

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

VDCF LDom Product

Administration Guide

Version 1.2
3. September 2010

Copyright © 2005-2010 JomaSoft GmbH
All rights reserved.

Table of Contents

1 Introduction.....	3
1.1 Overview.....	4
1.1.1 Consolidation.....	4
1.1.2 Virtualization.....	5
1.1.3 Solaris Containers.....	6
1.1.4 Solaris Logical Domains.....	7
1.1.5 Datacenter Architecture.....	8
1.1.6 Virtual Datacenter Control Framework (VDCF)	9
1.1.7 VDCF terminology.....	10
1.2 Supported Environments.....	11
2 VDCF Product Structure and Commands.....	12
2.1 Product Structure.....	12
2.1.1 Commands and Manual Pages.....	12
2.1.2 VDCF Datastore and Configuration.....	12
2.1.3 Logfiles.....	12
2.2 Commands.....	13
3 Logical Domain Management.....	14
3.1 Overview.....	14
3.1.1 Control Domain (cdom).....	14
3.1.2 Guest Domain (gdom).....	14
3.2 Control domain (cdom).....	15
3.2.1 cdom definition.....	15
3.2.2 cdom creation.....	15
3.2.3 cdom remove.....	15
3.3 Guest domain (gdom) configuration.....	16
3.3.1 gdom initial definition.....	16
3.3.2 gdom modifications.....	16
3.3.3 Disks (LUN).....	16
3.3.4 Network.....	16
3.3.5 Summary.....	17
3.3.6 Installation.....	17
3.4 Guest domain (gdom) operation.....	18
3.5 Guest domain (gdom) migration.....	18
3.6 Guest domain (gdom) cleanup.....	18
3.7 Virtual pools (vPools) - Permission to manage Guest domains.....	19
4 Runtime States (Control and Guest Domains).....	20
4.1 Overview.....	20
4.2 Cronjob.....	20
5 Configuration States (CDom, GDom, Disk, Network).....	21
5.1 Overview.....	21
5.2 Possible state values.....	21
5.3 State values explained.....	21
5.4 GDom state diagram.....	22
5.5 Supported GDom commands.....	22
6 Security Management.....	24
6.1 Management Server RBAC.....	24
7 Appendixes.....	25
7.1 Data File Overview.....	25
7.1.1 On VDCF Management Server.....	25
7.1.2 On Compute Nodes.....	25
7.2 Customization of VDCF environment.....	26
8 Manpages.....	27

1 Introduction

This documentation describes the Virtual Datacenter Control Framework (VDCF) LDom product for the Solaris Operating System, Version 1.2 and explains how to use the product.

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 – Quick Reference</i>	for a short command overview
<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 – 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/>

1.1 Overview

Virtualization is an approach to IT that pools and shares resources so that utilization is optimized and supply automatically meets demand. The case for Virtualization is compelling: industry analysts estimate that the average utilization rate of a typical IT data-center's resources is between 15 and 20 percent.

With Virtualization, IT resources dynamically and automatically flow toward business demand, driving up utilization rates and aligning IT closely with business needs.

Pooling and sharing are at the heart of Virtualization. The logical functions of server, storage, network and software resources are separated from their physical constraints to create pooled resources. Business processes can share the same physical infrastructure. As a result, resources linked with one function, such as ERP, can be dynamically allocated to another, such as CRM, to handle peaks in demand. IT services can also be provided as a utility model, on a pay-per-use basis.

Virtualization is more than the implementation of technology. It's a new way of thinking about the IT infrastructure. To manage the environment as a whole, IT processes must be standardized and people educated on how to deliver service levels across a shared infrastructure.

1.1.1 Consolidation

In many data centers, a small number of servers carry the bulk of the workload, while others run vastly under utilized, consuming your energy, time and resources.

Therefore a growing number of users have become interested in improving the utilization of their compute resources through consolidation and aggregation. Consolidation is already common concept in mainframe environments, where technology to support running multiple applications and even operating systems on the same hardware has been in development since the late 1960's. Such technology is now becoming an important differentiator in other markets (such as Unix/Linux servers), both at the low end (virtual web hosting) and high end (traditional data center server consolidation).

Virtualization technologies can help you achieve full asset utilization by identifying under performing assets and enabling asset consolidation. Consolidation means fewer assets to own and manage which in turn lowers the asset TCO.

1.1.2 Virtualization

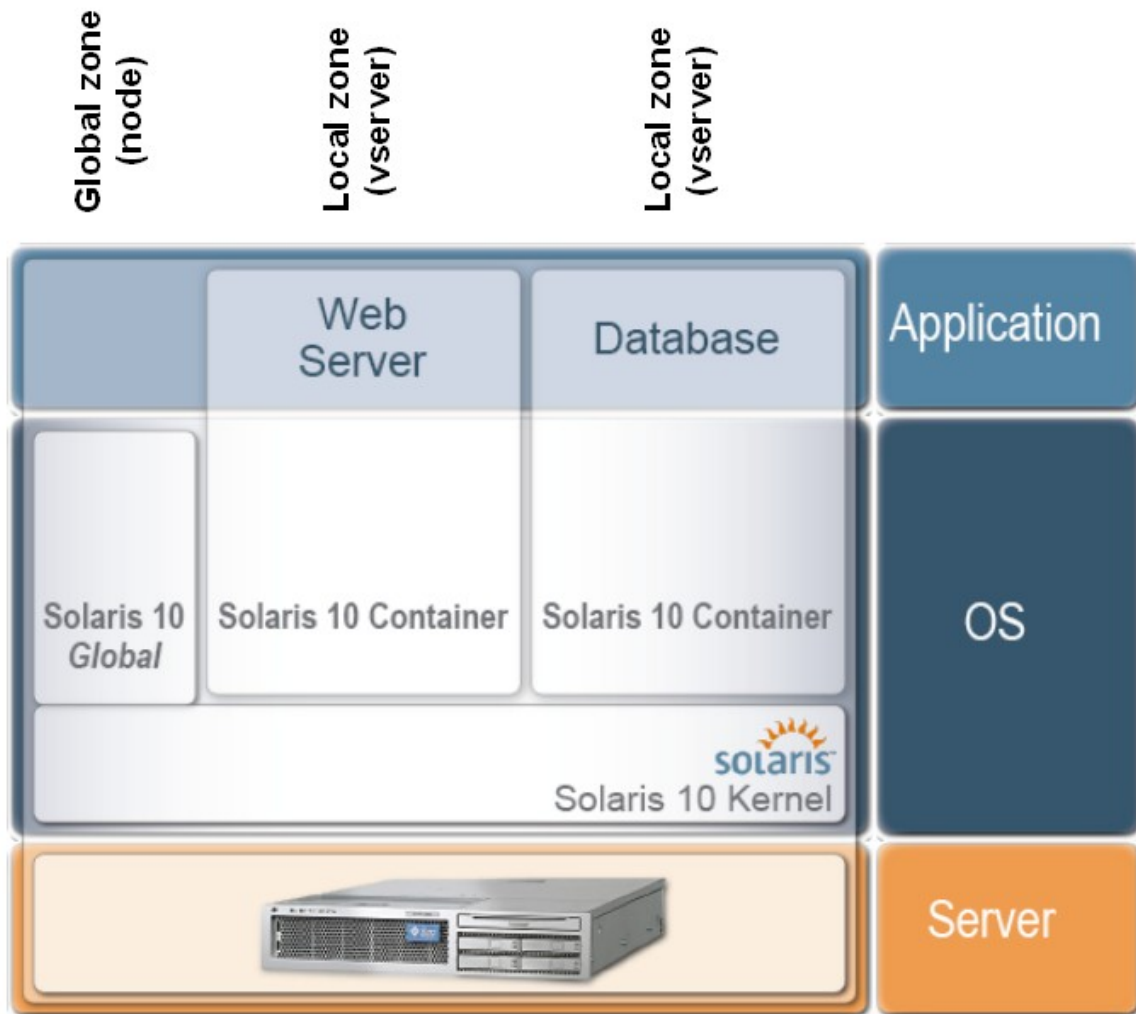
In computing terms, Virtualization is the creation of many digital abstractions that represent a real physical object.

So, in terms of servers, a virtual server may look like a single physical server to the end users and administrators. Each virtual server will be operate oblivious to the fact that it is sharing compute resources with other virtual servers. Virtual servers continue to provide the many benefits of their physical counterparts, only in a greatly reduced physical package.

Virtualization of the infrastructure addresses one of the most burning problems of today's data centers. It solves the dependencies between the numerous technology layers and creates transparency and flexibility. Resources will be administered in pools which are flexible to use and utilize.

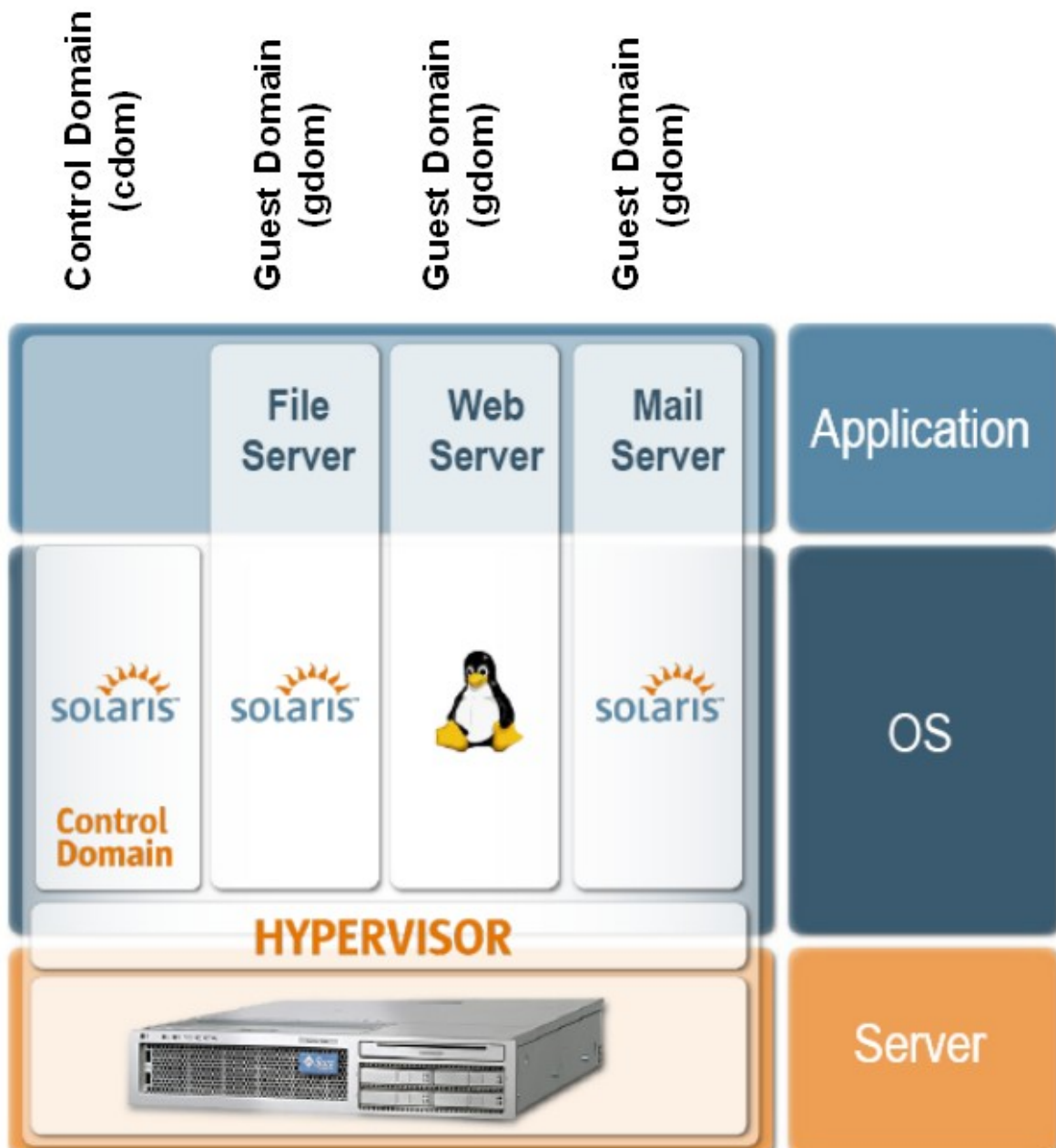
1.1.3 Solaris Containers

The VDCF vServer product builds on top of a Virtualization technology called Solaris Containers. A Solaris Container is logical abstraction of a Solaris application environment that can also reduce the overhead of administering multiple operating system instances. Each application running in a Container is isolated from what is happening in other Containers that may potentially be running within the same physical system. From an applications point of view, a Container looks exactly like a standard Solaris Operating Environment. VDCF manages both, the physical servers or nodes used in the form of a stateless carrier for Containers called vServers.



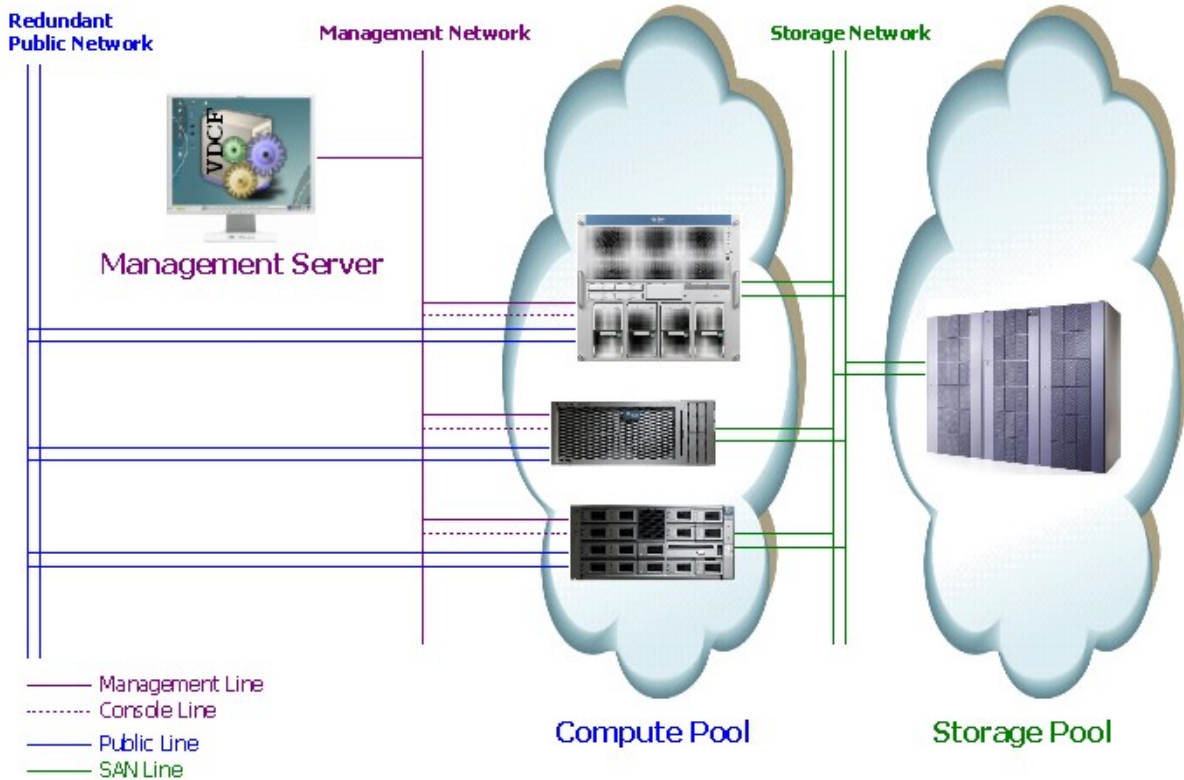
1.1.4 Solaris Logical Domains

The VDCF LDom product is based on another Oracle Virtualization technology called Oracle VM Server for SPARC (previously called Sun Logical Domains). A logical domain (LDM) is a full virtual machine that runs an independent operating system instance and contains virtualized CPU, memory, storage, console, and cryptographic devices. Within the logical domains architecture, the Hypervisor is a small firmware layer that provides a stable, virtualized machine architecture to which an operating system can be written. As such, each logical domain is completely isolated and may run different Solaris Operating Systems. On each LDM server there is one control domain which controls and servers the Guest Domains. Guest Domains may contain Solaris Containers. From an applications point of view, a Guest Domain looks like a standard Solaris Operating Environment. VDCF manages both, the control domain (cdom) and the guest domains (gdom).



1.1.5 Datacenter Architecture

Successful consolidation always relies on a standardized environment. VDCF follows a standard data center blueprint as a base to its architecture and design.



In the diagram above we show the generic data centers architecture complete with a management server, compute and storage pool. It also highlights the typical connections between the different entities. It separates management traffic from public and other data traffic. Data access is handled by the SAN and its associated fabrics and storage devices. A management server serves as a single point of control for the entire infrastructure.

Management Server

This system is the central operation cockpit to manage the Compute Server Pools. At a minimum it hosts the VDCF software but might be used for other system management products as well. The management server also serves as secure gateway to the Compute Pool infrastructure. It controls the access to the Compute Servers management interfaces and consoles.

Compute Pools

Services and applications run on virtual servers. Virtual servers are hosted on compute resources - servers - out of the compute pools.

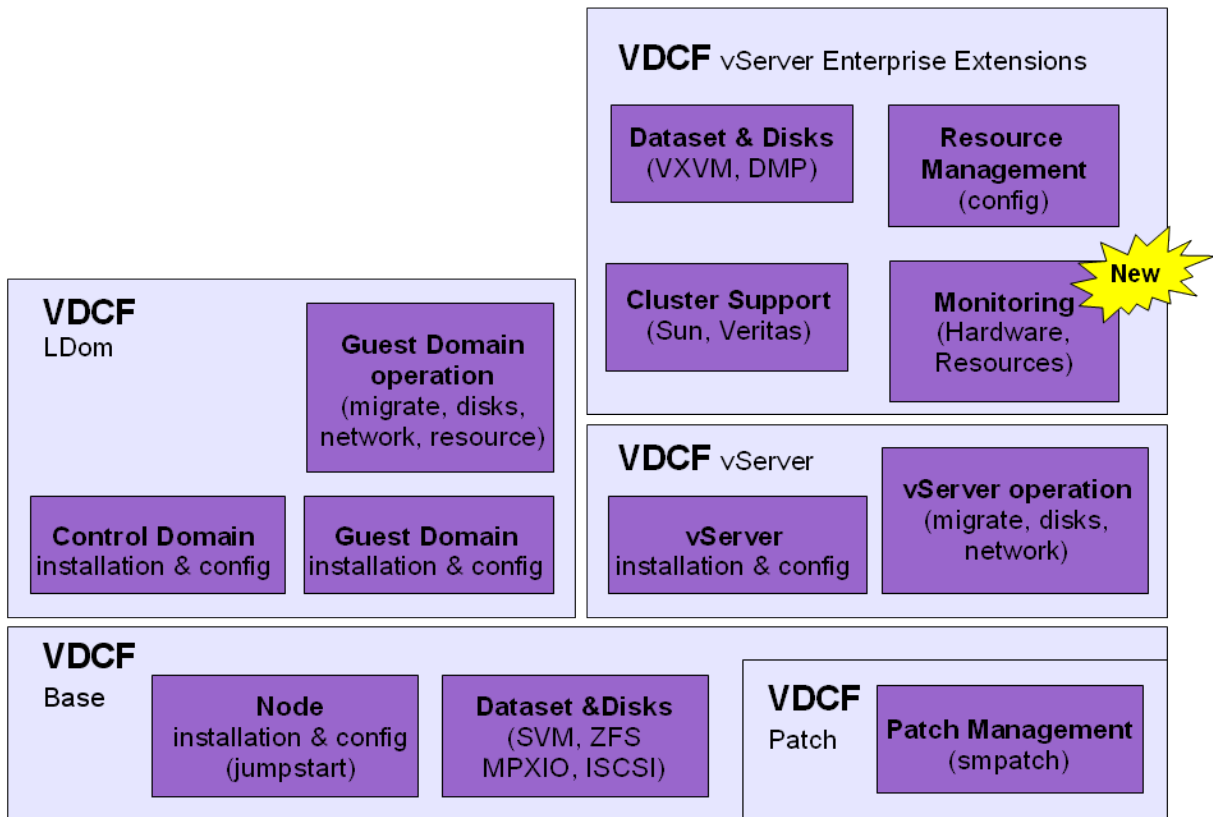
Storage Pool

Stateful data like a virtual servers root and data filesystems are stored on SAN storage. The SAN storage serves LUN's to the compute pool. These LUN's must be visible on all or at least a subset of the physical servers. The accessibility of these LUN's on multiple physical servers is what enables VDCF to control the compute pool Virtualization.

1.1.6 Virtual Datacenter Control Framework (VDCF)

VDCF is a platform management framework for the Solaris Operating System. VDCF allows you to run a virtualized data center using Solaris 10 Containers and/or Logical Domains controlled by a centralized management server.

With VDCF, JomaSoft offers a tool to simply and effectively operate your Solaris 10 based virtual data center. On a central management server you create definitions and configuration, which are stored in the Configuration Repository. This information is then used by VDCF to populate physical servers with a Solaris build from which virtual servers or logical domains are created.



1.1.7 VDCF terminology

In order to facilitate the virtualized environment created and managed by VDCF, a specific terminology is applied. This terminology strictly separates the physical servers (global zone) or nodes and virtual servers (non-global zone).

Node: The physical servers hardware plus the Solaris global zone.

The node is strictly used as a carrier for vServers. It is stateless and might be re-installed at any time. VDCF is responsible for installing and tracking the currently active build of a particular node.

vServer: The Solaris non-global zone.

The vServer is responsible for running the business applications. All state assigned to a particular application is contained within a vServer and its associated storage. A vServer is built on top of at least one dataset which in turn hosts at least one filesystem carrying the configuration, programs and data for it.

LDom: Control Domain and Guest Domains.

The Control Domain is managing and serving the Guest Domains installed on the same physical server hardware. Guest Domains may be used as a physical node to carry vServers. VDCF is responsible for installing and tracking the currently active build of a particular logical domain.

Dataset: A storage abstraction used to manage LUN's in the volume-manager hierarchies.

A Dataset abstracts and standardizes the storage handling for vServers. Datasets use volume manager technology to create the required quality of service by grouping LUN's into different RAID (0,1,0+1) constructs. By default datasets are available in two different implementations. One uses Solaris Volume Manager (SVM) technology while the other implements on top of ZFS.

VDCF is installed in the global zone of a Solaris 10 server called the management server. In a highly available VDCF environment it may be installed in a non-global zone. From this server you install and operate your Nodes, vServers or logical domains.

The modular structure of the management server and the VDCF software makes it possible to flexibly adapt to individual customer's requirements. Extensions to the basic functionality can be simply realized by the means of adjustment and addition of individual modules.

1.2 Supported Environments

Currently the following System Environments are supported:

- Management Server Oracle SPARC Server and x86 Server
- Compute Node/Server Oracle SPARC Server and x86 Server
- Solaris Operating System Solaris 10 Update1 (1/06) or later
- Logical Domains LDomS 1.1/1.2/1.3
- Branded Zones solaris8, solaris9
- Volume Manager Solaris Volume Manager (SVM), ZFS
- Filesystem Solaris UFS, lofs, ZFS
- SAN / iSCSI Storage and HBA's compatible to
SUN StorEdge SAN 4.4.x / Multipathing using STMS/MPXIO
iSCSI Targets compatible to Solaris iSCSI Initiator
- Terminal Server Blackbox, Cyclades, IOLAN
- System Controller SC/ALOM, RSC, SSC, 15K, XSCF, ALOMCMT, ILOM, ILOMx86
- Network Link aggregation, IPMP and tagged VLAN for LDomS and vServer

For VDCF vServer Enterprise Customers the following Extensions are available:

- Resource Management Administration of vServer Resource settings
- Monitoring Hardware and Resource Monitoring
- Veritas Dataset Volume Manager: VXVM, Filesystem: vxfs
- Sun Cluster Integration of vServers in Sun Cluster
- Veritas Cluster Integration of vServers in Veritas Cluster

Other environments may only need small enhancements. Send us your request !

2 VDCF Product Structure and Commands

2.1 Product Structure

2.1.1 Commands and Manual Pages

The VDCF framework base installation directory is `/opt/jomasoft/vdcf`. For Administrators the two major subdirectories are

<code>bin</code>	where the framework commands can be found
<code>man</code>	man pages about the framework commands and configuration files

2.1.2 VDCF Datastore and Configuration

All data is saved in the `/var/opt/jomasoft/vdcf` directory. The Main subdirectories are

<code>db</code>	database directory / configuration repository
<code>log</code>	where the framework logfiles are written
<code>conf</code>	various configuration files, like <code>customize.cfg</code> , partitioning, build profile, etc
<code>config</code>	files used for the system configuration, like scripts and packages
<code>discover</code>	configuration data about discovered nodes
<code>export</code>	data exports from the configuration repository

2.1.3 Logfiles

The framework writes its messages to two logfiles

<code>audit.log</code>	This log contains all executed commands along with user and timestamp
<code>framework.log</code>	INFO and ERROR messages about the operations executed. These messages are used by support staff and system administrators to debug problems.

2.2 Commands

All VDCF commands can be found in `/opt/jomasoft/vdcf/bin`.

The VDCF LDom package adds these additional commands:

<code>cdom</code>	Control Domain Setup and administration
<code>gdom</code>	Guest Domain Setup and administration

All commands are built using the same basic structure. There are switches to enable debugging, getting help and executing a particular operation.

```
USAGE: command [ -xhH ] -c <operation>
      The following options are supported:

      -x key|key=n[,key|key=n, ...] while key is:
          debug=<level>           as defined in debugMsg(3lib)
          noexec                  as defined in execCmd(3lib)
          verbose                 stdout verbosity for log
                                as defined in logMsg(3lib)
      -h                          issue this message
      -H <operation>              operation manual page viewing
      -c <operation>             operation to be executed (see below)
```

All operations are documented as manual pages. These detailed descriptions can be shown using the `'-H <operation>'` switch or by using the `man(1)` command directly. An overview about all possible operations supported by a specific command can be listed using the `'-h'` switch.

3 Logical Domain Management

3.1 Overview

To use the Logical Domain features of VDCF a CMT system needs to be setup as a VDCF Node using the VDCF Base framework. The required steps include Node Discovery, Profile and Node configuration and Node install. See the VDCF Base – Administration Guide for details.

The Logical Domain Software (SUNWldm package Version 1.1 or later) must be installed on the Node. Use the VDCF commands `config` and `serverconfig` to automate this package installation.

Example:

```
config -c add type=PKG name=SUNWldm pkgdevice=ldom/SUNWldm.pkg pkgs=SUNWldm.v
serverconfig -c add type=PKG name=SUNWldm server=computeA
```

3.1.1 Control Domain (cdom)

The Control Domain requires dedicated Resources, like CPU and Memory. Before creating guest domains (gdom) on a node a Control Domain must exist. As a first step the control domain is defined in the Configuration Repository held on the management server. A cdom has to be created on an existing node. CPU and Memory settings have to be defined.

After completion of the configuration, the cdom is created on the Node by the "commit" operation.

3.1.2 Guest Domain (gdom)

As a first step a new guest domain (gdom) is defined in the Configuration Repository.

Every gdom requires at least this minimal configuration settings:

- one disk where the Solaris OS is to be installed
- a partitioning profile, which defines how to setup the root disk and filesystems
- resource settings for CPU, Memory
- a minimum of one network configuration. A network configuration requires an IP address and the selection of a network type (management, public, ...).
- a Solaris build has to be assigned using the "flash" command.

After completion of the configuration, the gdom is deployed to the control domain (cdom) by the "commit" operation. With the "install" operation the OS will be installed into the gdom.

3.2 Control domain (cdom)

3.2.1 cdom definition

When defining a Control domain it must be defined on top of an existing node. The CPU and Memory resources are reserved for the Control domain. The remaining resource may be used for Guest Domains.

```
% cdom -c create name=computeA cpu=8 memory=2048
```

3.2.2 cdom creation

The Control domain is activated on the Node using the commit operation. The memory allocation takes place at the next node reboot. This reboot is automated, if the reboot flag is used.

Manual reboot

```
% cdom -c commit name=computeA  
% node -c reboot name=computeA
```

Automated reboot

```
% cdom -c commit name=computeA reboot
```

If the Physical node is re-installed later, VDCF will configure the Control Domain automatically.

3.2.3 cdom remove

If a physical node has no guest domains anymore and is planned to be used as standalone system, the Control Domain can be removed as follows

```
% cdom -c remove name=computeA  
  
% cdom -c commit name=computeA reboot
```

3.3 Guest domain (gdom) configuration

3.3.1 gdom initial definition

Guests domains may be defined on activated Control Domains.

```
% gdom -c create name=g0001 cdom=computeA cpu=8 memory=1024 comment="Test gdom"
```

3.3.2 gdom modifications

At any time a gdom definition can be changed using the gdom modify command. I.e. to change the partitioning profile of a guest domain you can use this command:

```
% gdom -c modify name=g0001 profile=ldom_partitioning.cfg
```

3.3.3 Disks (LUN)

Every guest domain requires at least its own root disk, where the Root-Filesystem can be placed. The LUN must be visible to the target node (i.e. control domain). Display the list of available LUNs with the following command:

```
% diskadm -c show free node=computeA
```

Name	Use-Type	Dev-Type	State	GUID	Serial	Size/MB
-	FREE	MPXIO	UNUSED	600015D00002EE0...040D0	03461147	16384
-	FREE	MPXIO	UNUSED	600015D00002EE0...040EA	03461147	4096
-	FREE	MPXIO	UNUSED	600015D00002EE0...040ED	03461147	4096
-	FREE	MPXIO	UNUSED	600015D00002EE0...040F0	03461147	4096
-	FREE	MPXIO	UNUSED	600015D00002EE0...040F3	03461147	4096

Assign a LUN to the guest domain with this command. A LUN of type root is required:

```
% gdom -c adddisk name=g0001 type=root guids=60060E80141AC70000011AC700000172
```

3.3.4 Network

A gdom needs networks assigned. A network of type management is required. If only one network is defined, management also serves as public interface:

```
% gdom -c addnet name=g0001 type=management ipaddr=192.168.1.20 netmask=255.255.255.0
```

```
% gdom -c addnet name=g0001 type=public ipaddr=g0001 netmask=255.255.255.0
```

For IPMP you need to provide probes addresses if you use LDom 1.1 or 1.2 using the 'probe' attribute. Starting with LDom 1.3 and Solaris 10 Update 8 probes are optional.

3.3.5 Summary

```
$ gdom -c show name=g0001

General Guest Domain Information for: g0001 (Test gdom)

  Name  cState  rState  cPool  Act-Date  Act-Time  Mod-Date  Mod-Time
g0001  DEFINED  -       default  2008-12-16  00:25:56  2008-12-16  00:25:06

  Active Build  Enabled Build  Partitioning Profile  Configuration Groups
5.10sv_u6_user  5.10sv_u6_user  ldom_partitioning.cfg  <UNDEFINED>

  Resources  CPUs  RAM/MB  MAUs  CPU%  RAM%  Control Domain
              8      1024    0     16   40   computeA

Disk Devices
Type  GUID  Dev-Type  State  Serial  Size
root  60060E80141AC70000011AC700000179  MPXIO  DEFINED  50_11AC70179  4096

Network Interfaces
  Type  Hostname  VID  Interface  IP  Netmask  State
public  g0001     -    vnet0->bge0  10.31.200.60  255.255.255.0  DEFINED
management  g0001-mngt  -    vnet1->bge1  192.168.0.60  255.255.255.0  DEFINED
```

3.3.6 Installation

As for physical nodes, the first step is to assign an existing Build to the guest domain:

```
% flash -c list_versions

  Build Version  OS Version  Platform Arch  Build Name  Archive
      5.10S_U6  5.10       sparc  sun4u  5.10_U6_SPARC
/export/./5.10S_U6/archive/vdcf.flar
      5.10S_U6_P1  5.10       sparc  sun4u  5.10_U6_P20060526_SPARC
nfs://192.168.0.27.../vdcf.flar

% flash -c enable_install node=g0001 version=5.10S_U6_P1

Found Server: g0001 Model: 2 (2) x UltraSPARC-II 450MHz 2048MB RAM
Found network boot device on management network: qfe0, 192.168.1.20/255.255.255.0
Installation enabled for Node: g0001 Version: 5.10S_U6_P1
```

Then use this command to install the guest domain:

```
% gdom -c commit name=g0001 install
```

3.4 Guest domain (gdom) operation

Apart from run level functionality (boot, shutdown and reboot) the framework offers the possibility to modify the disks and network interfaces. And also resources like CPU, Memory or MAUs.

- **Disks:** Adding and Removing
- **Network:** Adding and Removing
- **Resources:** CPU, Memory, MAU

```
% gdom -c adddisk name=g0001 type=data guids=60060E80141AC70000011AC700000173
% gdom -c remdisk name=g0001 guids=60060E80141AC70000011AC700000173

% gdom -c addnet name=g0001 ...
% gdom -c remnet name=g0001 ...

% gdom -c modify name=g0001 memory=2048
```

The changes are activated using the `commit` operation.

Some changes, like changing Memory allocation are “delayed-reconfiguration”. Such configuration is activated at the next reboot of the guest domain.

3.5 Guest domain (gdom) migration

A guest domain can be migrated to other CMT systems including all containing vServers. To do that, this command sequence has to be used:

```
% gdom -c shutdown name=g0001
% gdom -c detach name=g0001
% gdom -c attach name=g0001 cdom=c0002
% gdom -c boot name=g0001
```

3.6 Guest domain (gdom) cleanup

To destroy the guest domain, free the used resources and disks use the following commands:

```
% gdom -c shutdown name=g0001
% gdom -c remove name=g0001
% gdom -c commit name=g0001 remove
```

3.7 Virtual pools (vPools) - Permission to manage Guest domains

Virtual pools (vPools) are used to add an additional authorization layer over the vServer and Guest domain related commands (vserver, gdom, dataset, rcadm). With this feature it's possible to define who may manipulate which vServer and Guest domains.

This feature is disabled by default, which means everybody is allowed to manage all vServers and GDoms. Set the configuration variable VPOOL_ENABLED to "TRUE" in your customize.cfg to activate it.

See VDCF Base – Administration Guide Chapter 11.3 for details.

4 Runtime States (Control and Guest Domains)

4.1 Overview

The Runtime States (rState) of Control and Guest Domains is displayed using the respective 'node -c show', 'cdom -c show' and 'gdom -c show' commands.

Control Domain (cdom) rState

ACTIVE	The Node is active. A ssh connection could be established.
UNKNOWN	The Node state is unknown, because no ssh connection could be established. The Node may be down or a network problem may be the cause.

Guest Domain (gdom) rState

	The GDom rState is detected using the Solaris Idm command.
ACTIVE (Softstate)	The GDom is running. This does not mean that Solaris is up and running. ACTIVE only implies that the gdom process is started. The Idm softstate is displayed in brackets and shows the real GDom runtime state.
BOUND	GDom is defined on the system but is not running.
UNKNOWN	The GDom state is unknown, because no ssh connection could be established to the control domain. The control domain may be down or a network problem may be the cause.

4.2 Cronjob

The Runtime States (rState) are updated in the VDCF configuration repository using a cronjob. It is recommended to run the cronjob regularly. VDCF delivers the following recommended cronjob in /opt/jomasoft/vdcf/conf/sysconf/vdcf_base_crontab:

```
# add the entries to the root crontab on
# your management server
# JSvdcf-base cron
0,15,30,45 * * * * /opt/jomasoft/vdcf/sbin/repos_update -q >/dev/null 2>&1
```

5 Configuration States (CDom, GDom, Disk, Network)

5.1 Overview

The Configuration States (cState) are displayed using the respective show commands and is the state that an object has in the VDCF repository.

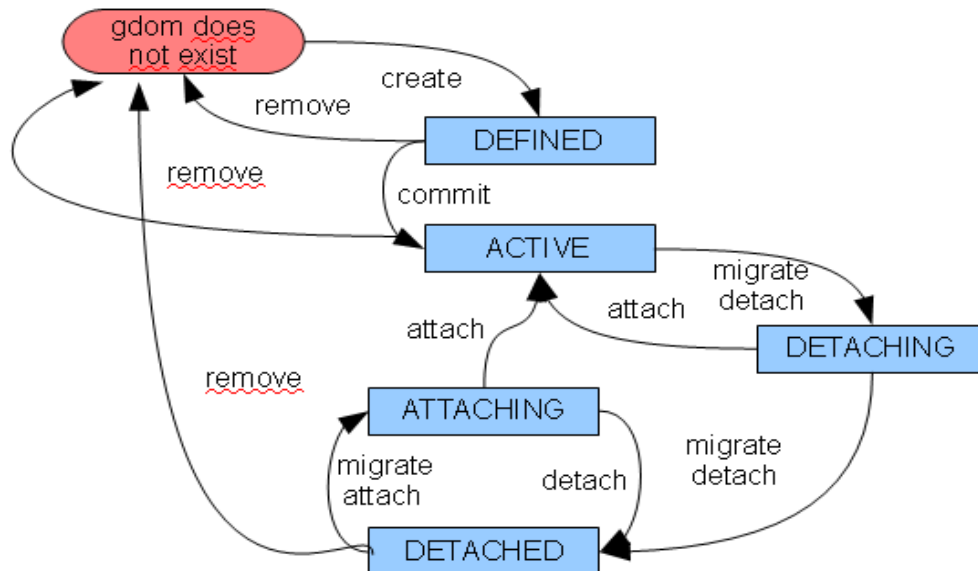
5.2 Possible state values

Node states:	UNINSTALLED, INSTALLING, ACTIVE, INACTIVE
CDom states:	DEFINED, ACTIVE, PURGING
GDom states:	DEFINED, ACTIVE, MODIFIED, DETACHING, DETACHED, ATTACHING, PURGING
Disk (LUN) states:	UNUSED, DEFINED, ACTIVATED, PURGING
Network states:	DEFINED, ACTIVATED, PURGING

5.3 State values explained

UNINSTALLED	Indicates that the node exists only in the VDCF repository.
INSTALLING	Indicates that the node is currently being installed by VDCF.
ACTIVE	Indicates that the object exists in the VDCF repository and is installed and running.
INACTIVE	Indicates that the node is marked inactive by administration staff
DEFINED	Indicates that the object exists only in the VDCF repository
MODIFIED	Indicates that the resource settings of the LDom was changed.
ACTIVATED	Indicates that the object exists in the VDCF repository and is activated on the node.
DETACHED	Indicates that the object was installed on a node, but is now detached (removed) from that node.
PURGING	Indicates that on the next commit operation this object will be deleted on the node and removed from the VDCF repository
DETACHING	Indicates that VDCF is or was trying to detach the GDom from a node.
ATTACHING	Indicates that VDCF is or was trying to attach the GDom to a node.

5.4 GDom state diagram



5.5 Supported GDom commands

Depending the state of a Guest Domain in VDCF different operations are possible. This overview shows the available operations:

These operations are always available:

- `gdom -c show`

While the GDom has the state **DEFINED** these gdom operations are possible:

- `gdom -c remove`
- `gdom -c addnet`
- `gdom -c adddisk`
- `gdom -c remnet`
- `gdom -c remdisk`
- `gdom -c commit`
- `gdom -c modify`

While the GDom has the state **ACTIVE** these gdom operations are possible:

- gdom -c addnet
- gdom -c remnet
- gdom -c commit
- gdom -c adddisk
- gdom -c remdisk
- gdom -c detach
- gdom -c boot
- gdom -c reboot
- gdom -c shutdown
- gdom -c remove

While the GDom has the state **DETACHING** these gdom operations are possible:

- gdom -c detach
- gdom -c attach

While the GDom has the state **DETACHED** these gdom operations are possible:

- gdom -c attach
- gdom -c remove

While the GDom has the state **ATTACHING** these gdom operations are possible:

- gdom -c attach
- gdom -c detach

6 Security Management

This chapter contains the information about security aspects of the VDCF framework.

6.1 Management Server RBAC

VDCF provides the following RBAC profiles which must be configured on the management server for your administration staff. Using the profiles you are able to permit a administrator appropriate access to the required VDCF commands.

RBAC profiles of VDCF LDom package:

VDCF ldom Module	cdom and gdom management & operations
------------------	---------------------------------------

RBAC profiles of VDCF base package:

VDCF Logger	required for all users, to be able to log framework messages
VDCF admin Module	vdcf administration
VDCF install Module	node installation
VDCF node Module	node operations
VDCF config Module	node and vServer customization
VDCF disks Module	disk management
VDCF dataset Module	dataset management
VDCF patches Module	patch management for nodes and vServer
VDCF computepool Manager	compute pool management
VDCF computepool User	compute pool display
VDCF vpool Manager	virtual pool management
VDCF vpool User	virtual pool display

Add the Profile entries to `/etc/user_attr` for the required administrators. All users with the above RBAC Profiles are allowed to execute the VDCF commands found in `/opt/jomasoft/vdcf/bin`.

Sample entry for an administrator user (see `/opt/jomasoft/vdcf/conf/sysconf/etc_user_attr`)

```
marcel::::type=normal;profiles=VDCF Logger,VDCF admin Module,VDCF install Module,VDCF
node Module,VDCF config Module,VDCF disks Module,VDCF dataset Module,VDCF virtual
Module,VDCF patches Module,VDCF computepool Manager,VDCF vpool Manager,VDCF ldom Module
```

If you would like to create a VDCF administration user, use the following command

```
useradd -d /export/home/vdcf -m -s /bin/bash -P "VDCF Logger,VDCF admin Module,VDCF
install Module,VDCF node Module,VDCF config Module,VDCF disks Module,VDCF dataset
Module,VDCF virtual Module,VDCF patches Module,VDCF computepool Manager,VDCF vpool
Manager,VDCF ldom Module" vdcf
```

7 Appendixes

7.1 Data File Overview

7.1.1 On VDCF Management Server

All data is saved in the `/var/opt/jomasoft/vdcf` directory. The main subdirectories are

<code>db</code>	database directory / configuration repository
<code>log</code>	where the framework logfiles are written
<code>conf</code>	various configuration files (see list below)
<code>config</code>	files used for the system configuration, like scripts and packages
<code>discover</code>	configuration data about discovered nodes
<code>export</code>	data exports from the configuration repository

Configfiles

<code>issue.cfg</code>	Issue message displayed while installing a system
<code>partitioning.cfg</code>	Partitioning definitions for node installations
<code>build.profile</code>	Build.profile used to define new Solaris Builds
<code>customize.cfg</code>	Customer dependent customizing of predefined values
<code>system.cfg</code>	Additions to <code>/etc/system</code> for node installations
<code>patch_issue.cfg</code>	Issue message displayed while patching a system
<code>wanboot_defaultrouter.cfg</code>	Network and Defaultrouter for WAN Boot

Logfiles

The framework writes its messages to two logfiles

<code>audit.log</code>	This log contains all executed commands along with user and timestamp
<code>framework.log</code>	INFO and ERROR messages about the operations executed. These messages are used by support staff and system administrators to debug problems.

7.1.2 On Compute Nodes

The data directory on Compute Nodes is `/etc/vdcfbuild` with the following subdirectories

<code>patches</code>	patch configuration and patching logfiles
<code>routes</code>	routes configuration files

VDCF internal configuration files for logical domains are stored under `/var/tmp/ldomcfg/<gdom>`.

These directories and files may be helpful while debugging a problem, modification should only be done with specific instruction from JomaSoft Support.



7.2 Customization of VDCF environment

These VDCF configuration values can be changed to adjust VDCF for a customer environment. To overwrite a VDCF variable add the appropriate value to `customize.cfg`:

Variable name	Description
<code>GDOM_DEFAULT_PROFILE</code>	Default partitioning profile file used to setup guest domains

8 Manpages

User Commands cdom -c <operation>(1)

NAME

cdom -c <operation> - Control Domain Operations Interface

SYNOPSIS

cdom -c <operation> <args ...>

DESCRIPTION

This command is used to manipulate Control Domain configurations and their state. The following operations are available:

cdom -c create

cdom -c show

cdom -c modify

cdom -c commit

cdom -c remove

Use the `cdom -H <operation>` command to get detailed information about how to use a particular operation.

RETURN VALUES

cdom returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

cdom_create(1M), cdom_show(1M), cdom_modify(1M),
cdom_commit(1M), cdom_remove(1M)

User Commands `cdom -c commit(1)`

NAME

`cdom -c commit` - Commits a control domain definition

SYNOPSIS

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

DESCRIPTION

The `commit` operation instantiates or removes a control domain or modifies the resource settings of a control domain. Depending on their state, a control domain is created ('DEFINED'), modified ('ACTIVE') or removed ('PURGING').

Use the 'reboot' option to immediately execute a reboot of the control domain after committing the modifications.

Use the 'force' option to commit changes in the resource settings of an already activated control domain.

EXAMPLES

Following an example for committing a control domain:

```
cdom -c commit name=S0010 reboot
```

RETURN VALUES

`cdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`cdom(1M)`, `cdom_show(1M)`, `cdom_create(1M)`, `cdom_modify(1M)`,
`cdom_remove(1M)`

User Commands

`cdom -c create(1)`

NAME

`cdom -c create` - Create a control domain

SYNOPSIS

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

DESCRIPTION

Creates a new control domain. 'name' defines the name of the node on which the control domain is created. The control domain and the node name are identical. The definition of 'ram' and 'cpu' is necessary.

EXAMPLES

Following an example for creating a control domain:

```
cdom -c create name=S0010 cpu=4 ram=512M
```

RETURN VALUES

`cdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`cdom(1M)`, `cdom_show(1M)`, `cdom_modify(1M)`, `cdom_commit(1M)`,
`cdom_remove(1M)`

User Commands `cdom -c modify(1)`

NAME

`cdom -c modify` - Modify a control domain

SYNOPSIS

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

DESCRIPTION

This command is used to modify the settings of a control domain.

EXAMPLES

Following an example for modifying a control domain:

```
cdom -c modify name=S0010 ram=1024M mau=2
```

RETURN VALUES

`cdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`cdom(1M)`, `cdom_show(1M)`, `cdom_create(1M)`, `cdom_commit(1M)`,
`cdom_remove(1M)`

User Commands `cdom -c remove(1)`

NAME

`cdom -c remove` - Removes a control domain

SYNOPSIS

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

DESCRIPTION

This command is used to remove a control domain. Prior to remove a control domain all guest domains sitting on that control domain must be removed or migrated to another control domain.

The remove command only sets the state of the control domain to 'PURGING'. The physical removal of it has to be confirmed by the `cdom -c commit` command.

EXAMPLES

Following an example for removing a control domain:

```
cdom -c remove name=computeA
```

RETURN VALUES

`cdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`cdom(1M)`, `cdom_show(1M)`, `cdom_create(1M)`, `cdom_modify(1M)`,
`cdom_commit(1M)`

User Commands `cdom -c show(1)`

NAME

`cdom -c show` - Shows control domain information

SYNOPSIS

```
cdom -c show  
[ name=<control domain name> ]  
[ verbose ]
```

DESCRIPTION

Shows the currently defined control domains. All control domains are listed, if no additional parameter is specified.

If 'name' is supplied, detailed information for this particular control domain will be shown. 'verbose' will display also information about the ldm configuration.

EXAMPLES

An example for showing information for a particular control domain:

```
cdom -c show name=S0010
```

RETURN VALUES

`cdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`cdom(1M)`, `cdom_create(1M)`, `cdom_modify(1M)`, `cdom_commit(1M)`, `cdom_remove(1M)`

User Commands `gdom -c <operation>(1)`

NAME

`gdom -c <operation>` - guest domain Operations Interface

SYNOPSIS

`gdom -c <operation> <args ...>`

DESCRIPTION

This command is used to manipulate guest domain configurations and their state. The following operations are available:

```
gdom -c create
gdom -c show
gdom -c modify
gdom -c revert
gdom -c adddisk
gdom -c remdisk
gdom -c addnet
gdom -c remnet
gdom -c remove
gdom -c commit
gdom -c install
gdom -c detach
gdom -c attach
gdom -c boot
gdom -c reboot
gdom -c shutdown
```

Use the `gdom -H <operation>` command to get detailed information about how to use a particular operation.

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal



User Commands

gdom -c <operation>(1)

0 if failed.

SEE ALSO

gdom_create(1M), gdom_show(1M), gdom_modify(1M),
gdom_adddisk(1M), gdom_remdisk(1M), gdom_addnet(1M),
gdom_remnet(1M), gdom_remove(1M), gdom_commit(1M),
gdom_install(1M), gdom_attach(1M), gdom_detach(1M),
gdom_boot(1M), gdom_reboot(1M), gdom_shutdown(1M)

User Commands `gdom -c adddisk(1)`

NAME

`gdom -c adddisk` - Add disks to a guest domain

SYNOPSIS

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

DESCRIPTION

Adds one or more disks to the specified guest domain. There are two 'type' of this command - one for creating the required root disk and another for optional data filesystems.

This command does not actually apply the disk to a node. Instead the newly configured resources are marked as DEFINED. Use the 'commit' operation to propagate configuration changes made with this command to the respective node and change there state to ACTIVE.

EXAMPLES

Following an example to add a root disk to a guest domain.

```
gdom -c adddisk name=gdomA type=root guids=XYX
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom(1M)`, `gdom_commit(1M)`, `gdom_remdisk(1M)`

NOTES

Note that all except the last 'commit' statement, only manipulates configuration values. Only the 'commit' operation performs actions on the actual node in order to instantiate the defined configuration.

User Commands gdom -c addnet(1)

NAME

`gdom -c addnet` - Add a network to a guest domain

SYNOPSIS

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

DESCRIPTION

Adds a network interface to a guest domain configuration. A network interface always has a 'type'. The selection of the physical interface will be based on this 'type'. The default mapping is defined as GDOM_NET_MAPPING. A different CDom network interface may be selected by specifying the CDom network type separated by a double point, for example: `type=management:PUBL`.

Moving a guest domain from one platform to another might cause a change of the underlying physical interface. Each interface also needs an 'ipaddr' and a 'netmask'.

'ipaddr' may be an ip address or hostname, which must be resolvable in the DNS.

Use the 'vlan' argument to assign the gdom network interface to a VLAN network interface. The VLAN ID (vid) is a value between 1 and 4094. The connected switch (port) must be configured accordingly.

To define networks with IPMP you may use these arguments: Use 'ipmp group name' to define your own IPMP group name, 'ipmp' to use the default IPMP group names and/or the 'probes' argument to define IPMP test addresses.

This command does not actually create an interface. Instead the configured resources are marked as DEFINED. Use the 'commit' operation to propagate configuration changes made with this command to the respective CDom and create the interfaces.

EXAMPLES

Below an example to add a 'public' network interface to a guest domain:

User Commands

```
gdom -c addnet(1)
```

```
gdom -c addnet name=gdomA type=public ipaddr=10.23.124.2
```

RETURN VALUES

gdom returns 0 if the operation was successful or not equal
0 if failed.

SEE ALSO

gdom(1M), gdom_commit(1M), gdom_remnet(1M),

NOTES

Note that all except the last 'commit' statement, only manipulates configuration values. Only the 'commit' operation performs actions on the actual node in order to instantiate the defined configuration.



User Commands `gdom -c attach(1)`

NAME

`gdom -c attach` - Attaches a guest domain to a control domain (node)

SYNOPSIS

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

DESCRIPTION

Attaches a previously detached guest domain. If one wants to change the currently configured target node - the node the guest domain was attached last - the 'cdom' parameter can be used to overwrite the current setting.

Attaching a guest domain is only allowed if the source and target control domain are assigned to the same Compute Pool.

EXAMPLES

Following an example that attaches a guest domain to a new target control domain:

```
gdom -c attach name=gdomA cdom=computeA
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom(1M)`, `gdom_detach(1M)`, `cpool(1M)`

NOTES

This command is a direct operation that actively changes state on the current hosting node for a particular guest domain.



User Commands

`gdom -c boot(1)`

NAME

`gdom -c boot` - Boots up a guest domain

SYNOPSIS

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

DESCRIPTION

Boots a guest domain.

EXAMPLES

Following an example for booting up a guest domain:

`gdom -c boot name=gdomA`

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal
0 if failed.

SEE ALSO

`gdom(1M)`, `gdom_reboot(1M)`, `gdom_shutdown(1M)`

User Commands `gdom -c commit(1)`

NAME

`gdom -c commit` - Commit a guest domain configuration

SYNOPSIS

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

DESCRIPTION

The commit operation instantiates or removes all resources in a intermediate state - this are resources in either DEFINED or PURGING state. Based on their state they are created (DEFINED) or removed (PURGING).

Activated resources are modified if their configuration changed, e.g. cpu, memory size changes.

NOTE: This operation does change a nodes configuration by directly invoking programs on it!

New guest domains can be installed after creation with the install option or later with `gdom -c install name=gdomA`
NOTE: The guest domain system configuration takes place at the first install.

For existing guest domains the configuration is updated on the node. Some configuration is activated on the next reboot of the guest domain.

EXAMPLES

Following an example that commits all pending changes for a given guest domain:

```
gdom -c commit name=gdomA
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom(1M)`, `gdom_install(1M)`, `flash_enable_install(1M)`

NOTES

This operation does change a nodes configuration by directly invoking programs on it!

User Commands gdom -c create(1)

NAME

`gdom -c create` - Create a guest domain

SYNOPSIS

```
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"> ]
```

DESCRIPTION

Creates a new guest domain. 'name' defines the name of the new guest domain. The name given must be unique within a management server. The 'cdom' defines the control domain where the guest domain will be created on. 'comment' is an arbitrary string used to describe the purpose of this guest domain.

The optional 'vpool' argument adds the new Guest Domain to the specified vPools. Every Guest Domain is automatically added to the 'ALL' vPool.

After completion of this command, the guest domain configuration will be inserted into the database with a state of DEFINED. To actually instantiate this DEFINED configuration a 'commit' command needs to be issued. 'commit' creates the guest domain configuration on its defined target 'node' and transitions the state of the guest domain configuration to ACTIVE if creation was successful. In order to 'commit' a guest domain configuration the following resources have to be added to the basic guest domain configuration:

Add root disk and optional data disks.

At least one network interface

EXAMPLES

A typical command sequence used to create above resources and therefor to create a working guest domain includes the following statements:

```
gdom -c create name=gdomA cdom=computeA cpu=4
```

```
gdom -c adddisks name=gdomA type=root guids=<GUID>
```

User Commands

`gdom -c create(1)`

```
gdom -c addnet name=gdomA type=management ipaddr=gdomA-  
mngt
```

```
gdom -c commit name=gdomA
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal
0 if failed.

SEE ALSO

`node(1M)`, `gdom_adddisk(1M)`, `gdom_addnet(1M)`,
`gdom_commit(1M)`, `vpool(1M)`

NOTES

'comment' must be enclosed in quotes ''' in order to accept
multiple blank separated words.

Note that all except the last 'commit' statement, only manipulates
configuration values. Only the 'commit' operation performs actions on
the actual node in order to instantiate the defined configuration.

User Commands `gdom -c detach(1)`

NAME

`gdom -c detach` - Detaches a guest domain from its current node

SYNOPSIS

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

DESCRIPTION

Detaches a guest domain from its current node. The guest domain must be halted - in the 'installed' state - to be able to perform this operation. This operation saves the guest domain configuration and detaches all vServer belonging to it. A detached guest domain is no longer visible to any node.

Use `gdom -c attach` to attach the guest domain to another node.

EXAMPLES

Following an example that detaches a guest domain from its current node:

```
gdom -c detach name=gdomA
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom(1M)`, `gdom_attach(1M)`

NOTES

This command is a direct operation that actively changes state on the current hosting node for a particular guest domain.

User Commands `gdom -c install(1)`

NAME

`gdom -c install` - Install a guest domain

SYNOPSIS

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

DESCRIPTION

Fresh installs a guest domain currently on the ok. prompt. The installation parameters will be extracted from the currently active setup (see `flash -c list_active`).

vServers should be detached or migrated from the guest domain node before installing.

EXAMPLES

Following an example used to install (fresh-bit) a particular guest domain:

```
gdom -c install name=gdomA
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`node(1M)`, `flash_enable_install(1M)`, `flash_list_active(1M)`

User Commands `gdom -c modify(1)`

NAME

`gdom -c modify` - Modify a guest domain

SYNOPSIS

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

DESCRIPTION

Modifies the existing guest domain with the given 'name'.

'comment' is an arbitrary string used to describe the purpose of this Guest domain.

EXAMPLES

The following command modifies the cpu attribute of the guest domain.

```
gdom -c modify name=server2 cpu=8
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom_create(1M)`, `gdom(1M)`

NOTES

The comment needs to be enclosed in quotes ''' in order to accept multiple blank separated words.



User Commands

`gdom -c reboot(1)`

NAME

`gdom -c reboot` - Reboots a guest domain

SYNOPSIS

```
gdom -c reboot
name=<guest domain name>
[ force ]
[ stop ]
```

DESCRIPTION

Reboots a currently running guest domain. A regular run-level change will be issued to shut down the running solaris operating system and in a second step the gdom will be stopped/started on the cdom.

The 'force' flag is required to shutdown or reboot a gdom, if there are running vServers and the config `NODE_SHUTDOWN_CHECK_VSERVER` is set to TRUE.

If a gdom is not reachable by the VDCF management server, the 'stop' flag can be used to reboot the gdom using the logical domain management tools on the cdom.

EXAMPLES

Following an example that reboot a currently running guest domain with running vServers in it:

```
gdom -c reboot name=gdomA force
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom(1M)`, `gdom_boot(1M)`, `gdom_shutdown(1M)`

User Commands `gdom -c remdisk(1)`

NAME

`gdom -c remdisk` - Removes disks from a guest domain

SYNOPSIS

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

DESCRIPTION

Removes one or more disks from the specified guest domain.

This command does not actually remove the disk to guest domain. Instead the disk is marked for PURGING. Use the 'commit' operation to propagate configuration changes made with this command to the respective node.

EXAMPLES

Following an example to remove a data disk from a guest domain.

```
gdom -c remdisk name=gdomA guids=XYX
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom(1M)`, `gdom_commit(1M)`, `gdom_adddisk(1M)`

NOTES

Note that all except the last 'commit' statement, only manipulates configuration values. Only the 'commit' operation performs actions on the actual node in order to instantiate the defined configuration.

User Commands `gdom -c remnet(1)`

NAME

`gdom -c remnet` - Removes networks from a guest domain

SYNOPSIS

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

DESCRIPTION

Removes a network interface configuration or indicates removal for already created interface. Interfaces to remove are either selected by 'type' or 'ipaddr'.

This command does not actually remove an interface from a node. Instead the unconfigured resources are marked as PURGING. Use the 'commit' operation to propagate configuration changes made with this command to the respective node and remove their traces from the configuration.

EXAMPLES

Below an example to remove an existing network interface from a guest domain based on its IP address:

```
gdom -c remnet name=gdomA ipaddr=10.23.124.2
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom(1M)`, `gdom_commit(1M)`

NOTES

Note that all except the last 'commit' statement, only manipulates configuration values. Only the 'commit' operation performs actions on the actual node in order to instantiate the defined configuration.

User Commands `gdom -c remove(1)`

NAME

`gdom -c remove` - Remove a guest domain

SYNOPSIS

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

DESCRIPTION

Removes a guest domain configuration. You have to first remove all resources, e.g. Disks and Networks to remove the guest domain completely.

To remove a detached guest domain, which cannot be attached to any node, you must use the force option. Use this option only with special care, because it deletes all resources (Disks, Networks) of the guest domain from the configuration repository.

EXAMPLES

A typical command sequence used to remove a existing guest domain is shown below:

```
gdom -c shutdown name=server1
gdom -c remnet name=server1 type=all
gdom -c remdisk name=server1 guids=XYZ
gdom -c commit name=server1
gdom -c remove name=server1
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom_remdisk(1M)`, `vserver_remnet(1M)`, `gdom_commit(1M)`



User Commands `gdom -c revert(1)`

NAME

`gdom -c revert` - Reverts guest domain configuration changes

SYNOPSIS

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

DESCRIPTION

Reverts un-committed changes. It only operates on configuration changes in the VDCF configuration repository - typically remove changes - and makes them undone.

EXAMPLES

Following an example that reverts an operation:

```
gdom -c revert name=gdomA
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom(1M)`, `gdom_show(1M)`, `gdom_remove(1M)`

User Commands `gdom -c show(1)`

NAME

`gdom -c show` - Shows guest domain information

SYNOPSIS

```
gdom -c show
[ name=<guest domain name> [ verbose ] ]
[ cdom=<control domain> ]
[ cpool=<computepool name> ]
[ all ]
```

DESCRIPTION

Shows the currently defined guest domains. If no search parameter is specified all Guest Domains are listed. But when the vPool feature is activated only Guest Domains assigned to the users vPools are listed. In that case the 'all' flag may be used to list all existing Guest Domains (even those assigned to other users).

The 'cdom' and 'cpool' argument limits the list to the guest domains assigned to the given control domain or computepool.

If 'name' is supplied, detailed information for this particular guest domain will be shown. 'verbose' will display also information about the ldm configuration on the current node.

EXAMPLES

An example for showing a list of guest domains a user may manipulate:

```
gdom -c show
```

An example for showing a list of all guest domains in a specific computepool:

```
gdom -c show cpool='Prod' all
```

RETURN VALUES

`gdom` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom(1M)`, `cpool(1M)`

User Commands `gdom -c shutdown(1)`

NAME

`gdom -c shutdown` - Shutting down a guest domain

SYNOPSIS

```
gdom -c shutdown
name=<guest domain name>
[ force ]
[ stop ]
```

DESCRIPTION

Shut down a currently running guest domain. A regular run-level change will be issued to shut down the running solaris operating system and in a second step the gdom will be stopped on the cdom.

The 'force' flag is required to shutdown a gdom, if there are running vServers and the config NODE_SHUTDOWN_CHECK_VSERVER is set to TRUE.

If a gdom is not reachable by the VDCF management server, the 'stop' flag can be used to shut down the gdom using the logical domain management tools on the cdom.

EXAMPLES

Following an example for shutting down a guest domain:

```
gdom -c shutdown name=gdomA
```

RETURN VALUES

gdom returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`gdom(1M)`, `gdom_boot(1M)`, `gdom_reboot(1M)`, `gdom_show(1M)`



User Commands `vpool -c add_gdom(1)`

NAME

`vpool -c add_gdom` - add Guest Domains to vPools

SYNOPSIS

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

DESCRIPTION

Used to add Guest Domains to vPools. All arguments accept a comma-separated list of names.

Use the argument 'gdom' to add selected Guest Domains to vPools. With the argument 'cpool' all Guest Domains defined in that cPool are added.

EXAMPLES

Following an example to add selected Guest Domains to the vPool 'business1':

```
vpool -c add_gdom name=business1
gdom=guest1,guest5,guest9
```

RETURN VALUES

`vpool` returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

`vpool_show(1M)`, `vpool_create(1M)`, `vpool_modify(1M)`,
`vpool_add_vserver(1M)`, `vpool_add_user(1M)`,
`vpool_remove_vserver(1M)`, `vpool_remove_user(1M)`,
`vpool_remove_gdom(1M)`, `vpool_remove(1M)`, `cpool(1M)`

User Commands `vpool -c remove_gdom(1)`

NAME

`vpool -c remove_gdom` - remove Guest Domains from vPools

SYNOPSIS

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

DESCRIPTION

Used to remove Guest Domains from vPools. All arguments accept a comma-separated list of names. Use the argument 'gdom' to remove selected Guest Domains from vPools. With the argument 'cpool' all Guest Domains defined in that cPool are removed.

EXAMPLES

Following an example to remove all Guest Domains of cPool 'prod' from vPools 'bu1' and 'bu2':

```
vpool -c remove_gdom name=bu1,bu2 cpool=prod
```

RETURN VALUES

vpool returns 0 if the operation was successful or not equal 0 if failed.

SEE ALSO

```
vpool_show(1M),      vpool_create(1M),      vpool_modify(1M),
vpool_add_vserver(1M),      vpool_add_user(1M),
vpool_add_gdom(1M),      vpool_remove_vserver(1M),
vpool_remove_user(1M), vpool_remove(1M), cpool(1M)
```