

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

## VDCF vServer Product

## Administration Guide

Version 4.2  
22. December 2011

Copyright © 2005-2011 JomaSoft GmbH  
All rights reserved.



## Table of Contents

1 Introduction.....	4
1.1 Overview.....	5
1.1.1 Consolidation.....	5
1.1.2 Virtualization.....	6
1.1.3 Solaris Containers.....	7
1.1.4 Solaris Logical Domains.....	8
1.1.5 Datacenter Architecture.....	9
1.1.6 Virtual Datacenter Control Framework (VDCF) .....	10
1.1.7 VDCF terminology.....	11
1.2 Supported Environments.....	12
2 VDCF Product Structure and Commands.....	13
2.1 Product Structure.....	13
2.1.1 Commands and Manual Pages.....	13
2.1.2 VDCF Datastore and Configuration.....	13
2.1.3 Logfiles.....	13
2.2 Commands.....	14
3 Virtual Server Management.....	15
3.1 Overview.....	15
3.1.1 Datasets.....	15
3.1.2 vServer.....	15
3.2 vServer - Initial Definitions.....	16
3.2.1 vServer .....	16
3.2.2 Dataset.....	17
3.2.3 Filesystems.....	19
3.2.4 Network.....	20
3.3 vServer - Installation.....	21
3.4 vServer - Operations.....	21
3.4.1 Mount / unmount filesystems.....	21
3.4.2 Rename filesystems.....	22
3.4.3 Manipulate Mirrors (ZFS and SVM).....	22
3.4.4 Display and manipulate ZFS.....	24
3.5 vServer - Migration.....	25
3.5.1 Typical Migration.....	25
3.5.2 Detach/Attach.....	27
3.6 vServer - Disaster Recovery.....	28
3.6.1 Reinstall the compute node .....	28
3.6.2 Migrate the vServer to another existing node .....	28
3.7 vServer - Cleanup, Re-Installation and Remove.....	29
3.7.1 Cleanup vServer.....	29
3.7.2 Remove vServer.....	29
3.7.3 Remove a DETACHED vServer.....	29
3.8 Virtual pools (vPools) - Permission to manage vServers.....	30
4 vServer Runtime States.....	31
4.1 Overview.....	31
4.2 Cronjob.....	31
5 Configuration States (vServer, Disk, Dataset, Filesystem, Network).....	32
5.1 Overview.....	32
5.2 Possible state values.....	32
5.3 State values explained.....	32
5.4 State diagram vserver.....	33
5.5 Supported vserver commands.....	33
6 Security Management.....	35
6.1 Management Server RBAC.....	35
7 Appendixes.....	36
7.1 Data File Overview.....	36
7.1.1 VDCF Management Server.....	36
7.1.2 On Compute Nodes.....	36



7.2 Customization of VDCF environment.....	37
7.3 Solaris 8 Containers (Branded Zones).....	38
7.3.1 Requirements.....	38
7.3.2 SOL8 vServer.....	38
7.4 Solaris 9 Containers (Branded Zones).....	39
7.4.1 Requirements.....	39
7.4.2 SOL9 vServer.....	39



## 1 Introduction

This documentation describes the Virtual Datacenter Control Framework (VDCF) vServer product for the Solaris Operating System, Version 4.1 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 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/>

## 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

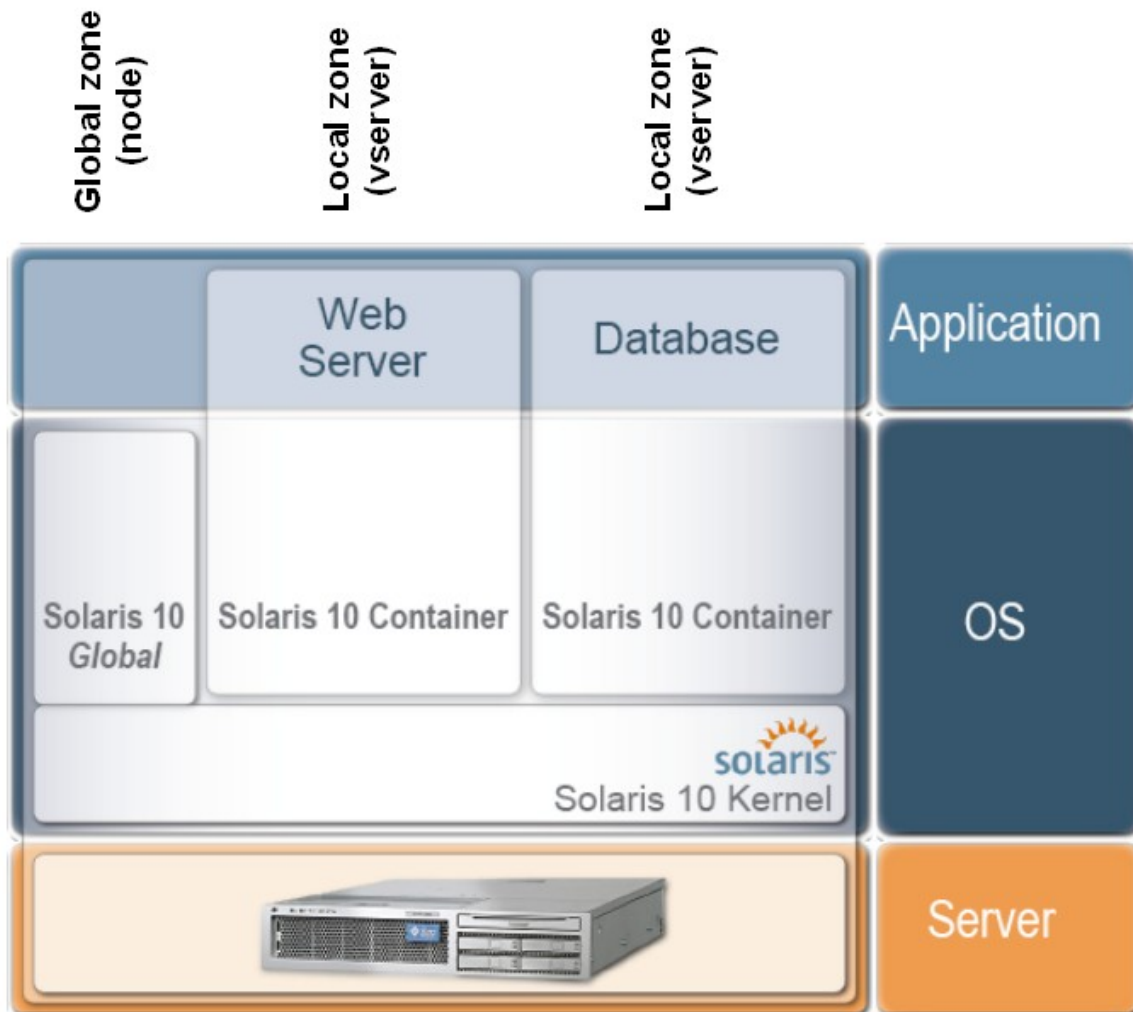
In computing terms, Virtualization is the creation of many digital abstractions that represent a real physical object.

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.

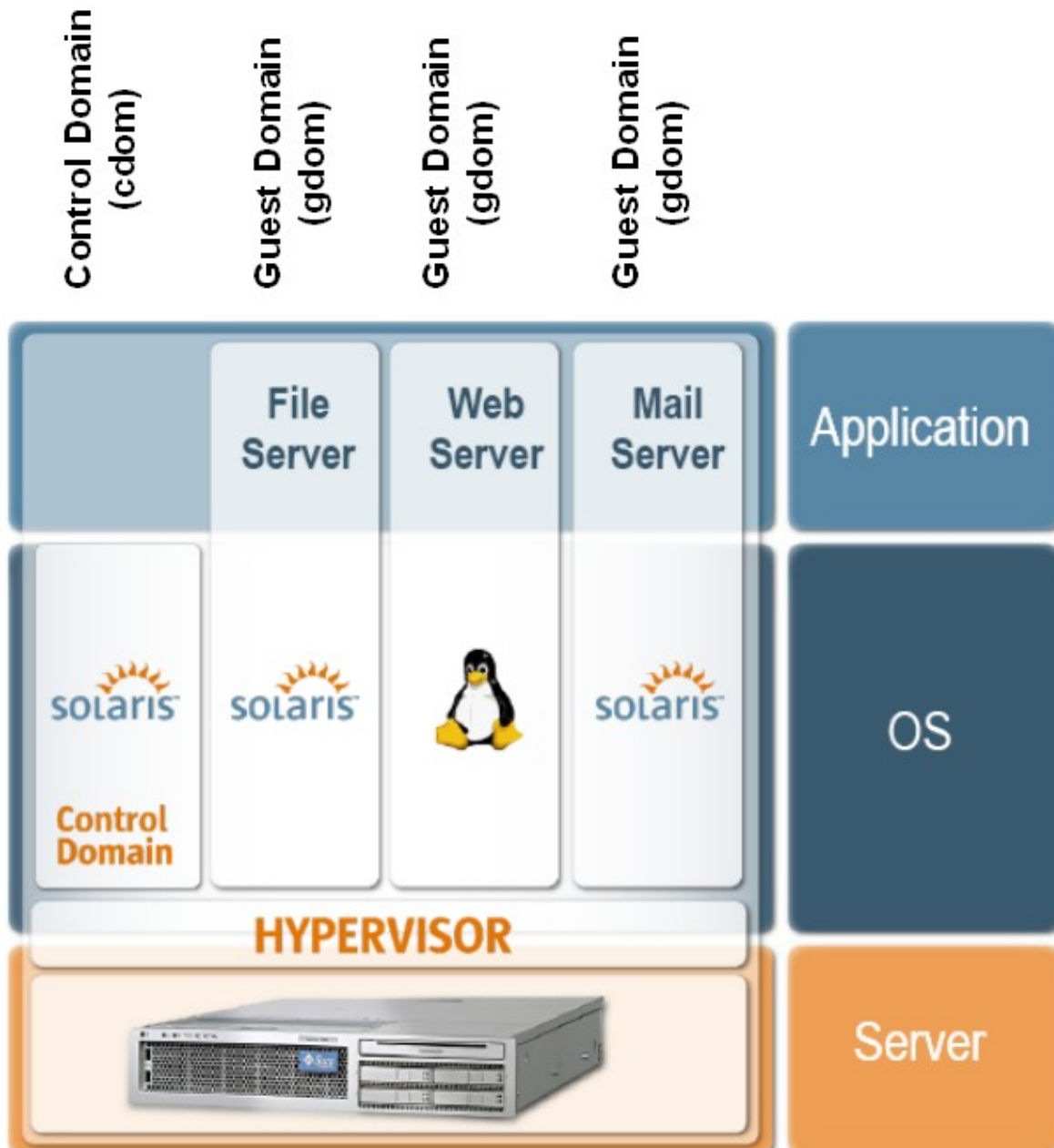
### 1.1.3 Solaris Containers

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.



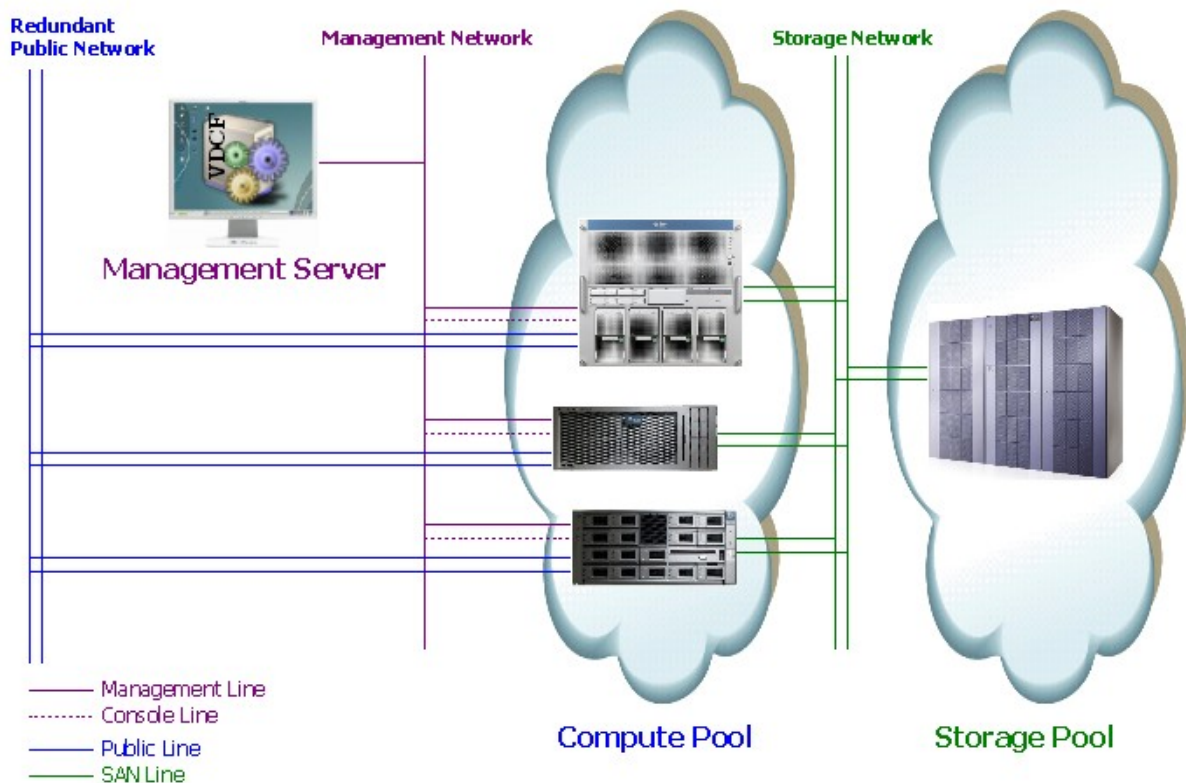
### 1.1.4 Solaris Logical Domains

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.

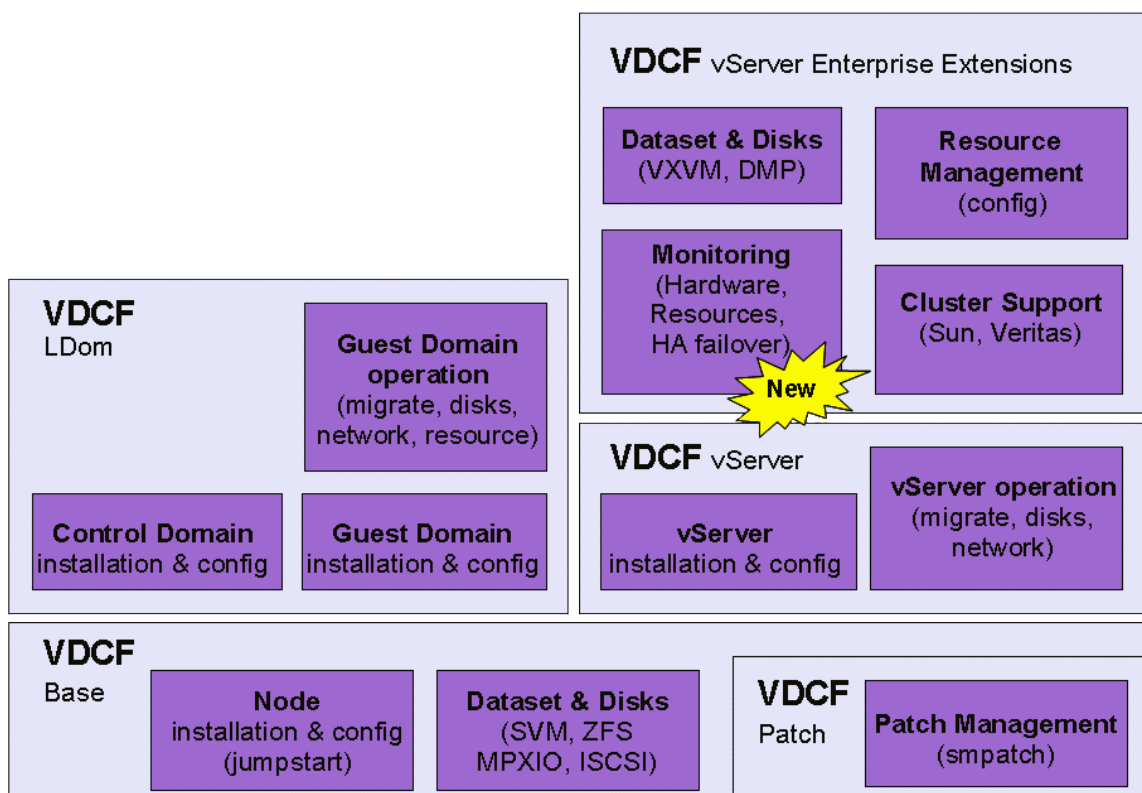
#### Storage Pool

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) **up to Update 10 (8/11)**
- Logical Domains LDoms 1.1/1.2/1.3/2.0/2.1
- 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  
vServer exclusive ip-stack

For VDCF vServer Enterprise Customers the following Extensions are available:

- Resource Management Administration of vServer Resource settings
- Monitoring Hardware and Resource Monitoring  
High Availability/Automated Failover
- 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 vServer package adds this additional commands:

<code>vserver</code>	Virtual Server Management (Zones), Configuration, Installation, Operations
<code>zfsadm</code>	Admin command for vServer related ZFS operations

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
    verbose                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 Virtual Server Management

### 3.1 Overview

#### 3.1.1 Datasets

All data of the virtual server is stored inside a data management abstraction layer called Datasets. Datasets are used to handle the quality of service aspects by providing standardized volume manager hierarchies. Datasets are implemented on top of a Volume Manager technology and allowed to create different storage qualities through different RAID levels. Currently implementations for Solaris Volume Manager (SVM) and ZFS are available in the base product. Other volume managers like VxVM are available as enterprise extensions or can be implemented as well. A dataset can also just be a group of raw devices which are delegated into a vServer to be used with Oracle ASM or other tools using raw devices.

A Dataset is assigned to one vServer. The Dataset names must be unique in the VDCF environment. The vServer Name is used as default prefix for the Dataset name.

#### 3.1.2 vServer

As a first step a new vServer is defined in the Configuration Repository held on the management server. Every vServer requires at least one Dataset for the vServers Root filesystem. On top of the Datasets file systems of different types (root,data or lofs) must be created. A minimum of one network configuration is also required.

A network configuration requires an IP addresses and the selection of a network type (management, public, backup).

After completion of the configuration, the vServer is deployed to the Node by the "commit" operation. At this time all the configured resources (Datasets, Filesystems, Networks) are created and the vServer will be installed. The vServer is configured at the first boot.

## 3.2 vServer - Initial Definitions

### 3.2.1 vServer

When defining a vServer it must be assigned to an existing node. This defines where the vServer will be created. The vServer must have a unique name, which is also used as the `hostname`, this usually matches the public ip address defined in the local DNS.

There are four types of vServer: FULL, SPARSE, SOL8 and SOL9. The types FULL and SPARSE are described within the Solaris Administration Collection for creating Zones. The `type` argument is optional. If not specified a default will be taken as defined in `customize.cfg`.

vServers of type SOL8 and SOL9 are “Solaris8 and Solaris 9 Container”. See Appendix 7.3 “Solaris 8 Branded Zones” for details and requirements.

```
% vserver -c create name=v0001 node=s0004 comment="App UnitTests"
```

### Category and Priority

You may assign a Category and/or Priority to your vServer. Categorizing allows you to separate Productive and Test/Development vServer.

The Category and Priority attributes are used from the Enterprise Features “Node Evacuation and HA Monitoring / Automated Failover”. The vServer are evacuated based on the configured Category order and inside the Category by Priority order. Using these attributes you define which vServer should be evacuated first.

Consult the “VDCF Base Administration Guide / chapter 6.7 Node Evacuation” and “VDCF Monitoring Guide” for more information about Node Evacuation and HA Monitoring.

### 3.2.2 Dataset

Every vServer (that is not residing on a node's local disk) requires at least his own Root-Dataset, where the root filesystem and optional data filesystems can be placed. Every vServer requires at least one LUN.

The LUN must be visible to the target node. Display the list of available LUNs with the following command:

```
% diskadm -c show free node=s0004
```

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

When creating a new dataset you can choose a name, type and your preferred layout. By default vserver is used as a prefix, resulting in a dataset name of v0001\_root. Use the globalname option to disable this. You define then the dataset name only using the name argument.

#### a) Single Lun

```
% dataset -c create name=root vserver=v0001 layout=600015D00002EE0...040EA
```

#### b) Mirror

```
% dataset -c create name=root vserver=v0001 \  

  layout="mirror 600015D00002EE0...040EA 00015D00002EE0...040F0"
```

#### c) Stripe

```
% dataset -c create name=myapp1_oradata vserver=v0001 globalname \  

  layout="600015D00002EE0...040EA 00015D00002EE0...040F0"
```

#### d) Concatenation

```
% dataset -c create name=root vserver=v0001 \  

  layout=600015D00002EE0...040EA  

% dataset -c commit name=v0001_root  

% dataset -c add name=v0001_root \  

  layout=00015D00002EE0...040F0  

% dataset -c commit name=v0001_root
```

### Delegated ZPOOL

The option 'delegated' is used to create a ZPOOL dataset which is delegated into a vServer. These kind of ZPOOLS can be manipulated within that vServer.

### RAW datasets

The dataset type RAW is used to build a group of disks, that are delegated as raw devices into a vServer to be used with Oracle ASM or other tools using raw devices. The file owner and group of the raw devices can be defined using the configuration variable DISKS\_OWNER\_RAW:

```
export DISKS_OWNER_RAW=owner[:group][,slices]
```

Slice can be 'all' or a specific disk slice number. This owner/group setting is applied to the devices on the first commit of that dataset only. To make the new raw devices available within the vServer the vServer must be rebooted.

## Disk location check for datasets

If you create or modify a dataset the following rules are checked, based on the dataset layout and the disk location configuration. These rules ensure your dataset keeps functional even though disks of one location are not available.

### 1. Non-Mirror Dataset

All GUIDs have to be in the same location

### 2. Mirror Dataset

All GUIDs of a submirror have to be in the same location

Submirrors have to be in at least 2 different locations

Non-compliant layouts are rejected by default:

```
$ diskadm -c show free

Use-Type Dev-Type State  GUID                               Serial  Size/MB  Location
FREE     MPXIO     UNUSED  6001438012599B9B0001100001B80000  PA0U06A  5120    HPEVA
FREE     MPXIO     UNUSED  6001438012599B9B0001100001BC0000  PA0U06A  5120    HPEVA

$ dataset -c create name=test vserver=v0100 layout="mirror
6001438012599B9B0001100001B80000 6001438012599B9B0001100001BC0000"

Creating vServer dataset <v0100_test>
ERROR: Submirrors have to be in at least 2 different locations
ERROR: could not create dataset: v0100_test
ERROR: failed to create dataset
```

To allow non-compliant datasets add the following configuration to the customize.cfg. It is not recommended to set this variable, because it allows to create mirrored datasets, which might fail in disaster scenarios.

```
export DATASET_CHECK_LOCATIONS_ENFORCE=FALSE
```

The conformance of the existing datasets is displayed using the `dataset show` operation. You should revise non-compliant datasets by replacing disks accordingly.

```
$ dataset -c show

Name          State      Size/MB  Avail/MB  Type    Owner   Node    Layout
v0100_data    ACTIVATED 10240    10240     ZPOOL   v0100   s0010
6001438012599B9B0000A000002F0000 (compliant)

v0100_test    DEFINED   5120     5120     ZPOOL   v0100   s0010   mirror
6001438012599B9B0001100001B80000 6001438012599B9B0001100001BC0000 (non-compliant)
```

### 3.2.3 Filesystems

#### root filesystem

Define the root filesystem on your new Dataset with enough space to install the Solaris Zone. For vServers of type SPARSE this takes approximately 500MB and for FULL types around 2GB, this is dependent on the build size.

```
% vsriver -c addfs name=v0001 type=root dataset=v0001_root size=4g
```

To create a vServer on a node's local disk you have to omit the dataset argument and use the `local` flag instead. You can't migrate such vServers to another node. This feature is useful for nodes without access to a central storage.

```
% vsriver -c addfs name=v0001 type=root local
```

#### data filesystem

It is recommended to store application data on separate data filesystems defined on the same or on a separate Dataset other than the root filesystem.

```
% vsriver -c addfs name=v0001 type=data \  
dataset=v0001_root size=5000 mountpoint=/export
```

#### lofs Filesystem

To share data from the Node to the vServer you may define lofs filesystems. The source directory will be created if it doesn't already exist. Sharing directories into a vServer introduces migration restrictions that require the source directory to be manually copied to the target node!

```
% vsriver -c addfs name=v0001 type=lofs \  
directory=/export/share mountpoint=/myshare
```

### 3.2.4 Network

A vServer typically has two IP addresses, one for the management network for server management and a second in the public network used by the applications. You may use ip-addresses or host names for the `ipaddr` argument.

```
% vsriver -c addnet name=v0001 type=management \  
    ipaddr=10.1.1.101 netmask=255.255.255.0  
  
% vsriver -c addnet name=v0001 type=public ipaddr=v0001
```

The available virtual network types and the mapping to the node interface type must be defined in the `customize.cfg` configuration file. To display the current definitions use the following command:

```
% vdcfadm -c show_config | grep VIRTUAL_NET_M  
VIRTUAL_NET_MAPPING management:MNGT public:PUBL backup:BACK
```

### Network Isolation

Multiple vServers on the same Node are by default allowed to communicate with each other using the Global Zone's Network stack. To isolate the vServer network interface from the other vServers use the optional `stack` argument. Using the value "private" isolates the vServer network interface from the other vServers on the same Node. The vServer network interface is not isolated from other vServers on different Nodes. This has to be done using Firewall technologies.

```
% vsriver -c addnet name=v0001 type=public \  
    ipaddr=v0001 netmask=255.255.255.0 stack=private
```

Set your preferred default value in the VDCF `customize.cfg` as `VIRTUAL_NETSTACK_DEFAULT` (SHARED, PRIVATE or EXCLUSIVE).

### Exclusive IP-Stack

If your nodes have enough network interfaces you may assign the physical or a VLAN interface exclusively to your vServer using the `stack=exclusive` setting. To use this setting, you need to unplug the interface on the Node. In the Node configuration clear the ip address using `nodecfg -c modify_net name= interface= ipaddr=-`.

Only the vServer is then able to use this network interface. The vServer get his own network stack, which means all the vServer interfaces need to be of type exclusive and the routing needs to be defined inside the vServer. Using this option the root user of the vServer gains access to the network, because he is able to snoop the interface.

### VLAN

If your are using the tagged VLAN technology (IEEE 802.1Q) in your network infrastructure, you may define the VLAN ID for your vServer network interfaces with the optional '`vlan`' argument.

```
% vsriver -c addnet name=v0001 type=public \  
    ipaddr=v0001 netmask=255.255.255.0 vlan=130
```

### 3.3 vServer - Installation

After completing the initial vServer definitions, you should display and review the vServer configuration using the following command:

```
% vserver -c show name=v0001
```

All the defined resources will be created on the target node using the commit operation:

```
% vserver -c commit name=v0001
```

After the commit, the vServer is installed on the node and ready to boot. Display the vServer Console to view the operation as the vServer boots:

```
% vserver -c console name=v0001
```

From another terminal:

```
% vserver -c boot name=v0001
```

### 3.4 vServer - Operations

Apart from run level functionality (boot, shutdown and reboot) the framework offers the possibility to modify the datasets, file systems and network interfaces. The administrator decides whether the changes are made real-time in the running vServer or at the next Reboot of the vServer.

- **Datasets:** Adding and Removing
- **Filesystems:** Adding, Growing, Renaming, Mounting, Removing
- **Network:** Adding and Removing

To activate a change on next reboot use the 'commit' command without options.

To destroy data, e.g. Removing a filesystem or dataset, the 'remove' option of the 'commit' command has to be used.

To activate a change on a running vServer you must use the 'exec' option.  
New filesystems are mounted and network interfaces activated or deactivated in the active vServer.

#### 3.4.1 Mount / unmount filesystems

Sometimes you have to unmount or mount filesystems on a running vServer. The mount operation is e.g. useful when you missed the 'exec' option while committing new filesystems. In this situation the vserver operation 'mount' can be used:

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

Arguments can be a single mountpoint or a dataset name to mount all filesystems of that dataset.

Same functionality is also available for unmounting filesystems:

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

It's not possible to mount or unmount a vServers root filesystem.  
Only data and loopback (lofs) filesystems are supported by this operations.

### 3.4.2 Rename filesystems

Using the rename filesystem operation it's possible to rename existing filesystem mountpoints. With the optional argument 'remount' the filesystem is renamed and remounted on the vServer.

```
$ vsverver -c renamefs      name=<vServer name>  
                           mountpoint=</directory> to=</newdirectory>
```

Only data and loopback (lofs) filesystems are supported by this operations.

### 3.4.3 Manipulate Mirrors (ZFS and SVM)

VDCF gives you some useful commands to manage mirrored or non-mirrored datasets.

#### Attach additional mirror to dataset

You may attach an additional mirror to an existing dataset by using the command `dataset -c attach_mirror`. This command only updates the VDCF repository. Uncommitted changes of the dataset layout are indicated by an asterisk (\*). To effectively change the dataset on the node you must use the command `dataset -c commit`.

```
% dataset -c attach_mirror name=<dataset name> layout=<mirror layout description>  
% dataset -c commit name=<dataset name>
```

Depending the type of the dataset you may respect the following rules when adding a mirror:

#### a) Dataset of type ZPOOL:

- The number of disks in the new submirror must fit the number of disks in the existing dataset
- The size of each disk must be equal or greater than the size of their counterpart in the existing dataset.

#### b) Dataset of type DISKSET:

- The total size of the to be added disks must be greater or equal the existing size of the dataset.

Example:

Dataset layout before update: "mirror disk1 disk2 mirror disk3 disk4"

```
% dataset -c attach_mirror name=<dataset name> layout="new_disk5 new_disk6"
```

Dataset layout after update: "mirror disk1 disk2 new\_disk5 mirror disk3 disk4 new\_disk6"

Additionally the disk locations are checked as described in chapter 3.2.2

### Detach mirror from dataset

Or you may remove a mirror from an existing dataset. Use the command `dataset -c detach_mirror` for this. As for the `attach_mirror` command you have to commit the change using the `dataset -c commit` operation.

```
% dataset -c detach_mirror name=<dataset name> mirror=<mirror>  
% dataset -c commit name=<dataset name>
```

The mirror argument is used to specify which mirror should be removed from the dataset layout.

Example:

dataset layout before update: "mirror disk1 disk2 mirror disk3 disk4"

```
% dataset -c detach_mirror name=<dataset name> mirror=2nd
```

dataset layout string after update: "disk1 disk3"

After the removal of a mirror the size of the dataset may grow if the detached mirror was smaller than the remaining mirrors of that dataset.

### 3.4.4 Display and manipulate ZFS

VDCF gives you some useful commands to manage your ZFS filesystems:

a) List ZFS filesystems and snapshots:

```
$ zfsadm -c show
```

b) Create a ZFS snapshots

```
$ zfsadm -c snapshot
```

c) Rollback ZFS filesystem to a previous snapshot

```
$ zfsadm -c rollback
```

d) Destroy ZFS filesystem snapshots

```
$ zfsadm -c destroy
```

e) List ZFS filesystems properties

```
$ zfsadm -c get
```

f) Set ZFS filesystems properties

```
$ zfsadm -c set
```

See the manpages for details about the zfsadm operations.

## 3.5 vServer - Migration

At installation time of a new vServer the performance of the target node should match the requirements. But after a while the requirements of the vServer may change. Or some maintenance work must be done on the node, which should not impact the availability of the vServer and the applications.

To solve these issues the framework is able to migrate a vServer from one node to another. The nodes must be installed with the same build and have the same patch level to use this feature. Additionally both nodes must have access to the Datasets the vServer uses.

A migration is only allowed to nodes which are in the same ComputePool (cPool) as the current Node.

### Migrate or Evacuate

There are two ways to select the target node where vServers should be migrated to. First option is to set the target Node manually using the `vserver migrate` operation. The other option is to let VDCF select the target Node when evacuating all vServer of a Node. Evacuating is only supported if the VDCF Monitoring (Enterprise Feature) is installed.

### 3.5.1 Typical Migration

You display the candidate nodes using the `vserver show` operation:

```
% vserver -c show name=s0100 candidates
```

vServer Name	Type	State	Node	cPool	Build	Patch-Level	Comment
s0100	FULL	ACTIVATED	s0054	PROD	5.10sv_u8w_req	142900-12 (U8+)	v 100

Potential Nodes	Dataset Access	Matching Patch-Level
s0003 (U9):	yes	no [upgrade]
s0051 (U8):	yes	no
s0053 (U8+):	yes	yes
s0055 (U8):	yes	no

Using the `migrate` operation you migrate one vServer or all vServers from a source node to a new node. The vServer must be stopped (rState: installed) to migrate.

#### a) Migrate one vServer

```
% vserver -c migrate name=v0001 node=compute2 shutdown
Migrate vServer v0001 from Node computel to compute2.
vServer v0001 is down.
vServer v0001 is detached from Node computel.
vServer v0001 is attached to compute2.
vServer v0001 boot issued ...
vServer migrated successful.
```

#### b) Migrate all vServers of a Node

```
% vserver -c migrate source=computel node=compute2 shutdown
Migrate vServer v0001 from Node computel to compute2.
vServer v0001 is down.
vServer v0001 is detached from Node computel.
vServer v0001 is attached to compute2.
vServer v0001 boot issued ...
vServer migrated successful.
```

Typically the vServer's are running and must be stopped as first step using the `'shutdown'` option. After the migration the vServer is booted automatically unless the `'noboot'` option is given.

### c) Evacuate all vServers of a node

The evacuation feature distributes the vServers from one Node to the other compatible Nodes which have enough resources (CPU and RAM).

For more information about `node -c evacuate` please consult the “VDCF Base – Administration Guide”.

```
% node -c evacuate name=s0051

evacuating node s0051 - this may take a moment ...
Starting evacuation of Node s0051.
Trying to evacuate vServers: s0246 s0186
Now we do a iteration of vServer distribution from Node s0051 ...
Target node for vServer <s0246> selected: <s0004>
Target node for vServer <s0186> selected: <s0004>
Doing normal detach of vServer <s0246> ...
Doing normal detach of vServer <s0186> ...
Attaching vServers s0246 s0186 to Node s0004 ...
Doing attach of vServer <s0246> to Node <s0004> ...
vServer <s0246> successfully attached. Now booting ...
Doing attach of vServer <s0186> to Node <s0004> ...
vServer <s0186> successfully attached. Now booting ...
There are no more vServers left on Node s0051. Finished
All vServers successfully evacuated
Evacuation of node s0051 finished.
node successfully evacuated
```



### 3.5.2 Detach/Attach

The framework additionally supports the execution of the 4 migration steps (shutdown, detach, attach, boot) as individual operations. This is useful if you don't need particular vServers running. You may detach the vServers from a Node for storage maintenance. After the maintenance you can re-attach the vServers again to the node.

A stopped vServer can be detached from the current node. This step also exports all datasets of the vServer from the Node.

```
% vserver -c detach name=v0001
```

The attach operation then imports the Datasets. The VDCF framework checks the Operating System Version compatibility and makes the necessary configuration adjustments in the vServer configuration. This operation ensures all the required data filesystems of your vServer are available.

```
% vserver -c attach name=v0001
```

#### NOTE: lofs

For lofs filesystems which reference a directory of the compute node, you must manually migrate the data to your target compute node before attaching the vServer!

#### Attach/Upgrade

If attaching to a node with installed Solaris 10 U6 or later, you may set the 'upgrade' flag to use the zoneadm upgrade (-u) option. This feature updates the dependent packages/patches to higher-revisions. See vserver\_attach(1M) and zoneadm(1M) for details.

If the vServer root filesystem resides on a ZFS filesystem, VDCF creates a ZFS snapshot. This snapshot can be used to revert the upgrade, if the vServer should be migrated back to the original Node.

You may use the upgrade argument on vserver migration or attach functions:

```
% vserver -c migrate name=s0243 node=s0052 upgrade
```

or

```
% vserver -c attach name=s0243 upgrade
```

```
attaching virtual server(s)
waiting for snapshot to complete ...
waiting for snapshot to complete ...
waiting for snapshot to complete ...
waiting for snapshot to complete ...
snapshot completed: s0243_root/root@vdcf_upgrade_2009_03_09-09:02:51
Updating vServer Patch-Level ...
node being checked: s0052
node checked successfully: s0052
patch deployment for node updated: s0052
Virtual Server successfully attached.
attach successful
```

```
% zfsadm -c show vserver=s0243 snapshots
```

```
Dataset list for vServer: s0243
```

Pool Name:	NAME	USED	AVAIL	REFER	MOUNTPOINT
s0243_root	s0243_root/root@vdcf_upgrade_2009_03_09-09:02:51	1.24M	-	24.1M	-

## 3.6 vServer - Disaster Recovery

If one of your compute nodes goes out of service two recovery options are available to recover the vServers which were running on the out of service compute node.

### 3.6.1 Reinstall the compute node

After a fresh install of your compute node (`node -c install`) the vServer are still in the state "ACTIVATED". You have to first detach the vServer and then re-attach them to the node. Without the node argument the vServer is automatically attached to the node to which it was last attached.

```
% vserver -c detach name=v0001
% vserver -c attach name=v0001
```

### 3.6.2 Migrate the vServer to another existing node

#### a) Recovery

Because your node isn't accessible at this time, you must execute a forced detach. This operation only updates the configuration repository. The old node still references the Datasets and has the vServers configured. To avoid conflicts you should not try to boot your old node.

```
% vserver -c detach name=v0001 force
% vserver -c attach name=v0001 node=<failover node>
% vserver -c boot name=v0001
```

The underlying dataset implementation may refuse to import to the new node, because the dataset belongs to the out of service node. You have to use the force option with the attach operation in this case.

Attention: Now you shouldn't boot the failed server. Because the node would boot the migrated zone as well and tries to import datasets twice. Which could lead to a corrupt dataset.

#### b) Failback

If you would like to reboot the original compute node and to avoid problems with the vServers installed there you have to make this failback first:

Shutdown and detach from the failover node:

```
% vserver -c shutdown name=v0001
% vserver -c detach name=v0001
```

Now reboot the original failed node:

```
{28} ok boot
```

Finally re-attach the vServer again:

```
% vserver -c attach name=v0001 node=<original node>
```

## 3.7 vServer - Cleanup, Re-Installation and Remove

### 3.7.1 Cleanup vServer

The last step in the vServer's life-cycle is its cleanup. It is possible to reuse vServer definitions for other applications. Doing so it is recommended to re-install the vServer because the existing applications may have modified the Solaris environment inside the vServer. Before re-installing the vServer you must destroy all data on the filesystems of the vServer.

The 'commit uninstall' operation destroys all data on the filesystems and changes the state of the vServer to DEFINED status.

```
% vserver -c shutdown name=v0001
% vserver -c commit name=v0001 uninstall
% vserver -c show name=v0001
```

To re-install you must first re-create the vServer using the `commit` operation. If you need to destroy all definitions of the vServer refer to the steps described in chapter 3.7.2

```
% vserver -c commit name=v0001
```

### 3.7.2 Remove vServer

To completely remove all definitions of the vServer you must first cleanup the vServer as described in chapter 3.7.1. Removal of the filesystems, network definitions, datasets and finally removal of the vServer itself is now possible.

```
% vserver -c remfs mountpoint=all name=v0001
% vserver -c commit name=v0001 remove
% vserver -c remnet type=all name=v0001
% dataset -c remove name=v0001_root
% dataset -c commit name=v0001_root
% vserver -c remove name=v0001
```

After this steps the vServer is completely destroyed.

### 3.7.3 Remove a DETACHED vServer

It is also possible to remove a vServer in a DETACHED state. This same procedure also applies if a vServer is no longer attached to a Node, for example because the Node has been reused/reinstalled. In this case the vServer has not reached a DETACHED state. Instead, it still might be in ACTIVATED state and needs to be placed in DETACHED state first. Use the detach force command to move a vServer into DETACHED state.

```
% vserver -c detach name=v0001 force
```

Once a vServer has reached a detach state it can be removed using the following commands.

```
% vserver -c remove name=v0001 force
% dataset -c remove name=v0001_root force
```



### **3.8 Virtual pools (vPools) - Permission to manage vServers**

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 vServer Runtime States

### 4.1 Overview

The Runtime States (rState) of vServers is displayed using the `'vserver -c show'` commands.

<b>vServer rState</b>	The vServer rState is taken from the zoneadm command. See zones(5) manpage for details.
UNKNOWN	The vServer state is unknown, because no ssh connection to the Node could be established.
CONFIGURED	Indicates that the configuration for the zone has been completely specified and committed to stable storage.
INCOMPLETE	Indicates that the zone is in the midst of being installed or uninstalled, or was interrupted in the midst of such a transition.
INSTALLED	Indicates that the zone's configuration has been instantiated on the system: packages have been installed under the zone's root path.
READY	Indicates that the "virtual platform" for the zone has been established. Network interfaces have been plumbed, file systems have been mounted, devices have been configured, but no processes associated with the zone have been started.
RUNNING	Indicates that user processes associated with the zone application environment are running.
SHUTTING_DOWN DOWN	Indicates that the zone is being halted. The zone can become stuck in one of these states if it is unable to tear down the application environment state (such as mounted file systems) or if some portion of the virtual platform cannot be destroyed. Such cases require operator intervention.

### 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 (vServer, Disk, Dataset, Filesystem, 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.

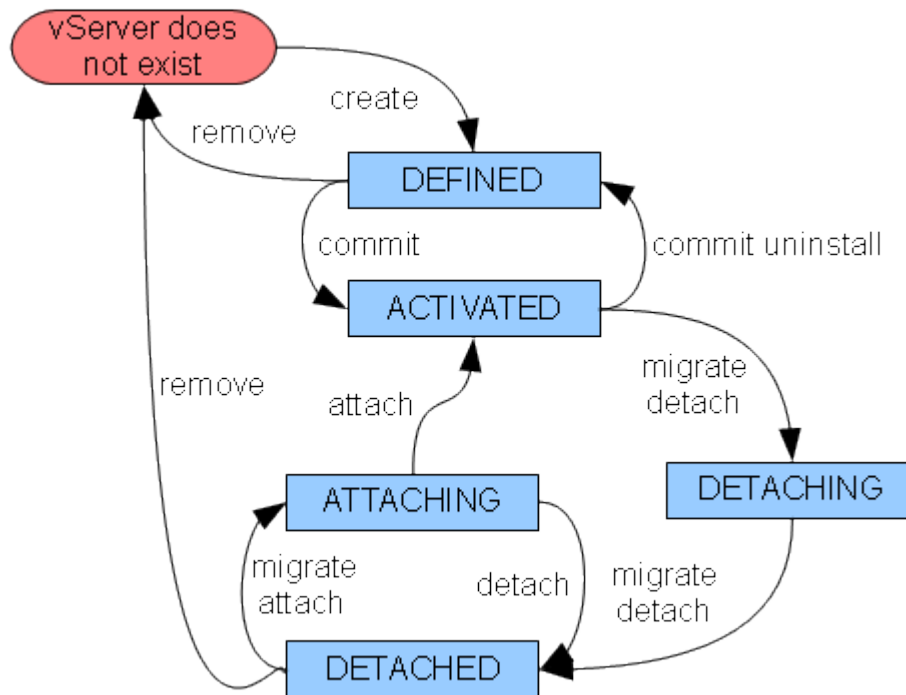
### 5.2 Possible state values

<b>vServer cStates:</b>	DEFINED, ACTIVATED, DETACHING, DETACHED, ATTACHING
<b>Disk (LUN) states:</b>	UNUSED, DEFINED, ACTIVATED, PURGING
<b>Dataset states:</b>	DEFINED, ACTIVATED, DETACHED, PURGING
<b>Filesystem states:</b>	DEFINED, ACTIVATED, PURGING
<b>Network states:</b>	DEFINED, ACTIVATED, PURGING

### 5.3 State values explained

DEFINED	Indicates that the object exists only in the VDCF repository
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 vServer from a node.
ATTACHING	Indicates that VDCF is or was trying to attach the vServer to a node.

## 5.4 State diagram vserver



## 5.5 Supported vsver commands

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

These operations are always available:

- `vserver -c show`
- `vserver -c modify`

While the vsver has the state **DEFINED** these vsver operations are possible:

- `vserver -c remove` (only if no datasets, filesystems or networks assigned)
- `vserver -c addfs`
- `vserver -c growfs`
- `vserver -c renamefs`
- `vserver -c remfs`
- `vserver -c addnet`
- `vserver -c remnet`
- `vserver -c revert`
- `vserver -c commit`
- `vserver -c zfs*` (deprecated, use `zfsadm`)

While the vServer has the state **ACTIVATED** these vserver operations are possible:

- `vserver -c addfs`
- `vserver -c growfs`
- `vserver -c renamefs`
- `vserver -c remfs`
- `vserver -c mount`
- `vserver -c unmount`
- `vserver -c addnet`
- `vserver -c remnet`
- `vserver -c revert`
- `vserver -c commit`
- `vserver -c migrate`
- `vserver -c detach`
- `vserver -c boot`
- `vserver -c reboot`
- `vserver -c shutdown`
- `vserver -c console`
- `vserver -c zfs*` **(deprecated, use zfsadm)**

While the vServer has the state **DETACHING** these vserver operations are possible:

- `vserver -c detach`
- `vserver -c migrate`

While the vServer has the state **DETACHED** these vserver operations are possible:

- `vserver -c attach`
- `vserver -c migrate`
- `vserver -c remove`

While the vServer has the state **ATTACHING** these vserver operations are possible:

- `vserver -c attach`
- `vserver -c detach`
- `vserver -c zfs*` **(deprecated, use zfsadm)**

## 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 an administrator appropriate access to the required VDCF commands.

RBAC profiles of VDCF vServer package:

VDCF virtual Module	vServer 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
```

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
```

## 7 Appendixes

### 7.1 Data File Overview

#### 7.1.1 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

Internal vServer Configuration Files and logfiles are stored in `/var/tmp/zonecfg/<vserver>`  
Post installation Scripts and logfiles are stored in `/var/tmp/vdcf`

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 vServer for a customer environment. To overwrite a VDCF variable add the appropriate value to `customize.cfg`:

Variable name	Description
DATASET_CHECK_LOCATIONS_ENFORCE	Enforce disk location check (TRUE/FALSE)
DATASET_DEFAULT_TYPE	Default Dataset type
DATASET_METASIZE	Dataset Metasize
DISKS_DEFAULT_METHODS	Methods for discovering disks. Valid options are: MPXIO ISCSI (VXVM: +DMP -MPXIO)
FS_VALID_OPTIONS	allowed filesystem options, when creating filesystems. see <code>vserver_adddfs(1M)</code> for details.
VIRTUAL_BRAND_IMAGEDIR	Directory for branded zones flash archives
VIRTUAL_ESCAPE_DEFAULT	Escape character to exit the console
VIRTUAL_NET_MAPPING	here you define the network types to be used when adding networks to a vserver using <code>vserver_addnet</code> . the format is <code>&lt;virtualnetwork&gt;:&lt;physicalnetwork&gt;</code> . a virtualnetwork with the name 'management' is required. supported physicalnetwork's are MNGT,PUBL,BACK or any IPMP alias (see <code>CONFIG_IPMP_ALIASES</code> variable).
VIRTUAL_NETSTACK_DEFAULT	default network stack used when adding networks to vServers. allowed values are: SHARED, PRIVATE. see <code>vserver_addnet(1M)</code> for details.
VIRTUAL_REMOVE_MNGT_NET	Remove management network from vserver automatically after installation
VIRTUAL_REMOVE_MNGT_NET_SLEEP	Sleep time before committing the auto remove of the management network
VIRTUAL_SGROUP_DEFAULT	default server configuration group used when creating vservers. see <code>vserver_create(1M)</code> for details.
VIRTUAL_TYPE_DEFAULT	default vserver type: SPARSE, FULL
VPOOL_ENABLED	if TRUE, vpool checks are enabled Default is FALSE
ZONE_ROOTDIR	Rootdirectory of zones

## 7.3 Solaris 8 Containers (Branded Zones)

Solaris 8 Branded Zones is a licensed feature of Oracle to create non-Global Zones on a Solaris 10 Node based on an image taken from an existing Solaris 8 System.

Read the Oracle documentation “System Administration Guide: Solaris 8 Branded Zones” for details.

### 7.3.1 Requirements

- Download and install the “Solaris 8 Containers 1.0”, formerly known as Solaris Migration Assistant from Oracle.
- Target Nodes must be installed using Solaris 10 Update 4 (8/07) or later
- Target Nodes require the Patch 127111-01 or later
- Target Nodes require the Solaris 8 Migration Assistant Packages installed  
(SUNWs8brandr SUNWs8brandu SUNWs8brandk)

### 7.3.2 SOL8 vServer

When committing the SOL8 vServer the Solaris 8 image must be available on the Target Node  
`as /var/tmp/images/<vserver>.flar`

This directory or file may be a link to another directory. For example

```
-bash-3.00$ cd /var/tmp
-bash-3.00$ ls -l images
lrwxrwxrwx  1 root    root          49 Dec 19 11:05 images ->
/net/masterserver/export/images
```

## 7.4 Solaris 9 Containers (Branded Zones)

Solaris 9 Containers is a licensed Feature of Oracle to create non-Global Zones on a Solaris 10 Node based on an image taken from an existing Solaris 9 System.

Read the Oracle documentation “System Administration Guide: Solaris 9 Containers ” for details.

### 7.4.1 Requirements

- Download and install the “Solaris 9 Containers 1.0” from Oracle.
- Target Nodes must be installed using Solaris 10 Update 4 (8/07) or later
- Target Nodes require the Patch 127111-01 or later
- Target Nodes require the Solaris 9 Containers Packages installed  
(SUNWs9brandr SUNWs9brandu SUNWs9brandk)

### 7.4.2 SOL9 vServer

When committing the SOL9 vServer the Solaris 9 image must be available on the Target Node  
`as /var/tmp/images/<vserver>.flar`

This directory or file may be a link to another directory. For example

```
-bash-3.00$ cd /var/tmp
-bash-3.00$ ls -l images
lrwxrwxrwx  1 root    root          49 Dec 19 11:05 images ->
/net/masterserver/export/images
```