

VDCF - Virtual Datacenter Control Framework for the Solaris™ Operating System

Quick Reference – How To

Version 4.2
22. December 2011

Copyright © 2005-2011 JomaSoft GmbH
All rights reserved.

Table of Contents

| | |
|---|----|
| 1 Introduction..... | 3 |
| 2 Quick Reference – VDCF base..... | 4 |
| 2.1 vdcfadm command..... | 4 |
| 2.2 cpool command..... | 5 |
| 2.3 nodecfg command..... | 6 |
| 2.4 console command..... | 6 |
| 2.5 config command..... | 7 |
| 2.6 serverconfig command..... | 8 |
| 2.7 build command..... | 9 |
| 2.8 flash command..... | 10 |
| 2.9 node command..... | 10 |
| 2.10 diskadm command..... | 11 |
| 2.11 dataset command..... | 12 |
| 2.12 patchadm command..... | 13 |
| 2.13 vpool command..... | 15 |
| 3 Quick Reference – VDCF vServer..... | 16 |
| 3.1 vsriver command..... | 16 |
| 3.2 zfsadm command..... | 19 |
| 4 Quick Reference – VDCF vServer enterprise extensions..... | 20 |
| 4.1 rcadm command (Resource Management)..... | 20 |
| 4.2 rcmon command (Resource Monitoring)..... | 21 |
| 4.3 hwmon command (Hardware Monitoring)..... | 21 |
| 4.4 hamon command (High Availability Monitoring)..... | 22 |
| 5 Quick Reference – VDCF LDom..... | 23 |
| 5.1 cdom command..... | 23 |
| 5.2 gdom command..... | 23 |
| 6 Solaris Builds..... | 25 |
| 6.1 Create a Solaris JumpStart Install and Boot Server..... | 25 |
| 6.2 Initial install of a node with a new Solaris Build (to create a flash archive)..... | 25 |
| 6.3 Create an installable build in VDCF for deployment on new nodes..... | 25 |
| 7 Node management..... | 26 |
| 7.1 Helper Commands..... | 26 |
| 7.2 Setup a new Compute Node..... | 26 |
| 8 vServer management..... | 27 |
| 8.1 Helper commands..... | 27 |
| 8.2 Create a new vServer..... | 27 |
| 9 LDom management..... | 28 |
| 9.1 Create a cdom..... | 28 |
| 9.2 Create a new gdom..... | 28 |
| 10 Sample Node / vServer configuration..... | 29 |

1 Introduction

This documentation describes the Virtual Datacenter Control Framework (VDCF) for the Solaris Operating System, Version 4.2 and contains the most important steps for VDCF administration and some helper scripts.

See these other documents for further information:

| | |
|--|--|
| <i>VDCF – Release Notes</i> | for details about new releases |
| <i>VDCF – Installation Guide</i> | for information about installing or upgrading |
| <i>VDCF Base – Administration Guide</i> | for information about VDCF base usage |
| <i>VDCF vServer – Administration Guide</i> | for information about VDCF vServer product usage |
| <i>VDCF LDom – Administration Guide</i> | for information about VDCF LDom product usage |
| <i>VDCF – Resource Management</i> | for information about VDCF Resource Management |
| <i>VDCF – Monitoring</i> | for information about VDCF Monitoring |

These and all other VDCF documents can be found at:

<http://www.jomasoft.ch/products/VDCF/docs/>

New and changed operations and arguments since VDCF 4.1 are marked **bold** and **green**.

2 Quick Reference – VDCF base

2.1 vdcfadm command

| | |
|--|--|
| vdcfadm -c show_log [follow] [tail=nn] | lists the content of the message log |
| vdcfadm -c show_audit [follow] [tail=nn] | lists the content of the audit log |
| vdcfadm -c clear_log [archive] | clears the message log |
| vdcfadm -c clear_audit [archive] | clears the audit log |
| vdcfadm -c show_version | shows the current VDCF version |
| vdcfadm -c show_config | shows the actual configuration |
| vdcfadm -c statistics | show VDCF statistics |
| vdcfadm -c clear_locks | clears eventually hung locks |
| vdcfadm -c dump_db | create dump files of current database |
| vdcfadm -c load_db date=<dump date> | load/initialize database from dump files |
| vdcfadm -c show_node node=<nodename> all | show client pkg version on nodes |
| vdcfadm -c update_node node=<nodename> all | update client pkg on nodes |

2.2 cpool command

```
cpool -c show      [ name=<compute pool name> ]  
  
cpool -c create    name=<compute pool name>  
                  comment=<comment>  
                  [ default ]  
                  [ node=<node name list> ]  
  
cpool -c set_default name=<compute pool name>  
  
cpool -c assign    name=<compute pool name>  
                  [ node=<node name list> ]  
  
cpool -c rename    name=<compute pool name>  
                  newname=<new pool name>  
  
cpool -c modify    name=<compute pool name>  
                  comment=<comment>  
  
cpool -c remove    name=<compute pool name>  
                  [ force ]  
  
cpool -c check     name=<compute pool name> | all
```

The following format rules apply to the below listed parameters:

```
lists    ::= < element,element,... >
```

2.3 nodecfg command

```
nodecfg -c discover      name=<node name>
                        [ hostname=<hostname> [ nonroot ] ]

nodecfg -c show          [ name=<node name> [ allif ] ]
                        [ cpool=<computepool name> ]

nodecfg -c show_profile [ profile=<platform profile> ]

nodecfg -c create_profile name=<node name>      (interactive)

nodecfg -c remove_profile profile=<platform profile>

nodecfg -c add           name=<node name>      (interactive)
                        profile=<platform profile>

nodecfg -c modify       name=<node name>
                        [ addgroup=<config group list> ]
                        [ remgroup=<config group list> ]
                        [ interface=<network interface>
                          speed=<network speed> ]
                        [ location=<physical location> |
                          serial=<serial no> |
                          hostid=<hostid> ]
                        [ comment=<comment> ]

nodecfg -c modify_net   name=<node name>
                        interface=<interface or ipmp group name>
                        [ ipaddr=<ip or hostname> ]
                        [ netmask=<netmask> ]
                        [ nettype=<MNGT|PUBL|...> ]

nodecfg -c remove      name=<node name>
```

The following format rules apply to the below listed parameters:

```
lists ::= < element,element,... >
```

2.4 console command

```
console -c show        [ node=<node name> ]

console -c add         node=<node name> (interactive)

console -c modify     node=<node name>
                      [ type=<console type> ]
                      [ user=<console user> ]
                      [ protocol=<protocol> ]
                      [ port=<port> ]
                      [ hostname=<hostname/IP> ]

console -c set_pwd    node=<node name>

console -c remove     node=<node name>
```

2.5 config command

```
config -c add      type=<config type>
                  name=<name>
                  <args ...>           depending on type

config -c modify  type=<config type>
                  name=<name>
                  <args ...>           depending on type

config -c remove  type=<config type>
                  name=<name>

config -c show    [ type=<config type>
                  [ name=<name> ] ]
```

Supported configuration types are:
COMMAND, DEFAULTROUTE, DNS, FILE, NTP, PKG, ROUTE, SCRIPT,
SCSI_VHCI, SERVICES

Type specific arguments:

```
type=COMMAND      command=<command with options>

type=DEFAULTROUTE ipaddr=<ip address of defaultrouter>

type=DNS          domain=<domain>
                  search=<search>
                  server=<server>

type=FILE         source=<file>
                  target=<directory or file>
                  owner=<fileowner>
                  mode=<filemode>

type=NTP          server=<serverlist>

type=PKG          pkgs=<pkg[,pkg,pkg]>
                  pkgdevice=<device>
                  [ options=<pkgadd options> ]

type=ROUTE        destination=<address[/prefix]>
                  gateway=<address>

type=SCRIPT       script=<script>

type=SCSI_VHCI    provider=<provider>
                  productid=<productid>

type=SERVICES     [ enable=<servicelist> ]
                  [ disable=<servicelist> ]
```

2.6 serverconfig command

```
serverconfig -c list      default | all | groups | servers  
                          [ type=<config type> ]  
  
serverconfig -c show     default | group=<config group> |  
                          server=<node or vserver>  
                          [ type=<config type> ]  
  
serverconfig -c show_members   group=<config group>  
  
serverconfig -c add      type=<config type>  
                          name=<baseconfig name>  
                          [ server=<node or vserver> ]  
                          [ group=<group> ]  
                          [ section=<section> ]  
  
serverconfig -c modify   type=<config type>  
                          name=<baseconfig name>  
                          section=<section>  
                          [ default | group=<config group> |  
                          server=<node or vserver> ]  
  
serverconfig -c remove   type=<config type>  
                          name=<baseconfig name>  
                          default | group=<config group> |  
                          server=<node or vserver>  
  
serverconfig -c create_group  
                          supergroup=<group>  
                          subgroups=<group,group,...>  
  
serverconfig -c modify_group  
                          supergroup=<group>  
                          subgroups=<group,group,...>  
  
serverconfig -c remove_group  
                          supergroup=<group>  
  
serverconfig -c exec      command=<command>  
                          server=<comma sep list> |  
                          servergroup=<config group> |  
                          serverfile=<abs. path to file>  
  
serverconfig -c exec      type=<COMMAND|SCRIPT|FILE|PKG|SERVICES>  
                          name=<config name>  
                          server=<comma sep list> |  
                          servergroup=<config group> |  
                          serverfile=<abs. path to file>  
  
serverconfig -c exec      group=<config group>  
                          server=<comma sep list> |  
                          serverfile=<abs. path to file>
```

Supported configuration types are:
COMMAND, DEFAULTROUTE, DNS, FILE, NTP, PKG, ROUTE, SCRIPT,
SCSI_VHCI, SERVICES

2.7 build command

```
build -c enable_install  hostname=<hostname>
                        macaddr=<mac address>
                        netmask=<netmask>
                        architecture=<kernel architecture>
                        install_server=<solaris install image>
                        [ be_name=<boot environment name> ]
                        [ profile=<profile name> ]

build -c add_bootserver  install_server=<solaris install image>
                        boot_server=<bootserver directory>

build -c create          version=<build version>
                        boot_server=<bootserver directory>
                        archive=<archive location>
                        [ architecture=<kernel architecture> ]

build -c update_archive  version=<build version>
                        archive=<archive location>
                        [ architecture=<kernel architecture> ]

build -c show            [ version=<build version> ]

build -c remove          version=<build version>
```

2.8 flash command

```
flash -c enable_install node=<node name>  
          [ version=<FLASH version> ]  
  
flash -c disable_install node=<node name>  
  
flash -c list_active [ node=<node name> ]
```

2.9 node command

```
node -c update      name=<node name> | all  
  
node -c inactivate name=<node name>  
          [ force ]  
  
node -c activate   name=<node name>  
          [ force ]  
  
node -c show       [ name=<node name> [ verbose ] [ allif ] ]  
          [ cpool=<computepool name> ]  
  
node -c install    name=<node name>  
          [ force ]  
  
node -c console    name=<node name>  
          [ escape=<escape character> ]  
  
node -c remove     name=<node name>  
          [ force ]  
  
node -c boot       name=<node name>  
  
node -c reboot     name=<node name>  
          [ force ]  
  
node -c shutdown   name=<node name>  
          [ force ]  
  
node -c evacuate   name=<node name>  
          [ upgrade ]  
          [ force ]  
  
node -c register   name=<node name>  
          build=<build version>
```

2.10 diskadm command

```
diskadm -c show      [ all | free ]  
                    [ comment ]  
diskadm -c show      name=<GUID>  
diskadm -c show      dataset=<dataset-name>  
diskadm -c show      node=<node-name> [ all | free | inuse ]  
                    [ comment ]  
  
diskadm -c register  node=<node-name> |  
                    cpool=<compute pool name> | all  
                    [ methods=<method-list> ]  
                    [ scan ]  
                    [ full | new ]  
  
diskadm -c mark      name=<GUID-list> foreign  
  
diskadm -c mark      name=<GUID-list> usable  
  
diskadm -c modify     name=<GUID-list>  
                    comment=<"comment"> |  
                    remove_comment  
  
diskadm -c deregister node=<node-name>  
                    name=<GUID-list> | all  
  
diskadm -c update  
  
diskadm -c remove     name=<GUID-list>
```

The following format rules apply to the below listed parameters:
lists ::= < element,element,... >

2.11 dataset command

```
dataset -c create name=<dset name>
                vservers=<vservers name>
                [ type=<DISKSET|ZPOOL|RAW|VXVM> ]
                [ globalname ]
                [ delegated ]
                layout=<layout description>

dataset -c create name=<dset name>
                node=<node name>
                [ type=<DISKSET|ZPOOL|VXVM> ]
                layout=<layout description>

dataset -c remove name=<dset name>
                [ force ]

dataset -c add name=<dset name>
                layout=<layout description>

dataset -c attach_mirror name=<dset name>
                layout=<layout description>
                [ force ]

dataset -c detach_mirror name=<dset name>
                mirror=<mirror> (i.e. 1st,2nd,3rd,4th)

dataset -c revert name=<dset name>

dataset -c commit name=<dset name>
                [ force ]

dataset -c show [ name=<dset name> [ verbose ] ]
                [ all | node=<node name> ]

dataset -c detach name=<dset name>
                [ force ]

dataset -c attach name=<dset name>
                [ node=<node name> ]
                [ force ]

dataset -c assign name=<dset name>
                vservers=<new vservers name>

dataset -c update name=<dset name>
```

2.12 patchadm command

```
patchadm -c create_set      name=<patch-set name>
                             node=<node name> to=<date>

patchadm -c create_set      name=<patch-set name>
                             file=<patch order file>

patchadm -c create_set      name=<patch-set name>
                             platform=<platform>
                             [ from=<date> to=<date>      ]

patchadm -c delete_set      name=<patch-set name>

patchadm -c modify_set      name=<patch-set name>
                             [ add    patches=<patch list> ]
                             [ delete patches=<patch list> ]

patchadm -c create_target   name=<target name>
                             desc=<description>
                             filter=<node-filter-spec>
                             [ patchset=<patch-set list> ]

patchadm -c delete_target   name=<target name>

patchadm -c modify_target   name=<target name> [ rescan ]
                             [ filter=<node-filter-spec> ]
                             [ add    patchset=<patch-set list> ]
                             [ delete patchset=<patch-set list> ]
                             [ add    node=<node list> ]
                             [ delete node=<node list> ]

patchadm -c show            [ id=<patch-id> | verbose ]

patchadm -c show_set        [ name=<set name> ]

patchadm -c show_target     [ name=<target name> ]
                             [ verbose           ]

patchadm -c show_node       node=<server name> | all
                             [ name=<set name> |
                             patchlevel ]

patchadm -c show_level      [ cpool=<compute pool name> ]

patchadm -c diff            server=<vserver1,node2>
```



```
patchadm -c analyze      node=<node name> | all  
                        [ localhost ] [ showonly ]  
  
patchadm -c check       node=<node name> | all  
  
patchadm -c download  
  
patchadm -c import      [ spool=<patch spool directory> ]  
  
patchadm -c prepare     target=<patch target>  
                        [ force ]  
  
patchadm -c install     target=<patch target>  
                        [ reboot ]  
                        [ force ]  
  
patchadm -c credentials  show |  
                          set=oracle|proxy |  
                          remove=oracle|proxy
```

The following format rules apply to the below listed parameters:

```
platform ::= < sparc | i386 >  
date      ::= < YYYY-MM-DD >  
lists     ::= < element,element,... >  
filter    ::= node:<platform>  
           node:<node list>  
           build:<build-version list>
```

2.13 vpool command

```
vpool -c show      [ name=<vPool name>
                  [ vservers | gdoms | nodes ] ]
                  [ user=<user name> ]
                  [ vserver=<vServer name> |
                    gdom=<Guest Domain name> |
                    node=<Physical node name> ]

vpool -c create    name=<vPool name>
                  comment=<"comment">
                  [ vserver=<vServer name list> |
                    gdom=<Guest Domain name list |
                    node=<Physical node name list> ]
                  [ user=<user name list> ]

vpool -c modify    name=<vPool name>
                  [ newname=<new vPool name> ]
                  [ comment=<comment> ]

vpool -c remove    name=<vPool name>
                  [ force ]

vpool -c add_user  name=<vPool name list>
                  user=<user name list>

vpool -c remove_user name=<vPool name list>
                  user=<user name list>

vpool -c add_vserver name=<vPool name list>
                  [ vserver=<vServer name list> |
                    cpool=<cPool name list> ]

vpool -c remove_vserver
                  name=<vPool name list>
                  [ vserver=<vServer name list> |
                    cpool=<cPool name list> ]

vpool -c add_gdom  name=<vPool name list>
                  [ gdom=<Guest Domain name list> |
                    cpool=<cPool name list> ]

vpool -c remove_gdom
                  name=<vPool name list>
                  [ gdom=<Guest Domain name list> |
                    cpool=<cPool name list> ]

vpool -c add_node  name=<vPool name list>
                  [ node=<Node name list> |
                    cpool=<cPool name list> ]

vpool -c remove_node
                  name=<vPool name list>
                  [ node=<Node name list> |
                    cpool=<cPool name list> ]
```

The following format rules apply to the below listed parameters:

```
lists ::= < element,element,... >
```

3 Quick Reference – VDCF vServer

3.1 vserver command

```
vserver -c create  name=<vServer name>
                  node=<physical node name>
                  [ type=<FULL|SPARSE|SOL8|SOL9> ]
                  [ sgroup=<server group> ]
                  [ comment=<"comment"> ]
                  [ vpool=<vPool name list> ]
                  [ priority=<integer, lower is more important> ]
                  [ category=<category name> ]
                  [ hostid=<hostid> ]

vserver -c remove name=<vServer name>
                  [ force ]

vserver -c addfs  [ type=data ]
                  name=<vServer name>
                  [ dataset=<dataset name> ]
                  mountpoint=</directory>
                  [ size=<size> ]
                  [ options=<mount options> ]

vserver -c addfs  type=root
                  name=<vserver name>
                  [ dataset=<dataset name> | local ]
                  [ size=<size> ]
                  [ options=<mount options> ]

vserver -c addfs  type=lofs
                  name=<vserver name>
                  directory=</directory>
                  mountpoint=</directory>
                  [ options=<mount options> ]

vserver -c growfs name=<vserver name>
                  mountpoint=</directory | root>
                  [ size=<size> ]

vserver -c mount  name=<vserver name>
                  mountpoint=</directory> |
                  dataset=<dataset name>

vserver -c unmount name=<vserver name>
                  mountpoint=</directory> |
                  dataset=<dataset name>

vserver -c renamefs name=<vserver name>
                  mountpoint=</directory>
                  to=</newdirectory>
                  [ remount [ commit | force ] ]

vserver -c remfs  name=<vserver name>
                  mountpoint=</directory | root | all>
```

```
vserver -c addnet  name=<vserver name>
                  type=<management|public|backup>
                  ipaddr=<ip address | hostname>
                  [ netmask=<network mask> ]
                  [ vlan=<vid> ]
                  [ stack=<shared|private|exclusive> ]

vserver -c remnet  name=<vserver name>
                  [ type=<management|public|backup|all> ]
                  [ ipaddr=<ip address | hostname> ]

vserver -c revert  name=<vserver name>
                  mountpoint=</directory | root | all>

vserver -c modify  name=<vserver name>
                  [ comment=<comment> ]
                  [ addgroup=<config group list> ]
                  [ remgroup=<config group list> ]
                  [ priority=<integer, lower is more important> ]
                  [ category=<category name> ]
                  [ hostid=<hostid> | clear_hostid ]

vserver -c commit  name=<vserver name>
                  [ boot ]
                  [ exec ]
                  [ remove ]
                  [ uninstall ]

vserver -c migrate name=<vserver name>
                  node=<new target node>
                  [ shutdown ]
                  [ upgrade [ full ] ]
                  [ noboot ]
                  [ nocheck ]
                  [ force ]

vserver -c migrate source=<source node>
                  node=<new target node>
                  [ shutdown ]
                  [ upgrade [ full ] ]
                  [ noboot ]
                  [ all ]
                  [ nocheck ]
                  [ force ]

vserver -c detach  name=<vserver name list>
                  [ force ]
                  [ shutdown ]

vserver -c detach  node=<node name>
                  [ force ]
                  [ shutdown ]

vserver -c attach  name=<vserver name list>
                  [ node=<new target node> ]
                  [ nocheck ]
                  [ force ]
                  [ upgrade [ full ] ]
                  [ boot ]
```

```
vserver -c reattach name=<vServer list> |
                    node=<node name> |
                    cdom=<control domain>
                    [ nocheck ]
                    [ force ]
                    [ upgrade [ full ] ]
                    [ boot ]

vserver -c show [ node=<node name> |
                cpool=<computepool name> |
                cdom=<control domain> ]
                [ all ]

vserver -c show name=<vServer name>
                [ verbose |
                candidates ]

vserver -c boot name=<vserver name list> |
                node=<node name> |
                cdom=<control domain>

vserver -c reboot name=<vserver name list> |
                node=<node name> |
                cdom=<control domain>

vserver -c shutdown name=<vserver name list> |
                    node=<node name> |
                    cdom=<control domain>
                    [ halt ]

vserver -c console name=<vserver name>
                  [ escape=<escape char> ]
```

3.2 zfsadm command

```
zfsadm -c show      vserver=<vServer name>
                   [ snapshots | all ]

zfsadm -c snapshot vserver=<vServer name>
                   filesystem=<filesystem name> |
                   mountpoint=</directory>
                   snapshot=<snapshot name>
                   [ recursive | rec ]

zfsadm -c rollback vserver=<vServer name>
                   snapshot=<snapshot name>
                   [ filesystem=<filesystem name> |
                   mountpoint=</directory> ]
                   [ recursive | rec ]
                   [ recursive_all | recall ]
                   [ force ]

zfsadm -c destroy  vserver=<vServer name>
                   snapshot=<snapshot name>
                   [ filesystem=<filesystem name> |
                   mountpoint=</directory> ]
                   [ recursive | rec ]
                   [ recursive_all | recall ]
                   [ force ]

zfsadm -c get      vserver=<vServer name>
                   filesystem=<filesystem name> |
                   mountpoint=</directory>
                   [ props=<property list> ]

zfsadm -c set      vserver=<vServer name>
                   filesystem=<filesystem name> |
                   mountpoint=</directory>
                   props=<property list>
```

The following format rules apply to the below listed parameters:

```
lists    ::= < element,element,... >
date     ::= <YYYY-MM-DD>
```

4 Quick Reference – VDCF vServer enterprise extensions

4.1 rcadm command (Resource Management)

```
rcadm -c show      [ name=<vServer or node name> ]
                  [ vservers=<vServer> ]

rcadm -c show_perf { node | cpu }

rcadm -c statistics

rcadm -c set       help

rcadm -c set       vservers=<vServer list>
                  <property>=<property value>
                  [ force ]

rcadm -c unset     vservers=<vServer list>
                  props=<property list>

rcadm -c revert    vservers=<vServer list>

rcadm -c remove    vservers=<vServer list>

rcadm -c clone     vservers=<vServer list>
                  template=<vServer>

rcadm -c commit    vservers=<vServer list>
                  [ push ]
                  [ force ]
```

Properties are defined as follows:

CPU_Shares/CPU_cap and CPUs/Importance Properties are mutually exclusive

| | |
|--------------|--|
| 'CPU_Shares' | Number of Base Units. If CPU_Shares are defined 'CPUs' and 'Importance' are not allowed and FSS based RM is activated |
| 'CPU_cap' | CPU capping in Base Units. 'CPU_Shares' and 'CPU_cap' can be but must not be specified together. They indicate a guaranteed and a maximum CPU entitlement. |
| -- or -- | |
| 'CPUs' | Number of CPUs. Requires 'Importance'. |
| 'Importance' | Relative importance of temp pool |

General Properties with no additional dependencies

| | |
|------------|---|
| 'RAM' | Physical RAM in K,M,G,T |
| 'SWAP' | Virtual Memory in K,M,G,T |
| 'Locked' | Maximum locked down Memory in K,M,G,T |
| 'LWP' | Maximum number of LWPs |
| 'MSG_ids' | Maximum number of Message Queues |
| 'SEM_ids' | Maximum number of Semaphores |
| 'SHM_ids' | Maximum number of Shared Memory Segments |
| 'SHM_Size' | Maximum size of all Shared Memory Segments in K,M,G,T |

Sizes are specified as n, n[bB], n[kK], n[mM], n[gG], n[tT] where n is megabytes. The minimum values are: 1048576b, 1024k, 1 or 1m, 1g, 1t.

Properties are not case sensitive!

4.2 rcmon command (Resource Monitoring)

```
rcmon -c status      [ verbose ] [ node ]

rcmon -c enable      aggregator | collector

rcmon -c enable      node=<node name> | node all

rcmon -c disable     aggregator | collector

rcmon -c disable     node=<node name> | node all

rcmon -c update      node=<node name> | node all

rcmon -c show        help

rcmon -c show        cpu | memory | memory_extended
                    hourly | daily | monthly | yearly
                    server=<server name>
                    [ verbose ]
                    [ gz_total | gzt ]

rcmon -c show        cpu | memory | memory_extended
                    from=<'time-spec'>
                    server=<server name>
                    [ to=<'time-spec'> ]
                    [ aggr=<aggr-spec> ]
                    [ verbose ]
                    [ gz_total | gzt ]

rcmon -c summary     [ node | vservice ]
                    [ cpool=<compute pool> ]
                    [ sortkey=server | cpu ]
                    [ asc ]
```

4.3 hwmon command (Hardware Monitoring)

```
hwmon -c enable

hwmon -c disable

hwmon -c status      [ verbose ]

hwmon -c show        [ node=<node name> ]
                    [ verbose ]

hwmon -c update      all | node=<node name>

hwmon -c show_locator node=<node name>

hwmon -c set_locator node=<node name>

hwmon -c clear_locator node=<node name>

hwmon -c clear_history node=<node name>

hwmon -c clear_state node=<node name>
```

4.4 hamon command (High Availability Monitoring)

```
hamon -c status
hamon -c show          [ node=<node name> ]
hamon -c show          [ on|off|fault|maint|susp|all ]
hamon -c enable        daemon
hamon -c enable        node=<node name>
hamon -c disable       daemon
hamon -c disable       node=<node name>
hamon -c suspend       node=<node name>
hamon -c resume        node=<node name>
hamon -c clear         node=<node name>
```

5 Quick Reference – VDCF LDom

5.1 cdom command

```
cdom -c create      name=<cdom name>
                   cpu=<no of virtual CPUs>
                   ram=<memory in K,M,G,T>
                   [ mau=<no of modular arithmetic units (MAU)> ]

cdom -c show        [ name=<cdom name> ]
                   [ verbose ]

cdom -c modify      name=<cdom name>
                   [ cpu=<no of virtual CPUs> ]
                   [ ram=<memory in K,M,G,T> ]
                   [ mau=<no of modular arithmetic units (MAU)> ]

cdom -c commit      name=<cdom name>
                   [ reboot ]
                   [ force ]

cdom -c remove      name=<cdom name>
```

5.2 gdom command

```
gdom -c create      name=<guest domain name>
                   cdom=<control domain name>
                   cpu=<no of virtual CPUs>
                   ram=<memory in K,M,G,T>
                   [ mau=<no of modular arithmetic units (MAU)> ]
                   [ vpool=<vPool name list> ]
                   [ profile=<partitioning profile> ]
                   [ comment=<"comment"> ]

gdom -c show        [ cdom=<control domain> |
                   cpool=<compute pool name> ]
                   [ all ]

gdom -c show        name=<guest domain name>
                   [ verbose |
                   candidates ]

gdom -c modify      name=<guest domain name>
                   [ cpu=<no of virtual CPUs> ]
                   [ ram=<memory in K,M,G,T> ]
                   [ mau=<no of modular arithmetic units (MAU)> ]
                   [ profile=<partitioning profile> ]
                   [ comment=<"comment"> ]

gdom -c revert      name=<guest domain name>

gdom -c adddisk     name=<guest domain name>
                   type=<root|data>
                   guids=<guid-list>

gdom -c remdisk     name=<guest domain name>
                   guids=<guid-list>
```

```
gdom -c addnet      name=<guest domain name>
                   type=<management|public|backup>
                   ipaddr=<ip address | hostname>
                   netmask=<network mask>
                   [ vlan=<vid> ]
                   [ ipmpgroup=<ipmp group name> | ipmp ]
                   [ probes=<test-ip | hostname,test-ip |
                     hostname,...> ]

gdom -c remnet     name=<guest domain name>
                   [ type=<management|public|backup> ]
                   [ ipaddr=<ip address | hostname> ]

gdom -c remove     name=<guest domain name>
                   [ force ]

gdom -c commit     name=<guest domain name>
                   [ install ]
                   [ remove ]

gdom -c install    name=<guest domain name>

gdom -c migrate    name=<guest domain name>
                   cdom=<target control domain>
                   live

gdom -c migrate    name=<guest domain name>
                   cdom=<target control domain>
                   [ shutdown ]
                   [ noboot ]

gdom -c detach     name=<guest domain list>
                   [ force ]
                   [ shutdown ]

gdom -c detach     cdom=<control domain name>
                   [ force ]
                   [ shutdown ]

gdom -c attach     name=<guest domain name>
                   [ cdom=<new target control domain> ]
                   [ boot ]

gdom -c reattach   name=<guest domain name> |
                   cdom=<control domain name>
                   [ boot ]

gdom -c boot       name=<guest domain list> |
                   cdom=<control domain name>

gdom -c reboot     name=<guest domain list> |
                   cdom=<control domain name>
                   [ force ]
                   [ stop ]

gdom -c shutdown   name=<guest domain list> |
                   cdom=<control domain name>
                   [ force ]
                   [ stop ]

gdom -c console    name=<guest domain name>
```

The following format rules apply to the below listed parameters:
lists ::= < element,element,... >

6 Solaris Builds

6.1 Create a Solaris JumpStart Install and Boot Server

```
# The install server is created from a Solaris DVD using the following command:
cd /cdrom/cdrom0/Solaris_10/Tools
setup_install_server <install-server-directory>

# Don't forget to share the install server directory for all:
share -F nfs -o ro,anon=0 <install-server-directory>

# Create a Boot Server for this Solaris Release using the VDCF build command
build -c add_bootserver install_server=<install-server-directory> \
boot_server=<boot-server-directory>
```

6.2 Initial install of a node with a new Solaris Build (to create a flash archive)

```
# prepare the install of a compute node with a new JumpStart build.profile
build -c enable_install hostname=nodexxx macaddr=0:8:20:x:x:x \
netmask=255.255.255.0 architecture=sun4u install_server=<install-server-directory>

# effectively install the node, enter on the nodes OBP prompt:
OK> boot net - install http://<mngt-ipaddr>/<nodexxx>.tar

# after installation of all required additional packages create
# a flash archive for reuse in VDCF
flarcreate -n <flashname> -S -c /var/tmp/build/<flashname>.flar
# flashname should represent the Solaris build and architecture:
# e.g. 5.10v_U4.flar
```

6.3 Create an installable build in VDCF for deployment on new nodes

```
# create a build with a locally saved archive:
build -c create version=<build-name> boot_server=<boot-server-directory> \
archive=/export/upload/<flashname>.flar
# --> VDCF will copy the flar into VDCF's flash directory

# create a build with a remote archive: (use nfs, http, ftp and IP address)
build -c create version=<build-name> boot_server=<boot-server-directory> \
archive=nfs://192.168.1.34/<flashname>.flar architecture=sun4u
# --> VDCF won't copy the flar
```

7 Node management

7.1 Helper Commands

```
# display defined system profiles
nodecfg -c show_profile

# display defined builds
build -c show

# display server configuration
serverconfig -c show server=<newnode>
```

7.2 Setup a new Compute Node

```
# discover the new system (system has to be up and running)
nodecfg -c discover name=<newnode> hostname=<newnode>

# only for first system of the same type create a system profile
nodecfg -c create_profile name=<newnode>

# add node configuration into VDCF database
nodecfg -c add name=<newnode> profile=<system-profile>

# add console
console -c add node=<newnode>

# map a predefined build to the new node
flash -c enable_install node=<newnode> version=<build-name>

# install the node
node -c install name=<newnode>
```

8 vServer management

8.1 Helper commands

```
# display defined nodes
node -c show

# display free disk devices (to copy Device-ID)
diskadm -c show free node=<targetnode>

# display serverconfigs and groups
serverconfig -c list all

# verify the vserver config
vserver -c show name=<newzone>
```

8.2 Create a new vServer

```
# create in VDCF database
vserver -c create name=<newzone> node=<targetnode> comment="<comment>"

# create dataset for vServer
dataset -c create name=root vserver=<newzone> layout=<Device_ID>

# create filesystem on dataset
vserver -c addfs name=<newzone> type=root

# add management network
vserver -c addnet name=<newzone> type=management ipaddr=<x.x.x.x>

# add public network
vserver -c addnet name=<newzone> type=public ipaddr=<newzone>

# add additional configuration groups to the vServer (optional)
vserver -c modify name=<newzone> addgroup=<config-groups>

# now initiate the installation, create the zone on the target node
vserver -c commit name=<newzone>

# boot the vServer to finish the zone setup
vserver -c boot name=<newzone>

# open the system console of the zone to see what's going on while booting:
vserver -c console name=<newzone>
```

9 LDom management

9.1 Create a cdom

For the prerequisite please consult the LDom Administration-Guide (3.1)

```
# create in VDCF database
cdom -c create name=<cdom name> cpu=<no of virtual CPUs> ram=<memory in K,M,G,T>

# now initiate the installation, create the cdom and reboot/activate it
cdom -c commit name=<cdom name> reboot
```

9.2 Create a new gdom

```
# create in VDCF database
gdom -c create name=<newgdom> cdom=<cdom name> cpu=<no of virtual CPUs> ram=<memory
in K,M,G,T> comment="Test gdom"

# modify partitioning.cfg if required
gdom -c modify name=<newgdom> profile=ldom_partitioning.cfg

# add root/boot disk to gdom
gdom -c adddisk name=<newgdom> type=root guids=<Device_ID>

# add management ip-address to gdom
gdom -c addnet name=<newgdom> type=management ipaddr=<x.x.x.x> netmask=<x.x.x.x>

# add public ip-address to gdom
gdom -c addnet name=<newgdom> type=public ipaddr=<newgdom> netmask=<x.x.x.x>

# enable required build version
build -c show

  Build Version  OS Version  Platform Arch  Method  Type  Build Name
      5.10u_U8   5.10 (U8)   sparc   sun4u  WAN    zfs   5.10_U8_SPARC
      5.10v_U6_P1 5.10 (U6+)  sparc   sun4v  STD    ufs   5.10_v6_P20060526_SPARC

flash -c enable_install node=<newgdom> version=5.10v_U6_P1

# now initiate the installation, create the gdom on the target cdom
gdom -c commit name=<newgdom> install

# watch console
gdom -c console name=<newgdom>
```

10 Sample Node / vServer configuration

```
# For non-Sun SAN storage, it is required to register the provider SCSI product id
# Example for an IBM storage:
config -c add type=SCSI_VHCI name=ds8300 provider=IBM productid=2107900
serverconfig -c add type=SCSI_VHCI name=ds8300 group=IBM

# enable SVM Solaris Volume Manager required services
config -c add type=SERVICES name=SVM enable=metainit,mdmonitor,meta
serverconfig -c add type=SERVICES name=SVM group=node

# Default route per Network
config -c add type=DEFAULTROUTE name=<yournet.name> ipaddr=<x.x.x.x>
serverconfig -c add type=DEFAULTROUTE name=<yournet.name> group=<yournet.name>

# DNS Settings
config -c add type=DNS name=prod server=<your-serverip> domain=<yourdomain.com> \
search=<yourdomain.com>
serverconfig -c add type=DNS name=prod

# enable FSS scheduler for every node
config -c add type=COMMAND name=FSS command="dispadmin -d FSS"
serverconfig -c add type=COMMAND name=FSS group=node

# Disable some insecure SMF services for all nodes / vServers
config -c add type=SERVICES name=hardening \
disable=telnet,sendmail,print/server,rstat
serverconfig -c add type=SERVICES name=hardening

# Define NTP servers
config -c add type=NTP name=prod server=<your-ntp-server>
serverconfig -c add type=NTP name=prod group=system-management

# add shell scripts to execute while installing node or vservers
# shell scripts must be saved in /var/opt/jomasoft/vdcf/config/script
config -c add type=SCRIPT name=add-users script=add_users.sh
serverconfig -c add type=SCRIPT name=add-users

# add files to be copied onto the node or vservers after setup
# files must be saved in /var/opt/jomasoft/vdcf/config/file
config -c add type=FILE name=root-profile source=std_profile
target=/root/.profile \
owner=root mode=444
serverconfig -c add type=FILE name=root-profile

# add packages to be installed onto the node or vservers after setup
# pkgs must be saved in /var/opt/jomasoft/vdcf/config/pkg
config -c add type=PKG name=xxx-prereq pkgs=all pkgdevice=xxx/xxx-prereq-sparc.pkg
serverconfig -c add type=PKG name=xxx-prereq group=node section=2
```