Version 3.0.0

**NOTICE:** Got the **NEWest Version?**  
Make sure by clicking here!

## Abstract:

This study guide will help you to prepare for Sun exam 310-012, Solaris 8 System Administration II. Exam topics include syslog, NFS, CacheFS, naming services, RBAC, Management Console, AdminSuite and JumpStart.

Find even more help here:

- > **Feedback & Discussion Board for this exam**
- > Read & Write Reviews of this study guide
- > Rate this Cramsession study guide



**Contents:**

Client-Server Relationship ..... 6

- List and define the types of servers used in the Solaris 8 Network Environment . 6
- List and define the types of clients used in the Solaris 8 network environment... 6

Solaris Network Environment..... 6

- Define the function of each layer within the seven-layer OSI model ..... 6
- Define the function of each layer in the five-layer TCP/IP model ..... 7
- List the features and functions of the Ethernet ..... 7
- Describe the characteristics of RARP and ARP ..... 8
- Identify the commands which display information about the local network interface ..... 8
- Describe the relationship between the RPC service and the rpcbind process ..... 8
- Recall how to list registered RPC services ..... 8
- Identify the steps necessary to start and stop network services via the command line..... 8

Solaris Syslog ..... 9

- Identify the functions of syslog ..... 9
- Recall the syntax of the syslog configuration file ..... 9
- Deduce syslogd behavior from its configuration file.....10
- Configure syslog messages by increasing the logging severity level for both the login and telnet daemons.....10
- Use the command line to update the system log .....11

Disk Management .....11

- List the utilities used to create, check, and mount file systems .....11
- Identify the logical path name differences between physical disks and virtual disks.....11
- List the advantages of a virtual disk management application .....11
- Identify the characteristics and functions of Solstice DiskSuite and the Sun StorEdge Volume Manager .....12

Solaris Pseudo File Systems and Swap Space .....12



**Sun Solaris 8 Certified Systems Administration II**

State the characteristics of the Solaris pseudo file system types.....12

List and define the commands used to extract information from entries in the /proc directory .....12

List the steps to create and add a swap file to the system swap space.....13

NFS .....13

State the functions of an NFS server and an NFS client .....13

List the steps required to make resources available and unavailable for mounting as a shared resource.....13

Identify the command used in the /etc/dfs/dfstab file on an NFS server to enable automatic sharing of resources .....14

Explain how entries in the /etc/vfstab file can enable automatic mounting of resources on an NFS client.....14

State the function of each of these commands: mountall, umountall, shareall, and unshareall .....14

Explain the new NFS server logging.....15

AutoFS .....15

List the benefits of using the automount utility.....15

State the purpose of each of the types of automount maps.....16

Identify the steps needed to set up automount to read a direct map .....16

Identify circumstances under which the automount daemon should be restarted .....16

CacheFS .....16

Given an existing client-server environment, explain how to configure the cache file system.....16

Use appropriate commands to check the status and consistency of the cache file system.....17

Identify the steps to set up cache file system logging .....17

List the steps necessary to perform a check of the cache file system .....18

Identify the steps to dismantle and delete a cache file system.....18

Naming Services .....18

State the purpose of a name service .....18

List the different name services and compare their functionality .....18



**Sun Solaris 8 Certified Systems Administration II**

Given a local area network with a need for a name service, identify the appropriate name service switch process and determine which configuration to use for your network.....19

NIS .....19

    List and define the processes and components of the NIS master server, NIS slave server, and NIS client .....19

    List the steps to configure an NIS master, slave, and client .....20

    Given an existing network using the NIS name service, list the steps to add a new NIS map .....21

    Given an existing network using the NIS name service, list commands to update and propagate an NIS map .....21

Role-Based Access Control (RBAC) .....22

    State the purpose of the Role-Based Access Control (RBAC) with regard to Solaris security .....22

    Select the statements that describe RBAC database features .....22

Solaris Management Console and Solstice AdminSuite .....24

    List the features of the Solaris Management Console .....24

    List the features of the Solaris AdminSuite .....24

JumpStart-Automatic Installation .....25

    State the purpose of the JumpStart server .....25

    Identify the main components of a Jumpstart server.....25

    List the parameters possible when using the add\_install\_client script syntax ....26

    State the purpose of the boot service function on a subnet .....26

    Identify the events that occur during the JumpStart Client boot sequence.....26

    List the files necessary to support the JumpStart boot operation.....27

    State the purpose of the sysidcfg file with and without name service support ...28

    Select the statements that describe the steps necessary to set up a JumpStart install server system to provide the Solaris 8 release software necessary to install a new system .....28

    State the purpose of the add\_to\_install\_server, modify\_install\_server and add\_install\_client scripts.....28

    Given the appropriate software source, explain how to create a configuration server with a customized rules file and class files .....29



**Sun Solaris 8 Certified Systems Administration II**

Given an NIS name service network environment, explain how to configure NIS name service support for JumpStart .....29



## Client-Server Relationship

### List and define the types of servers used in the Solaris 8 Network Environment

- **Application servers** - Share software applications across the network with clients.
- **Boot servers** - Provide startup configuration information to new clients booting up on the network. Used with Jumpstart.
- **Installation servers** - Provide client software images to new computers booting up on the network. Also used with Jumpstart.
- **Database servers** - Provide a platform for running database applications and sharing this data with clients.
- **Mail servers** - Store and forward servers that allow access to email to local clients.
- **License servers** - Manage application and system licenses using a special license daemon.
- **Print servers** - Share locally attached or network-attached printers with clients on a network.
- **Name services server** - Host one of the many naming services provided by Solaris (DNS, FNS, NIS, NIS+).
- **home Directory servers** - Provide storage of 'exported' home directories in a central location.

### List and define the types of clients used in the Solaris 8 network environment

Clients use the services provided by servers. A single client can request multiple services from various servers, like shared files, printers and data from a database. A specialized example would be a jumpstart client, which requests configuration and installation information from a jumpstart server in anticipation of a new operating system install.

Sun now has technology called SunRay™ that provides server based or thin-client computing to clients that have only keyboards, mice, display and simple CPUs.

## Solaris Network Environment

### Define the function of each layer within the seven-layer OSI model

- **Application layer**  
Represents the user level. TELNET, FTP, SMTP, NFS.
- **Presentation Layer**  
Different computers interpret information in different ways. The presentation layer facilitates the encoding and decoding required between platforms.



Examples would include ASCII and EBCDIC. XDR (external data representation) resides at this layer.

- **Session layer**  
The session layer manages services like authentication, dialog management and encryption between connected clients. It also reestablishes interrupted connections.
- **Transport Layer**  
Handles transport-specific functions like flow-control and quality between two communicating stations.
- **Network Layer**  
The network layer addresses, routes and delivers data traffic on a network. Routing tables are found at this layer.
- **Data Link Layer**  
This layer addresses the physical medium directly. This is the first location where bits are arranged into a recognizable format. Checksum error detection occurs here. MAC addresses are used here.
- **Physical Layer**  
Operating at the lowest level, this layer moves unstructured bit streams using electrical signals.

### Define the function of each layer in the five-layer TCP/IP model

Sun's implementation of the TCP/IP protocol stack includes five layers:

#### 1. Application Layer

User accessed application programs and network services.

#### 2. Transport Layer

Connection-oriented TCP and connectionless UDP data transfer.

#### 3. Internet Layer

Here data is fragmented, addressed and routed.

#### 4. Network Interface Layer

Error detection and packet framing. 802.3, 802.4, 802.5.

#### 5. Hardware Layer

Contains electrical signals that move raw bits through the ether.

### List the features and functions of the Ethernet

Ethernet is an implementation of **C**arrier **S**ense **M**ultiple **A**ccess with **C**ollision **D**etection (CSMA/CD). Hosts that share a subnet (usually a Bus) transmit data to other hosts at random intervals. If two hosts transmit at the same time, a collision occurs, and one host must rebroadcast after a random wait time. Ethernet is popular because it is easy to add hosts, and performs rather well on lightly loaded networks. Performance does begin to degrade as more hosts are added.





### **Describe the characteristics of RARP and ARP**

ARP and RARP (ReverseARP) are protocols to match a host's unique MAC address (hard-coded into the network interface card) with an assigned ethernet address (IP). To obtain a destination IP or Ethernet address, a host must send a broadcast alerting other hosts on the network to its request.

ARP maps a 32-bit IP to a 48-bit MAC (or Ethernet) address.

RARP maps a 48-bit MAC address to a 32-bit IP address.

### **Identify the commands which display information about the local network interface**

**ifconfig** - shows the status of configured interfaces. Includes information relating to the MTU, IP address, Netmask, Broadcast address, and MAC address.

```
#ifconfig -a
```

```
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
      inet 127.0.0.1 netmask ff000000
```

```
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
      inet 10.1.15.10 netmask ffffffff broadcast 10.1.15.255
```

**banner** - shows the MAC of a system from the OBP (Open Boot Prom).

### **Describe the relationship between the RPC service and the rpcbind process**

A network service must use an agreed-upon unique port number. To eliminate the problem of too many hosts and too many services to configure and maintain distinctive information for, Sun created an RPC service that does not require predefined port numbers to be established at boot time.

A process, `rpcbind`, interprets incoming requests and sends them to the appropriate server processes. Using RPC, clients are given the actual port number at connection time by `rpcbind` (listening at well-known port 111). RPC services register themselves with `rpcbind` when they start, and are assigned an available port number at that time. RPC services are named `rpc.<daemon>`.

### **Recall how to list registered RPC services**

The configured ports for RPC are listed in `/etc/rpc`.

To see which services are currently running, use the `rpcinfo -p` command.

### **Identify the steps necessary to start and stop network services via the command line**

`rpcinfo` can also start and stop network services. To reregister network services that have been stopped, send a hangup (HUP) signal to the `inetd` process.

```
# pkill -HUP inetd
```

and then verify the network service is available using `rpcinfo -p`

To stop a network service, use `rpcinfo` in the following manner:



```
# rpcinfo -d mountd 1
```

## Solaris Syslog

### Identify the functions of syslog

syslog is a system of routing messages generated by the system (kernel) or system utilities to appropriate, manageable log files. Messages can be sent to the console or a system log file, sent to a list of users logged on, or forwarded to other hosts on a network.

### Recall the syntax of the syslog configuration file

The syslog configuration file is located in `/etc/syslog.conf`. The `syslogd` daemon reads the file each time it is started.

Two fields of the `syslog` are: selector and action. The selector field is divided into two fields delimited by a period: `facility.level`.

The following table contains the facility definitions:

user	kern	mail	daemon
auth	lpr	news	uucp
cron	local0 - 7	mark	*

The following table contains the severity levels (highest to lowest):

emerg
alert
crit
err
warning
notice
info
debug
none



## Deduce syslog behavior from its configuration file

The syslog configuration file appears like this:

```
#ident "@(#)syslog.conf 1.5 98/12/14 SMI" /* SunOS 5.0 */
#
# Copyright (c) 1991-1998 by Sun Microsystems, Inc.
# All rights reserved.
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (`') names
# that match m4 reserved words. Also, within ifdef's, arguments
# containing commas must be quoted.
#
*.err;kern.notice;auth.notice /dev/sysmsg
*.info;mail.none;*.err;kern.debug;daemon.notice /var/adm/messages
mail.err;mail.info;mail.alert;mail.emerg;mail.notice /var/log/maillog
*.alert;kern.err;daemon.err operator
*.alert root
*.emerg *
# if a non-loghost machine chooses to have authentication messages
# sent to the loghost machine, un-comment out the following line:
#auth.notice ifdef(`LOGHOST', /var/log/authlog, @loghost)
mail.debug ifdef(`LOGHOST', /var/log/maillog, @loghost)
local6.debug /var/log/tacacs.log
local7.emerg;local7.alert;local7.debug /var/log/local7.log
#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef(`LOGHOST', ,
user.err /dev/sysmsg
user.err /var/adm/messages
user.alert `root, operator'
user.emerg *
)
```

Note that the syslog conf file is processed via the M4 macro processor. The `ifdef` statements are evaluated to determine where information is sent.

## Configure syslog messages by increasing the logging severity level for both the login and telnet daemons.

Changing the severity level of the login daemons requires changing the level associated with the auth facility:

```
auth.info /var/adm/messages
to
auth.crit /var/adm/messages
```



Changing the severity level of the login daemons requires changing the level associated with the daemon facility:

```
daemon.info /var/adm/messages  
to  
daemon.crit /var/adm/messages
```

### Use the command line to update the system log

The logger command updates entries in the system log.

```
logger [ -I ] [ -f file ] [ -p priority ] [ -t tag ] [ message ]  
# logger -p user.err "System Restart"
```

This is useful functionality when writing scripts.

## Disk Management

### List the utilities used to create, check, and mount file systems

`newfs` is a utility to create the ufs filesystem on a new partition (remember that it is a friendly front-end to the more configurable `makefs` command).

`fsck` is the utility used to check a new file system. It detects and repairs inconsistencies.

`mount` is the utility that is used to 'attach' a new file system to the existing hierarchy. Solaris™ 8 provides a new feature for adding devices (reconfiguration) without rebooting. The `devfsadm` utility combines the functions of `drvconfig`, `disks`, and `devlinks` for creating entries in `/dev/` and `/devices/`.

### Identify the logical path name differences between physical disks and virtual disks

Typically, file systems on Solaris™ are limited to just one partition or slice. Using tools like Solstice™ DiskSuite™ or Sun StorEdge™ Volume Manager, an administrator can span a file system over more than one partition.

Using DiskSuite™, a virtual file system path would look like:

```
/dev/md/rdsk/d10  
/dev/md/dsk/d10
```

Using Volume Manager, a virtual file system path would look like:

```
/dev/vx/rdsk/tools/binaries  
/dev/vx/dsk/tools/binaries
```

### List the advantages of a virtual disk management application

Using a virtual disk management application, a systems administrator can overcome disk capacity limitations, and improve performance and reliability by supporting various RAID configurations.



Most virtual disk management utilities are in the form of GUIs that make system setup and breakdown easier.

**Identify the characteristics and functions of Solstice DiskSuite and the Sun StorEdge Volume Manager**

DiskSuite™ combines disks that have been created using format. A collection of existing disks or partitions makes up a *metadevice*.

Volume manager manages disk space on *subdisks* by formatting into two initial partitions. Slice 3 is a private area that maintains information about the public file system. Slice 4 is used to provide disk space to create new volumes.

**Solaris Pseudo File Systems and Swap Space**

**State the characteristics of the Solaris pseudo file system types**

A pseudo file system is a file-system that is entirely resident in memory while the operating system is running. They provide an important increase in performance by using standard operating system semantics on data structures that are in faster physical memory, instead of on disk.

*procfs* - stores a list of active processes on the machine

*tmpfs* - a space for temporary system files that is cleared with each reboot

*fsfs* - maintains a list of file names and file descriptors for open files

*swapfs* - performs virtual addressing for the swap files on a disk

**List and define the commands used to extract information from entries in the /proc directory**

The *proc* file system contains a hierarchical directory structure for the state of each active process. Each entry is a decimal number that corresponds with the process ID. Each directory contains files that define the process. The owner of the process determines owner and group permissions.

```
# ps -ef
  UID    PID  PPIDC      STIME TTY          TIME CMD
  root     0     0        0    Dec 02 ?           0:00 sched
  root     1     0        0    Dec 02 ?           0:01 /etc/init -
  root     2     0        0    Dec 02 ?           0:00 pageout
  root     3     0        0    Dec 02 ?          24:13 fsflush
  root    310     1        0    Dec 02 ?           0:00 /usr/lib/saf/sac -t300

nobody  166     1  0    Dec 02 ?           0:59 /usr/local/sbin/iplog

  root     50     1  0    Dec 02 ?           0:00 /usr/.../devfsadmd
  root    143     1  0    Dec 02 ?           0:00 ipmon -sn
  root    179     1  0    Dec 02 ?           0:00 /usr/sbin/inetd -s
```



## Sun Solaris 8 Certified Systems Administration II

```
root 26980      1  0   Dec 13 ?           0:03 /usr/sbin/syslogd
root   199      1  0   Dec 02 ?           0:01 ntpd -A
root 21479      1  0   Dec 11 ?           0:30 /usr/sbin/nscd
root   194      1  0   Dec 02 ?           0:00 /usr/sbin/cron
root   914     360  0 17:11:10 ?           0:00 /usr/local/sbin/sshd
root   313     310  0   Dec 02 ?           0:00 /usr/lib/saf/ttymon
<...>
```

### List the steps to create and add a swap file to the system swap space.

File space can be added to swap on the fly using `swap -a` command:

Find a partition that has free space.

```
# df -k
# mkfile 100m /export/home/myswap
# swap -a /export/home/myswap
```

This will be cleared at next reboot. Make it permanent by adding it to the `/etc/vfstab` file.

## NFS

### State the functions of an NFS server and an NFS client

NFS stands for **N**etwork **F**ile **S**ystem, and provides a means of distributing files to clients across a network.

An NFS server stores the files or common programs on a local disk, and runs the appropriate daemons to share them across a network.

An NFS client mounts shared files or common programs from an NFS server across the network. The process is almost transparent, in that files on an NFS share appear to be local to the client machine.

### List the steps required to make resources available and unavailable for mounting as a shared resource

Add an entry to the `/etc/dfs/dfstab` file to make a directory available

```
share /usr/share/man
```

Start the NFS daemons on the server

```
/etc/init.d/nfs.server start
```

On the NFS client, use the `mount` command to connect the resource

```
# mount <servername>:/usr/share/man /usr/share/man
```

To unmount the resource from the client, run

```
# umount /usr/share/man
```

To stop sharing on the server, run

```
# unshareall
```

or

remove the share entry from the DFS tab and stop/start the NFS server



**Identify the command used in the /etc/dfs/dfstab file on an NFS server to enable automatic sharing of resources**

The /usr/sbin/share command makes files available for remote mounting.

```
share [ -F fs-type ] [ -o options ] [ -d description ] path_name  
# share -F nfs /export/home
```

will share out the /export/home directory. Data about the share is logged in the /etc/dfs/sharetab file.

Running share with no arguments will display the current shared resources.

/etc/dfs/dfstab is similar to the /etc/vfstab, in the sense that it is a hard and fast recording of shares to be shared out during the boot process, or by using commands like shareall. The file simply contains a listing of share commands.

```
share -F nfs -o ro /export/updates  
share -F nfs -o rw:sparty:beaumont /export/home
```

**Explain how entries in the /etc/vfstab file can enable automatic mounting of resources on an NFS client**

The /etc/vfstab file is read by the system startup scripts and is in charge of mounting specified file systems (including the standard /, /usr, /var, /opt, etc.) at boot time. Adding entries for NFS shares causes the server to attempt to mount the shared resources along with local mounts.

```
#device      device      mount      FS      fsck      mount      mount  
#to mount    to fsck     point      type    pass     at boot  options  
#  
#  
sparty:/export/home -      /export/home  nfs      -        yes      bg
```

Note that the device to fsck contains a -, because NFS shares are never fscked. Mount at boot is yes in this case, and the option bg will specify 'background' mounting (the system will retry the mount in the background if an initial attempt fails).

**State the function of each of these commands: mountall, umountall, shareall, and unshareall**

mountall, when executed on a client machine, will mount all the shares specified in the /etc/vfstab file as 'mount at boot'. When used with the -r option, only remote shares will be mounted.

```
# mountall -r
```

umountall will unmount the current mounted file systems. When used with the -r option, only remote shares will be unmounted.

```
# umountall -r
```



`shareall` is run on a server machine to automatically share all the filesystems listed in the `/etc/dfs/dfstab` file.

`unshareall` is run on a server machine to unshare all of the currently shared file systems. It reads from the `/etc/dfs/sharetab` file.

### Explain the new NFS server logging

NFS server logging, new to Solaris™ 8, enables record-keeping of connections made to shared file systems. A daemon, `nfslogd`, is in charge of the logging.

The logs store ASCII output of IP/hostnames and User IDs of clients. A file called the file-handle-to-path stores file handle information about requested files so that it does not need to be re-identified for each use.

The file `/etc/nfs/nfslog.conf` defines path, file name, and type of logging for `nfslogd`.

Logging for a particular file system is handled using the `log` option of `share`

```
# share -F nfs -o ro, log=global /export/home
```

`global` is the default value in the `nfslog.conf` file

This file looks like:

```
#ident "@(#)nfslog.conf 1.5 99/02/21 SMI"
#
# Copyright (c) 1999 by Sun Microsystems, Inc.
# All rights reserved.
#
# NFS server log configuration file.
#
# <tag> [ defaultdir=<dir_path> ] \
# [ log=<logfile_path> ] [ fhtable=<table_path> ] \
# [ buffer=<bufferfile_path> ] [ logformat=basic|extended ]
#
global defaultdir=/var/nfs \
log=nfslog fhtable=fhtable buffer=nfslog_workbuffer
```

## AutoFS

### List the benefits of using the automount utility

File mounts on demand – automatic mounts take place when referenced share is needed.

A timeout feature can be configured to unmount shares that haven't been used.

A name service can be configured to manage NFS mounts in conjunction with automount.

Handles load balancing and redundancy between servers when multiple servers share the same file systems.





### State the purpose of each of the types of automount maps

There are four types of maps used by the autofs:

1. **Master Maps** – automount reads these maps to determine what other maps are present for the autofs environment
2. **Direct Maps** – contain the absolute pathnames to mount points
3. **Indirect Maps** – contain the relative pathnames to mount points
4. **Special Maps** – maps that point to the `/etc/hosts` file or to FNS

### Identify the steps needed to set up automount to read a direct map

Share out data that is stored in an `/export/opt` directory

The `/etc/auto_master` map file must be modified to specify a new direct map file

```
+auto_master
/-          auto_direct
/net        -hosts      -nosuid,nobrowse
/home      auto_home   -nobrowse
/xfn       -xfn
```

A new file called `/etc/auto_direct` must be created, and an entry containing the absolute pathnames to the shared data must be entered.

Note that the `+` symbol refers the automounter to NIS/+. Comment this out for local files.

```
/export/opt -ro sparty:/export/opt
```

Re-run `automount` with the `-v` option to make sure the changes take effect. The autofs daemons may also be stopped and started. This must be done when making changes to either the master maps or the direct maps.

```
# automount -v
```

### Identify circumstances under which the automount daemon should be restarted

As stated above in the direct map example, the `automountd` daemons should be restarted whenever the master maps or direct maps are modified.

## CacheFS

### Given an existing client-server environment, explain how to configure the cache file system

CacheFS is available to speed up access to slow remotely mounted file systems or devices like CDRoms. Basically, a local cache is created on the hard disk, and requests for the remote data are redirected to the cache.

CacheFS has three main terms to remember:

1. **Back File System** – the original source of the data, be it network or CDRom
2. **Front File System** – the mounted, cached local file system



### 3. **Consistency** – the synchronization status between the front and back systems

To create a CacheFS file system, use the `cfsadmin` utility:

```
# cfsadmin -c /export/c_home_dirs
# mount -F cachefs -o backfstype=nfs,cachedir=/slow_remote_system, \
  cacheid=remote_system_0930 server1:/export/home /export/c_home_dirs
```

Where `server1` is the remote server sharing its `/export/home`, which is being cached on this server in `/export/c_home_dirs`.

#### **Use appropriate commands to check the status and consistency of the cache file system**

`Cachefsstat` is the command that displays CacheFS status information, and `cfsadmin` is the command that verifies consistency.

```
# cachefsstat /slow_data
```

```
/slow_data
cache hit rate:          100% ( 3 hits, 0 misses )
consistency checks:     18   ( 18 pass, 0 fail )
modifies:               0
garbage collection:     0
```

```
# cfsadmin -s /slow_data
```

invokes consistency checking. Note, consistency checking is not necessary for back file systems that do not change (i.e., CDROM). Mounting these file systems with `demandconst` specified in `mount` will disable consistency checking. The `cfsadmin` utility will force consistency check if it is disabled at mount time.

#### **Identify the steps to set up cache file system logging**

CacheFS logging enters the picture when the size of the cache file system is being determined. The `cachefswssize` (whoa!) command will display the amount of space allocated to the cache file system. It obtains this information from the cacheFS system logs, configured like the following:

```
# mkdir /var/cachelogs
# cachefslog -f /var/cachelogs/slow_data.log /slow_data
```

A file `slow_data.log` stores statistics about the `/slow_data` files being served.

```
# cachefslog /slow_data
/var/cachelogs/slow_data.log: /slow_data
displays the log file name for the cached file system.
```

```
# cachefswssize /var/cachelogs/slow_data.log
```

```
total for cache
  initial size:      30720k
  end size:          4096k
high water size:     4096k
```



### List the steps necessary to perform a check of the cache file system

fsck can be used to verify the cache file system.

```
# umount /slow_data
# fsck -F cacheefs -o noclean /cache/cache0
```

Remount using the mount command.

### Identify the steps to dismantle and delete a cache file system

Determine the cache ID for the file system that is being removed.

```
# cfsadmin -l /cache/cache0
cfsadmin: list cache FS information
    maxblocks      45%
    minblocks      0%
    threshblocks   75%
    maxfiles       45%
    minfiles       0%
    threshfiles    75%
    maxfilesize    6MB
    remote_system_0930
```

umount the file system using the cached data

```
# umount /c_home_dirs
```

Delete the cache file system

```
# cfsadmin -d remote_system_0930 /cache/cache0
```

## Naming Services

### State the purpose of a name service

Name services act as intermediaries between the numerical addressing scheme that computers use to identify themselves on a network, and alphabetical host names that are easy for humans to remember.

Special software on clients and servers translate between the two on the fly. More complex name services also share account information and machine configurations. Naming services offer a single point of administration, consistency, and immediate update capabilities.

### List the different name services and compare their functionality

- **Domain Name Service (DNS)** - Globally supported naming system used on TCP/IP systems. It is hierarchical in nature, and supports local delegation.
- **Network Information Service (NIS)** - Centrally-managed, domain-based Sun system of maintaining configuration files for clients on a set of designated name servers. Can update user accounts, host names, file maps, etc. on many client computers.



- **Network Information Service Plus (NIS+)** - An updated version of NIS that supports tighter security, cross-domain support, and hierarchical naming.
- **Lightweight Directory Access Protocol (LDAP)** - Combines standard naming service features with a directory that can store custom information about objects in the information bank. These objects can be queried in a number of ways.

**Given a local area network with a need for a name service, identify the appropriate name service switch process and determine which configuration to use for your network**

The name service switch is a file (`/etc/nsswitch.conf`) that controls how network information is obtained. Each client on the network has a local copy of this file. Entries in the file determine how a particular type of information is obtained (e.g., from NIS, NIS+, DNS, etc.)

Tables or objects are listed in the file and can be configured individually for each name service and in which order those name services should be queried when a lookup is necessary.

A network that uses DNS as a primary host resolver would specify DNS first in the list that trails the hosts (an entry is made in the `nsswitch.conf` file). By default, the file is configured to use NIS.

Template files exist to make the configuration easier. For example, to enable NIS, simply copy the `/etc/nsswitch.nis` to `/etc/nsswitch.conf`.

The five templates in Solaris™ 8 are:

- `/etc/nsswitch.files` - searches only local files for information
- `/etc/nsswitch.dns` - searches local files for everything but hosts
- `/etc/nsswitch.nis` - searches the local NIS database for information
- `/etc/nsswitch.nisplus` - searches the local NIS+ databases for information
- `/etc/nsswitch.ldap` - searches the LDAP database for information

## NIS

**List and define the processes and components of the NIS master server, NIS slave server, and NIS client**

An NIS system is comprised of servers that act as repositories for configuration information, which is shared with clients that are all members of the same domain. There are five main processes in an NIS environment:

1. `ypserv` - answers `ypbind` requests
2. `ypbind` - binds to domain server and stores binding information
3. `rpc.yppasswdd` - accounts for password changes and updates appropriate maps



4. `ypxfrd` – transfers NIS maps within the domain
5. `rpc.yppupdated` – also transfers NIS maps within the domain

`ypserv` is found on master and slave servers. `ypbind` is run on master, slave and client computers. `rpc.yppasswdd`, `ypxfrd` and `rpc.yppupdated` are all run on master servers only. The NIS master server is a single server that contains the master copies of configuration tables for the network. These map files are built from special ASCII files and stored in the `/etc/` directory. Having one server allows for a single configuration and control point for an entire domain. Remember that a NIS server is a client of itself in the domain.

The NIS slave server receives map files from the master server in the domain. They act as backups in the event that a master server becomes unavailable. In busy networks, their presence in the domain serves as load balancing for map requests. The NIS client does not contain any local maps. Processes on the client bind to the master or a slave server for configuration information. In the event that the host it is bound to should become unavailable, it can dynamically rebind to another working server.

### List the steps to configure an NIS master, slave, and client

For a master server:

- 1) Make sure the computer is configured for NIS
- 2) Set the domain name using the `domainname` command and editing the `/etc/defaultdomain` file
- 3) Make sure the text files with configuration information are up to date in the `/etc` directory (touch `ethers`, `bootparams`, `locale`, `timezone`, `netgroup` and `netmasks` files so they have a zero length)
- 4) Run `ypinit -m` and specify slave servers when asked
- 5) Start the NIS daemons: `/usr/lib/netsvc/yp/ypstart`

For a slave server:

- 1) Make sure the computer is configured for NIS
- 2) Set the domain name using the `domainname` command and editing `/etc/defaultdomain` file
- 3) Initialize the machine as a client: `ypinit -c`, enter in the names of the other master and slave servers when prompted
- 4) Start the NIS daemons: `/usr/lib/netsvc/yp/ypstart`
- 5) Initialize the machine as a slave: `ypinit -s <master name>`
- 6) Stop and start the NIS daemons on the new slave server



For a client:

- 1) Make sure the computer is configured for NIS
- 2) Make sure the `/etc/hosts` file contains the master and slave servers
- 3) Set the local domain name by using the `domainname` command
- 4) Initialize as a client: `ypinit -c`, and enter in the names of the master and slave servers when prompted

### Given an existing network using the NIS name service, list the steps to add a new NIS map

Network Information System maps are created from source files. On the master server where the maps are created, it is a good practice to separate the source maps from the distribution maps in the `/etc/` directory. Choose an alternate location, maybe `/usr/<domain-name>/maps`, and change the variable `PWDIR` to this new location in the `/var/yp/Makefile`.

NIS maps are built using the `make` utility. `Make` reads a file called `/var/yp/Makefile` which contains macros and other instructions for creating *targets*, which are the final maps. When adding a new map, the name of the map file must be entered into the `Makefile` at the end of the `all:` target list.

The commands to build NIS maps are:

```
# /var/yp
# make all
```

### Given an existing network using the NIS name service, list commands to update and propagate an NIS map

To change maps, first edit the files in the `/etc/` directory on the master server. Then run `make` in the `/var/yp` directory to compile these changes into a map file. To pull the map files to slaves, you must run the `ypxfr` command. For example, to retrieve an updated `ethers.byaddr` file, run:

```
# /usr/lib/netsvc/yp/ypxfr ethers.byaddr
```

Sun provides a few scripts to propagate maps for certain intervals. These can be scheduled to run in `cron`. Examine, for example:

```
/usr/lib/netsvc/yp/ypxfr_1perhour
```

or

```
/usr/lib/netsvc/yp/ypxfr_2perday
```



## Role-Based Access Control (RBAC)

### State the purpose of the Role-Based Access Control (RBAC) with regard to Solaris security

Role-Based Access Control can be thought of as a way to delegate system tasks to a combination of designated users and groups. The traditional UNIX model is one of a single computer system that shares its resources among multiple users. However, the management of the system is left to a single 'superuser' because the rights of this special account give access to the entire system. This could lead to problems of misuse or simply misunderstanding.

The RBAC system allows a subset of tasks that fall under 'root' access to be granted to the user community, in the hopes that savvy users can correct their own problems, and daily administrative tasks can be off-loaded by the (usually) very busy administrator.

*Privileges* are assigned to users and groups through roles. This role is really treated as a special account by the system. In fact, users, while in a role, execute commands in a special shell.

*Authorizations*, as they are in NIS, are assigned rights that grant access to a particular system function.

*Execution profiles* are a method of grouping commands and assigning them to users and roles.

### Select the statements that describe RBAC database features

The main features of RBAC lie in the four ASCII database files that make it up.

`/etc/user_attr` is the extended user attributes database. The file contains users and roles with authorizations and execution profiles.

In order to support roles, this file extends the `/etc/passwd` and `/etc/shadow` files by assigning the role to a system user. The fields of this file look like:  
username:::attribute

where `username` is the same as in the `/etc/passwd` file, and `attributes` are optional key-value pairs that may be any of the keywords `auths`, `profiles`, `roles` and `type`. The colons delimit each field; note the empty fields are not in use by Sun currently.

#### Example:

```
kchiotis:::type=normal;auths=solaris.system.date;roles=sysadmin;
profiles=All
```

`/etc/security/auth_attr` is the authorization attributes database. All system authorizations and their attributes are listed here.

Programs determine if the appropriate rights are set by checking authorizations in the file. They may be assigned to users or execution profiles. The file entries look like:

```
authname:::short-description:long-description:attributes
```





where `authname` is a short, unique string that defines the auth-type in the form `prefix.suffix`. Sun has used `solaris` as the prefix in all of its implementations. Short description is meant to be displayed in a GUI titlebar, long description is a helpful sentence describing the authorization, and the attributes is a key-value pair, most often the `help` keyword.

**Example:**

```
solaris.device.config::Configure Device Attribs::help=DevConfig.html
```

*note* that this defined authorization would be assigned to the user using the 'auths' keyword in the `/etc/user_attr` file.

`/etc/security/prof_attr` is the execution profile attributes database. Profiles on the system are defined here. Each profile has an associated authorization and help file.

Usually, the attribute just sets the UID to root (0).

```
profname::description:attribute
```

where `profname` is the case-sensitive profile name. The description is a definition of the purpose of the profile. The attributes is a key-value pair for the object; right now it can only be `help` or `auths`.

**Example:**

```
Device Management::Configure New \  
Devices:auths=solaris.device.*;help=DevMgmt.html
```

`/etc/security/exec_attr` is the profile execution attributes database. This file is where each profile is linked to its delegated, privileged operation.

The execution attributes file is where the commands that are run when a user or role is administering the system are defined.

```
name:policy:type::id:attribute
```

where `name` is the name of the associated profile. Policy -- `suser` -- is the only valid instance at this time. Type -- `cmd` -- is the only valid type at this time. Id is a string value that represents the commands to be run, and can be a script that executes a succession of commands. Attribute is a key-value pair that assigns security attributes to the executing commands. Attributes can be `eid`, `uid`, `egid` or `gid`. Specifying `eid` or `uid` is the same as having `setuid` applied. Specifying `egid` or `gid` is the same as having `setgid` applied.

**Example:**

```
Printer Management:suser:cmd::/usr/lib/lp/lpsched:euid=0  
Printer Management:suser:cmd::/usr/lib/lp/lpshut:euid=0  
Printer Management:suser:cmd::/usr/sbin/lpadmin:euid=0  
Printer Management:suser:cmd::/usr/sbin/lpsystem:euid=0
```

where `Printer Management` is an assigned profile in the `prof_attr` database:

```
Printer Management::Manage System Printing:help=Printmgt.html
```



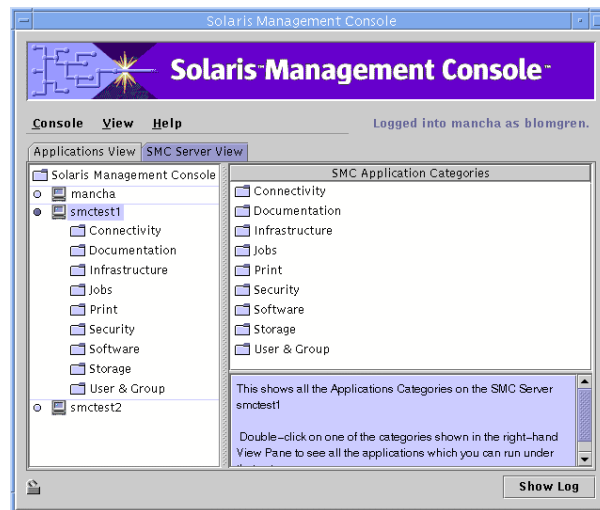


## Solaris Management Console and Solstice AdminSuite

### List the features of the Solaris Management Console

The Solaris™ management console lets administrators view and change attributes of systems on the network from any single console station. Applications can also be run and viewed from one computer to another.

- *Centralized Administration:* Any Solaris administration tools can be integrated and run from one location
- *Centralized Management:* Any server on a network can be managed centrally
- *Single Login:* Once a user is logged on using the SMC, credentials do not need to be provided to subsequent applications.
- *Instant access* to administration tools by running existing Solaris administration tools
- *Secure communication* via Secure Sockets Layer (SSL) support



### List the features of the Solaris AdminSuite

Solstice™ AdminSuite™ is a collection of GUI tools and commands used to perform administrative tasks such as managing hosts, users, groups, system files, printers, disks, file systems, terminals and modems. These tools and commands provide a graphical interface to the Solaris™ command-line tasks. When using AdminSuite™, system files are automatically adjusted to eliminate manual entry errors. AdminSuite™ can also manage hosts remotely.



## JumpStart-Automatic Installation

### State the purpose of the JumpStart server

Jumpstart™ is built into Solaris™ 8 to allow administrators to quickly and consistently install or upgrade the operating system on new or existing Sun computers.

Custom Jumpstart™ is a method to automatically install groups of identical systems. To customize Jumpstart™, a text file called *rules* must be created that lists one or more *profiles*. A *profile* is a text file that defines how Solaris™ is to be installed on a group of systems. Once these files are completed, they are validated using the *check* script. In a non-networked environment, the validated files are placed on a diskette in the *jumpstart* directory and the system is booted. Then the appropriate profile is selected to direct the installation of Solaris™. In a networked environment, the *jumpstart* directory is located on a network server.

### Identify the main components of a Jumpstart server

Jumpstart™ requires the presence of certain servers for the process to work smoothly. The main components of these servers for setting up networks for automatic install are:

- Boot and client identification services.
  - The boot services answer RARP queries and serve files via TFTP
  - The boot server answers the questions as if a user were answering them using interactive mode
- Installation services
  - Reside on a CDRom-equipped *install server* somewhere on the network. The OS CD is either copied to the server's hard disk or mounted and shared from the CD-ROM drive.



## Sun Solaris 8 Certified Systems Administration II

- If the *install server* is on a different subnet than the systems being installed, a *boot server* is required to boot the systems.
- It is a good idea to copy the OS to the disk, since the first CD of the two CD Solaris™ 8 set contains only Core and End User clusters.
- Configuration Services
  - Contains information about filesystems, partition sizes, and packages to install.
  - Class and rules files will reside here.
  - If appropriate, the Solstice™ Host Manager is used to add network information about the systems being installed to NIS or NIS+. Or, instead, the `add_install_client` is used to add network information to the `/etc` files of the install or boot server.
  - If the systems being installed are diskless or Autoclients, an OS server will be used to provide the Solaris™ operating system.

Remember, any of the four default install methods for Solaris™ (and not just JumpStart™) can be used when installing over the network. A single server provide boot, install and configuration services for Jumpstart™.

### List the parameters possible when using the `add_install_client` script syntax

Boot and install servers need clients added to them. Adding an install client involves running the `add_install_client` script.

```
# ./add_install_client -e Ethernet -i ip_addr -s install_svr:/dist \  
-c config_svr:/config_dir -p config_svr:/config_dir client_name \  
client_arch
```

-e specifies the Ethernet address

-i specifies the IP address of the client

-s specifies the name of the install server and path to the install distribution

-c specifies the configuration server and the path to the config directory

-p specifies the path to the sysidcfg file

### State the purpose of the boot service function on a subnet

The boot server contains information that the clients need to boot and contact other installation and configuration servers that exist on the network. Boot servers must be on the same subnet as the clients (because of the ARP nature of the clients' requests).

### Identify the events that occur during the JumpStart Client boot sequence

When a client boots, it sends out a RARP packet that contains its MAC address. The boot server, which is on the same subnet as the client, will receive this message and match it to an entry in its `/etc/ethers` file. A valid installation IP address is returned to the client.



The client will then issue a TFTP request to obtain boot information from the server. The request will be a returned symlink that will correspond with the boot program appropriate to its architecture. Once the client has obtained the boot program, it executes it.

As the boot program runs, it sends out a `whoami` request that contains its hostname. The boot server running `rpc.bootparamd` will look up the host name in `/etc/bootparam` and respond to the client with the location of its root and swap space.

Once the client receives the boot parameters and the root filesystem is mounted, the kernel is loaded and `init` begins and the client is redirected to the configuration server, where it runs `sysidtool` to determine where the installation files directory is and starts SunInstall™ to load the operating system.

### List the files necessary to support the JumpStart boot operation

There are several files associated with the boot server.

`/etc/ethers` contains the MAC addresses of each client on the network. A lookup in this file is performed when a client RARP request is received. Without this information, the clients will not be able to obtain a valid IP address.

```
08:00:a5:2d:90:3d snarg
```

`/etc/hosts` contains the IP address to be associated with a client computer.

```
192.168.10.15 snarg
```

`/tftpboot` is the directory that contains the binary files that contain the boot program for a specific architecture of client. It is created when the `add_install_client` script is run. For example, `sun4u` would look for a file called `inetboot.SUN4x.Solaris_8-1`. If the files in `/tftpboot` do not exist, the clients will fail to boot and no error will be displayed.

`/etc/bootparams` contains client specific information for each client that is going to boot off the server. Information in the file is added via the `add_install_client` script. It would look like:

```
snarg
root=bserver:/export/install/Solaris_8/Tools/Boot
install=bserver:/export/install
boottype=:in
sysid_config=bserver:/export/config
install_config=server1:/export/config
rootopts=:rsize=32768
```

`/etc/dfs/dfstab` is a listing of the local file systems that are going to share the install files with clients. It is also populated using options from the `add_install_client` script.



### State the purpose of the `sysidcfg` file with and without name service support

Once the Jumpstart™ installation has begun, there must be a way to answer the questions posed during the configuration identification setup. This is handled through the use of a `sysidcfg` file on the server specific to the client. If a name service does exist on a network, information like locale can be provided by the name server. If there is no name service, the `sysidconfig` file will provide this information to the client.

A `sysidcfg` file looks similar to the following:

```
system_locale=en_us
timezone=US/Pacific
terminal=vt100
timeserver=10.10.15.20
name_service=NONE
root_password=MPAdmhCdb4q
network_interface=hme0 (protocol_ipv6=no netmask=255.255.255.0)
```

The file location may be specified by the `-p` argument of the `add_install_client` script.

### Select the statements that describe the steps necessary to set up a JumpStart install server system to provide the Solaris 8 release software necessary to install a new system

1. Log in as root.
2. Mount the Solaris™ CD either by inserting it in the CD-ROM drive or mounting it via *NFS* from another system.
3. Change directory to the *Solaris\_8/Tools* directory on the Solaris™ 8 CD 1 and execute the `setup_install_server -b` command to boot the software from the Solaris™ CD to hard disk.

```
# ./setup_install_server -b /export/install/
```

### State the purpose of the `add_to_install_server`, `modify_install_server` and `add_install_client` scripts

Since Solaris™ 8 now requires two CDs for the full installation, a second script called `add_to_install_server` must be run to copy the second CD into the `install` directory. It is run from the CD 2:

1. Change directory to the *Solaris\_8/Tools* directory on the Solaris™ 8 CD 2 and execute:

```
# ./add_to_install_server /export/install/
```

2. To add install clients, `cd` to the client directory and run `add_install_client` script



If the `add_to_install_server` script is not run, only Core and End User clusters will be available.

The `modify_install_server` script will enable the Webstart style of Solaris™ 8 installation. Why you would do this for a jumpstart™ install is beyond me, because Webstart will require interactive answers.

Run the `add_install_client` script similar to the following:

```
# ./add_install_client -s bserver:/export/install -c \  
bserver:/export/config -p bserver:/export/config snarg sun4u
```

**Given the appropriate software source, explain how to create a configuration server with a customized rules file and class files**

The custom JumpStart™ files are accessed on the configuration servers via NFS or DFS. Preparing a custom JumpStart™ directory and files consists of:

1. Designating the JumpStart™ directory.
2. Create a rule for each group of systems in the *rules* file using the appropriate keywords and syntax. Example rules file entries:

```
network 10.10.16.0  && ! model 'SUNW, Sun 4_50' - class_net16 -  
memsize 16-32      && arch sparc - class_admin_support -
```

The rules classify machines on the network.

3. Create class files to categorize all the machines on the network, and specify how each system will be installed using the appropriate keywords and syntax.
4. Create check scripts to verify that the rules and class files are valid. A `rules.ok` file is created when check scripts finish. The `rules.ok` file is what is actually read during a live install.
5. Create optional begin and finish scripts to allow for advanced configuration options, like disabling the 'power-save' feature of new systems.

Be able to recognize valid rule file and profile files.

**Given an NIS name service network environment, explain how to configure NIS name service support for JumpStart**

If NIS exists on the network, certain files required for Jumpstart™ installation can be built and distributed in the form of maps. These files include:

```
/etc/ethers  
/etc/hosts  
/etc/bootparams
```

Solaris™, SunInstall™, SunRay™, Jumpstart™, Admintool™, Solstice™ and DiskSuite™ are registered trademarks of Sun Microsystems, Inc.



Special thanks to  
[Matthew Kortas](#)  
for contributing this Cramsession. Please  
visit his site at  
<http://acm.cse.msu.edu/~kortasma>