

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

## VDCF

## Administration Guide

Version 10.0  
2 December 2025

Copyright © 2005-2025 JomaSoft GmbH  
All rights reserved.



## Table of Contents

1 Introduction.....	7
1.1 Overview.....	8
1.1.1 Consolidation.....	8
1.1.2 Virtualization.....	8
1.1.3 Solaris Zones (Containers).....	9
1.1.4 Logical Domains.....	9
1.1.5 Datacenter Architecture.....	10
1.1.6 Virtual Datacenter Cloud Framework (VDCF).....	11
1.1.7 VDCF Terminology.....	12
1.2 Supported Environments.....	13
2 VDCF Product Structure and Commands.....	14
2.1 Product Structure.....	14
2.1.1 Commands and Manual Pages.....	14
2.1.2 VDCF Datastore and Configuration.....	14
2.1.3 Logfiles.....	14
2.2 Commands.....	15
2.2.1 Commands for Node, Dataset, Patch, Packages, Config, Security.....	15
2.2.2 Commands for vServer/Zone Administration.....	16
2.2.3 Commands for LDom Administration.....	16
3 VDCF Administration Command.....	17
3.1 Version.....	17
3.2 Statistics.....	17
3.3 Configuration.....	17
3.4 Configuration Repository.....	18
3.5 Logfiles.....	18
3.6 VDCF Client Package.....	18
3.7 Pending changes.....	19
4 Node and disk management.....	20
4.1 Installable Solaris Builds.....	20
4.1.1 Solaris 10: Builds and flash archives.....	20
4.1.1.1 Flash archives.....	20
4.1.1.1.1 Solaris JumpStart install server.....	20
4.1.1.1.2 Configure the profile.....	21
4.1.1.1.3 JumpStart installation.....	21
4.1.1.1.4 Create the flash archive.....	21
4.1.1.2 Creating a VDCF Build.....	22
4.1.1.2.1 Create a Bootserver.....	22
4.1.1.2.2 Create an installable Build.....	23
4.1.2 Solaris 11: Builds/Unified Archives and IPS repositories.....	24
4.1.2.1 IPS repositories.....	24
4.1.2.1.1 Create a local IPS Solaris repository.....	24
4.1.2.1.2 Sync new updates into a local IPS Solaris repository.....	24
4.1.2.2 AI install services.....	25
4.1.2.2.1 Create an AI install service.....	25
4.1.2.2.2 Remove an AI install service.....	25
4.1.2.3 Unified Archives (UAR).....	26
4.1.2.3.1 Creating an UAR.....	26
4.1.2.4 Creating a VDCF Build.....	27
4.2 System Configuration.....	28
4.2.1 Base Configuration.....	28
4.2.2 Server Configuration.....	30
4.2.3 Server Configuration execution.....	32
4.2.3.1 Supported Configuration types.....	32
4.2.3.2 Security (RBAC,vpool).....	32
4.2.3.3 Customization.....	32
4.2.3.4 Command usage.....	33
4.2.3.5 Messages.....	34
4.2.3.6 Examples.....	35



4.2.4 Network Route Configuration (Solaris 11).....	38
4.3 Physical Server Management.....	39
4.3.1 Compute Pools.....	39
4.3.1.1 RBAC profiles.....	39
4.3.1.2 Default compute pool.....	40
4.3.1.3 Consistency check.....	40
4.3.1.4 Patchlevel statistics for a Compute Pool.....	41
4.3.2 Node Discover.....	42
4.3.2.1 Using root (Solaris 8/9/10 only).....	42
4.3.2.2 Using nonroot.....	42
4.3.2.3 SAN Boot.....	42
4.3.3 Node configuration.....	43
4.3.3.1 Automated Node configure (non-interactive).....	44
4.3.3.2 Node configure based on a Profile (interactive).....	45
4.3.3.3 Node Console.....	46
4.3.3.4 ASR SNMP Integration for ILOM.....	47
4.3.4 Node Install.....	48
4.3.4.1 Solaris 10 installation.....	48
4.3.4.2 Solaris 11 installation.....	50
4.3.4.2.1 Installing sparc nodes.....	51
4.3.4.2.2 Installing x86 nodes.....	51
4.3.4.2.3 Customization of generated AI xml files.....	52
4.3.5 Solaris Node Import.....	55
4.3.6 Linux Node Import.....	56
4.3.7 Node Operation.....	56
4.3.8 Node visible VLAN Verification.....	56
4.3.9 Node Configuration Updates.....	57
4.3.10 Node Kernel Settings.....	58
4.3.11 Node Evacuation.....	59
4.3.11.1 Requirements.....	59
4.3.11.2 Overview.....	59
4.3.11.3 vServer shutdown on target Nodes.....	59
4.3.11.4 Evacuation.....	60
4.3.12 SPARC Node Firmware Upgrade.....	61
4.3.13 Node Remove.....	62
4.4 Patch Management (Solaris 10).....	63
4.4.1 Introduction.....	63
4.4.2 VDCF Patch Architecture.....	63
4.4.2.1 Patch Spooling.....	63
4.4.2.2 Patch Analyzing.....	63
4.4.2.3 Patch download.....	64
4.4.3 VDCF Patch Configuration.....	65
4.4.3.1 Patch Sets.....	65
4.4.3.2 Patch Targets.....	65
4.4.4 VDCF Patch Installation.....	66
4.4.4.1 Patch Prepare.....	66
4.4.4.2 Patch Install.....	66
4.4.4.3 Zones Parallel Patching.....	66
4.4.4.4 Checking Patch Status.....	68
4.4.5 VDCF Patch-Level.....	69
4.4.5.1 Display installed Patches.....	69
4.4.5.2 Patch differences.....	70
4.5 Upgrading Solaris 11.....	71
4.5.1 Upgrade Trial-Run.....	72
4.5.2 Upgrade node.....	73
4.5.3 Upgrade failback.....	74
4.5.4 Additional: Node upgrade check.....	75
4.5.4.1 Configure upgrade paths.....	75
4.5.4.2 Optional settings.....	76
4.5.4.3 Usage Step 1: Upgrade Check.....	76



4.5.4.4 Usage Step 2: Node upgrade.....	76
4.5.4.5 Usage Step 3: finish or fallback.....	77
4.6 Package Management.....	78
4.6.1 Package information data.....	78
4.6.2 Analyzing and importing packages.....	78
4.6.3 Compare package levels.....	79
4.6.4 Querying packages.....	80
4.6.4.1 Search for packages.....	80
4.6.4.2 Common package information.....	82
4.6.4.3 Package deployment information.....	83
4.7 Disk Management.....	84
4.7.1 Overview.....	84
4.7.2 ZFS Volumes (ZVOL).....	84
4.7.2.1 Preparation.....	84
4.7.2.2 ZVOL Registration and Usage.....	84
4.7.3 Disk Registration.....	85
4.7.4 Physical disk location.....	86
4.7.5 Show / modify disk location / storage box.....	86
4.7.6 Verify disks (assigned to nodes in more than one compute pool).....	86
4.7.7 Dataset.....	87
4.7.8 Dataset fast provisioning.....	88
4.7.9 Dataset verify.....	88
4.7.9.1 Dataset verify update_size.....	88
4.7.10 Disk location check for datasets.....	89
4.7.11 Node SWAP dataset.....	90
4.7.12 Node DUMP dataset.....	91
4.7.13 Dataset ZPOOL Log Disk.....	91
4.7.14 Encrypted ZPOOL dataset.....	92
4.7.15 Node data filesystems.....	93
4.8 Node runtime states.....	94
4.8.1 Overview.....	94
4.8.2 Cronjob.....	94
5 Virtual Server (vServer) Management.....	95
5.1 Overview.....	95
5.1.1 Datasets.....	95
5.1.2 Native vServer.....	95
5.1.3 Kernel vServer.....	95
5.2 vServer - Initial Definitions.....	96
5.2.1 vServer.....	96
5.2.1.1 vServer 'autoboot'.....	97
5.2.1.2 vServer 'limitpriv'.....	97
5.2.1.3 Category and Priority.....	98
5.2.2 Dataset.....	99
5.2.2.1 Delegated ZPOOL.....	99
5.2.2.2 RAW datasets.....	99
5.2.2.3 Disk location check for datasets.....	100
5.2.3 Kernel vServer root disk.....	101
5.2.4 Filesystems.....	101
5.2.4.1 root filesystem for native vServer.....	101
5.2.4.2 data filesystem.....	101
5.2.4.3 lofs Filesystem.....	101
5.2.5 Filesystem Usage.....	102
5.2.6 Network.....	103
5.2.6.1 Network Isolation.....	103
5.2.6.2 Exclusive IP-Stack.....	104
5.2.6.3 VLAN.....	104
5.2.6.4 IPMP Probe IPs.....	104
5.3 vServer - Installation.....	105
5.3.1 vServer console.....	106
5.4 vServer - Operations.....	107
5.4.1 Mount / unmount filesystems.....	107

5.4.2 Apply vServer configurations online.....	108
5.4.2.1 Solaris 11.2 and later vServer usage examples.....	108
5.4.2.1.1 Add an additional exclusive network to a vServer.....	108
5.4.2.1.2 Remove an exclusive network from a vServer.....	108
5.4.2.1.3 Add RAW devices to a vServer.....	109
5.4.2.1.4 Remove RAW devices from a vServer.....	109
5.4.3 Rename filesystems.....	110
5.4.4 Cloning ZFS data filesystems.....	111
5.4.5 Dataset ZPOOL Replication.....	112
5.4.6 Dataset ZPOOL Disk remove.....	113
5.4.7 Manipulate Mirrors (ZFS and SVM).....	114
5.4.7.1 Attach additional mirror to dataset.....	114
5.4.7.2 Conservative ZPOOL Mirroring.....	114
5.4.7.3 Detach mirror from dataset.....	115
5.4.8 Display and manipulate ZFS.....	116
5.4.9 Immutable zones.....	117
5.5 vServer - Migration.....	118
5.5.1 Migration.....	119
5.5.2 Detach/Attach.....	120
5.5.3 Kernel vServer – Live Migration.....	121
5.6 vServer - Disaster Recovery.....	122
5.6.1 Reinstall the compute node.....	122
5.6.2 Migrate the vServer to another existing node.....	122
5.7 Solaris Branded Zones.....	123
5.7.1 Solaris 8 Containers (Branded Zones).....	123
5.7.1.1 Requirements.....	123
5.7.1.2 SOL8 vServer.....	123
5.7.2 Solaris 9 Containers (Branded Zones).....	124
5.7.2.1 Requirements.....	124
5.7.2.2 SOL9 vServer.....	124
5.7.3 Solaris 10 branded zones.....	125
5.7.3.1 Requirements.....	125
5.7.3.2 SOL10 vServer.....	125
5.8 vServer - Cleanup, Re-Installation and Remove.....	126
5.8.1 Cleanup vServer.....	126
5.8.2 Remove vServer.....	126
5.8.3 Remove a DETACHED vServer.....	126
5.9 Virtual pools (vPools) - Permission to manage vServers.....	127
5.10 vServer Dependencies.....	127
5.11 vServer Import – import existing zones into VDCF.....	128
5.12 vServer verify – verify the vServer configuration.....	129
5.13 vServer Runtime States.....	130
5.13.1 Overview.....	130
5.13.2 Cronjob.....	130
5.14 vServer Configuration States.....	131
5.14.1 Overview.....	131
5.14.2 vServer cState diagram.....	131
5.14.3 Supported vserver commands.....	132
6 Logical Domain Management.....	134
6.1 Overview.....	134
6.1.1 Control Domain (cdom).....	134
6.1.2 Guest Domain (gdom).....	134
6.1.3 Stages.....	135
6.2 Control domain (cdom).....	136
6.2.1 cdom definition.....	136
6.2.2 cdom creation.....	136
6.2.3 cdom remove.....	136
6.2.4 cdom discover.....	136
6.2.5 cdom show.....	137
6.2.6 cdom evacuation.....	137



6.3 Guest domain (gdom) configuration.....	138
6.3.1 gdom initial definition.....	138
6.3.2 gdom modifications.....	138
6.3.3 Disks (LUN).....	138
6.3.4 Network.....	139
6.3.4.1 Network type definitions.....	139
6.3.5 Summary.....	140
6.3.6 Installation.....	140
6.3.7 Console.....	141
6.4 Root IO Domains and Split IO GDom.....	142
6.4.1 Overview.....	142
6.4.2 Root IO Domain Import.....	142
6.4.3 Split IO GDom Setup.....	144
6.5 Guest domain (gdom) operation.....	145
6.5.1 Readonly GDoms.....	145
6.6 Guest domain (gdom) migration.....	146
6.6.1 Candidates.....	146
6.6.2 Migrate (live and cold).....	147
6.6.2.1 Live Migration of a guest domain.....	147
6.6.2.2 Cold Migration of a guest domain.....	147
6.6.3 Detach / Attach.....	147
6.7 Guest domain (gdom) cleanup / destroy.....	148
6.8 Virtual pools (vPools) - Permission to manage Guest domains.....	149
6.9 Runtime States (CDom and GDom).....	149
6.9.1 Overview.....	149
6.9.2 Cronjob.....	149
6.10 Configuration States (CDom and GDom).....	150
6.10.1 Overview.....	150
6.10.2 GDom cState diagram.....	150
6.10.3 Supported GDom commands.....	150
7 Configuration States.....	152
7.1 Overview.....	152
7.2 Possible cState values.....	152
7.3 cState values explained.....	152
8 Security Management.....	153
8.1 Management Server RBAC.....	153
8.2 Remote Execution / SSH Environment.....	154
9 Virtual pools (vPools) - Permission to manage Servers.....	154
9.0.1 Definition.....	154
9.0.2 Usage.....	155
10 Appendixes.....	156
10.1 Data File Overview.....	156
10.1.1 On VDCF Management Server.....	156
10.1.1.1 ntp.cfg.....	157
10.1.2 On Compute Nodes.....	157
10.2 Customization of the VDCF environment.....	158

## 1 Introduction

This documentation describes the main features of the Virtual Datacenter Cloud Framework (VDCF) for the Solaris Operating System, Version 10.0 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 Solaris 11</i>	for information about installing VDCF on Solaris 11
<i>VDCF – Proxy</i>	for information about running VDCF using Proxies
<i>VDCF – Quick Reference</i>	for a short command overview

See these documents for additional VDCF components:

<i>VDCF – Resource Management</i>	for information about VDCF Resource Management
<i>VDCF – Monitoring</i>	for information about VDCF Monitoring (HW, Resource, OS, Security Compliance and Hardening)
<i>VDCF – High Availability</i>	for information about VDCF HA (Automated Failover for Zones and LDOMs)
<i>VDCF – Linux</i>	for information about Redhat and Oracle Linux

These and all other VDCF documents can be found at:  
<https://www.jomasoft.ch/vdcf/#js-docu>

## 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 Zones (Containers)

The VDCF vServer component builds on top of a Virtualization technology called Solaris Zones (Containers). A Solaris Zone 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.

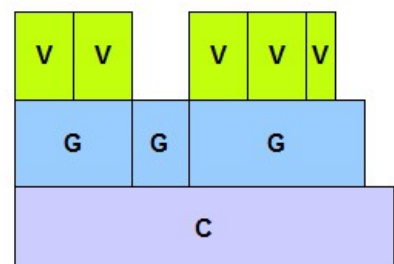
Since VDCF 8.0 Solaris Kernel Zones can also be deployed.

### 1.1.4 Logical Domains

The VDCF LDom component 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).

Usage	Technology	VDCF Command
Application Environment	Solaris Zone	vServer
Solaris OS	Guest Domain (LDM)	GDom (Node)
Hardware (I/O)	Control Domain (LDM)	CDom (Node)

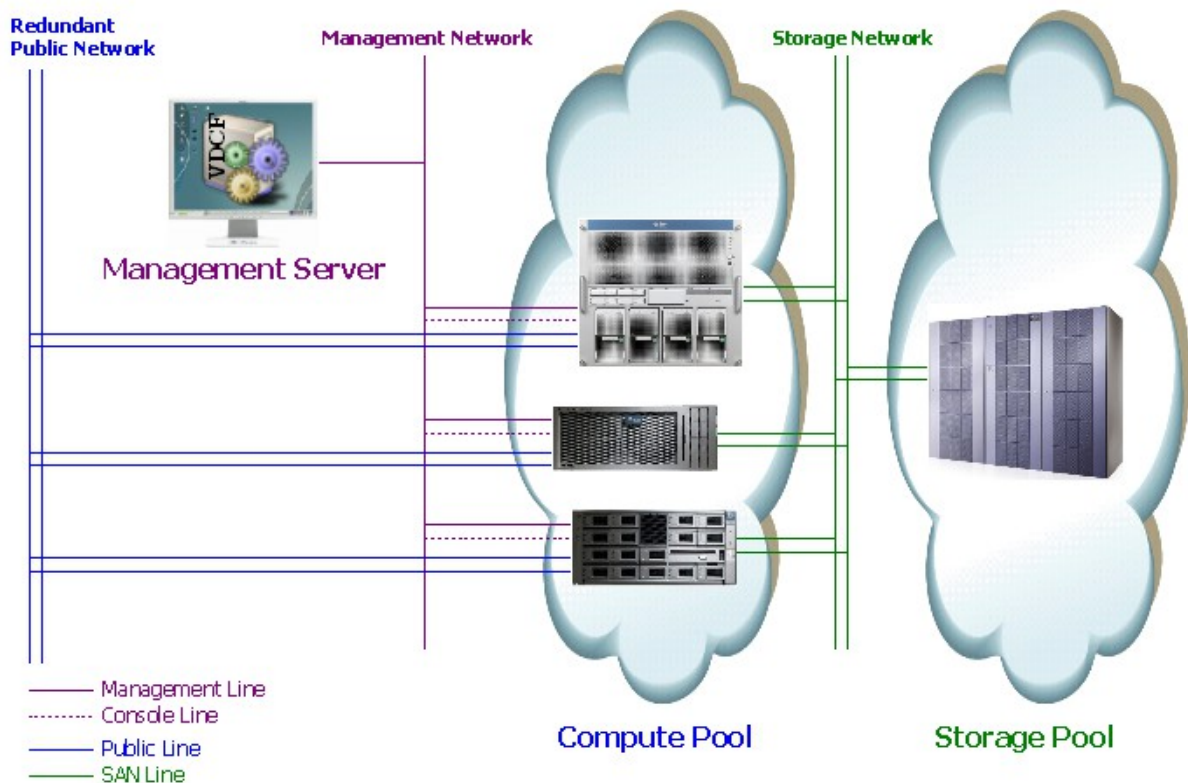


Oracle SPARC Server Setup using VDCF



### 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.

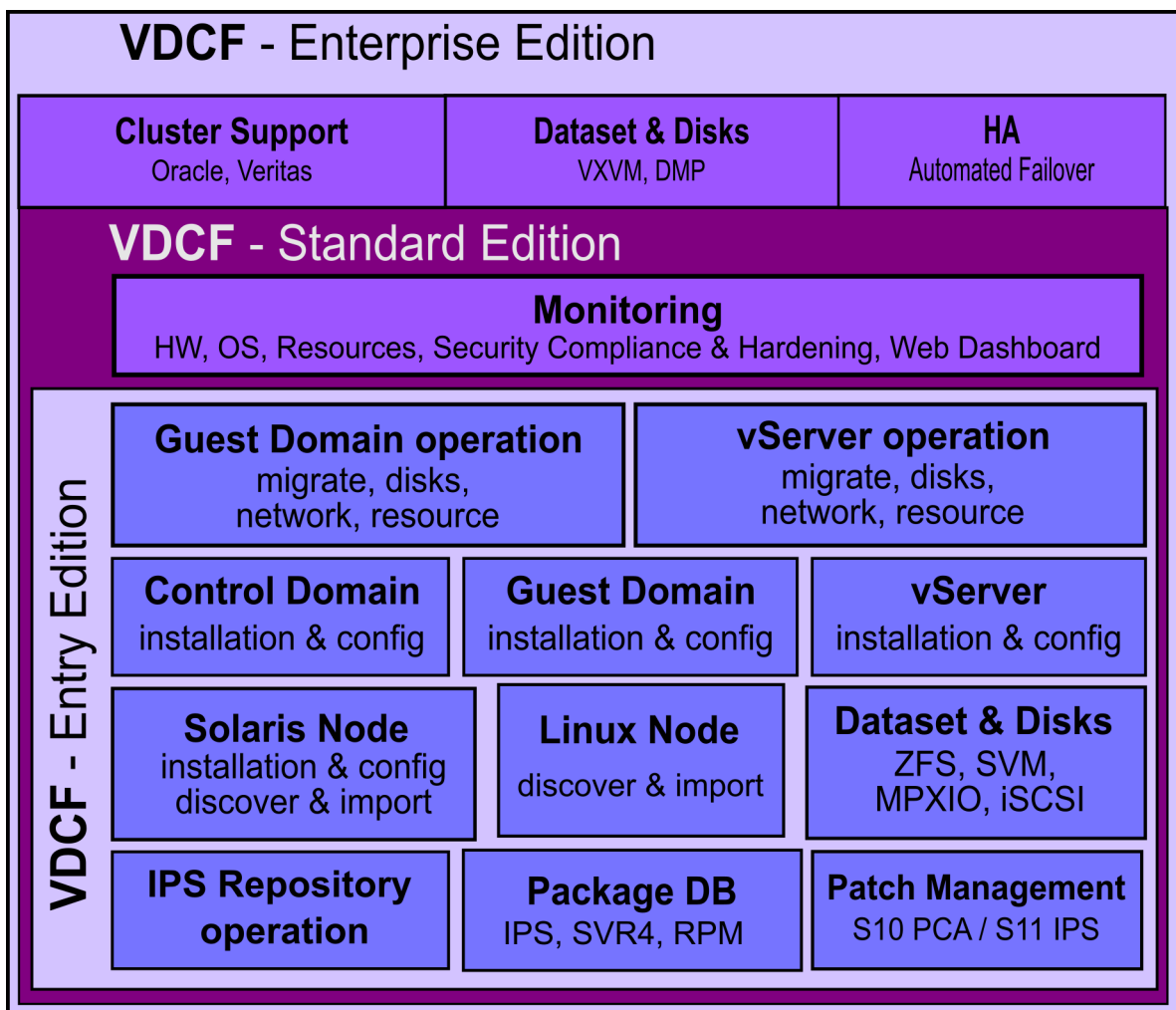
#### Storage Pool

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 Cloud 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 or 11 Zones and/or Logical Domains (Oracle VM Server for SPARC) controlled by a centralized management server.

With VDCF, JomaSoft offers a tool to simply and effectively operate and monitor your Solaris 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.



#### VDCF on Solaris 11.4

You can install VDCF only on Solaris 11.4. VDCF is able to install and operate your physical servers (Nodes / Control Domains), Guest Domains (LDoms) and your virtual servers (Containers/Zones) running on Solaris 10 or 11.

### 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:** Solaris non-global zone or kernel 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 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  
Fujitsu SPARC M10 and M12 Server
  - Solaris 11 Solaris 11.4
- Compute Node/Server Oracle SPARC Server and x86 Server  
Fujitsu SPARC M10 and M12 Server
- Solaris Operating System Solaris 10 Update 1 (1/06) up to Update 11 (1/13)  
Solaris 11.1, 11.2, 11.3 and 11.4
- Linux RedHat 8/9/10, Oracle Linux 8/9/10
- Logical Domains LDomS 1.1/1.2/1.3/2.0/2.1/2.2/3.0/3.1/3.2/3.3/3.4/3.5/3.6
- Branded Zones solaris8, solaris9, solaris10
- Kernel Zones Solaris 11.4
- Volume Manager ZFS, Solaris Volume Manager (SVM)
- Filesystem ZFS, Solaris UFS, lofs
- 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 ILOM, XSCF, SC/ALOM, RSC, SSC, 15K, ALOMCMT, ILOMx86
- Network Link aggregation, IPMP and tagged VLAN for LDomS and vServer  
vServer exclusive ip-stack

For VDCF Entry customers the following Extensions are available:

- Resource Management Administration of vServer Resource settings

For VDCF Standard and Enterprise customers the following Extensions are available:

- Monitoring Hardware, Resource and OS Monitoring, Compliance & Hardening,  
Web Dashboard

For VDCF Enterprise customers the following Extensions are available:

- HA High Availability/Automated Failover
- Veritas Dataset Volume Manager: VXVM, Filesystem: vxfs
- Sun/Solaris Cluster Integration of vServers in Sun Cluster  
Integration of Ldoms/GDomS in Solaris 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
<code>ai</code>	xml data and template files for Solaris Automated Installer (AI)

#### 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 following commands are available from the **VDCF base** package

### 2.2.1 Commands for Node, Dataset, Patch, Packages, Config, Security

<code>vdcfadm</code>	Admin command to display version, logfiles and configuration
<code>cpool</code>	Manages the compute pools, where the nodes/vServers are isolated
<code>nodecfg</code>	Analyzes and configures the physical servers (Nodes)
<code>console</code>	Manage the console settings of the physical servers (Nodes)
<code>config</code>	Used to manage basic system configuration (DNS,NTP,...)
<code>serverconfig</code>	Manages server configuration (Connecting system configs to nodes and vServers)
<code>routecfg</code>	Network Route configuration and verification
<code>build</code>	Tools to create and manage flash archives (builds) For Solaris 10 Nodes only
<code>flash</code>	Utility to manage flash archives and connections between builds and nodes. For Solaris 10 Nodes only
<code>ipsadm</code>	Utility to manage Solaris IPS repositories, AI services and builds
<code>node</code>	Node administration: Display, Install, Boot, Shutdown, etc.
<code>diskadm</code>	Manages Disks (LUNs)
<code>dataset</code>	Manages Datasets (Volumes)
<code>patchadm</code>	Patch Management (Analyse, Download, Configuration, Installation)
<code>vpkgadm</code>	Package Management (Analyse, Diff)
<code>vpool</code>	Manages the virtual pools, where vServers/GDoms and Physical Nodes are assigned to users

## 2.2.2 Commands for vServer/Zone Administration

vserver	Virtual Server Management (Zones), Configuration, Installation and Operations
zfsadm	Admin command for vServer related ZFS operations
dependadm	Admin command for vServer Dependencies

See **chapter 5** for more details.

## 2.2.3 Commands for LDom Administration

cdom	Control Domain Setup and administration
gdom	Guest Domain Setup and administration

See **chapter 6** for more details.

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 VDCF Administration Command

Using the command 'vdcfadm' you are able to display information about the installed VDCF version and its configuration. It allows you to manage the Configuration Repository and the Framework logfiles.

#### 3.1 Version

The 'show\_version' operation displays the currently installed VDCF components on your central management server.

```
% vdcfadm -c show_version

Package   Version  Arch.   Install-Date      Name
JSvdcf-base  5.7.0    sparc   Mar 31 2016 10:20  JomaSoft VDCF - Base
JSvdcf-monitor 2.5.0    sparc   Mar 31 2016 10:20  JomaSoft VDCF - Monitor
```

#### 3.2 Statistics

The 'statistics' operation counts the virtual and physical server objects defined in the VDCF repository. Virtual objects are vServers and Guest domains. Physical objects are the bare metal servers (nodes). Control domains are counted as physical nodes.

```
% vdcfadm -c statistics

VDCF Statistics (management1 / 02.12.2025 10:07:27)
(10.0.0 / 5d64746177dc8bdd6af260e461f914f7e8700108)

VDCF Objects (required Licenses): 22

WARN: There are still 3 Virtual Objects running Solaris 10. Migration to Solaris 11 is highly
recommended.

Virtual Servers (vserver) - 3 on Solaris 10

      DEFINED      ACTIVATED      DETACHED      OTHER      TOTAL
        2           7           0           0           9

Guest Domains (gdom)

      DEFINED      ACTIVATED      DETACHED      OTHER      TOTAL
        0           5           0           0           5

Physical Nodes (node)

      sun4u      sun4v      TOTAL (Cdom)
        2         6           8 (4)
```

#### 3.3 Configuration

The active framework configuration contained in the files framework.cfg, patch.cfg and customize.cfg in the directory /opt/jomasoft/vdcf/conf and /var/opt/jomasoft/vdcf/conf may be displayed as follows:

```
% vdcfadm -c show_config

VDCF Configuration Values
Variable Value
CONFIG_DEFAULTS server.group=node;server.location=RZ
CONFIG_IPMP_ALIASES MNGT:management,PUBL:public,BACK:backup
DATASET_DEFAULT_TYPE ZPOOL
DATASET_FILESYS DISKSET:ufs ZPOOL:zfs
...
```

### 3.4 Configuration Repository

The information in the configuration repository (database) may be dumped into data files using the `dump_db` operation. These dump files are stored in `/var/opt/jomasoft/vdcf/db/dump`. The dump files may be reloaded later into the configuration repository using the `load_db` operation.

### 3.5 Logfiles

The VDCF framework writes messages into two logfiles

```
/var/opt/jomasoft/vdcf/log/audit.log  
/var/opt/jomasoft/vdcf/log/framework.log
```

The logfile content is display using the operations `show_audit` and `show_log`. To manage the logfiles the two operations `clear_audit` and `clear_log` are used. Without additional options the logfiles are cleared. Using the `archive` option you can create archive copies of your logs.

```
% vdcfadm -c clear_audit archive  
log archived to: /var/opt/jomasoft/vdcf/log/audit.log_20061110_082933  
  
% ls -l /var/opt/jomasoft/vdcf/log/audit.log*  
-rw-r--r--  1 root    root          0 Nov 10 08:29 /var/opt/jomasoft/vdcf/log/audit.log  
-rw-r--r--  1 root    root       26151 Nov 10 08:29  
/var/opt/jomasoft/vdcf/log/audit.log_20061110_082933
```

### 3.6 VDCF Client Package

The VDCF client package (JSvdcf-client) contains tools that have to be present on all nodes and vServers managed by VDCF. Without or using an old version of the client package VDCF won't operate correctly. The client package are delivered within the VDCF base package.

Use this command to show the version of the client package on all nodes:

```
% vdcfadm -c show_node all  
Node: s0002      Version: 2.3.18   Install-Date: Aug 26 2010 17:42  
Node: s0003      Version: 2.3.18   Install-Date: Aug 27 2010 09:14  
Node: s0004      Version: 2.3.18   Install-Date: Aug 26 2010 16:54  
...
```

And to upgrade the client package on a particular node:

```
% vdcfadm -c update_node node=s0003  
  
getting framework packages ... done  
found existing client package version: 2.3.18  
replacing with new client package: 3.0.0  
installing framework packages ... done  
Check /var/tmp/vdcf/install_config.log for details.
```

When upgrading the VDCF base package, normally you also have to upgrade the VDCF client package. See the **VDCF Release Notes** for more information about upgrading VDCF.

### 3.7 Pending changes

To find unfinished work, e.g. uncommitted node, vserver, dataset changes you can use operation `vdcfadm -c show_pending`

```
-bash-5.2$ vdcfadm -c show_pending
```

```
WARN: Modified vServer with uncommitted network changes
```

Name	Type	cState	rState	Node	Comment
v0116	FULL	DEFINED	-	g0091	Test Server

Type	Hostname	Interface	State
management	v0116-mngt	management0->vnet0->i40e0	DEFINED

Consult the manpage of `vdcfadm` for additional details about the available functions.

## 4 Node and disk management

### 4.1 Installable Solaris Builds

A Build is a carefully defined set of Software packages used for provisioning a node with its required software. Typically, a Build contains only what is needed by the infrastructure and tends to be as small as possible. The process of assembling a Build out of a standard OS distribution is also known as *minimization*. It allows for a lightweight, more secure and faster install of the operating environment.

Builds also form the base for a standardized environment. All installations are done using a particular Build-Version. All systems installed with the same Build are known as having exactly the same OS software level.

#### 4.1.1 Solaris 10: Builds and flash archives

On Solaris 10 nodes the build concept of VDCF uses the Solaris Flash technology for provisioning a node. A Flash archive therefore represents a specific Build. Multiple build versions are kept in different Flash archives.

##### 4.1.1.1 Flash archives

A Flash archive is created from a previously installed and running master system. After creation, the Flash archive contains all the software of the master system.

The recommended approach to create a master system is to install a system with the required software using JumpStart technology. This is controlled via the VDCF framework using the `build` command.

You may also use an existing Flash archive, this must be based on Solaris 10 Update 1 or later.

For more details on using existing Flash archives, read **chapter 4.1.1.2** which explains how to integrate your Flash archives within the VDCF framework.

##### 4.1.1.1.1 Solaris JumpStart install server

To be able to install a compute server using JumpStart technology you must create an install server. During creation of a install server, all Solaris packages are copied from the Solaris CD/DVD to your management server.

For details about the JumpStart technology read the Oracle Manuals “*Solaris 10 Installation Guide: Network-Based Installations*” and “*Solaris 10 Installation Guide: Custom JumpStart and Advanced Installations*”.

The install server is created from a DVD using the following command:

```
% cd /cdrom/cdrom0/Solaris_10/Tools
% setup_install_server <install-server-directory>
```

The target `install-server-directory` (for example `/export/install/OS/5.10s_U3`) must be empty and have sufficient free disk space.

Don't forget to share the install server directory for all:

```
% share -F nfs -o ro,anon=0 <install-server-directory>
```

#### 4.1.1.1.2 Configure the profile

In order to be able to initially install a master server Solaris JumpStart needs to be provided with certain configuration information. These required configuration parameters are recorded in a profile. Within the VDCF configuration directory `/var/opt/jomasoft/vdcf/conf` you can find a sample JumpStart profile `build.profile`. The profile defines what software to install and where. Because we are in the process of installing a master server from which we intend to capture a Build it is important that the definitions used select the required software distribution.

**NOTE:** The VDCF framework requires the following two packages `SUNWwgetr` and `SUNWwgetu` (or Cluster `SUNWCwget`). You should not remove them from the profile.

For details about the syntax of the JumpStart profile, see the Oracle Manual *“Solaris 10 Installation Guide: Custom JumpStart and Advanced Installations”*.

#### 4.1.1.1.3 JumpStart installation

You prepare the management server to install a compute node with the defined JumpStart `build.profile` using the following build command :

```
% build -c enable_install hostname=nodexxx macaddr=0:8:20:x:x:x \
netmask=255.255.255.0 architecture=sun4u install_server=<install-server-directory>
```

A JumpStart/Build installation on sparc requires booting using ARP. If you enabled the Node to use WANBOOT before, make sure you remove the network-boot-arguments

```
OK> set-default network-boot-arguments
```

To install the target compute server use the following command:

```
OK> boot net - install http://<mngt-ipaddr>/<nodexxx>.tar
```

at the nodes OBP prompt.

#### 4.1.1.1.4 Create the flash archive

After successful installation of the Build, you need to create a flash archive using the Solaris `flarcreate` command. VDCF allows you to use your own naming standard for flash archives. It is recommended that this should be tagged with the intended Build name for identification. Build names should be standardized and indicate OS-Version, Update-Level, Patch-Level and architecture.

```
% mkdir -p /var/tmp/build
% flarcreate -n 5.10S_U1_P1 -S -c /var/tmp/build/5.10S_U1_P1.flar
```

The installed software will be copied into the archive. After successful creation, copy the flash archive to the management server into a temporary directory such as `/var/tmp`.

To speed up flash creation use `-S` and `-c` to create a smaller compressed archive.

### 4.1.1.2 Creating a VDCF Build

The VDCF framework installs nodes based on Builds. A Build consists of a boot server configuration and a flash archive. You may use your own flash archive or create a flash archive using the build command (see **chapter 4.1.1.1**)

Typically, the Build configuration are stored in the directory `/export/install/flash`.

#### 4.1.1.2.1 Create a Bootserver

To be able to install a compute node, a boot server environment is required on the management server. A boot server must be created per Solaris OS release. It is supported to use a full Solaris installation image instead of a boot server environment.

Because the VDCF framework has special requirements, you have to use the `build -c add_bootserver` command to create the boot server. It is required to run this command in the global zone. The target directory specified using the `boot_server` argument may be inside a non-global zone (vServer), for example `/zones/vdcf/root/export/install/boot/5.10S_U5`.

To get a list of available boot servers use the command `build -c show_bootserver` and to remove existing boot server directories there is the command `build -c remove_bootserver`.

### WAN Boot

You may globally enable WAN Boot in the VDCF configuration by setting `FLASH_BOOT_METHOD` to "wanboot" in your `customize.cfg`. This will produce WAN Boot miniroot's and setup the web server environment.

There are two ways to install a boot server:

a) from an already installed Solaris image

```
% build -c add_bootserver install_server=<install server directory> \  
boot_server=/export/install/boot/5.10S_U1
```

b) from a CD/DVD

```
% build -c add_bootserver install_server=/cdrom/cdrom0 \  
boot_server=/export/install/boot/5.10S_U1
```

The WAN Boot technology allows to install Nodes anywhere in your network, but you must configure the default routers and your networks in the following VDCF configuration file:

```
% pwd  
/var/opt/jomasoft/vdcf/conf  
  
% cat wanboot_defaultrouter.cfg  
  
# VDCF wanboot_defaultrouter.cfg  
# =====  
# network,defaultrouter  
# 192.168.1.0,192.168.1.1  
192.168.0.0,192.168.0.2
```

#### 4.1.1.2.2 Create an installable Build

Use the build command to create a new build and to copy the new flash archive into the build environment.

```
% build -c create version=5.10S_U1_P1 \  
boot_server=/export/install/boot/5.10S_U1 \  
archive=/export/upload/5.10S_U1_P1.flar
```

If you have an installed Solaris installation image you may use this image instead of creating a separate boot server environment.

You may store your flash archives on a different server other than the management server and configure the location of the flash archive. In this case, the flash archive is not copied to the management server. The syntax of the archive argument must be compatible with the JumpStart profile syntax of the keyword `archive_location`.

```
% build -c create version=5.10s_U1_user \  
boot_server=/export/install/boot/5.10S_U1 \  
archive=nfs://<ip-addr>/export/arch/location/5.10S_U1_P1.flar  
  
% build -c create version=5.10s_U1_user \  
boot_server=/export/install/boot/5.10S_U1 \  
archive=ftp://anonymous:@<ip-addr>/arch/location/5.10S_U1_P1.flar \  
architecture=sun4u  
  
% build -c create version=5.10s_U1_user \  
boot_server=/export/install/boot/5.10S_U1 \  
archive=http://<ip-addr>/arch/location/5.10S_U1_P1.flar \  
architecture=sun4u
```

## 4.1.2 Solaris 11: Builds/Unified Archives and IPS repositories

For Solaris 11 Nodes the build concept of VDCF is based on IPS repositories and the Solaris Automated Installer. A Build represents a specific Solaris software version provided by a IPS repository and installed thru a defined AI install service.

With Solaris 11.2 and later builds can also be based on a Unified Archive. But an IPS repository must be still defined for the build. Find more details on Unified Archives in **chapter 4.1.2.3**.

Note: The VDCF command `ipsadm` is using the Solaris `pkg` command. If your `pkg publisher` list has no valid repositories configured, `pkg` command may not work correctly just as `ipsadm` which depends on it. You should always pay attention to your `pkg publisher` list. It must contain at least one accessible Solaris repository.

### 4.1.2.1 IPS repositories

Solaris 11 installations in VDCF are based on Automated Installer (AI) and local IPS repositories. Therefore VDCF offers tools (command `ipsadm`) to manage local IPS repositories.

#### 4.1.2.1.1 Create a local IPS Solaris repository

VDCF can build IPS repositories from Solaris ZIP files and configures the corresponding repository server SMF service (application/pkg/server):

```
% ipsadm -c create_repo name=prod dir=<absolute path to directory with zipped SRU files>
```

The repository target directory is configured in the VDCF variable `AI_REPO_DIR` and defaults to `"/ips/repo/"`. The http port for the repository server is configured in VDCF variable `IPS_REPO_PORT`. If this port is already used by another service the `create_repo` command increments the port number till a free port is found.

#### Alternative to integrate an existing Repository directory:

If there is already a local repository but no http repository server is configured you can use this command to configure the SMF service only:

```
% ipsadm -c config_repo name=prod dir=/export/myrepo
```

#### 4.1.2.1.2 Sync new updates into a local IPS Solaris repository

It is recommended to download periodically new Solaris repository updates from <https://pkg.oracle.com/solaris/support/> and sync it into the local repository.

You need a valid Solaris support contract to get these updates. Your support private key and certificate file must be stored on the VDCF management server. The location of these files are configured in these VDCF configuration variables: `SOL11_SUPPORT_KEY` and `SOL11_SUPPORT_CERT`.

```
% ipsadm -c update_repo name=prod [ all-versions ]
```

### New Feature in VDCF 8.3

Update your repository to the next available SRU or update the repository with a specific SRU:

```
% ipsadm -c update_repo name=prod [ next ] [ sru= ]
```

If you don't have internet access on your ips repository server, you can download SRU zip files from [support.oracle.com](https://support.oracle.com) (search for 'Solaris Support Repository Updates') and import the zip files into your repository using this command:

```
% ipsadm -c update_repo name=prod dir=<absolute path to directory with zipped SRU files>
```

#### 4.1.2.2 AI install services

AI install services are used in Solaris 11 to install nodes. For each major Solaris release and platform we need a specific install service. (Similar to the Boot-Server on Solaris 10 Jumpstart).

Normally AI requires DHCP to start the installation process. For sparc nodes VDCF has implemented a feature to use wanboot instead of DHCP. For x86 nodes DHCP configuration is always required and is generated automatically by VDCF (Only if the DHCP server is located on the VDCF management server).

##### 4.1.2.2.1 Create an AI install service

VDCF is registering the service in AI using the `installadm` command. Use this command to create an AI install service from a solaris AI iso image:

```
% ipsadm -c create_service name=sol11-11-sparc \  
isofile=/isofiles/sol11/sol-11_1-ai-sparc.iso
```

Or you may create the service directly from the AI install package (`install-image/solaris-auto-install`) using this form of the command:

```
% ipsadm -c create_service name=sol11-11-sparc \  
patchlevel=1.1 os=11 arch=sparc repository=http://your.repo.ch:8284
```

##### 4.1.2.2.2 Remove an AI install service

Use this command to remove an AI install service:

```
% ipsadm -c remove_service name=sol11-11-sparc
```

Remove of an install service is only possible if there are no depending clients (nodes) defined on it.

### 4.1.2.3 Unified Archives (UAR)

#### New Feature in VDCF 5.4

Since Solaris 11.2 you can also create archives of installed systems, similar to the well known Flash Archives of Solaris 10. This Unified Archives can be used to rapidly clone and deploy new Nodes, Gdoms or vServers. Or you can create them on a timely base for disaster recovery.

Unified Archives is highly recommended to be used for node installation, because it is much faster than an installation from an IPS repository.

To use the Unified Archives the VDCF Management Server must run on Solaris 11.2+.

#### 4.1.2.3.1 Creating an UAR

To create a build based on a UAR you have to create a master installation on a GDom or a physical machine. To reduce complexity do not add datasets or zones to this installation before you create the archive from it.

After successful installation by IPS, you need to create a Unified Archive using the Solaris `archiveadm` command. VDCF allows you to use your own naming standard for unified archives. It is recommended that this should be tagged with the intended Build name for identification. Build names should be standardized and indicate OS-Version, Update-Level, Patch-Level, architecture and software group.

You have to make sure the UAR does only include the global zone with it's rpool and no other zone or dataset information. If there is other data around, you have to exclude it accordingly on archive creation. To reduce space, we can also exclude the AI ISO image from the UAR and save around 1Gb in size. This can be done by adding `'--exclude-media'` to the `archiveadm` command.

This will create a Unified Archive from a large-server installation:

```
% archiveadm create --exclude-media /var/tmp/s11.2-sru0-s_large.uar
Initializing Unified Archive creation resources...
Unified Archive initialized: /var/tmp/s11.2-sru0-s_large.uar
Logging to: /system/volatile/archive_log.1518
Executing dataset discovery...
Dataset discovery complete
Creating install media for zone(s)...
Media creation complete
Preparing archive system image...
Beginning archive stream creation...
Archive stream creation complete
Beginning final archive assembly...
Archive creation complete
```

You can verify the Archive like this:

```
% archiveadm info -v /var/tmp/s11.2-sru0-s_large.uar
Archive Information
    Creation Time: 2014-08-11T13:54:03Z
    Source Host: g0058
    Architecture: sparc
    Operating System: Oracle Solaris 11.2 SPARC
    Recovery Archive: No
    Unique ID: b591876b-7015-c1ad-ce8a-f8431baa6c92
    Archive Version: 1.0

Deployable Systems
    'global'
    OS Version: 0.5.11
    OS Branch: 0.175.2.0.0.42.2
    Active BE: solaris
```

```
Brand: solaris
Size Needed: 2.2GB
Unique ID: b79c36ef-4a5e-477b-fa7c-ec479c849c74
Root-only: Yes
```

#### 4.1.2.4 Creating a VDCF Build

Use this command to create a new build. A build defines the Solaris version to be installed (defined by the Solaris patchlevel / SRU), the AI install service to be used and the IPS repository as source for the installation. VDCF is searching all IPS repositories defined as pkg publisher or as smf pkg server for a matching Solaris version and if found selects that repository for installation.

```
% ipsadm -c create_build name=s11.1-sru10-sparc \
  service=sol11-11-sparc patchlevel=1.10

Repo server http://localhost:8282 with patchlevel 1.10.0.5.0 (U1.SRU10) selected
Build s11.1-sru10-sparc successfully created
```

You can also define a build based on a Unified Archive. This build will still need an IPS repository and a service as well. Normally the repository is not used during installation (only if some basic packages are missing in the UAR), but it will be configured as active pkg publisher after the installation has finished. Make sure the UAR is available by a webserver. To define a UAR build simply add the archive location to the `ipsadm` command.

```
% ipsadm -c create_build name=s11.2-s-u-nomedia \
  service=s11u2 repository=http://192.168.20.76:8282/ \
  archive=http://g0076-mngt/uarch/s11.2-sru0-s_large.uar
```

A node's enabled / active build is always a combination of this build and the enabled group installation packages (i.e. large-server, small-server, auto-install, mini-server). See node **chapter 4.3.4** for more information. For Unified Archives the software group is part of the archive and cannot be changed during enable of the build.

To successfully migrate vServers from one Node to another build and group installation packages should be the same.

#### AI ISO for Kernel vServer

For proper installation of a Kernel vServer with a different Solaris Version than the underlying Node, you need to provide an AI ISO file for the used VDCF Build. The AI ISO image files can be downloaded from My Oracle Support.

The AI ISO can be specified using `ipsadm -c modify_build`

```
% ipsadm -c modify_build name=s11u4-sru15 aiiso=/iso/sol-11_4_15_5_0-ai-sparc.iso
AI ISO file '/iso/sol-11_4_15_5_0-ai-sparc.iso' added to Build 's11u4-sru15'
Build s11u4-sru15 successfully modified
```

Don't move the ISO files, because they are used at the time of the Kernel vServer Installation.

## 4.2 System Configuration

System Configuration are used to configure and customize your Nodes and Virtual Servers. The Configuration are stored in the Configuration Repository. They are applied automatically to the target systems during initial server installation or later using the server configuration execution command.

You define Base and Server Configuration before you install your systems. Base Configuration contain the configuration values. The Base Configuration are the reusable building blocks of the configuration information. Every Base Configuration has a specific type and unique name. Optionally you can also add an OS or platform attribute to separate for example Solaris 10 and 11 configurations. The connection between target systems and the Base Configuration is created by adding Server Configuration.

### 4.2.1 Base Configuration

The following Base Configuration Types are supported:

#### a) Solaris Configuration

SCSI_VHCI	To configure I/O multipathing for non-Sun symmetric storage devices. (Applies only to Nodes)
NTP	To configure the Time Service (Applies only to Nodes) Template files can be used. See <b>chapter 10.1.1.1</b> for details
DEFAULTROUTE	To configure the Defaultroute (Applies only to Nodes)
DNS	To configure the DNS
ROUTE	To configure Network Routes
SERVICES	Used to enable and disable SMF Services

#### b) System Customization

COMMAND	To execute a simple Command with arguments
FILE	To copy files to the target system
SCRIPT	To execute a script on the target system
PKG	To add packages

Use the `config` command to show, add and remove Base Configuration. When adding a Base Configuration you choose a Type and give a Name, which must be unique for the Type. For the four System Customization Types you can also add OS and/or Platform information. The configs are then only installed/executed on the OS/Platform defined, even if they are configured to a group with mixed systems inside.

Depending on the Base Configuration Type you must provide several arguments. For example for the COMMAND type you must enter the command including arguments. Use the `comment` argument to describe the purpose of a Base Configuration.

```
% config -c add type=COMMAND name=FSS command="dispadmin -d FSS" comment="enable FSS"
```

Use the manpages to learn more about the required arguments:

```
% config -H PKG
```

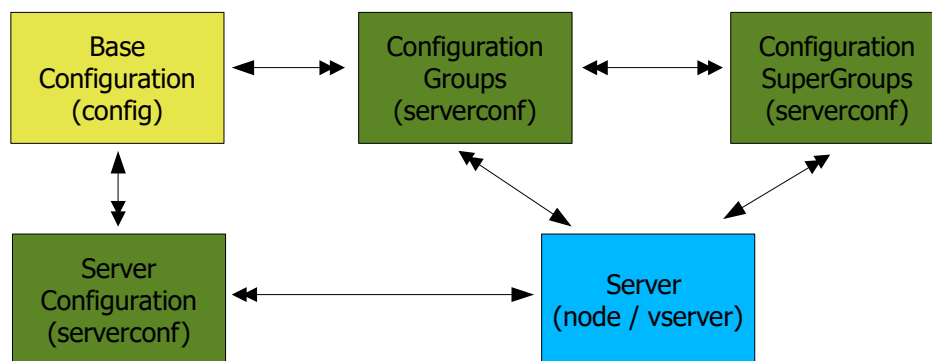
## 4.2.2 Server Configuration

A Server Configuration is used to connect a Base Configuration to a Node, vServer or a group. You manipulate the Server Configuration using the `serverconfig` command.

### Configuration Groups

Node's and vServer's are added to one or more Configuration Groups when you create them using `vserver -c create group=mygroup` or `nodecfg -c add`. Additional groups may be added using the modify operations (`vserver -c modify`, `nodecfg -c modify`). The Configuration Groups are created when you add a Server Configuration. To list the Configuration Groups with attached Server Configuration use the `serverconfig -c list` command.

It's possible to build groups of configuration groups (we call them Configuration Supergroups):



### Default Server Configuration

For base configuration items which should be applied to all servers, you should add it to the Default Server Configuration (i.e an VDCF internal configuration group). The Default Server Configuration may be overwritten or combined with Server specific Server Configuration.

Such default configurations are automatically applied to systems at installation time, but never used when you execute Server configurations to existing system (See **chapter 4.2.3** for more details).

Candidates for Default Server Configuration are the SERVICES and SCSI\_VHCI configuration. Use these commands to create the base configuration and add it to the default Server Configuration:

```
% config -c add type=SERVICES name=DEFAULT \  
  disable=telnet,sendmail,print/server,rstat comment="Default hardening for all"  
  
% serverconfig -c add type=SERVICES name=DEFAULT  
  
% config -c add type=SCSI_VHCI name=DEFAULT \  
  provider=YOURPROVIDER productid=YOURPRODID  
  
% serverconfig -c add type=SCSI_VHCI name=DEFAULT
```

## Sections

The "System Customization" Base Configuration types COMMAND, PKG, SCRIPT, FILE can be defined in section order when adding server configuration. The Server Configuration is then used in the order, where Section 1 is used first, then the Configuration of Section 2 and so on.

Sections are not supported for the "Solaris Configuration" Base configuration items (DNS, NTP, DEFAULTROUTE, SCSI\_VHCI, SERVICES, ROUTE)

The section number can be changed using the command `serverconfig -c modify`.

## Usage

The server configuration is used depending on the base configuration type

a) Usage: First Found (for configuration types DNS, NTP, DEFAULTROUTE)

Summary: server -> groups -> default

At node/vServer installation, a search is started with the required configuration type in the configuration repository. If a server configuration is found for the target server, the search stops and the corresponding configuration is used. If no server specific configuration is found, all servergroups the server belongs to, are searched for the configuration. The first found servergroup configuration is used. If the search does not succeed then the default server configuration is used.

This algorithm allows you to override default server configuration by adding server configuration to servergroups and servers.

b) Usage: Combine (for configuration of type SERVICES, COMMAND, PKG, SCRIPT, FILE, SCSI\_VHCI, ROUTE)

Summary: default -> +group -> +server

At node/vServer installation, a search is started with the required configuration type. The server configuration is "executed" in the order they are found, first the server configuration of type default, then from all servergroups and finally from server specific configuration.

For every section (1-'SRVCFG\_HIGHEST\_SECTION\_NO') this search loop is executed.

## Recommended Configuration

Refer to chapter 3.2 of the *VDCF Installation Guide* for information about required and recommended System Configuration.

### 4.2.3 Server Configuration execution

Server Configuration execution is a feature for applying existing Server Configurations to already installed systems. You specify exactly what configuration is executed on the target systems using a single configuration item or a whole configuration group. Additionally it's possible to execute an arbitrary Solaris command. Server Configuration execution is not a re-deployment tool for the originally Server Configuration used at installation time.

#### 4.2.3.1 Supported Configuration types

The following configuration types are currently supported:

- COMMAND
- SCRIPT
- FILE
- PKG
- SERVICES
- DNS

#### 4.2.3.2 Security (RBAC,vpool)

To use the Server Configuration execution you need to have the RBAC profile “vDCF serverconfig exec” assigned . You may use vPool authorizations to restrict applying configurations to selected servers (see [chapter 9](#)).

#### 4.2.3.3 Customization

The following VDCF settings can be changed to your needs (default is FALSE):

```
export CONFIG_EXEC_FILE_SAVE=TRUE
```

Used for config type=FILE. Files already existing on the target system are always overwritten. If this variable is TRUE then VDCF will create a backup copy of the existing file. The backup copy is named like this: <filename>.vdcf.YYYYMMDD:HHMMSS.

```
export CONFIG_EXEC_PKG_REPLACE=TRUE
```

Used for config type=PKG. This variable defines the behavior when replacing packages on the target system. If TRUE, the package is always replaced (even when the installed package is newer than the one to be installed). By default (FALSE) an already installed package won't be replaced.

```
export CONFIG_EXEC_COMMAND_DEFAULTUSER="nobody"
```

By default all commands are executed on the remote server by the root user. You may change this by setting another default user for command remote execution.

This default can be overwritten using the arguments `user` and `root`:

```
serverconfig -c exec command=<command> server=<server> [ user=<username> | root ]
```

#### 4.2.3.4 Command usage

The server configuration execution command has three distinct forms:

##### Execute a single command:

```
serverconfig -c exec command=<command>  
server=<comma sep list> |  
servergroup=<config group> |  
serverfile=<abs. path to file>  
servertype=all|node|vserver|gdom|cdom|solaris|linux  
[ stage=<stage> ]
```

##### Execute an existing server configuration:

```
serverconfig -c exec type=<COMMAND|SCRIPT|FILE|PKG|SERVICES|DNS> name=<config name>  
server=<comma sep list> |  
servergroup=<config group> |  
serverfile=<abs. path to file>  
servertype=all|node|vserver|gdom|cdom|solaris|linux  
[ stage=<stage> ]
```

##### Execute a server configuration group:

```
serverconfig -c exec group=<config group>  
server=<comma sep list> |  
serverfile=<abs. path to file>  
servertype=all|node|vserver|gdom|cdom|solaris|linux  
[ stage=<stage> ]
```

##### Arguments:

- server:** comma separated list of server names. e.g.: s0002,s0003,s0004
- servergroup:** selects servers belonging to the configuration group  
use `serverconfig -c list groups`, resp.  
`serverconfig -c show_members group=<config group>`  
to list configuration groups and their members
- serverfile:** absolute path to an input file containing server names on separate lines
- servertype:** selects servers based on their type
- group:** configuration group to deploy to selected targets.  
Use `serverconfig -c show group=<config group>` to list the referenced base configurations.

#### 4.2.3.5 Messages

In this example the user does not have the required permission (RBAC Profile "VDCF serverconfig exec"):

```
% serverconfig -c exec command="ps -ef | grep java" server=s0005,s0009  
ERROR: Permission denied for this operation. RBAC profile is required.
```

In this example the user has no vPool permission for node s0009:

```
% serverconfig -c exec command="ps -ef | grep java" server=s0005,s0009  
ERROR: No permission for node s0009 (Prod Bank). Check your vPools.
```

In this example the vServer v0100 is in state DETACHED. Execution is only allowed for vServers in state ACTIVATED:

```
% serverconfig -c exec command="ps -ef | grep java" server=s0005,s0009,v0100  
ERROR: Ignoring vServer v0100 (Prod Bank01) in State DETACHED
```

In this example the node s0005 is ACTIVE but currently not reachable:

```
% serverconfig -c exec command="ps -ef | grep java" server=s0005,s0009,v0101
```

```
Executing on Node s0005 (Test Bank)  
WARN: Ignoring not reachable Node s0005 (ssh connection failed).
```

→ the command will be executed on all other servers.

### 4.2.3.6 Examples

#### Execute a command on a single server:

```
% serverconfig -c exec command="uptime" server=v0100

Executing on vServer v0100 (Prod Bank01) on Node s0009
Executing command: uptime
 5:50pm up 3 day(s),  5:44,  0 users,  load average: 0.04, 0.01, 0.01
Exit-Code: 0

execution successful
```

#### Execute server configuration of type COMMAND:

```
% config -c show type=COMMAND name=swapinfo
                        Name Value
                        swapinfo /opt/MSP/serverdocu/bin/swapinfo.d

% serverconfig -c exec type=COMMAND name=swapinfo server=s0009

Executing on Node s0009 (Prod Bank)
Executing command: /opt/MSP/serverdocu/bin/swapinfo.d
RAM   _____ Total 1024 MB
RAM   _____ Unusable 22 MB
RAM   _____ Kernel 222 MB
RAM   _____ Locked 2 MB
RAM   _____ Used 76 MB
RAM   _____ Free 699 MB

Swap  _____ Total 8845 MB
Swap  _____ Resv 97 MB
Swap  _____ Avail 8747 MB
Swap  (Minfree) 125 MB

Exit-Code: 0

execution successful
```

#### Executing a server configuration of type FILE:

```
% config -c show type=FILE name=customer_inv
                        Name Value
                        customer_inv
sysdoc/customer_inv,/opt/MSP/serverdocu/data/customer_inv,root:other,0640

% serverconfig -c exec type=FILE name=customer_inv server=s0009

Executing on Node s0009 (Prod Bank)
copy file sysdoc/customer_inv to /opt/MSP/serverdocu/data/customer_inv
Existing File saved as: /opt/MSP/serverdocu/data/customer_inv.vdcf.20111205_175748
execution successful
```

### Executing a server configuration of type SCRIPT:

```
% config -c show type=SCRIPT name=exec_cmd
      Name Value
      exec_cmd exec_cmd/exec_1

% serverconfig -c exec type=SCRIPT name=exec_cmd server=s0009

Executing on Node s0009 (Prod Bank)
Executing script: exec_cmd/exec_1
exec_1
SunOS s0009 5.10 Generic_142909-17 sun4u sparc SUNW,UltraAX-i2
config SW
update root shell
Exit-Code: 0

execution successful
```

### Executing a server configuration of type PKG:

```
% config -c show type=PKG name=sysdoc

      Name OS Platform Comment Options PkgDevice Packages
      sysdoc - - -G sysdoc/MSPserdoc.pkg MSPserdoc

% serverconfig -c exec type=PKG name=sysdoc server=s0009

Executing on Node s0009 (Prod Bank)
Removing Package MSPserdoc (2.0.0 (patchdiag.xref 01.05.2011)) ...
Installing Package MSPserdoc (2.2.0 (patchdiag.xref 06.08.2011)) ...
Exit-code: 0 (Successful completion)
execution successful
```

### Executing a server configuration of type SERVICES:

```
% config -c show type=SERVICES name=DEFAULT

      Name Value
      DEFAULT enable=:disable=telnet,sendmail,sendmail-client,print/server,rstat,ftp

% serverconfig -c exec type=SERVICES name=DEFAULT server=s0009

Executing on Node s0009 (Prod Bank)
SMF service disabled: telnet
SMF service disabled: sendmail
SMF service disabled: sendmail-client
ERROR: SMF service disable failed: print/server
svcadm: Pattern 'print/server' doesn't match any instances
SMF service disabled: rstat
SMF service disabled: ftp
ERROR: service update failed
ERROR: execution failed
```

### Executing a whole server configuration group:

```
% serverconfig -c show group=MSP

section: 2 type: PKG          name: sysdoc          group: MSP
value: MSPserdoc@sysdoc/MSPserdoc.pkg

section: 3 type: FILE        name: customer_inv    group: MSP
value: sysdoc/customer_inv,/opt/MSP/serverdocu/data/customer_inv,root:root,0644

section: 3 type: COMMAND     name: check_change    group: MSP
value: /opt/MSP/serverdocu/bin/check_change.ksh pre

% serverconfig -c exec group=MSP server=s0009

Executing on Node s0009 (Prod Bank)
Removing Package MSPserdoc (2.0.0 (patchdiag.xref 01.05.2011)) ...
Installing Package MSPserdoc (2.2.0 (patchdiag.xref 06.08.2011)) ...
Exit-code: 0 (Successful completion)
copy file sysdoc/customer_inv to /opt/MSP/serverdocu/data/customer_inv
Existing File saved as: /opt/MSP/serverdocu/data/customer_inv.vdcf.20111206_092013
Executing command: /opt/MSP/serverdocu/bin/check_change.ksh pre
collecting data .... (please be patient)
Exit-Code: 0

execution successful
```

## 4.2.4 Network Route Configuration (Solaris 11)

### New Feature in VDCF 7.0

VDCF can discover, display and change network route configuration on your systems using the new command 'routecfg'. This feature is only available for Solaris 11 and require persistent routes on the systems. The goal of this feature is to track the changes of routes and in case of a re-install the previously deployed routes are re-applied to the freshly installed system.

It's important to mention that this feature is an optional replacement for the existing server configuration types ROUTE/DEFAULTROUTE. To deploy the routes based on the routecfg information instead of the ROUTE/DEFAULTROUTE config the following variable must be set in customize.cfg.

```
export INSTALL_USE_ROUTECFG="TRUE"
```

Be aware of the following restrictions:

- only persistent routes are imported/verified
- vServer must be configured with 'exclusive-ip'
- vServer must be running for verification, import and commit operations

The following operations are supported:

#### **routecfg -c import**

You can use the import function to import existing route configuration.

Be aware, an import will remove route configuration information from VDCF if they are not deployed on the node. You should always use 'routecfg -c verify' to compare the existing routes on the system with the routes stored in VDCF. And only import them if your are sure that they are correct.

#### **routecfg -c verify**

Compares the route information in the database with the effective routes discovered on the requested system.

#### **routecfg -c show**

Use the 'node', 'vserver', 'destination' and 'gateway' arguments to search routes. By using the 'node' argument with the 'full' flag also routes of vServer running on it are listed.

#### **routecfg -c add**

Adds a new persistent route configuration to the VDCF Repository. The route definition is added as DEFINED and will be configured on the system by the command 'routecfg -c commit'.

#### **routecfg -c remove**

Set the state of one or more routes to PURGING. To set all routes to PURGING you can use the 'destination=all' argument. The routes will be removed from the system during the next 'routecfg -c commit'.

#### **routecfg -c revert**

Use the revert command to change the state of PURGING routes back to ACTIVATED.

#### **routecfg -c commit**

Commits pending configuration changes. The target node will be updated to reflect changes indicated in the configuration.

All resources in state 'DEFINED' will be configured and brought to state 'ACTIVATED'. All resources in state 'PURGING' will be removed from the system. Otherwise no operation takes place.

#### **routecfg -c diff**

Use the diff command to compare two different systems.

Starting with Solaris 11.4, persistent routes have always a name. You can name them by your own with the 'routecfg -c add name=xy' or let the system set a default name (route-X).

## 4.3 Physical Server Management

### 4.3.1 Compute Pools

Every physical server belongs to one compute pool. You can create multiple compute pools to define your environment in your data center. Typically there is a compute pool for production, training, development, etc. VDCF ensures that no virtual server can migrate across the border of a compute pool.

#### 4.3.1.1 RBAC profiles

Only users with the RBAC Profile “VDCF computepool Manager” are allowed to manipulate compute pools and to assign Nodes to a different compute pool.

```
% profiles | grep computepool
VDCF computepool Manager

% cpool
USAGE: cpool [ -xhH ] -c <cmd>
...
    Operations on ComputePools

    cpool -c show          [ name=<computepool name> ]
                        [ parsable [ header ] ]
                        [ patches [ summary ] ]

    cpool -c create        name=<computepool name>
                        comment=<comment>
                        [ default ]
                        [ node=<node name>[,<node name>,... ] ]

    cpool -c set_default  name=<computepool name>

    cpool -c assign       name=<computepool name>
                        node=<node name>[,<node name>,...]

    cpool -c rename       name=<computepool name>
                        newname=<new pool name>

    cpool -c modify       name=<computepool name>
                        comment=<comment>

    cpool -c remove       name=<computepool name>
                        [ force ]

    cpool -c check        name=<computepool name> | all
```

Users with the RBAC Profile “VDCF computepool User” only are allowed to use the cpool show command:

```
% profiles | grep computepool
VDCF computepool User

% cpool
USAGE: cpool [ -xhH ] -c <cmd>
...
    Operations on ComputePools

    cpool -c show        [ name=<computepool name> ]
```

### 4.3.1.2 Default compute pool

One of the compute pools is marked as “default”. This is where new physical server configuration is usually added.

```
% cpool -c show

      Name  Default      Creation Date      Comment
  default  *           2007-08-29 07:12:39  Default ComputePool
      prod           2007-08-29 18:58:00  Production
      dev           2007-09-07 21:36:23  Development

% cpool -c show name=default

General information for ComputePool:

default (default) - Default ComputePool - creation date 2007-08-29 07:12:39

Pool Nodes

      Node  Model              Location      Comment
  computel0  SUNW,Sun-Fire-480R  RZ            Project XY
  computel1  SUNW,Sun-Fire-480R  RZ            Testing ZFS
```

### 4.3.1.3 Consistency check

With the 'check' operation you may verify if your compute pools are consistent in terms of SAN disk visibility, network (type and VLAN) and patch level:

```
% cpool -c check name=default

ERROR: ComputePool default (Default ComputePool) is not consistent:

List of missing (unregistered) disks

Disks not registered on Node <s0052>:
60060E80141AC70000011AC700000176
60060E80141AC70000011AC700000177
60060E80141AC70000011AC700000178
60060E80141AC70000011AC700000179
60060E80141AC70000011AC70000032B
60060E80141AC70000011AC70000032C
60060E80141AC70000011AC70000032D
60060E80141AC70000011AC70000032E

Disks not registered on Node <s0058>:
01000003BA1D3B0F00002A0048D9308F
60060E80141AC70000011AC700000171
60060E80141AC70000011AC700000172
60060E80141AC70000011AC700000173
60060E80141AC70000011AC700000174

Systems with Patch-Level 237E25E9292B09477816C7674D2ED276 / Kernel: 141444-09 (U8)
vServers: s0111 s0102

Systems with Patch-Level 49885E15CEA673C0882426E8D67A38 / Kernel: 141444-09 (U8)
vServers: s0245

Systems with Patch-Level 8161B1340B68ED03898F9F54EFBF0B25 / Kernel: 141444-09 (U8)
Nodes: s0052
```



#### 4.3.1.4 Patchlevel statistics for a Compute Pool

##### New Feature in VDCF 8.1

This feature shows a statistic of all OS versions and OS updates used on the systems in a compute pool.

```
% cpool -c show name=default patches

Solaris Patch Statistic 22.02.2021

-----
  ComputePool: <default>          Comment: 'Default ComputePool'
  Total Systems: 22
  =====
  Total Nodes: 14
  OS          Count
  10           5          35.7%
  11           9          64.3%   100.0%
  |--> U3      4          |----> 44.4%
  |--> U4      5          |----> 55.6%
  Total vServers: 8
  OS          Count
  10           3          37.5%
  11           5          62.5%   100.0%
  |--> U4      5          |----> 100.0%
  -----
```

Use the optional 'summary' argument to show the statistics over all compute pools.

```
% cpool -c show patches summary

Solaris Patch Statistic 22.02.2021

-----
  ComputePool: <clustergdoms>    Comment: 'Clustered GDomns'
  Total Systems: 15
  =====
  ...
  ...
  Total vServers: 3
  OS          Count
  11           3          100.0% 100.0%
  |--> U3      3          |----> 100.0%
  -----

Solaris OS-Level Statistics
Total Systems: 155
=====
  OS          Count
  10           30          19.3%
  11          125          80.6%
  |--> U1      10          |--> 6.4%
  |--> U2       4          |--> 2.5%
  |--> U3      28          |--> 18.0%
  |--> U4      83          |--> 53.5%
```

For systems installed with Solaris 11 the OS update is also listed. Compute pool based statistics are calculated on the number of Solaris 11 systems in that compute pool.

In the summary view the Solaris 11 systems percentage is based on 'Total Systems' in our example 155

##### New Feature in VDCF 8.3

With the additional flag 'sru' the statistic is broken down to the SRU (Support Repository Update) level.

## 4.3.2 Node Discover

Before a physical server can be installed and operated using VDCF, it is necessary to determine the physical devices of the server. Information about the existing hardware is discovered, e.g. CPUs, Memory, local disks, network interfaces and then loaded into the VDCF Configuration Repository.

It is supported to automatically add the Node using the discover operation with the 'add' flag using default values. See **chapter 4.3.3.1** for details.

With VDCF 5.4 or later the node discover does not discover SAN LUNs anymore by default.

### 4.3.2.1 Using root (Solaris 8/9/10 only)

The target server must already running Solaris 8 or later and `ssh` must be enabled for root. Configure "PermitRootLogin" to yes in `/etc/ssh/sshd_config`.

```
% nodecfg -c discover hostname=192.168.4.251 name=s0004
```

### 4.3.2.2 Using nonroot

There is an alternative solution, if you have to avoid ssh logins for the root user or if you are discovering a Solaris 11 node. This solution requires to install the `JSvdcf-client` package on the target Node.

Use the URL which is configured on the VDCF Management Server:

```
% vdcfadm -c show_config | grep FLASH_WEBSERVER_URL  
FLASH_WEBSERVER_URL http://192.168.0.2:80
```

On the Node:

```
# export FLASH_WEBSERVER_URL=http://192.168.0.2:80  
# wget $FLASH_WEBSERVER_URL/pkg/`uname -p`/JSvdcf-client.pkg  
# yes | pkgadd -d ./JSvdcf-client.pkg all
```

To allow ssh communication between VDCF and the Node, ssh keys must be deployed using a VDCF client tool. Execute as root on the target node:

```
# /opt/jomasoft/vdcf/client/sbin/update_key -u $FLASH_WEBSERVER_URL
```

The discover operation is then executed using the 'vdcfexec' user if the 'nonroot' flag is provided.

```
% nodecfg -c discover hostname=192.168.4.251 name=s0004 nonroot
```

### 4.3.2.3 SAN Boot

VDCF supports SAN Boot if Standard Solaris Multipathing (MPXIO) is used.

If you plan to install your Node on a SAN LUN, MPXIO must be enabled before you run the `nodecfg -c discover` command. This can be achieved by installing a build on the Node using `build -c enable_install`. If you use Non-Sun/Oracle storage you must manually add your Storage ProductID's to `scsi_vhci.conf` on your Node and in your Boot Server environment.

On Solaris 10: `/kernel/drv/scsi_vhci.conf`

On Solaris 11: `/etc/driver/drv/scsi_vhci.conf`

To enable discover of SAN LUNs the following configuration must be set in the `customize.cfg` file:  
`export CONFIG_DISCOVER_SANDISK=TRUE`

### 4.3.3 Node configuration

Before a Node can be used by VDCF, it must be configured in the VDCF Configuration Repository. A Type needs to be assigned to the network interfaces, to let VDCF know how the network interfaces are to be used (MNGT, BACK, PUBL ...). Each network type must be unique for the node.

The Configuration of a Node is always based on the information discovered by `node -c discover` (See **chapter 4.3.2**).

To add a Node there are two ways:

#### Automated node configure

The node is added only based on the information discovered. This results in a Node with some basics configured. You need to manually complete the configuration: Set network types and comment, assign cpool, etc)

#### Node configure based on a Profile

If you add multiple, standardized systems this is the recommended way, where you define a Standard Profile for each System Model. In the Profile the Network Types are assigned to the network interfaces once. The Nodes are then added based on this Profile.

#### Node Console

It is highly recommended to add a console configuration. This allows VDCF to connect to the System Controller for automated installation and for Hardware Monitoring.

### 4.3.3.1 Automated Node configure (non-interactive)

It's possible to add a node in a non-interactive way and without the usage of node profiles at all. This is useful when you like to import a node into VDCF that doesn't conform to any existing node profile. This auto-add feature can discover network link aggregations and ipmp settings. After adding the node you may change some node and network settings using `nodecfg -c modify` and `nodecfg -c modify_net`.

To add a node automatically you need to discover and then add it to the VDCF database using the `add` flag:

```
% nodecfg -c discover nonroot hostname=192.168.4.251 name=s0004 add
discovering new client : s0004
Discover Systeminfo ...
Discover Diskinfo ...
This may take some time, it depends on the number of disks
.....
Discover Rootdiskinfo ...
Discover Netinfo ...
discover successful
Node configuration successfully added.
nodecfg add successful
```

If you already have discovered the node you can use just the `add` command.

```
% nodecfg -c add name=s0004 noprofile
Node configuration successfully added.
```

After successfully adding your compute node you can display the configuration.

```
-bash-4.1$ nodecfg -c show name=s0004
```

Name	Model	HostId	Serial	cPool	DataCenter	Location	Comment
s0004	ORCL,SPARC-T4-1	85e94498	1147BDYE2A	default	ZUERICH	RZ	Discovered

Configuration Groups		Technical specification	
node		32256MB RAM	CPU: 1 64 (SPARC-T4) x 2848MHz

Disk Devices				Device Path	
Usage	Media	Name			
ROOTDISK	fibre	c0t5000CCA012AFA844d0		/scsi_vhci/disk@g5000cca012afa844	
ROOTMIRR	fibre	c0t5000CCA012B66658d0		/scsi_vhci/disk@g5000cca012b66658	

Network Interfaces						
Name	Type	Speed	Usage	MAC-Address	IP-Address	Device Path
igb0	ETH	AUTO	<b>MNGT</b>	0:21:28:e9:44:98	192.168.20.24/255.255.255.0	/pci@400/pci@..
igb1	ETH	AUTO	<b>NONE</b>	0:21:28:e9:44:99	192.168.100.24/255.255.255.0	/pci@400/pci@..
igb2	ETH	AUTO	NONE	0:21:28:e9:44:9a	-	/pci@400/pci@..
igb3	ETH	AUTO	NONE	0:21:28:e9:44:9b	-	/pci@400/pci@..

Ensure command output reflects what has been configured. Then complete the configuration, with the following additional steps:

```
% nodecfg -c modify name=s0004 comment="Oracle DB Prod"
Node s0004 / Comment updated from Discovered Node to Oracle DB Prod
node modified successfully.

% nodecfg -c modify_net name=s0004 interface=igb1 nettype=PUBL
node network configuration modified successfully.

% cpool -c assign name=prod node=s0004
assigning nodes to computepool: prod
computepool modified successfully
```

#### 4.3.3.2 Node configure based on a Profile (interactive)

For every Standard Platform (Solaris Server Model) you create a profile, where you define which disks and network interfaces are to be used. A Compute Node needs at least one root disk, but it is recommended a root mirror and a management network interface and IP address also exist. Available network types are MNGT, PUBL and BACKUP. Per type of network one IPMP Group is also configurable, by assigning type PROBE to the physical network interfaces. To define network link aggregation please assign type AGGR.

You may define your own network types by setting the variable NODE\_NET\_ALIAS.

This command goes interactive and requires manual input. Defaults are marked with '['...']' and are accepted by simply pressing return.

```
% nodecfg -c create_profile name=s0004

Modify the values from Node: s0004 to define your standard hardware profile.

The following disk devices have been detected:
c0t0d0 /pci@1f,4000/scsi@3/sd@0,0
c0t1d0 /pci@1f,4000/scsi@3/sd@1,0

Enter 'Disk Device for ROOTDISK' [Device-Path]: /pci@1f,4000/scsi@3/sd@0,0
Enter 'Disk Name for ROOTDISK' [c1t0d0]:
Enter 'Disk Device for ROOTMIRR' [Device-Path]: /pci@1f,4000/scsi@3/sd@1,0
Enter 'Disk Name for ROOTMIRR' [c1t1d0]:

Network interface definitions: Assign a network type.
Choose a network type out of MNGT PUBL BACK PROBE AGGR or NONE
MNGT: Used for the installation of the node. One device with type MNGT is mand.
PROBE: Used to select devices for IPMP groups
AGGR: Used to select devices for link aggregations
NONE: Used to skip the device

Enter 'bge0: /pci@7c0/pci@0/network@4' []: MNGT
Enter 'bge1: /pci@7c0/pci@0/network@4,1' []: NONE
Enter 'nxge0: /pci@7c0/pci@0/network@2' []: AGGR
Enter 'nxge1: /pci@7c0/pci@0/network@2,0' []: AGGR
Enter 'nxge2: /pci@7c0/pci@0/network@2,1' []: AGGR
Enter 'nxge3: /pci@7c0/pci@0/network@2,2' []: AGGR
Enter 'nxge4: /pci@7c0/pci@0/network@2,3' []: PROBE
Enter 'nxge5: /pci@7c0/pci@0/network@2,4' []: PROBE

Define link aggregations:

Enter 'Aggregation key' [1]:
Enter 'interfaces' [nxge0,nxge1,nxge2,nxge3]: nxge0,nxge1
Enter 'Network Type (PROBE PUBL BACK)' []: PROBE

Enter 'Aggregation key' [2]:
Enter 'interfaces' [nxge2,nxge3]: nxge2,nxge3
Enter 'Network Type (PROBE PUBL BACK)' []: BACK

Define IPMP Interface Groups:

Enter 'Network Type (MNGT PUBL BACK)' [PUBL]:
Enter 'IPMP Group Name' [ipmppublic]:
Enter 'interfaces' [nxge4,nxge5,aggr1]: nxge4,nxge5,aggr1

Enter 'Please enter the Name of the Profile: ' [SUNW,M5000]:

New profile created /var/opt/jomasoft/vdcf/conf/platforms/SUNW,M5000
```

A new node is added to the configuration repository based on a standard profile and the discover results. Here you manually assign the system configuration.

```
% nodecfg -c add name=s0004 profile=SUNW,Ultra-60
```

This command go interactive and allow you to fine tune the values for a particular node based on the generic profile created before.

### 4.3.3.3 Node Console

The console definition is dependent of the Server Model.

There are two configuration variables to define your defaults in customize.cfg

```
Postfix of the Console Hostname          export CONFIG_CONSOLE_POSTFIX="-sc"
User of the Console/System Controller    export CONFIG_CONSOLE_USER="admin"
```

#### SSH-Key for Console User

For the Console types ILOM, ILOMx86 and XSCF VDCF supports the authentication using a SSH-Key.

To enable this feature you must set the absolute path to the SSH private key file:

```
export CONFIG_CONSOLE_PRIVKEY="/root/.ssh/console/id_rsa"
```

```
% console -c add node=s0004
```

```
adding console for ORCL,SPARC-T4-1
```

```
Enter 'console type <ILOM>' [ILOM]:
Enter 'console hostname/IP' [s0004-sc]:
Enter 'console protocol <SSH/TELNET>' [SSH]:
Enter 'tcp/ip port' [22]:
Enter 'admin user' [admin]:
Enter 'admin password' :
Re-enter 'admin password' :
```

```
Checking Console connectivity
Firmware Version changed to 8.4.0.b for Node s0004
console added successfully
```

After updating your system controller firmware you should execute a 'console -c check' to update the console firmware and serial number information in VDCF:

```
% console -c check node=s0004
Firmware Version changed to 8.4.0.b for Node s0004
Serial Number changed from serial to 1147BDYE2A for Node s0004
Console check successful for s0004
```

Firmware version and last Update is displayed using the 'console -c show' command:

```
% console -c show node=s0004
```

```
Console attached at Node s0004 (Testserver)
```

Type	FW-Version	FW-Update	Hostname/IP	Port	Protocol	User
ILOM	8.4.0.b	2018-02-01 09:12:34	s0004-sc	22	SSH	admin

#### 4.3.3.4 ASR SNMP Integration for ILOM

VDCF can automatically configure SNMP on your ILOM or ILOMx86 console, to integrate the ILOM into your ASR (Oracle®, Auto Service Request) infrastructure.

The SNMP configuration is done when you execute `console -c add`.

To enable this feature you need to set, update the following variables:

<b>ASR_MANAGER</b>	You need to define where your ASR Manager is running Allowed values: VDCF_MNGT, VDCF_PROXY or IP-Address
<b>ASR_ILOM_VALUES</b>	Default Values: 1:162:2c:public  <code>alertrule(1):destination_port(162): snmp_version(2c):community_or_username(public)</code>
<b>ASR_ENABLE_TEST_RULE</b>	Define if a test service request should be created. Default Value: TRUE

If the ASR Manager is running on a VDCF Proxy or specific host, the VDCF Management server must be able to login to that system (JSvdcf-client package and SSH-Key deployed)

If SNMP version 3 is used, the username, password must be configured manually.

## 4.3.4 Node Install

### 4.3.4.1 Solaris 10 installation

When installing a Node with Solaris 10, the first step is to assign an existing Build to the Node.

```
% build -c show

Build Version  OS Version  Platform Arch  Method  Type  Build Name
      5.10v_z8   5.10 (U8)   sparc  sun4v  STD    zfs   5.10_u8_SPARC
      5.10u_u6_P1 5.10 (U6+)  sparc  sun4u  WAN    ufs   5.10_u6_P20060526_SPARC

% flash -c enable_install node=s0004 version=5.10u_u6_P1

Found Server: s0004 Model: 2 (2) x UltraSPARC-II 450MHz 2048MB RAM
Found network boot device on management network: qfe0, 192.168.0.4/255.255.255.0
Installation enabled for Node: s0004 Version: 5.10u_u6_P1
```

The new SPARC CPUs (SPARC S7,M7 and M8) require a minimum Solaris Patch Level, this will be checked during the enable\_install.

The required install command to issue the install is displayed using

```
% flash -c list_active

Node  Version  Install Command
s0004 5.10u_u6_P1 node -c install name=s0004
computel 5.10x_U4 <unavailable - no console>
```

For Nodes without a configured Console/System Controller in VDCF, the required OBP boot command is displayed as follows

```
% flash -c list_active node=s0004

Node  Version  Install Command
s0004 5.10u_u6_P1 node -c install name=s0004

Native OBP Install Command
boot /pci@7c0/pci@0/network@4,1 - nowin http://192.168.0.2:81/s0004.tar

Native OBP WANBOOT settings
setenv network-boot-arguments host-ip=192.168.0.4,router-
ip=192.168.0.2,subnet-mask=255.255.255.0,hostname=s0004-
vdcf,file=http://192.168.0.2:81/scripts/wanboot.cgi
```

To issue the install command, the Server must be brought down to the OK> prompt (init 0 state).

### x86 Nodes

To install a x86 node we have to use PXE boot. PXE boot requires a DHCP configuration for each node to be installed. When VDCF the configuration variable DHCP\_CONFIG\_MODE is set to EXEC, VDCF is configuring the local DHCP server and adds all required DHCP macros.

If your DHCP server is not on your VDCF management server you have to add these macros in your DHCP server manually:

- Boot server IP (BootSrvA) : <IP of VDCF management server>
- Boot file (BootFile) : <mac addr of server to be installed>

## WAN Boot

If your Build is based on a WANBoot boot server and your node have a configured system controller VDCF will configure the OBP automatically before installing the node. Otherwise you have to configure OBP manually:

```
{0} ok setenv network-boot-arguments host-ip=<mngt-ip-of-node>,  
router-ip=<your-router-ip>,subnet-mask=<your-netmask>,  
hostname=<nodename>,file=http://<mngt-webserver-ip:port>/scripts/wanboot.cgi
```

As an alternative you may use the `eeprom` command if the node is running:

```
% eeprom network-boot-arguments="host-ip=<mngt-ip-of-node>,router-ip=<your-router-ip>,\  
subnet-mask=<your-netmask>,hostname=<nodename>,\  
file=http://<mngt-webserver-ip:port>/scripts/wanboot.cgi"
```

VDCF displays the required arguments with `flash -c list_active node=<yournode>`

If you have configured a supported System Controller or Terminal server the command

```
% node -c install name=s0004
```

can be used. As an alternative you may enter the OBP command on the `OK>` prompt.

The Solaris installation takes place on the selected disks of the Node under control of ZFS or the Solaris Volume Manager (SVM) depending on your settings in the `partitioning.cfg` file. After the Solaris installation the System configuration will be applied to the node, including the Remote Execution Environment (SSH). Finally the visible disks are automatically registered for the node in the configuration repository.

#### 4.3.4.2 Solaris 11 installation

When installing a Node with Solaris 11, the first step is to assign an existing build to the Node. To display existing builds use this command:

```
% ipsadm -c show_build
```

Build	PatchLevel	Install Service	#A	#E	IPS Repository	Platform	Method
s11.1-sru19	1.19.0.6.0 (U1.SRU19)	sol11.1-18	0	1	http://localhost:8282	sparc	WAN
s11.1-sru21	1.21.0.4.1 (U1.SRU21)	sol11.1-21	2	2	http://localhost:8282	sparc	WAN
s11.2-uar	2.0.0.42.0 (U2)	s11u2	3	2	http://localhost:8282	sparc	WAN

If you have builds based on Unified Archives, the archive location will also be printed out.

Enable the node for installation with a specific build:

```
% node -c enable_install name=s0021 build=s11.2-uar
```

```
Found Server: s0021 Model: 1 (32) x UltraSPARC-T1 1200MHz 16256MB RAM
Found network boot device on management network: e1000g0, 192.168.20.21/255.255.255.0
Client 0:14:4f:9d:5b:2 added to AI servise s11u2
Install your node using 'node -c install name=s0021'
```

The `node -c enable_install` command creates all required settings in the Solaris `installadm` database and activates `wanboot` for `sparc` and `dhcp` for `x86` installations. The generated node manifest and system configuration profile files are stored as backup copy in the directory `/var/opt/jomasoft/vdcf/ai/`.

The new SPARC CPUs (SPARC S7, M7 and M8) require a minimum Solaris Patch Level, this will be checked during the `enable_install`.

The required install command to issue the install is displayed using

```
% node -c show_enabled
```

Node	Build	Group	Pkg	OS Version	Install Method	Install Command
g0051	s11.1-sru19	large-server	11	U1.SRU19	WAN	node -c install name=g0051
g0062	s11.1-sru21	large-server	11	U1.SRU21	WAN	node -c install name=g0062
s0021	s11.2-uar	large-server	11	U2	WAN	node -c install name=s0021

For Nodes without a configured Console/System Controller in VDCF, the required OBP boot command is displayed as follows

```
% node -c show_enabled node=s0021
```

Node	Build	Group	Pkg	OS Version	Install Method	Install Command
s0021	s11.2-uar	large-server	11	U2	WAN	node -c install name=s0021

```
Native OBP Install Command
boot /pci@780/pci@0/pci@1/network@0 - install

Native OBP WANBOOT settings
setenv network-boot-arguments host-ip=192.168.20.21,subnet-mask=255.255.255.0,
hostname=s0021-mngt,file=http://192.168.20.3:5555/cgi-bin/wanboot-cgi
```

#### 4.3.4.2.1 Installing sparc nodes

For sparc nodes VDCF enables always WANBoot. If your node has a configured system controller VDCF will configure the OBP automatically before installing the node. Otherwise you have to configure OBP manually:

```
{0} ok setenv network-boot-arguments host-ip=<mngt-ip-of-node>, \  
subnet-mask=<your-netmask>,hostname=<nodename>, \  
file=http://<mngt-webserver-ip:5555/cgi-bin/wanboot-cgi
```

As an alternative you may use the `eeprom` command if the node is running:

```
% eeprom network-boot-arguments="host-ip=<mngt-ip-of-node>, \  
subnet-mask=<your-netmask>,hostname=<nodename>, \  
file=http://<mngt-webserver-ip:5555/cgi-bin/wanboot-cgi"
```

VDCF displays the required arguments with `node -c show_enabled node=<yournode>`.

To issue the `install` command, the Server must be brought down to the `OK>` prompt (init 0 state). If you have configured a supported System Controller or Terminal server the command

```
% node -c install name=<yournode>
```

can be used. As an alternative you may enter the OBP command on the `OK>` prompt.

#### 4.3.4.2.2 Installing x86 nodes

To install x86 nodes we have to use PXE boot. PXE boot requires a DHCP configuration for each node to be installed. When VDCF the configuration variable `DHCP_CONFIG_MODE` is set to `EXEC`, VDCF is configuring the local DHCP server and adds all required DHCP macros.

If your DHCP server is not on your VDCF management server you have to add these macros in your DHCP server manually:

- Boot server IP (BootSrvA) : <IP of VDCF management server>
  - Boot file (BootFile) : <your install servers pxegrub file>
- e.g. `dhcptab` for a node with mac addr `01080027C1827D`:

```
root@intell11:~# dhtadm -P  
Name                               Type                               Value  
=====                               =====  
01080027C1827D                     Macro                               :BootSrvA=192.168.1.46:BootFile=sol11-ga-  
x86/boot/grub/pxegrub:  
localhost                           Macro                               :Include=Locale:Timeserv=127.0.0.1:LeaseTim=86400:LeaseNeg:DNSdmain="islikon.net":DNSserv=192  
.168.1.1:  
192.168.1.0                          Macro                               :Subnet=255.255.255.0:RDiscvyF=1:Broadcst=192.168.1.255:  
Locale                               Macro                               :UTCoffst=3600:
```

#### 4.3.4.2.3 Customization of generated AI xml files

The xml files for AI are generated by VDCF using predefined template files stored in `/opt/jomasoft/vdcf/conf/`. If required you may overwrite these template files. To do so, copy the template file to `/var/opt/jomasoft/vdcf/ai/` and change them accordingly. You may create node specific or global template files. Please do not change the placeholders (marked with `%name%`) which are used to fill the correct values.)

Type	Template files by search order	Description
Base Manifest template file used for: <code>\${node}_manifest.xml</code>	<code>/var/.../ai/\${node}_AI.templ</code>	Customer's node specific
	<code>/var/.../ai/s11_node_AI.templ</code>	Customer's global default
	<code>/opt/.../conf/s11_node_AI.templ</code>	VDCF global default
Include file for manifest logical xml tag	<code>/var/.../ai/\${node}_manifest_logical.templ</code>	Customer's node specific
	<code>/var/.../ai/gdom_manifest_logical.templ</code>	Customer's setup for GDOMs
	<code>/var/.../ai/physical_manifest_logical.templ</code>	Customer's setup for Physical Nodes
	<code>/var/.../ai/s11_manifest_logical.templ</code>	Customer's global default
	<code>/opt/.../conf/s11_manifest_logical.templ</code>	VDCF sample (not used). VDCF generates XML tag corresponding to the nodecfg.
Include file for manifest facet xml tags	<code>/var/.../ai/\${node}_manifest_facets.templ</code>	Customer's node specific
	<code>/var/.../ai/s11_manifest_facets.templ</code>	Customer's global default
	<code>/opt/.../conf/s11_manifest_facets.templ</code>	VDCF global default file (only english localized files and manpages are installed)
Include file for additional ips packages	<code>/var/.../ai/{node}/ips-pkg.list</code>	Customer's node specific package list
	<code>/var/.../ai/all-nodes/ips-pkg.list</code>	Customer's global package list
Global System Config file used for <code>base_SC.xml</code>	<code>/var/.../ai/s11_base_SC.templ</code>	Customer's default
	<code>/opt/.../conf/s11_base_SC.templ</code>	VDCF default
Node System Config file used for <code>\$node_SC.xml</code>	<code>/var/.../ai/\${node}_SC.templ</code>	Customer's node specific
	<code>/var/.../ai/s11_node_SC.templ</code>	Customer's global default
	<code>/opt/.../conf/s11_node_SC.templ</code>	VDCF global default
Include files for additional System configuration profiles	<code>/var/.../ai/{node}/*.xml</code>	Customer's node specific System Configuration Profiles
	<code>/var/.../ai/all-nodes/*.xml</code>	Customer's global System Configuration Profiles

## rpool zfs settings

The Solaris installation takes place on the selected disks of the Node under control of ZFS.

By default VDCF is using these ZFS settings for the root pool:

- rpool\_vdev redundancy: none or mirror (depending on your nodecfg root disk definition)
- zvol for swap (only if rpool has enough free space)
- zvol for dump (only if rpool has enough free space)

See generated manifest files in `/var/opt/jomasoft/vdcf/ai` for more details.

If these default settings aren't good enough you may create a xml file containing the logical tag for your node manifest. And store it in `/var/opt/jomasoft/vdcf/ai` using this name pattern:

`<node>_manifest_logical.templ`. e.g.:

```
# more /var/opt/jomasoft/vdcf/ai/g0072_manifest_logical.templ
<logical noswap="false" nodump="false">
  <zpool name="rpool" is_root="true" action="create">
    <vdev name="rpool_vdev" redundancy="none"/>
    <zvol name="swap" use="swap">
      <size val="12g"/>
    </zvol>
    <zvol name="dump" use="dump">
      <size val="6g"/>
    </zvol>
  </zpool>
</logical>
```

## Facets

The install manifest files generated by VDCF are using facet properties to select which files should be installed to the system. By default only English localized files are installed and beside manpages no other doc files are installed. To change this default you have to create a file called `s11_manifest_facets.templ` (for all installations) or `${node}_manifest_facets.templ` (for a specific node only) and store it in

`/var/opt/jomasoft/vdcf/ai/`.

The allowed content of this file are facet xml tags:

```
# more /var/opt/jomasoft/vdcf/ai/s11_manifest_facets.templ
<!-- install only english localized files -->
<facet set="false">facet.locale.*</facet>
<facet set="true">facet.locale.en</facet>
<facet set="true">facet.locale.en_US</facet>
<!-- install only manpages -->
<facet set="false">facet.doc.*</facet>
<facet set="true">facet.doc.man</facet>
```

After the Solaris installation the System configuration will be applied to the node, including the Remote Execution Environment (SSH). Finally the visible disks are automatically registered for the node in the configuration repository.

## Additional packages

You can add additional IPS packages to the manifest using a custom package list stored in this file:

`ips-pkg.list`. All packages from this list are added to the package list in the `<software_data>` tag of the node manifest file.

See above to see where you have to store this file. It depends if you want to use it for all installations or just for a specific node. The allowed content of `ips-pkg.list` is a list of pkg fmri:

```
# cat /var/opt/jomasoft/vdcf/ai/all-nodes/ips-pkg.list
pkg://solaris/web/curl
pkg://solaris/system/management/puppet
```

## Additional System Configuration Profiles

It's possible to add additional System Configuration Profiles to enable additional smf services at installation time. These profiles are written to the node system configuration into the `$node_SC.xml` file.

Add these profile definitions as xml files into the related AI configuration directories.  
See above to view the exact directory for these files.

The file names must end with a `*.xml` suffix and the content should be xml code representing your service definitions. i.e.:

```
# cat /var/opt/jomasoft/vdcf/ai/node99/puppet.xml

<service name='application/puppet' version='1' type='service'>
  <instance name='master' enabled='true'>
    <property_group name='start' type='application'>
      <propval name='timeout_seconds' type='count' value='10' />
    </property_group>
  </instance>
</service>
```

### 4.3.5 Solaris Node Import

#### New Feature in VDCF 5.7

VDCF supports to integrate existing Nodes, if they are running Solaris 10 or 11. Use this import operation when the Node wasn't installed by VDCF but you like to integrate it into VDCF. You must be aware, that this import does no configuration on the Node itself. It just adds the system information of that Node in the database. The import operation does discover information for the Node, Control Domain, Guest Domain, vServer (Zones), disks and ZPOOL datasets.

Use the 'nodeonly' flag if virtual objects should not be imported with the initial import of the node.

To successfully import a Node the following task must be done before:

Install VDCF Client Pkg and add ssh Key on the node (→ **chapter 4.3.2.2**) this includes:

- JSvdcf-client package installation
- deploy ssh key (/opt/jomasoft/vdcf/client/sbin/update\_key)
- If importing a Guest Domain, the Control Domain has to be imported first.

Then you can import the Node:

```
$ node -c import name=s0023

-bash-4.1$ node -c import name=s0023
Importing new Node s0023 ...
Warning: Permanently added 's0023,192.168.100.23' (RSA) to the list of known hosts.
Discover Systeminfo ...
Discover Diskinfo ...
This may take some time, it depends on the number of disks

Discover Rootdiskinfo ...
Discover Netinfo ...
Node configuration successfully added.
System registration done for s0023.
node with all vservers being checked: s0023
check on node s0023 successful
patch deployment updated from node s0023
registering disks from node s0023
New visible Lun 6001438012599B9B0000A000002F0000 Size: 10240 MB
New visible Lun 6001438012599B9B0001100001AC0000 Size: 5120 MB
Registered new Lun: 6001438012599B9B0001100001B00000 Size: 5120 MB
Registered new Lun: 6001438012599B6200011000118D0000 Size: 10240 MB
No node datasets found on Node s0023
No vServer found on Node s0023.
WARN: Add console configuration manually using: console -c add node=s0023
Node s0023 import finished
```

If your Node uses multiple subnets you need to modify the network settings and specify additional networks using `nodecfg -c modify_net` (**Chapter 4.3.3.1** contains sample output).

If the first import ignored some network interfaces of imported vServers you need to re-run the node import to update the vServer network definition.

Now the system is known to VDCF and can be managed as a normal VDCF Node:

The Node can be monitored, vServers can be installed and you can roll out patches.

If your Node is configured as a Control Domain new GDoms can be deployed and managed.

You can automatically set an imported Guest Domain to the READONLY State after import. To enable this feature set `GDOM_IMPORT_SET_READONLY` to TRUE. See **chapter 6.5.1** for more information about the READONLY State for Guest Domains.

### 4.3.6 Linux Node Import

Existing Redhat and Oracle Linux server can be imported as well. Consult the VDCF Linux Guide for details.

### 4.3.7 Node Operation

Once the Node has been installed with a particular build, it registers its presence within the framework.

Installed Nodes are managed by the `node` command. Remember that the Node – the physical server – only acts as a carrier for the vServer. It manages the environment needed by vServers. However, because the node performs critical operations on behalf of the management server, it should normally not be required to log into the physical server for its operation. All day to day operational tasks on the nodes should be performed using the `node` command.

See the `node` command and `manpage` for more details about possible operations.

### 4.3.8 Node visible VLAN Verification

#### New Feature in VDCF 7.1

VDCF supports to verify the visible VLANs on network interfaces, when you add additional network configurations or migrate a virtual object (vserver or gdom) to another system.

Update VLAN configuration:

```
$ node -c update vlan name=s0024

updating node: s0024 - vlan update takes several minutes ...
VLAN change detected on Node s0024 for interface igb1. Old VLANs: 20 New VLANs: 20,150 / Change
to 20,150
node updated successfully
```

To clear existing VLANs on the node interfaces, use the additional flag 'clear':

```
$ node -c update vlan clear name=s0024

updating node: s0024 - vlan update takes several minutes ...
Existing VLANs cleared for Node s0024
VLAN change detected on Node s0024 for interface igb1. Old VLANs: New VLANs: 20,150 / Change to
20,150
node updated successfully
```

Be aware, as soon as a node has VLANs stored on its interfaces, VDCF commands will check against these settings.

e.g. : `vserver/gdom -c addnet` with `vlan` options, `vserver/gdom` candidates and `cpool -c check`

### 4.3.9 Node Configuration Updates

You may need to update your node configuration, e.g. different network type for a node interface, IPMP changes, Default Scheduler. With the 'nodecfg' command you are able to modify the configuration of a node. Be aware, such a modification does not change the configuration on the node itself. It does update the VDCF database and would apply these settings if the system would be re-installed.

#### Scheduler Class for Node

##### New Feature in VDCF 7.1

You may need a specific scheduler running on a node.  
Update the actual scheduler for an existing node.

```
$ node -c update name=s0024
updating node: s0024 - this may take a moment ...
registering disks from node s0024
Node 's0024' --> Scheduler class has changed! - Current value: 'FSS', old value: ''
node updated successfully
```

Change the scheduler class for a node:

```
$ nodecfg -c modify name=s0024 scheduler=TS
Node s0024 / Scheduler attribute updated to TS
node modified successfully.
```

To set a default scheduler for nodes, configure the VDCF variable:  
NODE\_SCHEDULER\_DEFAULT: allowed values are: FSS, TS

#### IPMP Standby Configuration for Nodes

To update an IPMP Configuration from active-active to active-standby or vice-versa

```
$ nodecfg -c modify_net name=s0024 interface=management standby
IPMP Usage set to 'standby' for IPMP Group management on Node s0024
node network configuration modified successfully.
```

```
$ nodecfg -c modify_net name=s0024 interface=management clear_standby
IPMP Usage set to 'active' for IPMP Group management on Node s0024
node network configuration modified successfully.
```

#### Update Network Type (Usage)

To change a network usage use the nodecfg -c modify\_net command.  
For example, the IPMP Group Usage and ip-address need to change for a node:

Old Settings:

Network Interfaces						
Name	Type	Speed	Usage	MAC-Address	IP-Address	Device Path
public100	IPMP	-	<b>NONE</b>	-	-	qfe1,qfe3

```
$ nodecfg -c modify_net name=s0013 interface=public100 nettype=PUBL ipaddr=10.1.0.223
netmask=255.255.255.0
Net-Type changed from NONE to PUBL for Interface public100 on Node s0013
node network configuration modified successfully.
```

Updated Settings:

Name	Type	Speed	Usage	MAC-Address	IP-Address	Device Path
public100	IPMP	-	<b>PUBL</b>	-	<b>10.1.0.223/255.255.255.0</b>	qfe1,qfe3

### 4.3.10 Node Kernel Settings

The node show kernel operation provides an overview of a few typically used settings.  
The values are updated by node -c verify

```
-bash-5.2$ node -c show kernel
```

Name	Type	Model	Comment	Total RAM/GB	Free RAM/GB	ZFSArc Max/GB	ZFSvDev MaxPend	Max OpenFiles
g0042	GDOM	GDom on s0003	OpsCenter	16.0	1	2.8	10	4095
g0069	GDOM	GDom on s0003	VDCF test1	18.0	4	4.0	20	4095
g0071	GDOM	GDom on s0003	CBE42 postgres	16.0	7	2.0	20	4095
g0078	GDOM	GDom on s0003	CBE81 postgres	10.0	4	1.0	20	4095
s0003	CDOM	ORCL,SPARC-S7-2	S7-2 Server	24.0	6	7.0	10	4095
s0028	CDOM	ORCL,SPARC-T8-1	T8-1 512GB	16.0	3	4.0	10	4095

### 4.3.11 Node Evacuation

#### 4.3.11.1 Requirements

Because this evacuation feature is based on resource usage information, it is only supported, if the VDCF Monitoring and HA features are installed and the resource monitoring (rcmon) enabled on each participating Node.

The VDCF HA feature is available to VDCF Enterprise customers.

#### 4.3.11.2 Overview

The evacuation feature distributes the vServers from one Node to the other compatible Nodes which have enough resources (CPU and RAM). This feature may be used for planned maintenance and if a node fails. This evacuation feature is used by the VDCF High Availability Monitoring (hamon). Consult the VDCF HA Guide for more information about hamon.

Because of the potential limitation of resources on the Nodes, not all vServer may be evacuated. Therefore it is highly recommended to define categories and priorities to your vServer, to make sure the production vServer are evacuated first. See the **chapter 5.2.1** for more information about categories and priorities.

It is highly recommended to regularly check your compute pools using `cpool -c check` to avoid incompatibilities.

#### 4.3.11.3 vServer shutdown on target Nodes

##### New Feature in VDCF 6.0

Your target Nodes may not have enough free resources for the evacuated vServers. In such environments you can define the Categories for less important vServers, which VDCF can shutdown to free resources. The vServers are shutdown only when required and ordered by the vServer Priority.

Define the Categories in `VIRTUAL_EVACUATION_SHUTDOWN_CATEGORIES` and use the 'shutdown' flag of `node -c evacuate` if you want to shutdown the less important vServers.

#### 4.3.11.4 Evacuation

The evacuation operation offers three optional flags: upgrade, force and shutdown

'upgrade' needs to be used, if you would like to migrate the vServers to Nodes, which are on a higher Patch Level or are of a different architecture (sun4u and sun4v).

Only if your Node failed and is not reachable you need to use the 'force' flag. Wrong usage for 'force' may damage your data!

Here a sample of a successful evacuation.

```
-bash-5.2$ node -c evacuate name=g0094  
evacuating node g0094 - this may take a moment ...
```

Name	T	cState	rState	Node	CDom	cPool	OS	Patch-Level	Comment
v0142	K	ACTIVATED	RUNNING	g0094	s0003	hamon-demo	11	4.15.0.1.5.0 (U4.SRU15)	kzone
v0173	F	ACTIVATED	RUNNING	g0094	s0003	hamon-demo	11	4.71.0.1.170.2 (U4.SRU71)	full zone

This operation may shutdown and detach the vServers listed above

```
Are you sure (yes/no) ? [no]: yes  
Starting evacuation of Node g0094.  
Trying to evacuate vServers: v0173 v0142  
Now we do an iteration of vServer distribution from Node g0094 ...  
Target node for vServer v0173 selected: g0048  
Target node for vServer v0142 selected: g0048  
Doing normal detach of vServer v0173 ...  
Doing normal detach of vServer v0142 ...  
Attaching vServers v0173 v0142 to Node g0048 ...  
Doing attach of vServer v0173 to Node g0048 ...  
vServer v0173 successfully attached. Now booting ...  
Doing attach of vServer v0142 to Node g0048 ...  
vServer v0142 successfully attached. Now booting ...  
There are no more vServers left on Node g0094. Finished  
All vServers successfully evacuated  
Evacuation of node g0094 finished.  
node successfully evacuated
```

### 4.3.12 SPARC Node Firmware Upgrade

The firmware of current Oracle SPARC servers (T7,S7,T8) can be upgraded online.

Oracle delivers ILOM firmware in two different formats. One option is to download an IPS package from the Oracle Support repository and the second option is to download a zip file from MyOracle Support.

VDCF supports both firmware formats.

```
node -c upgrade_fw name=<node name>
                fw_version=<firmware version (ips)> |
                patch=<patch_no (file)>
                [ trial-run | verbose ]
```

Downloaded firmware zip files must be placed into `/var/opt/jomasoft/vdcf/firmware`

```
-bash-5.2$ node -c upgrade_fw name=s0003 fw_version=9.10.8.a

Checking the available firmware packages ...
Installing firmware/system/S7-2/sysfw9-10@9.10.8.1 ... DONE
Executing firmware update (takes around 30 minutes)
/usr/sbin/fwupdate update all -T lanplus -q -x /var/firmware/system/S7-
2/sysfw9-10/p36549129_9108a/Firmware/Sun_System_Firmware/metadata.xml -o
/var/tmp/vdcf/fw_update.9050.log

Details from logfile /var/tmp/vdcf/fw_update.9050.log

-----
Monday, November 18, 2024 at 4:35:18 PM CET
/usr/sbin/fwupdate update all -T lanplus -q -x /var/firmware/system/S7-
2/sysfw9-10/p36549129_9108a/Firmware/Sun_System_Firmware/metadata.xml

Firmware upgrade to version 9.10.8.a was successful on Node s0003
Checking console ...
Firmware Version changed from 9.10.7 to 9.10.8.a for Node s0003
```

### 4.3.13 Node Remove

If you plan to take a Node completely out of service, remove all objects on the Node first (vServer, Guest Domains and node datasets). Then after the final shutdown (node -c shutdown) you remove all the Node definitions from the VDCF repository using

```
$ node -c remove name=s0004
removing node: s0004
node removed successfully

$ nodecfg -c remove name=s0004
Node configuration removed successfully

$ console -c remove name=s0004
removing console for node <s0004>
console removed successfully
```

## 4.4 Patch Management (Solaris 10)

### 4.4.1 Introduction

VDCF Patch Management is used to install and compare Patch-Levels of Nodes, GDoms and vServers running on Solaris 10. Solaris 11 introduces a new concept using Package updates instead for installing patches.

#### Patch Check Advanced (PCA)

Patch Check Advanced (<http://www.par.univie.ac.at/solaris/pca/>) written by Martin Paul is used internally to download patches from Oracle. Dependency checking capabilities of PCA are used based on the xref file provided by Oracle Support.

#### Solaris 10 Patch Policy

Oracle introduced a new patch policy for access to Solaris 10 patches. Now that Solaris 10 is freely available, support services require the purchase of an Oracle Service Plan. Under this new policy, access to patches, or what are now called Software Updates, is restricted. Access to the patches requires an Oracle Service Plan and an Oracle Support User.

### 4.4.2 VDCF Patch Architecture

#### 4.4.2.1 Patch Spooling

The Management Server is used to download the patches from the Oracle Support portal. The downloaded patches are stored in the default spool directory (`/var/sadm/spool`). This spool directory is defined as `PATCH_SPOOL` in the framework configuration (`conf/customize.cfg`).

#### 4.4.2.2 Patch Analyzing

The collection of patches to be downloaded is determined by analyzing each ACTIVE node at a regular interval. The resulting collection of patches will be downloaded by using the 'pca' utility. Downloaded patches are spooled and then imported into the VDCF Configuration Repository.

This procedure should be triggered regularly using the recommended crontab configuration from `/opt/jomasoft/vdcf/conf/sysconf/vdcf_patch_crontab`. It ensures that patches are accumulated in all possible revisions available.

Use the following interface to manage patch spooling:

<code>patchadm -c analyze</code>	used to analyze nodes
<code>patchadm -c download</code>	used to download missing patches identified through analyze or only one specific patch.
<code>patchadm -c import</code>	used to import new downloaded patches from spool into DB
<code>patchadm -c show</code>	shows current spool patches

The nightly job `patchadm_nightly` executes the analyze, download and import operations.

#### 4.4.2.3 Patch download

##### Patch Check Advanced (PCA)

PCA is used to download patches to the VDCF patch spool directory (as defined by `PATCH_SPOOL`). The following VDCF variables can be set and affect the command `'patchadm -c download'`:

`HTTP_PROXY`.....A HTTP proxy may optionally be used to connect to the Oracle patch repository.

The credentials required to access the Oracle patch repository and optional credentials to get access to a HTTP proxy can be managed with the help of `'patchadm -c credentials'`.

<code>patchadm -c credentials set=oracle proxy</code>	interactive definition of credentials
<code>patchadm -c credentials remove=oracle proxy</code>	removal of existing credentials
<code>patchadm -c credentials show</code>	displays users of existing credentials

##### Example for an Oracle User definition

```
$ patchadm -c credentials set=oracle  
  
Enter 'Oracle Support user' []: name@mycompany.ch  
Enter 'name@mycompany.ch password' :  
Re-enter 'name@mycompany.ch password' :
```

## 4.4.3 VDCF Patch Configuration

### 4.4.3.1 Patch Sets

A Patch-Set is a defined collection of patches. Patch-Sets are applied to nodes (targets). A Patch-Set is created through the 'patchadm -c create\_set' command. This command creates a collection of patches based on a start- and an end-date. The initial set can be customized by adding or deleting individual patches. The following command creates a set that contains all patches released since December 1. until January 31. All other patches in the spool (which is all patches released by Oracle) will not be part of that set.

```
% patchadm -c create_set name=myset platform=sparc from=2005-12-01 to=2006-01-31
```

If the same patch with different revisions are available, the newest will be selected.

Use the following interface to manage Patch-Sets:

patchadm -c create_set	used to create a patch set
patchadm -c delete_set	used to delete a patch set
patchadm -c modify_set	used to add or delete individual patches from an existing patch set
patchadm -c show_set	used to show detail about an existing patch set

### 4.4.3.2 Patch Targets

A Patch-Target combines one or more Patch-Sets with their install targets. A Patch-Target is created using the 'patchadm -c create\_target' command. This command populates a Patch-Target with a number of Nodes based on the search criteria given to it. The following command creates a Patch-Target that contains all Nodes with a particular Build installed and connects it to the previously created Patch-Set 'myset':

```
% patchadm -c create_target name=mytarget \  
filter=build:5.10_U1 patchset=myset desc="my first"
```

This Target can now be modified to contain additional Patch-Sets. The following command attaches the Target 'mytarget' to a Patch-Set called 'otherset':

```
% patchadm -c modify_target name=mytarget add patchset=otherset
```

Patch-Targets are the unit of installation. Patching is started by specifying a particular Patch-Target. The Target will build the list of patches being installed based on the attached Patch-Sets. The patches will then be applied to all systems registered within the Target.

Use the following interface to manage Patch-Targets:

patchadm -c create_target	creates a Patch-Target and assigns Nodes and Patch-Sets
patchadm -c delete_target	deletes an existing Patch-Target
patchadm -c modify_target	modifies Patch-Sets and Nodes of a Patch-Target
patchadm -c show_target	shows Patch-Target information

#### 4.4.4 VDCF Patch Installation

How the Patches are installed depends on the Patch Types. There are 3 types STANDARD, NON\_STANDARD and SAFE\_MODE.

STANDARD	Patch does not require Single-User Mode and Reboot
NON_STANDARD	Patch requires installation in Single-User Mode and Reboot
SAFE_MODE	Kernel Patch requires installation in Single-User Mode and Reboot (Deferred Activation Patch)

If all the Patches of a Patch Target are of type STANDARD, the Patch Target is installed while the Node and vServers are running. NON\_STANDARD and SAFE\_MODE Patches require installation in Single-User Mode followed by a Reboot. While such NON\_STANDARD Patch Targets are installed, the vServers are automatically rebooted into Single-User Mode by VDCF. You must use the `reboot` Option of the `install` operation for such Patch Targets.

##### 4.4.4.1 Patch Prepare

It is not required but recommended to prepare a Patch Target as the first step. Preparing means all required patches are downloaded to the target nodes. If all patches are already installed the downloaded patches are deleted on the nodes and the user is informed.

```
% patchadm -c prepare target=development
preparing patch target: development
WARN: nothing to patch on node s0005 - all patches already installed
WARN: nothing to patch on node s0006 - all patches already installed
prepare successful for target: development
```

##### 4.4.4.2 Patch Install

The Patches are installed on all Nodes defined in the Patch Target in parallel. Installation of a Patch Target may require multiple reboots, which is automatically executed by VDCF.

```
% patchadm -c install target=development reboot
installing patch target: development
installation started for target: development
```

##### 4.4.4.3 Zones Parallel Patching

If VDCF detects the Solaris feature “Zones Parallel Patching”, which is available in the patch utilities patch 119254-66 (sparc) and 119255-66 (x86) or later, the feature is activated automatically. This is leading to significant performance gains in patching operations.

If required this feature may be disabled by setting the following value in the `customize.cfg`

```
export PATCH_NUM_PROC=1
```





#### 4.4.4.4 Checking Patch Status

Once the installation of a Patch-Target completes, which involves the application of a number of patches to a number of nodes and vServers, an automatic check operation will be issued. The check operation reflects the state of the installed Patch-Sets on either a particular node or on all nodes managed by the VDCF Management Server. The check operation might also be issued manually by using the `patchadm -c check` command. The `patchadm -c show_node` command can be used to verify the actual state of all installed Patch-Sets for a specified node.

```
% patchadm -c show_node node=s0004
```

Server	Patch-Set	State	Date
s0004	otherset	OK	2006-07-15_18:50:52
s0004	20060526	OK	2006-05-26_12:17:36

This command lists the installed Patch-Set name, its state (OK, INSTALLING or FAILED) and install timestamp. For a more detailed report about a particular Patch-Set supply the `name` parameter to the above command.

```
% patchadm -c show_node node=s0004 name=20060526
```

Server	Patch-Set	State	Date
s0004	20060526	OK	2006-05-26_12:17:36

Patches successfully installed:

118367-03	118375-07	118557-03	118676-02	118708-12	118712-09
118733-03	118777-04	118812-03	118815-02	118822-30	118833-03
118981-03	119254-19	119280-04	119315-05	119374-13	119470-07
119557-09	119582-04	119596-03	119681-07	119685-05	119689-07
119712-04	119764-03	119810-02	119828-05	119850-14	119955-03
119974-03	119981-09	119982-04	119984-03	119985-02	119992-02
120036-03	120048-03	120050-02	120056-02	120094-05	120182-02
120195-02	120199-04	120201-02	120253-01	120254-02	120256-01
120258-02	120272-02	120329-02	120467-04	120469-04	120560-02
120706-02	120780-02	120830-04	120849-04	120887-05	120928-07
120986-03	120990-02	120996-02	121268-01	121286-02	121288-01
121296-01	121298-01	121302-01	121308-03	121430-04	121474-01
121487-01	121556-01	121557-01	121558-01	121559-01	121561-03
121563-02	121580-01	121620-02	121693-02	121694-01	121721-01
121734-04	121802-01	121894-01	121901-01	121905-01	121921-02
121944-01	121946-01	121975-01	121977-01	122029-01	122031-01
122032-02	122034-01	122064-01	122079-01	122081-01	122083-01
122085-01	122087-01	122119-01	122176-01	122183-01	122195-01
122231-01	122235-01	122237-01	122239-01	122241-01	122242-01
122243-01	122251-01	122253-01	122255-01	122261-01	122735-01
122750-01	122856-01				

Patches failed to install:

The output includes two relevant sections. The first section lists the patches successfully installed by this Patch-Set. The second section contains patches that failed to install. This allows for a detailed verification of all Patch-Sets installed on each node.

For more details about the patch installation, consult the Logfile `/etc/vdcfbuild/patches/smpatch_<date>` on the Target Node.

A `'patchadm_check'` utility is available that be cron initiated to perform a patch check at regular intervals.



## 4.4.5 VDCF Patch-Level

### 4.4.5.1 Display installed Patches

VDCF allows you to use your own Patch Management Tool to apply patches. In this case VDCF will display the Patch-Level of Nodes and vServer independent of the Patch Management Tool used.

The Patch-Level is updated in the VDCF configuration repository when executing the `patchadm_check` utility/cronjob or using the `patchadm -c check` command.

Patch-Level details are displayed using the `patchadm -c show_node patchlevel` command. If the ID of one or more servers is identical the same relevant Patches are installed a migration of the vServer should be possible.

```
% patchadm -c show_node patchlevel all
```

Server	Patch-Level	Type	Date	ID
compute12	127112-07	CORE	2008-01-28_19:43:09	E1E04C732B56CF55B8A07AB17A46DF4
compute20	127128-06	CORE	2008-01-22_10:51:45	8424182A5060B1060AB78A4E3BAC40
compute2	118855-19	CORE	2008-01-15_00:11:42	E0382DFABD5F5E0C5A84F5634A1339C6
server3	118855-19	CORE	2008-01-15_00:11:42	E0382DFABD5F5E0C5A84F5634A1339C6
compute4	120012-14	CORE	2007-08-16_11:55:11	0E07275D34D855EF0C09A640C6554B

```
% patchadm -c show_node patchlevel node=compute2
```

Server	Patch-Level	Type	Date	ID
compute2	118855-19	CORE	2008-01-15_00:11:42	E0382DFABD5F5E0C5A84F5634A1339C6

Patches successfully installed:

113000-07	117181-01	117435-02	117448-01	117464-01	118344-11
118347-04	118368-03	118372-07	118374-01	118561-01	118567-01
118732-01	118734-03	118736-01	118778-04	118816-03	118825-01
118844-30	118855-19	118871-01	118873-02	118919-16	118960-02
118997-08	119043-09	119074-03	119078-08	119080-12	119089-06
119091-19	119093-07	119131-20	119144-02	119471-05	119575-02
119594-01	119649-01	119686-05	119713-04	119765-03	119825-01
119827-01	119853-03	119975-02	119987-03	119989-01	120010-01
120033-02	120037-03	120045-01	120047-01	120051-02	120053-01
120063-01	120069-01	120086-01	120102-01	120111-02	120129-02
120183-02	120223-09	120225-02	120312-01	120313-01	120314-01
120330-02	120345-01	120347-03	120468-04	120474-01	120536-11
120630-02	120808-01	120810-01	120817-01	120831-04	120846-01
120890-01	120901-03	120933-01	120935-01	120985-01	120987-04
120989-01	120991-02	121003-02	121005-01	121007-01	121009-01
121011-01	121013-01	121062-01	121082-05	121119-06	121127-02
121131-01	121134-01	121216-01	121230-01	121234-01	121240-01
121264-01	121287-01	121289-02	121297-01	121300-01	121334-04
121407-01	121454-02	121475-01	121562-03	121604-01	121696-02
121779-10	121781-10	121805-02	121902-01	121922-02	121948-01
122030-01	122033-02	122035-03	122082-01	122084-01	122086-01
122173-04	122175-03	122184-01	122196-03	122216-01	122226-01
122232-01	122240-01	122248-01	122252-01	122264-01	122522-01
122528-01	122530-01	122532-01	122534-01	122536-01	122638-01
122641-06	122647-03	122653-02	122655-04	122659-03	122661-01
122663-04	122665-04	122746-01	122748-01	122753-03	122755-01
122829-02	122857-02	123016-01	123018-01	123067-01	123124-02

#### 4.4.5.2 Patch differences

With the command `patchadm -c diff` you may compare the installed patches of 2 nodes or between a Node and a vServer.

```
% patchadm -c diff server=s0003,s0055
```

```
Compare PatchLevel of Node s0003 Test System 3 Build: 5.10sv_u7w_req  
and Node s0055 Prod System 55 Build: 5.10sv_u7_req
```

```
PatchLevel Summary for s0003  
Patches installed: 245  
Patches only installed on s0003: 2
```

Patch-ID	Type	Issue_Date	Retrival_Date	Description
139983-04	NON_STANDARD	2009-06-15	2009-07-01 11:56:13	ds patch
Revision on s0055 : 139983-03				
141778-01	STANDARD	2009-06-25	2009-07-01 11:48:45	vntsd patch

```
PatchLevel Summary for s0055  
Patches installed: 244  
Patches only installed on s0055: 1
```

Patch-ID	Type	Issue_Date	Retrival_Date	Description
139983-03	Patch is not registered in VDCF Repository			
Revision on s0003 : 139983-04				

#### Explanation:

The system s0055 is missing 2 patches which are installed on s0003:

- Patch 139983-04 (on s0055 we have an older one: 139983-03)
- Patch 141778-01

Using the optional `verbose` argument you may also display all architecture dependent patches.

## 4.5 Upgrading Solaris 11

### New Feature in VDCF 5.7

The 'node -c upgrade' command can be used to upgrade one or multiple Solaris 11 Nodes/Guests to a newer Solaris version. Choose a build with the desired entire version. The repository of that build is used as source for the pkg update on the system.

With Solaris 11 an Upgrade does not need much downtime, since the new package versions will be upgraded on a new boot environment, which is created by a snapshot on the ZFS filesystem. The current running installation will not be harmed like this. All we need to is a reboot to make run from the upgraded boot environment.

### Requirements

To make sure the upgrade works, VDCF includes a free space check, which will request at least `NODE_UPGRADE_FREE_GB` (defaults to 10GB) of free space in the root zpool of the global zone as well as for all rpools of the vServers running on this node. You can find the variable in the `customize.cfg` file.

#### 4.5.1 Upgrade Trial-Run

It is possible make a dry-run test of the upgrade process to be sure it will work from a Solaris point of view. This will do a pre-flight run of the upgrade procedure with the pkg command.

Note: You can also use a list of servers to run the test on in parallel. The amount of runs started in groups can be defined by the variable `NODE_UPGRADE_VSERVER_PARALLEL`.

```
% node -c upgrade name=g0104 build=s11u3-srul trial-run
Node Upgrade Trial-Run started for Node g0104 ...
doing a 'pkg update -n -C 5 --accept --be-name s11.3.1.0.5.0 entire@0.5.11,5.11-
0.175.3.1.0.5.0' now ...
  Startup: Refreshing catalog 'jomasoft' ... Done
  Startup: Refreshing catalog 'solaris' ... Done
  Planning: Solver setup ... Done
  Planning: Running solver ... Done
  Planning: Finding local manifests ... Done
  Planning: Fetching manifests: 0/573 0% complete
  ...
  Planning: Fetching manifests: 573/573 100% complete
  Planning: Package planning ... Done
  Planning: Merging actions ... Done
  Planning: Checking for conflicting actions ... Done
  Planning: Consolidating action changes ... Done
  Planning: Evaluating mediators ... Done
  Planning: Planning completed in 99.60 seconds
-----

      Packages to remove: 47
      Packages to install: 82
      Packages to update: 492
      Packages to change: 1
      Mediators to change: 7
      Create boot environment: Yes
      Create backup boot environment: No
      Trial-Run of node upgrade (to 3.1.0.5.0) finished for node g0104
```



## 4.5.2 Upgrade node

The beginning of the output of the real upgrade does look the same as the one from the trial-run. You will see the download and installation of the packages in addition.

With the option 'reboot' the system is rebooted after a successful pkg update. If there are running vServers on the node you have to allow the reboot with the 'force' flag.

Note: You can also use a list of servers to run the upgrade on in parallel. The amount of runs started in groups can be defined by the variable `NODE_UPGRADE_VSERVER_PARALLEL`.

```
% node -c upgrade name=g0104 build=s11u3-srul
Node Upgrade started for Node g0104 ...
doing a 'pkg update -C 5 --accept --be-name s11.3.1.0.5.0 entire@0.5.11,5.11-
0.175.3.1.0.5.0' now ...
Startup: Refreshing catalog 'jomasoft' ... Done
Startup: Refreshing catalog 'solaris' ... Done
Planning: Solver setup ... Done
Planning: Running solver ... Done
Planning: Finding local manifests ... Done
Planning: Package planning ... Done
Planning: Merging actions ... Done
Planning: Checking for conflicting actions ... Done
Planning: Consolidating action changes ... Done
Planning: Evaluating mediators ... Done
Planning: Planning completed in 66.99 seconds
```

```
-----
Packages to remove: 47
Packages to install: 82
Packages to update: 492
Packages to change: 1
Mediators to change: 7
Create boot environment: Yes
Create backup boot environment: No
Download: 0/22899 items 0.0/546.8MB 0% complete
Download: 1851/22899 items 76.7/546.8MB 14% complete (15.3M/s)
...
Download: 22882/22899 items 545.8/546.8MB 99% complete (7.0M/s)
Download: Completed 546.76 MB in 55.35 seconds (9.8M/s)
Actions: 1/38757 actions (Removing old actions)
Actions: 6676/38757 actions (Installing new actions)
Actions: 20749/38757 actions (Installing new actions)
Actions: 21430/38757 actions (Updating modified actions)
...
Actions: 38643/38757 actions (Updating modified actions)
Actions: Completed 38757 actions in 98.31 seconds.
Done
```

A clone of s11.2.10.0.5.0 exists and has been updated and activated.  
 On the next boot the Boot Environment s11.3.1.0.5.0 will be  
 mounted on '/'. Reboot when ready to switch to this updated BE.

```
Done
BE          Flags Mountpoint Space  Policy Created
--          -
s11.2.10.0.5.0 N      /          187.0K static 2016-01-08 11:00
s11.3.1.0.5.0 R      -          7.10G static 2016-01-08 17:12
GDom g0104 updated to entire@0.5.11,5.11-0.175.3.1.0.5.0.
```

### 4.5.3 Upgrade failback

When the Upgrade was not successful, you can failback to the previous BootEnvironment.

If VDCF is able to connect to the Node, it is using beadm to activate the previous BootEnvironment.

If the Node is down on the OBP OK prompt, a boot command is executed to load the previous BootEnvironment. In both cases the Target BootEnvironment will not be deleted. You can check the BootEnvironment and delete it manually using 'beadm destroy badBE' after analysis.

Because the upgrade\_failback will always boot or reboot the node, the 'reboot' flag is required.

```
% node -c upgrade_failback name=g0221 reboot
BootEnvironment 's11.2.2.0.8.0' activated on Node g0221
BE Flags Mountpoint Space Policy Created
-- -----
s11.2.2.0.8.0 R - 5.12G static 2016-02-14 14:30
s11.2.5.0.5.0 N / 1.15G static 2016-02-14 14:43
Rebooting Node g0221 ...
updating /platform/sun4v/boot_archive
Node g0221 failback successfully issued to BootEnvironment s11.2.2.0.8.0 ...
```

#### 4.5.4 Additional: Node upgrade check

Additionally you can enforce an upgrade check before doing a node upgrade. To enable this feature add the following line to `customize.cfg`:

```
export NODE_UPGRADE_CHECK_REQUIRED="TRUE"
```

The feature allow to check your nodes before you upgrade. Upgrade multiple nodes in parallel and easy failback if the upgraded environment doesn't work as expected.

The upgrade check includes the following tests:

- is there enough free disk space on the ZFS root pools (default: 10GB free)
- not allowed to upgrade if an IDR package is installed on the node
- is upgrade path allowed? (only some source/target SRUs are allowed)
- is server uptime not to long
- a pkg trial update is executed

Only if all these tests are successful a node is ready for upgrading.

##### 4.5.4.1 Configure upgrade paths

To keep control over Solaris versions, which are installed in your environment, you can optionally configure from which Solaris version (Build) you can upgrade to another one. Just add builds, which you really want to distribute in your environment. You can set the allowed versions in the following configuration file.

```
# cat /var/opt/jomasoft/vdcf/conf/allowed_sru_upgrades.cfg
#TargetSRU:SourceSRU,SRU,...
11.2.5:11.2.2,11.2.3
11.2.13:11.2.8
11.3.2:11.2.8,11.2.13
```

In the first column you have the target version to which it is allowed to upgrade to. The version numbers are the concatenation of the OS Version and the Patch-Level number, which is shown in the build info from the `ispadm` command. You do not have to specify all five digits from the Patch-Level, since a new SRU is defined by the first two numbers in most cases.

In the second column you have all the versions which are allowed to be upgraded to the version in the first column. If you try to update to something which is not listed here the upgrade command will fail with an error.

With the sample above, it is allowed to upgrade:

- from 11.2.2 or 11.2.3 to 11.2.5 "Update 2 SRU2 or SRU3 to Update 2 SRU5"
- from 11.2.8 to 11.2.13 "Update 2 SRU8 to Update 2 SRU13"
- from 11.2.8 or 11.2.13 to 11.3.2 "Update 2 SRU8 or SRU13 to Update 3 SRU2"

#### 4.5.4.2 Optional settings

These are the default settings which may be overwritten, by adding other values in customize.cfg

Number of Nodes to Update in parallel:  
`export NODE_UPGRADE_BATCH=10`

Number of vServer to Update in parallel on each Node:  
`export NODE_UPGRADE_VSERVER_PARALLEL=5`

Required free disk space on Global and vServer RootPool in GB:  
`export NODE_UPGRADE_FREE_GB=10`

Number of Days allowed Uptime:  
`export NODE_UPGRADE_UPTIME_DAYS=2`

ZPOOL upgrade behavior during 'node -c upgrade\_finish'  
TRUE will update all imported ZPOOLS to the latest version  
`export NODE_UPGRADE_FINISH_ZPOOL=FALSE|TRUE`

#### 4.5.4.3 Usage Step 1: Upgrade Check

A few days before upgrading the target nodes need to be checked. Multiple Nodes can be checked in parallel:

```
% node -c upgrade_check name=g0104,g0221 build=s11.2-sru5-s
```

##### New Feature in VDCF 7.2

After a successful check, the Node upgrade state is set to 'checked' (to avoid changes like vServer Migration or similar on the system). The upgrade state is stored in the VDCF Repository.

The state of the upgrade check is shown in the `node -c show` output:

Next Build	Check Status	Check-Date	Check-Time
s11.2-sru5-s	OK	2016-02-14	14:13:00

#### 4.5.4.4 Usage Step 2: Node upgrade

Only successfully checked systems can be upgraded when the check feature is enabled!

Start upgrade by executing the `node -c upgrade` command (see [chapter 4.5.2](#)).

After a reboot the node is running on the new Solaris version. The active and the last used BootEnvironments are displayed in the `node -c show` output:

Active BootEnv	Previous BootEnv	Upgrade State
s11.2.5.0.5.0	s11.2.2.0.8.0	upgraded

##### New Feature in VDCF 7.2

The upgrade state is stored in VDCF Repository. You can use the following command to list nodes in state 'checked' or 'upgraded': `node -c show [ upgraded | checked ]`

A node in state 'upgraded' does not allow vServer Migration or similar on the system to ensure an upgrade fallback is possible.

#### 4.5.4.5 Usage Step 3: finish or failback

If all applications are running fine, the upgrade can be finished. The `node -c upgrade_finish` command removes the previous BootEnvironment and clears the upgrade check information of the nodes. The upgrade finish can be executed for multiple nodes in parallel.

New Feature in VDCF 7.0

By setting the VDCF variable "NODE\_UPGRADE\_FINISH\_ZPOOL" to 'TRUE' the 'upgrade\_finish' process will upgrade all attached ZPOOL's to the latest ZPOOL version.

New Feature in VDCF 7.1

Add the 'keep' flag if you want to keep the previous BootEnvironment.

```
% node -c upgrade_finish name=g0104,g0123
```

When the upgrade was not successful you can failback to the previous BootEnvironment. See **chapter 4.5.3** for details.

After a successful failback the `upgrade_check` information is removed. To re-run the upgrade you have to start using the `upgrade_check` again.

## 4.6 Package Management

### 4.6.1 Package information data

Package information for packages installed on Nodes and vServers are recorded in the VDCF database:

- **Name:** The unique package instance name
- **Publisher:** The publisher of the package
- **Version:** The version string of the package. For IPS packages the version value consists of the component version and the branch version (component-branch)
- **Summary:** A short description of the package
- **Zones:** Whether the package has to be installed in all zones (A or -)

With the exception of 'Zones' these terms are based upon the ones shown by the subcommand 'info' of the IPS retrieval client pkg(1). For System V Packages they correspond to the following package variables:

- Name → PKGINST
- Publisher → VENDOR
- Version → VERSION
- Summary → NAME
- Zones → SUNW\_PKG\_ALLZONES

Depending on the package system the VDCF package ID is composed of:

- IPS → FMRI without pkg://, Build Release and timestamp
- System V → PKGINST@VERSION@PSTAMP

### 4.6.2 Analyzing and importing packages

Packages of nodes (and containing vServers) are automatically analyzed at server installation time and can also be manually imported with the help of vpkgadm -c analyze:

```
vpkgadm -c analyze node=<node list> | all
```

Two mutually exclusive modes are provided:

- **Selected nodes:** The argument 'node' defines a comma-separated list of nodes to be analyzed.
- **All active nodes:** With the argument 'all' all active nodes are analyzed.

It is recommended to activate the “vpkgadm\_nightly” cronjob, to analyze the packages of all nodes once a day. A sample is provided at `/opt/jomasoft/vdcf/conf/sysconf/vdcf_base_crontab`



### 4.6.3 Compare package levels

The command `vpkgadm -c diff` compares package levels of servers (nodes and vServers) of the same operating system version:

```
vpkgadm -c diff      server=<server1,server2> [full]
```

Lists the packages which are not installed on both servers (nodes and vServers).

Without the optional 'full' argument, only packages which have to be kept in sync between the global and the non-global zones are taken into account.

Example:

```
# vpkgadm -c diff server=node1,vserver1

Compare package level of Node      node1      VDCF Node 1      Build: 5.10s_U10
                        and vServer vserver1    VDCF vServer 1    Build: 5.10s_U10

Package summary for node1 with Patch-Level: 147440-04 (U10+)
Packages installed: 204
Packages only installed on node1: 2

      Name  Publisher                Version                Zones  Summary
-----  -
SUNWsndmr Sun Microsystems, Inc.  11.10.0,REV=2005.01.21.15.53  A      Sendmail (root)
SUNWsndmu Oracle Corporation      11.10.0,REV=2005.01.21.15.53  A      Sendmail (/usr)

Package summary for vserver1 with Patch-Level: 147440-04 (U10+)
Packages installed: 202
Packages only installed on vserver1: 0

Packages installed in different versions: 1
      Name  Publisher                Version@node1
-----  -
Version@vserver1                Zones  Summary

      SUNWcakr Oracle Corporation  11.10.0,REV=2005.08.25.02.12@on10-patch20111101055928
11.10.0,REV=2005.08.25.02.12@on10ptchfeat20110620074824  A      Core Solaris Kernel Architecture
(Root)
```

#### 4.6.4 Querying packages

The package data recorded in the VDCF database may be queried with the following commands:

- `vpkgadm_search(1M)` complex searches for packages
- `vpkgadm_show(1M)` shows common package information
- `vpkgadm_show_server(1M)` shows package deployment information

##### 4.6.4.1 Search for packages

The search command supports the following options:

```
vpkgadm -c search [ name=<name> ]  
                  [ version=<version> ]  
                  [ publisher=<publisher> ]  
                  [ summary=<summary> ]  
                  [ equal ]
```

Per default substring queries are performed and all package information attributes have to be found (conjunction). If 'equal' is specified, a query for equality is performed for all defined arguments. All recorded packages are shown if no search arguments are defined.

Solaris 11 Information:

Solaris System V packages are not imported on Solaris 11 systems, because these Solaris 11 packages (IPS based) are also listed with the `pkginfo` command and duplicate entries for one package would be the result.

## Examples:

### Search for SSH server packages (substring match)

```
$ vpkgadm -c search name=network/ssh summary=Server
          PKG-ID  Version
solaris/service/network/ssh@0.5.11-0.175.0.0.0.2.1 0.5.11-0.175.0.0.0.2.1 SSH Server
solaris/service/network/ssh@0.5.11-0.175.0.4.0.2.1 0.5.11-0.175.0.4.0.2.1 SSH Server
```

### Search for all SSH packages of a certain version (substring match)

```
$ vpkgadm -c search name=network/ssh version=0.4.0.2.1
          PKG-ID  Version
solaris/network/ssh/ssh-key@0.5.11-0.175.0.4.0.2.1 0.5.11-0.175.0.4.0.2.1 SSH Common
solaris/network/ssh@0.5.11-0.175.0.4.0.2.1 0.5.11-0.175.0.4.0.2.1 SSH Client
solaris/service/network/ssh@0.5.11-0.175.0.4.0.2.1 0.5.11-0.175.0.4.0.2.1 SSH Server
```

#### 4.6.4.2 Common package information

The show command supports the following options:

```
vpkgadm -c show      server=<server name>
```

```
vpkgadm -c show      name=<name> [ version=<version> ] [ equal ] |  
                    id=<pkg-id>
```

Three mutually exclusive modes are provided:

- **All installed packages on a server:** The argument 'server' defines a Node or a vServer
- **Detailed information about a package:** The argument 'name' defines a package name substring and the optional argument 'version' defines a package version substring. If 'equal' is specified, a query for equality is performed for all defined arguments
- **Detailed information about a package instance:** The argument 'id' defines a VDCF package ID. The ID can be learned with `vpkgadm_show_server(1M)` or with the help of the second mode of this command

Example:

Show common package information of all SSH packages of a certain version:

```
$ vpkgadm -c show name=network/ssh version=0.4.0.2.1  
      Name  Publisher  Version  Zones  Summary  
      PKG-ID  
network/ssh/ssh-key  solaris    0.5.11-0.175.0.4.0.2.1  -      SSH Common  
solaris/network/ssh/ssh-key@0.5.11-0.175.0.4.0.2.1  
      network/ssh  solaris    0.5.11-0.175.0.4.0.2.1  -      SSH Client and utilities  
solaris/network/ssh@0.5.11-0.175.0.4.0.2.1  
service/network/ssh  solaris    0.5.11-0.175.0.4.0.2.1  -      SSH Server  
solaris/service/network/ssh@0.5.11-0.175.0.4.0.2.1
```

#### 4.6.4.3 Package deployment information

The `show_server` command supports the following options:

```
vpkgadm -c show_server name=<name> [ version=<version> ] [ equal ] |  
id=<pkg-id>
```

Shows package deployment information per package instance. Two mutually exclusive modes are provided:

- **Detailed information about a package:** The argument 'name' defines a package name substring and the optional argument 'version' defines a package version substring. If 'equal' is specified, a query for equality is performed for all defined arguments.
- **Detailed information about a package instance:** The argument 'id' defines a VDCF package ID. The ID can be learned with `vpkgadm_show(1M)` or with the help of the first mode of this command.

Example:

Show package deployment information of all SSH packages of a certain version:

```
$ vpkgadm -c show_server name=network/ssh version=0.4.0.2.1  
  
Package: network/ssh/ssh-key - SSH Common  
PKG-ID : solaris/network/ssh/ssh-key@0.5.11-0.175.0.4.0.2.1  
Version: 0.5.11-0.175.0.4.0.2.1 is installed on:  
  Name  Type      Comment  
  node1 Node      S11 Node  
  vserver1 vServer  S11 vServer  
  
Package: network/ssh - SSH Client and utilities  
PKG-ID : solaris/network/ssh@0.5.11-0.175.0.4.0.2.1  
Version: 0.5.11-0.175.0.4.0.2.1 is installed on:  
  Name  Type      Comment  
  node1 Node      S11 Node  
  vserver1 vServer  S11 vServer  
  
Package: service/network/ssh - SSH Server  
PKG-ID : solaris/service/network/ssh@0.5.11-0.175.0.4.0.2.1  
Version: 0.5.11-0.175.0.4.0.2.1 is installed on:  
  Name  Type      Comment  
  node1 Node      S11 Node  
  vserver1 vServer  S11 vServer
```

## 4.7 Disk Management

### 4.7.1 Overview

vServers and guest domains (Virtual Objects) are stored on SAN LUN's or disks. Keeping a Virtual Objects data on SAN storage allows for migration of the vObject from one Node to another Node. Because this is an essential feature while maintaining a Server Pool, placement of vObject data on external SAN (or iSCSI) storage is highly recommended with VDCF.

The LUN's used to place a vObject's data should be visible to more than one Node. For optimal flexibility, all LUN's should be visible to all Nodes within a particular Server Pool. Access to a set of LUN's belonging to a specific vObject is controlled by its target Node and managed through VDCF. VDCF knows which LUN is available and at which Node. In larger Server Pools it might be useful to group LUN access so that only a subset of Nodes are able to access a particular set of LUN's.

For Physical Nodes without access to a central storage ZFS Volumes (ZVOL) on local Disks are supported. Of course the Virtual Objects (vServers and GDomS) using such ZFS Volumes cannot be migrated to another Physical Nodes. Failover and Disaster Recovery must be implemented at the Application Level.

### 4.7.2 ZFS Volumes (ZVOL)

#### New Feature in VDCF 5.5

##### 4.7.2.1 Preparation

The System Administrator can prepare the required ZFS Volumes on the Physical Node. VDCF only uses ZFS Volumes which contain **"/vdcf\_zvol/** to avoid conflicts with other ZFS Volumes. Such Volumes are for VDCF use only. Substructures are supported. ZFS Volumes may be placed on different zpools, but the Volume Name must be unique on the Physical Node.

Sample to create VDCF compatible ZFS Volumes:

```
# zfs create rpool/vdcf_zvol
# zfs create rpool/vdcf_zvol/prod
# zfs create rpool/vdcf_zvol/test
# zfs create -V 10g rpool/vdcf_zvol/prod/ldom1
# zfs create -V 10g rpool/vdcf_zvol/prod/ldom2
```

#### New Feature in VDCF 6.0:

For datasets the ZFS Volume can be created dynamically using the `newzvol` flag.

```
-bash-4.1$ dataset -c create name=mydataset node=s0024 size=1g newzvol
Creating Node dataset <mydataset>
new zvol 's0024.datasets/1_mydataset' created on Node s0024
Dataset mydataset (ZPOOL) created successfully
```

##### 4.7.2.2 ZVOL Registration and Usage

VDCF compatible ZFS Volumes are registered as DeviceType ZVOL in the VDCF disk database.

```
% diskadm -c register node=compute1 methods=ZVOL
registering disks from node compute1
Registered new Lun: compute1.prod/ldom1 Size: 10240 MB
register successful
```

To keep the Disk GUID short, it does not contain the zpool name and `vdcf_zvol`. VDCF ZVOL Disks can be used as GDom Disks, ZPOOL and RAW datasets.

### 4.7.3 Disk Registration

The visible Disks (LUN's) on a Node are registered with VDCF when a Node is installed or registered. After adding additional disks to the Node you must register the new Disks with VDCF using the `diskadm -c register` command. If the Node doesn't automatically detect the newly added disks, use the `'scan'` option. This will find and configure new disks on the Node.

```
% diskadm -c register node=compute2 scan new
registering new disks from node compute2
New visible Lun 6001438012599B9B0000A000002F0000 Size: 10240 MB
Registered new Lun: 6001438012599B9B0000A000003F0000 Size: 512000 MB
register successful
```

SANBoot disks of Nodes are registered as type `BOOTDISK` in VDCF. Of course, you can't use such Disks for vServer or GDomS.

To list the known disks the `diskadm -c show` offers various options. To display the unused/free LUN's which may be used to create datasets use the `'free'` option. To filter the visible disks of a node use the `'node'` argument.

If you disconnect a disk from a node, you should also deregister the disk from the VDCF configuration repository using the `diskadm -c deregister` command. The `diskadm -c register` command reports invisible disks if the `'new'` option is omitted.

```
% diskadm -c register node=compute2
registering disks from node compute2
WARN: Not visible anymore: 600015D00002EE0...040EB.
register successful

% diskadm -c deregister node=compute2 name=600015D00002EE0...040EB
deregistering disks from node compute2
deregister successful
```

To remove all unused disks from a node use the `deregister` with the `'all'` option.

Some disks may be visible to your nodes that you do not wish VDCF to control. Mark such disks as foreign using the `'mark'` command. Such foreign disks cannot be used in VDCF anymore and are only listed from the `diskadm -c show` command if the `'all'` option is given.

Additionally you may set a comment on a disk. This way you can clarify the usage of that foreign disks.

```
% diskadm -c modify name=600015D00002EE0...040EB comment="used for nfs project 1"
```

#### 4.7.4 Physical disk location

VDCF can display the location of your LUNs. This feature is implemented using a config file (disklocation.cfg). In this file mappings between data center locations and search patterns for LUNs have to be defined. VDCF is updating the location attribute while registering new LUNs.

Rename the sample file `/var/opt/jomasoft/vdcf/conf/disklocation.cfg_template` to `disklocation.cfg` and add your mappings. Each line in the config file defines a location. The following config convention is used:

```
LOCATION.<locationname1>=<LUN-search-pattern>[:<LUN-search-pattern>: ..]  
LOCATION.<locationname2>=<LUN-search-pattern>[:<LUN-search-pattern>: ..]
```

...

Run this command once to update the location of all disks:

```
diskadm -c update
```

#### New Feature in VDCF 8.3:

New variable “DISKS\_UPDATE\_LOCATION” to specify if 'diskadm -c update' command should update all disk locations or only disks with an empty location.

```
export DISKS_UPDATE_LOCATION="ALL | EMPTY"
```

#### 4.7.5 Show / modify disk location / storage box

#### New Feature in VDCF 8.3:

You can limit the list of disk / lun for 'diskadm -c show' with the optional arguments 'location' or 'box'.

Example limit the output to location 3PAR1:

```
diskadm -c show location=3PAR1
```

To modify the location for one or multiple disks specify the new location or storage box name:

```
diskadm -c modify name=GUID1,GUID2 location=3PAR1 box=BOX999
```

To remove an existing location or box definition use “-” as value:

```
diskadm -c modify location="-" name=GUID1,GUID2
```

To list only disks that have no location or box definition use “EMPTY” as argument value:

```
diskadm -c show location=EMPTY
```

#### 4.7.6 Verify disks (assigned to nodes in more than one compute pool)

#### New Feature in VDCF 8.3:

You can verify if the disks in VDCF are assigned to more than one compute pool.

This should not be the case. The 'diskadm -c verify' command will list the affected disks.

```
$ diskadm -c verify  
Found LUNs that are visible to multiple ComputePools:
```

GUID	Count	cPool Names
60002AC00000000000000002F60001507B	8	default firewall proxy sol10_guest sol11 test vdcf
60002AC00000000000000002F70001507B	8	default firewall proxy sol10_guest sol11 test vdcf
600140569416732DBB80D4DFCD9DBFD1	6	default firewall hamon-demo ist-stand prodrepo sol11
60014058CE18C9FD80A3D4BC1D805CD9	6	default firewall hamon-demo ist-stand prodrepo sol11
600140593127DD0D3E20D4BBFDB50CDE	6	default firewall hamon-demo ist-stand prodrepo sol11
...		

#### 4.7.7 Dataset

A dataset is a collection of SVM meta-devices, a ZFS pool or just a group of raw devices. Normally datasets are used to build vServers on top of it (see **chapter 5.2.2**). But it's also possible to define datasets directly on a node or to give a vServer access to the raw devices to use them for example with Oracle ASM or other tools using raw devices.

The LUN must be visible to the 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 for a node you can choose a name, type and your preferred layout. Using the layout construct you can build different types of datasets:

##### a) Single Lun

```
% dataset -c create name=softwarelib node=s0004 layout=600015D00002EE0...040EA
```

##### b) Mirror

```
% dataset -c create name=softwarelib node=s0004 \  
layout="mirror 600015D00002EE0...040EA 00015D00002EE0...040F0"
```

##### c) Stripe

```
% dataset -c create name=softwarelib node=s0004 \  
layout="600015D00002EE0...040EA 00015D00002EE0...040F0"
```

##### d) Concatenation

```
% dataset -c create name=softwarelib node=s0004 \  
layout=600015D00002EE0...040EA  
% dataset -c commit name=softwarelib  
% dataset -c add name=softwarelib \  
layout=00015D00002EE0...040F0  
% dataset -c commit name=softwarelib
```

### 4.7.8 Dataset fast provisioning

If you like to create a single-LUN dataset you may use the dataset create command with the argument size. VDCF will automatically choose a LUN that is appropriate for your needs.

```
% dataset -c create name=smalllds node=s0024 size=2g
Creating Node dataset <smalllds>
Disk 6001438012599B62000110000A580000 (MPXIO) with Size 2048 MB selected.
Dataset smalllds (ZPOOL) created successfully
```

### 4.7.9 Dataset verify

#### New Feature in VDCF 5.7

Compares the layout information in the database with the effective values discovered on the system. Use the 'update' argument to fix the information stored in the database.

Currently this command supports ZPOOL and VXVM datasets

```
% dataset -c verify node=s0024
Real layout and DB layout of dataset 'v0177_root' are equal.
Real layout and DB layout of dataset 'v0177_data' are equal.
WARN: Log layout of dataset 'v0177_data2' has changed from '600144F04D260060000003BA2D33B400
600144F04D260062000003BA2D33B400' to 'mirror 600144F04D260062000003BA2D33B400
600144F04D260060000003BA2D33B400'
WARN: Dataset verify partially failed on Node S0013.

$ dataset -c verify node=s0024 update
Real layout and DB layout of dataset 'v0177_root' are equal.
Real layout and DB layout of dataset 'v0177_data' are equal.
Updating ZPOOL Log Layout from <600144F04D260060000003BA2D33B400
600144F04D260062000003BA2D33B400> to <mirror 600144F04D260062000003BA2D33B400
600144F04D260060000003BA2D33B400> for dataset v0177_data
Dataset successfully updated
```

#### 4.7.9.1 Dataset verify update\_size

#### New Feature in VDCF 8.3

Add the optional flag 'update\_size' to detect and update dataset size changes in VDCF.

```
% dataset -c verify all update_size
Verifying Datasets on Node node01 ...
Verifying Datasets on Node node02 ...
Verifying Datasets on Node node03 ...
Real layout and size are equal to the values stored in the DB for dataset 'data1'.
Real layout and size are equal to the values stored in the DB for dataset 'data2'.
Real layout and size are equal to the values stored in the DB for dataset 'data3'.
Updating size of dataset 'topf' from 4.00 GB to 9.97 GB
Real layout and size are equal to the values stored in the DB for dataset 'topf'.
Updating size of dataset 'topf1' from 4.00 GB to 9.97 GB
Real layout and size are equal to the values stored in the DB for dataset 'topf1'.
```

A detected dataset layout change has still to be fixed by executing dataset verify with arguments dataset or node.

#### 4.7.10 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. (See **chapter 4.7.4**) 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)
```

## 4.7.11 Node SWAP dataset

### New Feature in VDCF 5.4

For easy SWAP management for a Node the dataset command supports the “swap” flag. This swap flag is supported for dataset type ZPOOL. When committing a new SWAP dataset the existing SWAP is automatically extended.

```
-bash-4.1$ dataset -c create name=g0074_swap node=g0074 size=2g swap

Creating Node SWAP dataset <g0074_swap>
Disk 6001438012599B62000110000A500000 (MPXIO) with Size 2048 MB selected.
Dataset g0074_swap (ZPOOL) created successfully

-bash-4.1$ dataset -c commit name=g0074_swap

committing dataset changes: g0074_swap
dataset changes committed successfully

-bash-4.1$ dataset -c show name=g0074_swap verbose

      Name      State      Size/MB   Avail/MB   Type      Owner      Node
g0074_swap  ACTIVATED  2048      2048      ZPOOL/s   g0074      g0074

      Layout: 6001438012599B62000110000A500000
      Location check: compliant

      Dataset Use-Type Dev-Type State      GUID                               Serial  Size/MB
g0074_swap ZPOOL      MPXIO    ACTIVATED 6001438012599B62000110000A500000 x12376 2048

Volume Manager details from node g0074

      pool: g0074_swap
      state: ONLINE
      scan: none requested
      config:

      NAME      STATE      READ WRITE CKSUM
      g0074_swap ONLINE      0     0     0
      c0d3      ONLINE      0     0     0

errors: No known data errors

swapfile      dev  swaplo blocks  free
/dev/md/dsk/d10      85,10      16 1048560 1048560
/dev/zvol/dsk/g0074_swap/swap1 256,1      16 3889136 3889136
```

Because you can't remove Disks from a ZPOOL, to reduce SWAP you need to remove the current dataset, after creating an additional smaller SWAP dataset. Removing SWAP is dependent if the SWAP is in use.

Starting with Solaris 11.4 it is supported to remove Disk from ZPOOLS.

#### 4.7.12 Node DUMP dataset

##### New Feature in VDCF 8.0

For easy DUMP management for a Node the dataset command supports the “dump” flag. This dump flag is supported for dataset type ZPOOL. When committing a new DUMP dataset the existing DUMP device on the root pool is automatically deleted.

```
$ dataset -c create name=g0074_dump node=g0074 size=10g dump
$ dataset -c commit name=g0074_dump
```

Dump datasets can't be modified. To use other LUNs for dump the existing dump dataset must be removed and recreated.

#### 4.7.13 Dataset ZPOOL Log Disk

##### New Feature in VDCF 7.0

For better write performance of a ZPOOL additional log disks/devices (e.g. SSD flash) can be added. The dataset must already exist to add a log device.

If a mirror layout is specified a maximum of 2 GUIDs are allowed. The location of the GUIDs are checked.

```
% dataset -c addlog name=oradb layout="mirror 600015D0000...040EA 00015D0000...040F0"
% dataset -c remlog name=oradbtest guides="00015D0000...040F0"
```

This command only manipulates configuration information. Use the 'commit' operation to set up physical incarnation of changed configurations. Use the 'revert' operation to remove uncommitted changes.

#### 4.7.14 Encrypted ZPOOL dataset

##### New Feature in VDCF 8.4

ZPOOLS can be encrypted. The encryption key must be manually prepared on the target server. The default location is globally defined as

```
DATASET_ZPOOL_KEYSTORE passphrase,file:///root/.zpool.key
```

The available keystore options are documented in the [Oracle Solaris 11.4 ZFS manual](#)

The algorithm to be used is configurable with the following default

```
DATASET_ZPOOL_ALG aes-256-ccm
```

Encryption can be enabled at the first commit when the zpool dataset is created using the 'encrypt' flag. All the filesystems on the zpool will be encrypted.

```
% dataset -c commit name=mydata_enc encrypt
```

#### 4.7.15 Node data filesystems

##### New Feature in VDCF 7.2

node data filesystems can be added and removed similar to filesystems for vServers.

Add them to the VDCF repository first and produce them using the commit operation.

```
% node -c addfs name=g0087 dataset=g0087_root mountpoint=/oradata
% node -c commit name=g0087 fs
```

To remove a filesystem the commit remove flag must be used

```
% node -c remfs name=g0087 mountpoint=/oradata
% node -c commit name=g0087 fs remove
```

## 4.8 Node runtime states

### 4.8.1 Overview

The Runtime States (rState) of Nodes is displayed using the 'node -c show' commands.

#### Node 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.

### 4.8.2 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  
0 2 * * * /opt/jomasoft/vdcf/sbin/vpkgadm_nightly >/dev/null 2>&1  
0 0 1 * * /opt/jomasoft/vdcf/sbin/diskusage_update -F -q >/dev/null 2>&1  
15 21 * * 0 /opt/jomasoft/vdcf/mods/config/console_check >/dev/null 2>&1  
# JSvdcf-base cron
```

## 5 Virtual Server (vServer) Management

### 5.1 Overview

#### 5.1.1 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.

#### 5.1.2 Native vServer

As a first step a new vServer is defined in the Configuration Repository held on the management server. Every native 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.

#### 5.1.3 Kernel vServer

Kernel vServer are independent of the underlying Node. It runs its own kernel and has an own root zpool. When creating a new Kernel vServer the root disk(s) needs to be added using `vserver -c adddisk`.

After the network configuration is configured the Kernel vServer is installed using the "commit" operation.

## 5.2 vServer - Initial Definitions

### 5.2.1 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 six types of vServer: FULL, KERNEL, SPARSE, SOL8, SOL9 and SOL10:

On a Solaris 11 node: only type FULL and SOL10 are available

On a Solaris 10 node: the types FULL, SPARSE, SOL8 and SOL9 are supported.

On a Solaris 11.4 node: the type KERNEL is additionally supported

The types FULL and SPARSE are described within the Solaris Administration Collection for creating Zones. The other types are described in the Solaris branded zones documentation. The `type` argument is optional. If not specified a default will be taken as defined in `customize.cfg`.

vServers of type SOL8, SOL9 and SOL10 are “Solaris 8, Solaris 9 and Solaris 10 Container”. See **chapter 5.7** for details and requirements.

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

A native vServer (type FULL or SPARSE) is installed with the same Solaris Software Version as the target node. The build depends on the Solaris Release. On Solaris 10 the packages which are installed on the Node are deployed into the vServer. On Solaris 11 the build can be defined by an IPS package group. Default group for vServers is the `'group/system/solaris-small-server'` package. Use the `vserver -c modify` command to select another group package or build to be used for the installation.

For KERNEL vServer use the `type=KERNEL` and specify the required resources.

```
% vserver -c create name=mykzone node=s0004 type=KERNEL ram=4g cores=1 comment="Test Zone"
```

RAM and cores (or cpus) are dedicated resources used by the Kernel Zone.  
By default 4 GB RAM is the required minimum.

You can specify the Build to be used with the `vserver modify` operation.

This modify is only required if you need to install a different Solaris Update/SRU then running on the underlying Node or when installing a Unified Archive (uar) based build.

```
% vserver -c modify name=mykzone build=s11u4-sru19
```

### 5.2.1.1 vServer 'autoboot'

By default all vServer are created with zonecfg autoboot flag enabled. You can change this attribute for a vServer using the `vserver -c modify` command. To overwrite the default value you can set the configuration variable `VIRTUAL_AUTOBOOT_DEFAULT`.

```
export VIRTUAL_AUTOBOOT_DEFAULT="FALSE"
```

If you wish to see the configured autoboot setting at `vserver -c show` add the following setting to `customize.cfg`:

```
export VIRTUAL_SYSATTR_SHOW="AUTOBOOT"
```

### 5.2.1.2 vServer 'limitpriv'

#### New Feature in VDCF 8.3

New deployed vServers are configured with the 'default' `limitpriv` and existing `limitpriv` values are imported into VDCF. You can modify the `limitpriv` setting for a vServer using the `vserver -c modify` command.

To define a new default value you can set the configuration variable `VIRTUAL_LIMITPRIV_DEFAULT`.

```
export VIRTUAL_LIMITPRIV_DEFAULT="default"
```

#### Modifying `limitpriv` value for vServer

To change a `limitpriv` value for a vServer use the `modify` command for vServer

```
vserver -c modify name=<vserver name> limitpriv=<comma serperated list>  
vserver -c modify name=<vserver name> clear_limitpriv
```

#### **ATTENTION:**

`VIRTUAL_LIMITPRIV_DEFAULT` will overwrite any `limitpriv` values configured in `/var/opt/jomasoft/vdcf/conf/default.zonecfg` or `/var/opt/jomasoft/vdcf/conf/<VSERVER-NAME>.zonecfg`

Use the '`vserver -c revert`' command to revert a modified `limitpriv` value for a vServer

```
vserver -c revert limitpriv name=<vserver name>
```

### 5.2.1.3 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.

The Categories are also taken into account when shutting down vServers on the target Nodes is required for free resources.

Consult **chapter 4.3.11** (Node evacuation) and the **VDCF HA Guide** for more information about Node Evacuation and Automated Failover.

## 5.2.2 Dataset

Every native 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...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=myappl_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
```

### 5.2.2.1 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.

### 5.2.2.2 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 on Solaris 10 and till Solaris 11.1. On Solaris 11.2 and later the devices are activated while the vServer is running.

### 5.2.2.3 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)
```

### 5.2.3 Kernel vServer root disk

For a native vServer you define a root filesystem, which is mounted on the Node. This is different for Kernel vServers which are using their own root zpool.

Therefore you have to add one or two root disks to define the root zpool for the vServer. As the vServer has his own root zpool, the root filesystem is only visible inside the Kernel vServer.

VDCF currently supports MPXIO SAN, ISCSI LUNs or predefined ZVOLs.

```
% diskadm -c show free node=s0004
% vservers -c adddisk name=mykzone size=40g or guids=600000XXX
```

### 5.2.4 Filesystems

#### 5.2.4.1 root filesystem for native vServer

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.

```
% vservers -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.

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

On Solaris 11 a vServers zonpath must not be in global zone root filesystem. For local vServer you have to make sure manually, that the zonpath is created within a separate filesystem. Dataset based vServers get their own filesystem by default.

#### 5.2.4.2 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.

```
% vservers -c addfs name=v0001 type=data \
dataset=v0001_root size=5g mountpoint=/export
```

#### 5.2.4.3 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!

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

## 5.2.5 Filesystem Usage

### New Feature in VDCF 5.3

The filesystem usage is shown with the “`vserver -c show name=<vServer name>`” command

```
% vserver -c show name=v0153
```

Virtual	C	Type	cState	rState	Build	Patch-Level	Comment
v0153		FULL	ACTIVATED	RUNNING	5.10svz_u10	147440-01 (U10+)	s10 exkl vlan

cPool	Node	nState	Hardware	DataCenter	Comment
prod	s0024	ACTIVE	ORCL,SPARC-T4-1	ZUERICH	Production 1

Filesystem Definitions

```
Dataset: v0153_root (ZPOOL/ACTIVATED) Size/MB: 15360 Unallocated/MB: 5120 Used: 17% warn-over: 80%
```

Name	State	Size/MB	Used	warn-over	Type	Options	Mountpoint
-	ACTIVATED	<undefined>	16% Z	80% (default)	zfs	rw	/zones/v0153 (root)
-	ACTIVATED	<undefined>	0% Z	80% (default)	zfs	rw	/export/wdkp
-	ACTIVATED	<undefined>	0% Z	80% (default)	zfs	rw	/oraexports

For customers not using the VDCF Monitoring feature, the filesystem usage can be manually updated by executing: `/opt/jomasoft/vdcf/sbin/fsmon_update -a | -n <node>`

You may add a crontab entry for the user root to update the usage with your preferred interval:

```
30 * * * * /opt/jomasoft/vdcf/sbin/fsmon_update -qa > /dev/null 2>&1
```

For Standard and Enterprise customers the VDCF Monitoring feature includes an additional command “osmon” which covers Operating System Monitoring. This osmon command can be used to display filesystem usage overviews and to generate alarming eMails, if the defined limits are reached. Because the osmon has it's own cronjob, the fsmon\_update above is not required!

Consult the **VDCF Monitoring Guide** for details about OS Monitoring.

## 5.2.6 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.

```
% vserver -c addnet name=v0001 type=management \  
    ipaddr=10.1.1.101 netmask=255.255.255.0  
  
% vserver -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_MAPPING  
VIRTUAL_NET_MAPPING management:MNGT public:PUBL backup:BACK  
VIRTUAL_NET_MAPPING_S11 management:MNGT public:PUBL backup:PUBL access:ACCESS
```

Description: This definition is used as default when building new vServer networks. It can be overwritten by providing virtual and node type at the `vserver -c addnet` command:

```
% vserver -c addnet name=v0001 type=backup:PUBL ipaddr=v0001
```

Format: "Virtual-Usage:Node-Usage Virtual-Usage:Node-Usage ..."

Explanation: Virtual-Usage is the Net-Type of the vserver.  
 Node-Usage is the Net-Type used in the node configuration.

Hint: The underlying node net types are displayed when using this `vserver` command:

`vserver -c show name=<vserver> verbose:`

```
Network Definitions  
Type vServer->Node Hostname VID Interface IP  
management->MNGT v0153-mngt 20 vnet1 192.168.20.153  
public->PUBL v0153 100 vnet0 192.168.100.153  
backup->BACK v0153-back - vnet2 192.168.200.153
```

### 5.2.6.1 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.

```
% vserver -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` (for Solaris 10 vServers) and `VIRTUAL_NETSTACK_DEFAULT_S11` (for Solaris 11 vServers). Possible values are: SHARED, PRIVATE or EXCLUSIVE.

### 5.2.6.2 Exclusive IP-Stack

To assign a network interface exclusively to your vServer use the `stack=exclusive` setting. 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.

Solaris 10:

Only recommended if your node has enough free network interfaces. You may assign physical or VLAN interfaces. To use this setting, you need to unplumb the interface on the Node. In the Node configuration clear the ip address using

```
% nodecfg -c modify_net name= interface= ipaddr=--
```

Solaris 11:

On Solaris 11 virtual network interfaces (vnics) are used to define exclusive IP-Stack vServers. The generated interface names correlate to the vServer network types i.e. public0 or management0.

See the `anet zonecfg` property for more information about this Solaris feature.

### 5.2.6.3 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.

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

### 5.2.6.4 IPMP Probe IPs

#### New Feature in VDCF 7.1

To activate Probe-Based IPMP on Solaris 11 vServer with exclusive IP-Stack the probe ip addresses can be configured using the `probes` argument:

```
% vserver -c addnet name=v0001 type=public \  
ipaddr=v0001 netmask=255.255.255.0 probes=v0001-if1,v0001-if2
```



### 5.3 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
```

On Solaris 11 the vServer is installed using the Solaris Automated Installer. The commit operation creates all required settings in the Solaris installadm database. The generated manifest and system configuration files are stored as backup copy in the directory `/var/opt/jomasoft/vdcf/zonecfg/${vserver}/` (Files `Manifest.xml` respectively `profiles/base_sc.xml` and `profiles/${vserver}_sc.xml`).

The Automated Installer needs access to an IPS repository. The node's pkg publisher is used as source repository. This repository must be available otherwise the installation will fail.

#### Customization of generated AI xml files

The xml files for AI are generated by VDCF using predefined template files stored in `/opt/jomasoft/vdcf/conf/`. If required you may overwrite these template files. To do so, copy the template file to `/var/opt/jomasoft/vdcf/ai/` and change them accordingly. You may create node specific or global template files. Please do not change the placeholders (marked with `%name%`) which are used to fill the correct values.)

Type	Template files by search order	Description
Base Manifest template file used for: <code>manifest.xml</code>	<code>/var/.../ai/\${vserver}_AI.templ</code>	Customer's vServer specific
	<code>/var/.../ai/s11_vserver_AI.templ</code>	Customer's global default
	<code>/opt/.../conf/s11_vserver_AI.templ</code>	VDCF global default
Include file for additional ips packages	<code>/var/.../ai/{vserver}/ips-pkg.list</code>	Customer's vServer specific package list
	<code>/var/.../ai/all-vservers/ips-pkg.list</code>	Customer's global package list
Global System Config file used for <code>base_SC.xml</code>	<code>/var/.../ai/s11_base_SC.templ</code>	Customer's default
	<code>/opt/.../conf/s11_base_SC.templ</code>	VDCF default
vServer System Config file used for <code>\${vserver}_SC.xml</code>	<code>/var/.../ai/\${vserver}_SC.templ</code>	Customer's vServer specific
	<code>/var/.../ai/s11_vserver_SC.templ</code>	Customer's global default
	<code>/opt/.../conf/s11_vserver_SC.templ</code>	VDCF global default
Include files for additional System configuration profiles	<code>/var/.../ai/{vserver}/*.xml</code>	Customer's vServer specific System Configuration Profiles
	<code>/var/.../ai/all-vservers/*.xml</code>	Customer's global System Configuration Profiles

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
```

### 5.3.1 vServer console

The interactive system console can be accessed using the following command:

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

If Solaris Version 11.2 or later is used, the history of the console output may be displayed using additional flags.

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

or

```
% vserver -c console name=v0001 tail=50
```

or

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

## 5.4 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, Cloning, 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 shared network interfaces activated or deactivated in the active vServer.

For vServers running on Solaris 11.2 or later VDCF does always activate exclusive network interfaces and RAW devices changes online. There is not need to use the 'exec' option there.

See **chapter 5.4.2** for details.

### 5.4.1 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.

## 5.4.2 Apply vServer configurations online

### New Feature in VDCF 5.7

On Nodes installed with Solaris 11.2 or later vServer configurations can be applied to a running vServer. This Solaris feature is called “Live Zone Reconfiguration (LZR)”. It allows to add or remove devices without rebooting the vServer. VDCF uses this feature automatically when you create or remove new exclusive network interfaces or RAW Devices.

### 5.4.2.1 Solaris 11.2 and later vServer usage examples

#### 5.4.2.1.1 Add an additional exclusive network to a vServer

```
-bash-4.1$ vserver -c addnet name=v0178 type=access ipaddr=v0178-access
adding network
Using Netmask 255.255.255.0 from Node g0091
network definitions added

-bash-4.1$ vserver -c commit name=v0178
committing datasets for vServer v0178
dataset commit successful
committing filesystems for vServer v0178
filesystem commit successful
committing vServer v0178 - this may take a moment ...
The following exclusive network interfaces will be updated now:
Adding new interface: access0, 10.1.200.178/24
zone 'v0178': Checking: Adding anet linkname=access0
zone 'v0178': Applying the changes
vServer successfully committed.
commit successful
```

#### 5.4.2.1.2 Remove an exclusive network from a vServer

```
-bash-4.1$ vserver -c remnet name=v0178 type=access
removing network
network definitions removed

-bash-4.1$ vserver -c commit name=v0178
committing datasets for vServer v0178
dataset commit successful
committing filesystems for vServer v0178
filesystem commit successful
committing vServer v0178 - this may take a moment ...
The following exclusive network interfaces will be updated now:
Removing old interface: access0, 10.1.200.178
zone 'v0178': Checking: Removing anet linkname=access0
zone 'v0178': Applying the changes
vServer successfully committed.
commit successful
```

#### 5.4.2.1.3 Add RAW devices to a vServer

```
-bash-4.1$ dataset -c create name=raw type=RAW vserver=v0178 size=5g
Creating vServer dataset <v0178_raw>
Disk 6001438012599B6200011000074E0000 (MPXIO) with Size 5.0 GB selected.
Dataset v0178_raw (RAW) created successfully

-bash-4.1$ dataset -c commit name=v0178_raw
committing dataset changes: v0178_raw
dataset changes committed successfully
```

#### 5.4.2.1.4 Remove RAW devices from a vServer

```
-bash-4.1$ dataset -c remove name=v0178_raw
removing dataset: v0178_raw
dataset removal successful

-bash-4.1$ dataset -c commit name=v0178_raw
committing dataset changes: v0178_raw
dataset changes committed successfully
```

### 5.4.3 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.

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

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

#### New in VDCF 7.0

Using the additional 'keepzfs' argument only the mountpoint will be renamed, the underlying ZFS filesystem keeps its name.

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

## 5.4.4 Cloning ZFS data filesystems

### New Feature in VDCF 6.0

vServers running on the same Solaris 11 Node can share a ZPOOL dataset. ZFS data filesystems can be cloned and mounted on another vServer using the clonefs operation.

The clonefs operation creates a snapshot using a timestamp, if the snapshot argument is not used.

Sample: [v0123\\_db/data/test@20170302-074906](#). Using the optional 'snapshot' argument an existing snapshot can be reused. Then a clone is created and mounted on the vServer or optional target vServer. There is no commit required.

There are tree 'types' of this command. One for creating a clone from a single mountpoint

```
vserver -c clonefs name=<vServer name>  
          mountpoint=</source directory>  
          to=</target directory>  
          [ snapshot=<existing source> ]  
          [ tovserver=<target vServer> ]
```

one for cloning all filesystems on a dataset.

```
vserver -c clonefs name=<vServer name>  
          dataset=<source dataset>  
          basedir=</target base directory>  
          [ snapshot=<existing source> ]  
          [ tovserver=<target vServer> ]
```

### New in VDCF 7.0

and another for cloning only one zfs filesystem or snapshot.

```
vserver -c clonefs name=<vServer name>  
          filesystem=<zfs filesystem or snapshot>  
          to=</target directory>  
          [ tovserver=<target vServer> ]
```

ZFS Properties to be copied from the Source ZFS to the ZFS Clone are globally defined.

```
-bash-4.1$ vdcfadm -c show_config | grep CLONEFS  
VSERVER_CLONEFS_COPY_PROPERTIES logbias,recordsize,compression,primarycache
```

The snapshots can be reused and are never automatically removed. Use zfsadm -c destroy, if the snapshot is not required anymore.

clonefs Sample:

```
-bash-4.1$ vserver -c clonefs name=v0148 dataset=v0148_db tovserver=v0124 basedir=/oradata/newdb  
ZPOOL dataset 'v0148_db' is now shared between vServer v0148 and v0124. You can't detach or migrate  
this vServers anymore  
Cloning filesystem /oradata/prod/data on vServer v0148  
ZFS snapshot created for vServer v0148  
ZFS Property recordsize=8192 set for v0148_db/data/oradata_newdb_data  
Filesystem </oradata/newdb/data> mounted on vServer v0124  
Cloning filesystem /oradata/prod/undo on vServer v0148  
ZFS snapshot created for vServer v0148  
Filesystem </oradata/newdb/undo> mounted on vServer v0124  
vServer successfully committed.  
Cloned all 2 filesystems from Dataset v0148_db of vServer v0148
```

## 5.4.5 Dataset ZPOOL Replication

### New Feature in VDCF 7.1

If you need to reorganize your zpool (fragmentation, amount of disk, ...) you can use the replication feature of VDCF. VDCF will create a snapshot of all filesystems on that dataset and replicate them to another dataset. Actually the replication is only possible on the same node.

1) create a new zpool with the required size and layout

```
$ dataset -c create name=replication vserver=v0177 size=5g  
$ dataset -c commit name=v0177_replication
```

2) start the replication (may take long depending on the amount of data)

```
$ dataset -c replicate name=v0177_data target=v0177_replication
```

3) run replicate again to transfer new changes (delta)

```
$ dataset -c replicate name=v0177_data
```

4) activate replication dataset

This will run the last replication after unmount (to make sure we have consistent data), then commit and mount new dataset

```
$ dataset -c activate_replica name=v0177_data
```

or

4) cancel replication

```
$ dataset -c cancel_replication name=v0177_data
```

This will remove the relationship from the source dataset (v0177\_data) to the replication target dataset (v0177\_replication). The replicated filesystems still exists on the target dataset (v0177\_replication).

Now you could start replication again (step 2). To cleanup the existing filesystem use the 'destroy' option. Or assign the replicated dataset to another vServer

## 5.4.6 Dataset ZPOOL Disk remove

### New Feature in VDCF 7.1 / Disabled by default in VDCF 8.2

With Solaris 11.4 it is possible to remove one or more devices from an existing zpool. Use the 'dataset -c remdisk' to remove a lun from an existing dataset:

Because this operation may have major impact to read performance it is disabled by default. Enable using DATASET\_ALLOW\_REMDISK=TRUE

Details in ZFS Guide Solaris 11.4 / [Removing Devices From a Storage Pool](#)

```
$ dataset -c remdisk name=v0134_data guids=6001438012599B62000110002D100000  
successfully updated dataset layout, use commit to activate it.
```

```
$ dataset -c commit name=v0134_data  
Current State of pending Disk Remove:  
  pool: v0134_data  
  state: ONLINE  
status: One or more devices are being removed.  
action: Wait for the resilver to complete.  
  Run 'zpool status -v' to see device specific details.  
  scan: resilver in progress since Fri Mar 15 13:45:31 2019  
        453K scanned out of 9.67G at 53.5K/s, 2d04h to go  
        0 resilvered  
config:
```

NAME	STATE	READ	WRITE	CKSUM
v0134_data	ONLINE	0	0	0
c1d10	ONLINE	0	0	0
c1d0	REMOVING	0	0	0

```
errors: No known data errors
```

```
WARN: Disk Remove started ... Dataset will be updated in Background after Remove is finished  
dataset changes committed successfully
```

The zpool must have enough space to reallocate the space of the removing devices.

## 5.4.7 Manipulate Mirrors (ZFS and SVM)

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

### 5.4.7.1 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 5.2.2**.

### 5.4.7.2 Conservative ZPOOL Mirroring

#### New Feature in VDCF 7.1

By default `dataset -c attach_mirror/commit` does attach all disks at once. For very large zpools with many disks this may produce too much load on the target system.

You can define how many disk should be attached at once (`DATASET_ATTACH_MIRROR_MAX_DISK`). Additional disks are then attached in the background.

Define at dataset/zpool size, when this feature should be activated.  
(`DATASET_ATTACH_MIRROR_MAX_DISK_MINSIZE_GB`)

### 5.4.7.3 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.

### 5.4.8 Display and manipulate ZFS

VDCF gives you the command `zfsadm` 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
```

#### New Feature in VDCF 6.0

g) Rename a ZFS snapshots

```
$ zfsadm -c rename
```

See the manpages for details about the `zfsadm` operations.

## 5.4.9 Immutable zones

### New Feature in VDCF 5.7

With the Solaris 11 immutable zones feature it's possible to set a zone's root filesystem to read-only to protect the zone.

Using VDCF you can set a vServer to immutable using the proper `file-mac-profile` property using this command. Only already installed vServer can be set to immutable!

```
$ vserver -c modify name=<vserver name> file-mac-profile=<profile name>
```

The value of `file-mac-profile zonecfg` property can be one of these:

<code>strict</code>	Read-only file system, no exceptions
<code>fixed-configuration</code>	Permits updates to <code>/var/*</code> directories, with the exception of directories that contain system configuration components
<code>flexible-configuration</code>	Permits modification of files in <code>/etc/*</code> directories, changes to root's home directory, and updates to <code>/var/*</code> directories
<code>none</code>	Standard, read-write zone (the default)

To activate the `file-mac-profile` change, the vServer has to be committed and rebooted:

```
$ vserver -c modify name=<vserver name>
$ vserver -c commit name=<vserver name>
$ vserver -c reboot name=<vserver name>
```

## 5.5 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 (LUNs) and Networks 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.

### Solaris 11 Zone boot environments (ZBE)

On Solaris 11 a vServer can have multiple boot environments. This can lead to unexpected behavior when migrating vServers to other nodes. While attaching a vServer Solaris tries to select a boot environment that matches best on the new target node. Sometimes Solaris does not select the last active ZBE (this could be the case if the vServer was already on that node before).

To avoid the unintentional activation of an old ZBE VDCF does always attach the last active ZBE. If you would like to attach with a different ZBE use the `vserver detach` and `attach` operation instead.

## 5.5.1 Migration

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

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

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	is candidate	Disk access	Net access	Patch-Level	Packages
s0053 (U6)	NO	ok	ok	nok	nok
s0057 (U7)	NO	ok	nok	nok	nok
s0023 (U10+)	YES [upgrade]	ok	ok	nok	nok

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 compute1 to compute2.
vServer v0001 is down.
vServer v0001 is detached from Node compute1.
vServer v0001 is attached to compute2.
vServer v0001 boot issued ...
vServer migrated successful.
```

### b) Migrate all vServers of a Node

```
% vserver -c migrate source=compute1 node=compute2 shutdown
Migrate vServer v0001 from Node compute1 to compute2.
vServer v0001 is down.
vServer v0001 is detached from Node compute1.
vServer v0001 is attached to compute2.
vServer v0001 boot issued ...
vServer migrated successful.
```

Typically the vServers 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 **chapter 4.3.11** (Node evacuation)

## 5.5.2 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: s0243_root	USED	AVAIL	REFER	MOUNTPOINT
s0243_root/root@vdcf_upgrade_2009_03_09-09:02:51	1.24M	-	24.1M	-

### 5.5.3 Kernel vServer – Live Migration

#### New Feature in VDCF 8.0

KERNEL vServer can be cold or live migrated to another Node.

Use the `vserver show candidates` command for potential target Nodes and the `vServer migrate` operation to execute.

```
$ vserver -c show name=v0191 candidates full

vServer Type      State      Node      cPool      Build      OS      Patch-Level
v0191  KERNEL    ACTIVATED  g0103      sol11      s1lu4-sru16  11  4.16.0.1.4.0 (U4.SRU16)

Potential Nodes      is candidate  Disk access  Net access      CPU/RAM resource
g0081 (S11 U4.SRU17)  NO           ok           ok              nok
s0003 (S11 U4.SRU17)  YES [live]   ok           ok              ok
s0013 (S11 U4.SRU12)  NO           nok          nok (MNGT)      ok

$ vserver -c migrate name=v0191 node=s0003 live

Preparing Target Node s0003
Executing Trial Migration ... OK
zoneadm: zone 'v0191': Using existing zone configuration on destination.
zoneadm: zone 'v0191': Attaching zone.
zoneadm: zone 'v0191': Booting zone in 'migrating-in' mode.
zoneadm: zone 'v0191': Checking live migration compatibility.
zoneadm: zone 'v0191': Performing initial copy (total 5120MB).
zoneadm: zone 'v0191': 0.00% done: 0MB copied @ 0.0MB/s, skipped 0MB
zoneadm: zone 'v0191': 49.95% done: 512MB copied @ 102.4MB/s, skipped 2045MB
zoneadm: zone 'v0191': 88.27% done: 1024MB copied @ 102.4MB/s, skipped 3495MB
zoneadm: zone 'v0191': 100.00% done: 1302MB copied @ 55.6MB/s, skipped 3817MB
zoneadm: zone 'v0191': Performing copy of recently modified memory.
zoneadm: zone 'v0191': Suspending zone on source host.
zoneadm: zone 'v0191': Waiting for migration to complete.
zoneadm: zone 'v0191': Migration successful.
zoneadm: zone 'v0191': Halting and detaching zone on source host.
Live Migration of Kernel Zone v0191 done. Cleaning up
```

## 5.6 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.

### 5.6.1 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.

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

### 5.6.2 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.

```
% vsriver -c detach name=v0001 force
% vsriver -c attach name=v0001 node=<failover node>
% vsriver -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:

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

Now reboot the original failed node:

```
{28} ok boot
```

Finally re-attach the vServer again:

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

## 5.7 Solaris Branded Zones

Solaris provides a feature called 'Branded zones'. This allows individual vServer to have another Solaris Release installed than the global zone (Node).

### 5.7.1 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.

#### 5.7.1.1 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)

#### 5.7.1.2 SOL8 vServer

To create a new Solaris8 vServer use:

```
vserver -c create name=myserver type=SOL8 node=xy comment="MyApp"
```

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
```

## 5.7.2 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.

### 5.7.2.1 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)

### 5.7.2.2 SOL9 vServer

To create a new Solaris9 vServer use:

```
vserver -c create name=myserver type=SOL9 node=xy comment="MyApp"
```

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
```

### 5.7.3 Solaris 10 branded zones

Solaris 10 branded zones is a feature of Solaris 11 to create non-global zones on a Solaris 11 Node based on an image taken from an existing Solaris 10 System or zone.

Read the Oracle documentation “Oracle Solaris Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones” for details.

#### 5.7.3.1 Requirements

- Source system installed with Solaris 10 U9 or later
- Target Nodes must be installed using Solaris 11 GA or later
- Install the “system/zones/brand/brand-solaris10” package
- A system image of the zone to be installed:  
Make a flar archive if the source system is a Solaris 10 global zone.  
Use a cpio.gz archive if the source system is a Solaris 10 non-global zone.

#### Steps to create an archive of a vServer:

Put vServer in ready state:

```
source# zoneadm -z my-zone halt
source# zoneadm -z my-zone ready
```

and create the cpio archive:

```
source# cd /zones/my-zone
source# find root -print | cpio -oP@ | gzip >/zones/my-zone.cpio.gz
```

#### Steps to create an archive of a Node:

Detach all vServers from node and then create the flar archive:

```
source# zoneadm list -cv
  ID NAME      STATUS   PATH          BRAND   IP
   0 global    running  /              native  shared
source# cd /

source# flarcreate -S -n my-node -x /var/spool/flar/my-node.flar -L pax
/var/spool/flar/my-node.flar
```

#### 5.7.3.2 SOL10 vServer

To create a new Solaris10 vServer use:

```
vserver -c create name=myserver type=SOL10 node=xy comment="MyApp"
```

When committing the SOL10 vServer the Solaris 10 image must be available on the Target Node as /var/tmp/images/<vserver>.flar respectively /var/tmp/images/<vserver>.cpio.gz.

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
```

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

### 5.8.1 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.

```
% vsserver -c shutdown name=v0001
% vsserver -c commit name=v0001 uninstall
% vsserver -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 5.8.2**

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

### 5.8.2 Remove vServer

To completely remove a vServer from the node you may use the `vsserver -c destroy` operation. This will cleanup the vServer as described in **chapter 5.8.1** and remove all definitions from the database (be careful!). Or if you prefer you can execute these commands separately:

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

After this steps the vServer is completely destroyed.

### 5.8.3 Remove a DETACHED vServer

The following procedure should not be used in daily operations, because the data on the filesystems and disks cannot be deleted. This may lead to troubles if the disks are reused later.

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.

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

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

```
% vsserver -c remove name=v0001 force
```

## 5.9 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 **chapter 9** for details.

## 5.10 vServer Dependencies

### New Feature in VDCF 5.3

Dependencies for vServer can be used to make sure other vServers are running or stopped, when a dependent vServer is booted or shutdown. If the variable 'VSERVER\_CHECK\_DEPEND' in customize.cfg is set to 'TRUE', VDCF does check the dependencies always if a command does request a running state change for a vServer. For example it does not allow to shutdown a master, if a slave vServer is still running. This applies also to boot requests. You cannot boot a vServer if the masters are not running. You are not allowed to add or remove dependencies if you don't have the slave vServer in one of your vPools.

As an example an application server on vServer 'appsrv' does need a database server on vServer 'dbsrv'. To create this dependency you define it like this:

```
$ dependadm -c add master=dbsrv slave=appsrv
```

And the webserver 'websrv' needs the application server to be running:

```
$ dependadm -c add master=appsrv slave=websrv
```

Show the direct dependencies:

```
$ dependadm -c show vserver=appsrv
      Name      rState      Node      Comment
Master(s):  dbsrv      RUNNING     NodeA     DB Srv
|
vServer:    appsrv     RUNNING     NodeB     App Srv
|
Slave(s):   websrv     RUNNING     NodeC     Web Srv
```

If you try to shutdown the application server you will get this ERROR:

```
$ vserver -c shutdown name=appsrv
shutdown command being issued for vServer(s)
shutdown vServer appsrv ...
ERROR: Slave 'websrv' of Master 'appsrv' has rState 'running'. Must not be running.
ERROR: Some Slaves are running. Not allowed to shutdown vServer appsrv.
ERROR: could not shutdown vServer(s)
```

You can show all defined dependencies with:

```
$ dependadm -c show
```

## 5.11 vServer Import – import existing zones into VDCF

This command can be used to import existing Solaris zones from a running system into VDCF. Imported vServer can be managed the same way as vServers installed by VDCF.

To get all the functionality for vServer operations, the imported vServer should be on a SAN LUN with ZFS. vServer installed on the boot/local disks of the node, will be imported as local vServer and can't be migrated to other nodes. The import function detects filesystems, network configurations and also resource settings from the imported vServer.

It is possible to import all vServers of a Node or just one of them. An example:

```
$ vserver -c import node=s0010

vServer v0111 (imported vServer) is created.
dataset successfully created: v0111
Filesystem /zones/v0111 defined on dataset v0111 (ZPOOL).
IP-Address 192.168.20.111 resolves to v0111-mngt.
Network management for vServer v0111 defined.
IP-Address 192.168.100.111 resolves to v0111.
Network public for vServer v0111 defined.
vServer v0112 (imported vServer) is created.
dataset successfully created: v0112
Filesystem /zones/v0112 defined on dataset v0112 (ZPOOL).
IP-Address 192.168.20.112 resolves to v0112-mngt.
Network management for vServer v0112 defined.
IP-Address 192.168.100.112 resolves to v0112.
Network public for vServer v0112 defined.
Checking PatchLevel of Node s0010 ...
Analyzing PKGs of Node s0010 ...
```

If not specified the vServers are assigned to the default vPool. The User who executes the import, has to be an administrator assigned to that vPool. If you assign the vServer to more than one vPool, the user must belong to at least one of these vPools.

The vServer import can be run more than once for a vServer/node. With each run only the missing resources are imported into the database. Do not leave a partially imported vServer in the VDCF database. Run the import again or remove the vServer completely.

## 5.12 vServer verify – verify the vServer configuration

This command can be used to compare the information in the database with the effective values in the zonecfg discovered on the system.

This command reports differences in:

- assigned node for vServer
- filesystem (zoneroot and data)
- network
- resource settings

Following an example where additional settings have been detected:

```
$ vserver -c verify name=v0134
```

```
Verify for vServer v0134 is running, please wait
```

```
WARN: Filesystem with mountpoint /app/data15 (zfs) was found on vServer v0134 (g0061) but is missing in VDCF
```

```
WARN: vServer v0134 (g0061) Resource setting: zone.max-swap / (priv=privileged,limit=1073741824,action=deny) (1024MB) is not defined on VDCF
```

```
WARN: vServer v0134 (g0061) Resource setting: zone.max-locked-memory / (priv=privileged,limit=3221225472,action=deny) (3072MB) is not defined on VDCF
```

Use `vserver -c import` to add these additional values to VDCF repository.

### New Feature in VDCF 7.2

Changed values can be updated in the VDCF repository with the 'update' flag.

```
$ vserver -c verify node=g0081
```

```
Verify for vServer on node g0081 is running, please wait
```

```
WARN: Filesystem with mountpoint /export (zfs) has a different size. vServer v0161 (Node g0081): 1024MB / VDCF: 0MB
```

```
WARN: Filesystem with mountpoint /export/home (zfs) has a different size. vServer v0161 (Node g0081): 1024MB / VDCF: 0MB
```

```
WARN: Filesystem with mountpoint /export/home/admin (zfs) has a different size. vServer v0161 (Node g0081): 100MB / VDCF: 0MB
```

```
$ vserver -c verify node=g0081 update
```

```
Verify for vServer on node g0081 is running, please wait
```

```
Filesystem attribute for mountpoint /export (zfs) has been updated: Size: 0MB -> 1024MB
```

```
Filesystem attribute for mountpoint /export/home (zfs) has been updated: Size: 0MB -> 1024MB
```

```
Filesystem attribute for mountpoint /export/home/admin (zfs) has been updated: Size: 0MB -> 100MB
```

## 5.13 vServer Runtime States

### 5.13.1 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.
UNAVAILABLE (since Solaris 11.1)	Indicates that the zone has been installed but cannot be booted. A zone enters the unavailable state when the zone's storage is unavailable while <code>svc:/system/zones:default</code> is on-line or when the zone tries to boot; when archive-based installations fail after successful archive extraction; and when the zone's software is incompatible with the global zone's software, such as after an improper forced attach.

### 5.13.2 Cronjob

The Runtime States (rState) are updated in the VDCF configuration repository using a cronjob.

See **chapter 4.8.2** for more details.

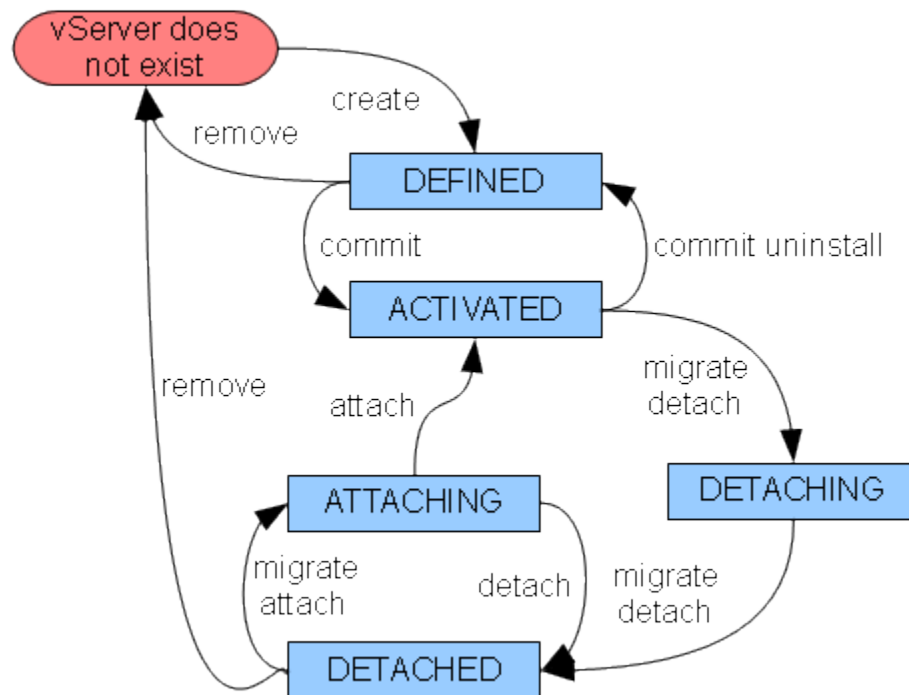
## 5.14 vServer Configuration States

### 5.14.1 Overview

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

See **chapter 7** for details about possible configuration states and the meaning of them.

### 5.14.2 vServer cState diagram



### 5.14.3 Supported vserver commands

Depending the cState of a vServer 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 vServer has the state **DEFINED** these vserver operations are possible:

- `vserver -c remove` (only if no datasets, filesystems or networks assigned)
- `vserver -c adddisk`
- `vserver -c adddfs`
- `vserver -c growfs`
- `vserver -c renamefs`
- `vserver -c remdisk`
- `vserver -c remfs`
- `vserver -c addnet`
- `vserver -c remnet`
- `vserver -c revert`
- `vserver -c modify`
- `vserver -c commit`
- `vserver -c destroy`

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

- `vserver -c attach_root_mirror`
- `vserver -c adddfs`
- `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 modify`
- `vserver -c commit`
- `vserver -c apply`
- `vserver -c migrate`
- `vserver -c detach`
- `vserver -c boot`
- `vserver -c reboot`
- `vserver -c shutdown`
- `vserver -c console`
- `vserver -c destroy`
- `vserver -c verify`
- `vserver -c upgrade`
- `vserver -c upgrade_check`
- `vserver -c upgrade_finish`
- `vserver -c upgrade_failback`

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`

## 6 Logical Domain Management

### 6.1 Overview

To use the Logical Domain features of VDCF a CMT system needs to be setup as a VDCF Node. The required steps include Node Discovery, Profile and Node configuration and Node install. See **chapter 4.3.2 ff** for details.

The Logical Domain Software (SUNWldm package / Version 3.2 is recommended) must be installed on Solaris 10 nodes. 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
```

**It is highly recommended to use the latest Solaris Version for the Control Domain.**

The current VDCF Release still supports Solaris 10 for a Control Domain, but the use is deprecated. Support for Solaris 10 Control Domains will be removed from VDCF in the future.

On Solaris 11 this package is already preinstalled.  
LDom Version 3.3 and later are only available for Solaris 11.

#### 6.1.1 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.

#### 6.1.2 Guest Domain (gdom)

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

A gdom requires these minimal configuration settings:

- a SAN disk where the Solaris OS is to be installed
- 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 for Solaris 10 or using the "`node`" command for Solaris 11.

for Solaris 10

- a partitioning profile, which defines how to setup the root disk and filesystems

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.

### 6.1.3 Stages

#### New Feature in VDCF 9.0

Using the customize.cfg variable "STAGES" you can define your own stages, for example dev, test and prod:

```
export STAGES="dev,test,prod"
```

On a control domain you can assign all the stages which are allowed for that control domain. On guest domains you only can assign one stage. The stage of the guest domain must also be assigned on the control domain.

To make sure every guest and control domain has an assigned stage, you can make the 'stage' argument of cdom -c create and gdom -c create mandatory with the variable STAGE\_REQUIRED.

```
export STAGE_REQUIRED="NODE"
```

If you want to make the stage mandatory for all vServer and all Nodes, you can set it to

```
export STAGE_REQUIRED="ALL"
```

To reject migration of guest domains to control domains which don't allow the assigned guest domain stage you can define

```
export GDOM_MIGRATE_CHECK_STAGE="TRUE"
```

In the following scenario with  
GDom1 with stage dev  
CDomA with stage prod  
CDomB with stages dev,test

You can migrate GDom1 only to CDomB and not to CDomA.

Stage is currently optional for vserver.

serverconfig -c exec servertype has an optional 'stage' argument

## 6.2 Control domain (cdom)

### 6.2.1 cdom definition

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

```
% cdom -c create name=computeA cores=2 ram=5120
```

### 6.2.2 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, if the LDom software is deployed using VDCF (serverconfig).

### 6.2.3 cdom remove

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

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

### 6.2.4 cdom discover

This command can be used to discover control domains not created by VDCF or to refresh the control domain services stored in VDCF.

Before using this command, the control domain to be discovered must already be registered as a node.

After the discover the control domain is visible in VDCF and can be used to deploy new guest domains on it.

```
% cdom -c discover name=computeB
```

The command analyzes a running physical node for ldm configuration. The following ldm information is discovered: ldm version, IO domains, details about ldm services like virtual console concentrator (VCC), virtual disk servers (VDS) and virtual network switches (VSW).

## 6.2.5 cdom show

Using 'cdom -c show' you see the current allocation, the available resources and the configured virtual services of the server.

```
% cdom -c show name=s0013
```

### Server Information

```
ORCL,SPARC-T4-2 CPU Socket: 2 VCPUs: 128 x SPARC-T4 2848MHz on 16 Cores (8 Threads/Core)
```

Domain Information			Ldom Version: 3.5.0.2.1							
Type	Name	State	OS	Cores	VCPUs	RAM/MB	CPU%	RAM%	#GDom	#vServer
NODE	s0013	ACTIVE	-	16	128	261568	100	100	4	2
CDOM	primary	ACTIVE	11	2	16	9216	12	3		
IODOM	secondary	ACTIVE	11	3	24	2048	18	0		
GDOM	(summary)	-	-	5	40	12288	31	5		
GDOM	g0067	ACTIVE	10	1	8	2048	6	0		1
GDOM	g0088	ACTIVE	11	2	16	8192	12	3		1
GDOM	g0092	ACTIVE	11	2	16	2048	12	0		0
LEFT	-	-	-	6	48	238076	38	91		

Domain Services	Name	Owner	Interface	Usage
Disk Service (VDS)	primary-vds0	primary		
Console Service (VCC)	primary-vcc0	primary		
Disk Service (VDS)	secondary-vds0	secondary		
Console Service (VCC)	secondary-vcc0	secondary		
Network Switch (VSW)	management-vsw0	primary	igb0	MNGT
Network Switch (VSW)	management-vsw02	primary	nxge0	MNGT
Network Switch (VSW)	public-vsw0	primary	igb1	PUBL
Network Switch (VSW)	secondary-mngt-vsw0	secondary	igb0	MNGT
Network Switch (VSW)	secondary-public-vsw0	secondary	igb1	NONE

## 6.2.6 cdom evacuation

### New Feature in VDCF 9.0

CDom evacuation is a VDCF HA feature which is available to VDCF Enterprise customers.

The evacuation feature distributes the GDoms/LDoms from one CDom to the other compatible CDoms which have enough resources (CPU and RAM). This feature may be used for planned maintenance or if a CDom fails. This evacuation feature is used by the VDCF High Availability Monitoring (hamon). Consult the VDCF HA Guide for more information about hamon.

## 6.3 Guest domain (gdom) configuration

### 6.3.1 gdom initial definition

Guests domains may be defined on activated Control domains.

Variable: **GDOM\_CPU\_ARGS** (New since VDCF 8.0)

Description: set the allowed CPU args for gdom creation/modification

Default: "VCPUS, CORES, MAXCORES"

Explanation: VCPUS set CPU threads

CORES set whole cores

MAXCORES set the max-cores property to be compliant with hard partition licensing

```
% gdom -c create name=g0001 cdom=computeA max-cores=2 ram=1g comment="Test gdom"
```

### 6.3.2 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
```

### 6.3.3 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/GB
-	FREE	MPXIO	UNUSED	600015D00002EE0...040D0	03461147	16.0
-	FREE	MPXIO	UNUSED	600015D00002EE0...040EA	03461147	4.0
-	FREE	MPXIO	UNUSED	600015D00002EE0...040ED	03461147	4.0
-	FREE	MPXIO	UNUSED	600015D00002EE0...040F0	03461147	4.0
-	FREE	MPXIO	UNUSED	600015D00002EE0...040F3	03461147	4.0

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
```

Or you can let VDCF choose an appropriate disk for you using the size argument:

```
% gdom -c adddisk name=g0001 type=root size=16g
```

If you define multiple root disks, matching Size and different Locations are checked.

### 6.3.4 Network

A gdom needs networks to be assigned. A network of type management is always required. If only one network is defined, the management interface also serves as the 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.

Depending on the Control Domain configuration the IPMP is automatically configured for the GDom.

#### 6.3.4.1 Network type definitions

If you define new network types you have to add the new ones to these configuration variables as well:

Variable: **GDOM\_NET\_MAPPING**

Description: Network name mapping definition for the different network types.  
 Format: "Gdom-Type:Node-Usage:Cdom-Usage Gdom-Type:Node-Usage:Cdom-Usage ..."  
 Sample: "management:MNGT:MNGT public:PUBL:PUBL backup:BACK:BACK"  
 Explanation: Gdom-Type is the Net-Type used in the gdom -c addnet command.  
 Node-Usage is the Net-Type used in the node configuration of the Guest domain.  
 Cdom-Usage is the Net-Type used in the CDom Node configuration.

For each Net-Type used on control domains the CDOM\_VSW\_NAME\_MAPPING variable must be modified:

Variable: **CDOM\_VSW\_NAME\_MAPPING**

Description: Mapping of Idm Virtual Switch (VSW) names to VDCF cdom network types. This definition is used when creating new or discovering existing control domains.  
 Format: "Cdom-Usage:vnet:VSW-Name Cdom-Usage:vnet:VSW-Name ..."  
 Sample: "MNGT:vnet0:management-vsw0 PUBL:vnet1:publ-vsw0 BACK:vnet1:back-vsw0"  
 Explanation: Cdom-Usage is the Net-Type used in the CDom Node configuration.  
 vnet: do not change this value, vnet0 is reserved for management net, all others should be set to vnet1  
 VSW-Name is the name as it's used to define Idm virtual switches.

Hint: The underlying node net types are displayed when using the gdom -c show name=<gdom> command in verbose mode:

```
Network Interfaces
Type GDom->CDom Hostname pVID Interface
management->MNGT g0069-mngt - vnet0->e1000g0
public->PUBL g0069 - vnet1->e1000g1
backup->MNGT g0069-sc-access 10 vnet2->e1000g0
access->BACK g0069-access - IPMP: vnet3,vnet4 (access)
```



### 6.3.5 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   ACTIVE   ACTIVE   default 2014-07-31 00:25:56 2014-07-31 00:25:06

  Active Build   Enabled Build   #V   Partitioning Profile   Configuration Groups
s11u2-s s11u2-s         4   ldom_zfs_partitioning.cfg   node,MSP

Resources

- - - Guest Domain - - - - - - - - - - - - - - - Control Domain usage - - - - - - - - -
Cores  Max-Cores VCPUs   RAM/MB MAU   CPU%   RAM%   Control Domain   Ldm Version
1      0         8      1024   0     12     3     s0024 ORCL,SPARC-T4   3.1.1.0.5

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

Network Interfaces
Type  Hostname   pVID  Interface           IP           Netmask       State
public g0001      -     vnet0->e1000g0     10.31.200.60 255.255.255.0 DEFINED
management g0001-mngt -     vnet1->e1000g1     192.168.0.60 255.255.255.0 DEFINED
```

### 6.3.6 Installation

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

For Solaris 10 using the build and flash commands:

```
% build -c show

  Build Version  OS Version  Platform Arch   Method  Type   Build Name
5.10sv_U9w_all  5.10 (U9)   sparc     sun4v  WAN     ufs    s10_v9_Xall
5.10sv_u8w_all  5.10 (U8)   sparc     sun4v  STD     ufs    s10_v8_Xall
5.10sv_u8w_req  5.10 (U8)   sparc     sun4v  WAN     zfs    s10_v8_req

% flash -c enable_install node=g0001 version=5.10sv_U9w_all

Found GDom: g0001 Model: (8) x UltraSPARC-T2 thread at 1165MHz 2048MB RAM
Found network boot device on management network: vnet1, 192.168.0.60/255.255.255.0
Installation (WAN boot) enabled for Node: g0001 Version: 5.10sv_U9w_all
```

For Solaris 11 the node -c enable\_install command is used:

```
% node -c enable_install name=g0001 build=s11.1-srul

Found GDom: g0001 Model: ORCL,SPARC-T4-1 (8) x SPARC-T4 thread at 2848MHz 2048MB RAM
Found network boot device on management network: vnet0, 192.168.20.76/255.255.255.0
Client 00:14:4f:fa:30:81 added to AI service 's11u1'
Install your node using 'node -c install name=g0001'
```

Then use this command to install the guest domain:

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

### 6.3.7 Console

The interactive system console can be accessed using the following command:

```
% gdom -c console name=g0001
```

If Version 3.0 or later of the Oracle LDom Software is used on Solaris 11+ control domains, the history of the console output may be displayed using additional flags.

```
% gdom -c console name=g0001 history
```

or

```
% gdom -c console name=g0001 tail=50
```

or

```
% gdom -c console name=g0001 follow
```

## 6.4 Root IO Domains and Split IO GDom

### New since VDCF LDom 5.6

#### 6.4.1 Overview

On physical Servers with multiple PCI Buses multiple Root IO Domains may be defined, where each Root IO Domain uses one or more PCI Bus exclusively. If a Root IO Domain has PCI Buses assigned with PCI Cards for SAN Disk Access and Networking it runs independent of the Control Domain. If the Control Domain is down for maintenance, the Root IO Domains still runs.

Before doing Maintenance on the Control Domain in traditional environments you (live) migrate the guests to other Control Domains. If the environment has only a few large physical servers with low free resource such a guest evacuation is no option. In such environments Split IO GDom's are used. Split IO GDom's use Disk and Network IO from both CDom and Root IO Domain. If the CDom or Root IO Domain is down the Guest switches to the IO Resources from the other running Domain.

Root IO Domains must be setup manually. See details in the Oracle LDom Admin Guide [https://docs.oracle.com/cd/E48724\\_01/html/E48732/configurepciexpressbusesacrossmultipleldoms.html#scrolltoc](https://docs.oracle.com/cd/E48724_01/html/E48732/configurepciexpressbusesacrossmultipleldoms.html#scrolltoc)

VDCF is able to discover such Root IO Domains. VDCF uses the term "IODOM" for such Root IO Domains. On Server with such IODOM's VDCF deploys the GDom's automatically as Split IO GDom's.

Current Limitation: Split IO Domains are supported if MPXIO LUNs are used.

To enable this features add to customize.cfg  
`export IODOM="TRUE"`

#### 6.4.2 Root IO Domain Import

Root IO Domains are added to VDCF similar to Physical Nodes.

Use the URL which is configured on the VDCF Management Server:

```
% vdcfadm -c show_config | grep FLASH_WEBSERVER_URL
FLASH_WEBSERVER_URL http://192.168.0.2:80
```

Install the VDCF Client Package on the Root IO Domain:

```
# export FLASH_WEBSERVER_URL=http://192.168.0.2:80
# wget $FLASH_WEBSERVER_URL/pkg/`uname -p`/JSvdcf-client.pkg
# yes | pkgadd -d ./JSvdcf-client.pkg all
```

To allow ssh communication between VDCF and the Root IO Domain, ssh keys must be deployed using a VDCF client tool. Execute as root:

```
# /opt/jomasoft/vdcf/client/sbin/update_key -u $FLASH_WEBSERVER_URL
```

```
-bash-4.4$ node -c import name=g0088
Importing new Node g0088 ...
Discover Systeminfo ...
Discover Rootdiskinfo ...
Discover Diskinfo ...
This may take some time, it depends on the number of disks
.....
Discover Netinfo ...
Root IO Domain secondary (Hostname g0088) detected on CDom s0013. Node
Discover file saved as s0013_secondary
Root IO Domain detected. Adding node with Name s0013_secondary
Node configuration successfully added.
Found Control Domain s0013 (primary).
Found Root IO Domain secondary on Node s0013.
Discovering Root IO Domain secondary on Node s0013 ...
Successfully discovered Root IO Domain s0013_secondary as a GDom.
Root IO Domain detected. Using s0013_secondary as Node Name ...
WARN: Node has S11 Patchlevel 3.4.0.5.0 (U3.SRU4). Patchlevel 3.4.0.5.0 set
as build name.
System registration done for s0013_secondary.
registering disks from node s0013_secondary
New visible Lun 6001438012599B62000110002A0E0000 Size: 20.00 GB
New visible Lun 6001438012599B6200011000293A0000 Size: 20.00 GB
New visible Lun 6001438012599B62000110002A120000 Size: 20.00 GB
New visible Lun 6001438012599B62000110002A160000 Size: 20.00 GB
New visible Lun 6001438012599B62000110001A410000 Size: 10.00 GB
New visible Lun 6001438012599B620001100023230000 Size: 20.00 GB
New visible Lun 6001438012599B620001100028810000 Size: 20.00 GB
New visible Lun 6001438012599B620001100028890000 Size: 20.00 GB
New visible Lun 6001438012599B620001100029420000 Size: 20.00 GB
Node s0013_secondary import finished
```

Root IO Domains are typically named “secondary”, “alternate” or similar. To make sure the Root IO Domain can be uniquely identified VDCF concatenates the CDom and IODom Names.

In the sample above the CDom s0013 and IODom secondary is concatenated to s0013\_secondary.

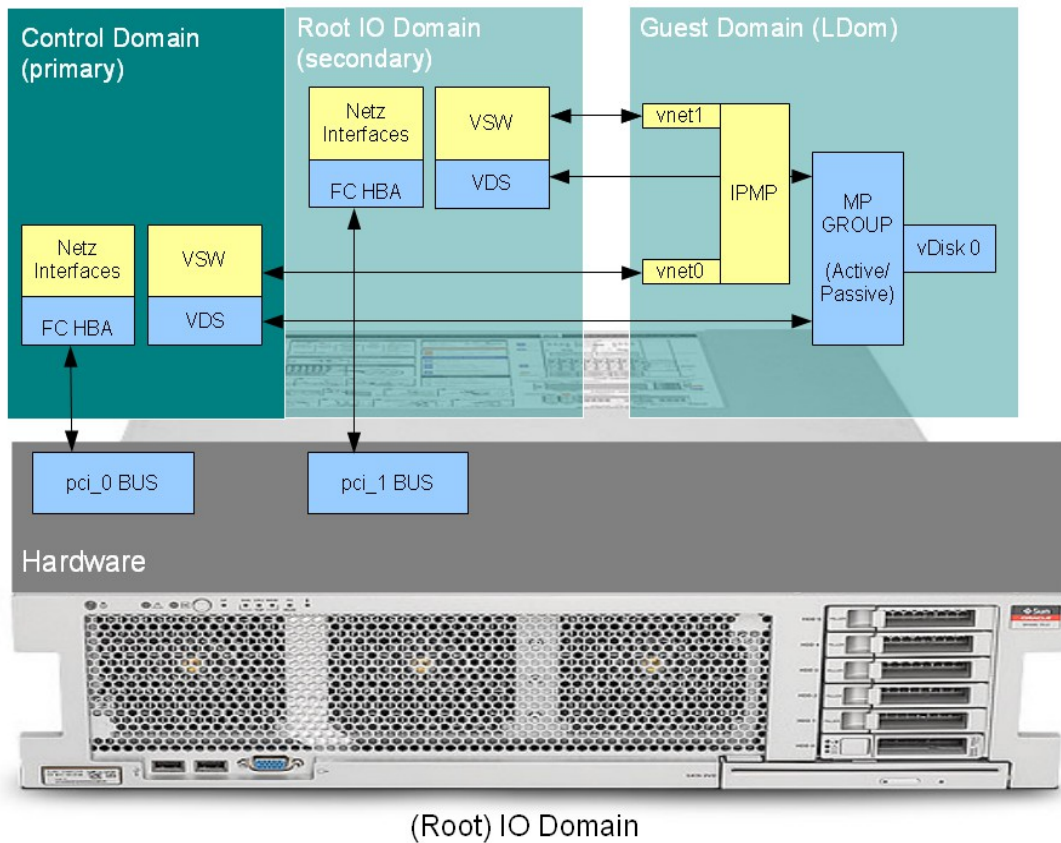
The Network Types must be configured to make VDCF aware which Networks are available in the IODom.

```
-bash-4.1$ nodecfg -c modify_net name=s0013_secondary interface=igbl
nettype=PUBL
Changed Net-Type of VirtualSwitch (VSW) 'secondary-public-vsw0' on CDom
's0013' to 'PUBL'
node network configuration modified successfully.
```

The IODom is added as GDom in the VDCF Repository. Operational Tasks like modify RAM, CPU Resources, Rebooting, Solaris Upgrades are supported.

### 6.4.3 Split IO GDom Setup

Guest Domains deployed on Control Domains with discovered IO Dom are setup using IPMP for Network Redundancy and MP GROUPS for the Setup of the SAN LUNs.



## 6.5 Guest domain (gdom) operation

Apart from run level functionality (boot, shutdown and reboot) the framework offers the possibility to modify system resources (disks, network interfaces, CPUs, Memory and MAUs).

- **Disks:** Adding and Removing
- **Network:** Adding and Removing
- **Resources:** vCPU, CPU cores/max-cores, 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 ram=8g
```

The changes are activated using the `commit` operation.

Some changes, like modifying the Memory allocation are so called “delayed-reconfigurations”. Such configurations are activated at the next reboot of the guest domain. Dynamic Memory changes are supported with LDom version 2.0 or later and Solaris 10 Update 9 or later.

Setting the CPU max-cores constraint is available with LDOM version 2.0 or later. Using max-cores the Oracle Software Licensing requirements for Virtualization using LDomS can be full-filled.

### 6.5.1 Readonly GDomS

A Guest Domain can be set to the READONLY State (cState) to prevent changes made to the GDom. This makes sense, if imported Gdoms are managed by another software and should not be changed within VDCF. This means all VDCF commands concerning these GDOMS are blocked. Rebooting and updating (refreshing the state) the Guest Domain is still possible with READONLY State. Setting State to READONLY:

```
% gdom -c modify name=g0001 readonly=true
```

READONLY State can also be set automatically after import. See **chapter 4.3.5**

## 6.6 Guest domain (gdom) migration

A guest domain may require better performance or some maintenance work is planned on the current control domain (for example patching or firmware upgrades). Typically this should not have a negative impact on the availability of the guest domain.

VDCF offers a solution to this problem and facilitates gdom migration between CMT systems of the same ComputePool (cPool). Both source and destination systems need access to the LUN's and Networks a gdom (and its containing vServers) relies on. And the target system needs to have enough free resources to support the guest domain.

If a gdom has physical IO devices configured (aka IO domain), then the migration to another control domain is not allowed by default. To disable the IO domain check use the optional flag 'noiocheck'.

### 6.6.1 Candidates

You can display the candidate control domains using the gdom show candidates operation:

```
% gdom -c show candidates name=g0001 full
```

Name	rState	CDom	VCPUs	RAM/GB	vServer	Comment
g0001	ACTIVE (RUNNING)	c0001	8	10	no	Guest domain g0001

Potential CDoms	is candidate	Disk access	Net access	CPU/RAM Resource	Node Info
c0002	<b>[live]</b> : YES	ok	ok	ok	SPARC-T4-1
c0003	NO	nok	ok	ok	T5440
c0004	NO	ok	ok	nok (free CPU:10 RAM:4096)	

**[live]** indicates the possibility to migrate while the gdom remains running. There is no downtime for applications and vServers inside this gdom. Such live migrations are supported, if both cdoms contain the same CPUs, run Solaris 10 Update 9 or later and LDOM software version 2.1 or later. Live Migration between different CPUs is supported for Solaris 11+ Guest Domains running on LDOM 2.2 or later. For migrations to different hardware it's required to shutdown the gdom first.

The details why a Cdom is not a candidate can be found in the framework.log

```
% vdcfadm -c show_log tail=20
```

## 6.6.2 Migrate (live and cold)

Using the migrate operation you transfer a guest domain from a source control domain to a new control domain.

### 6.6.2.1 Live Migration of a guest domain

Migrating a guest domain to another CMT System using Live Migration:

```
$ gdom -c migrate name=g0001 cdom=c0002 live
Migrate guest domain g0001 from CDom c0001 to c0002 (live).
Guest domain g0001 migrated successfully.
```

The duration of a Live Migration depends on the number of CPU's used by the control domain (2 cores is recommended) and the amount of Memory used by the guest domain.

### 6.6.2.2 Cold Migration of a guest domain

The guest domain must be stopped (rState: BOUND) prior to migration. Be aware, that running vServers will be stopped with the shutdown flag!

```
% gdom -c migrate name=g0001 cdom=c0002
or
% gdom -c migrate name=g0001 cdom=c0002 shutdown

Migrate guest domain g0001 from CDom c0001 to c0002 (cold).
Guest domain g0001 is down.
Guest domain g0001 is detached from CDom c0001.
Guest domain g0001 is attached to c0002.
Guest domain g0001 migrated successfully.
```

## 6.6.3 Detach / Attach

In addition the framework supports the execution of the four migration steps (shutdown, detach, attach, boot) as individual operations.

A guest domain can be migrated to other CMT systems including all its containing vServers.

To do that, this command sequence has to be applied:

```
% gdom -c shutdown name=g0001
% gdom -c detach name=g0001

or

% gdom -c detach name=g0001 shutdown

% gdom -c attach name=g0001 cdom=c0002
% gdom -c boot name=g0001

or

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

The new SPARC CPUs (SPARC S7,M7 and M8) require a minimum Solaris Patch Level, this will be checked during an attach of a guest domain.



## 6.7 Guest domain (gdom) cleanup / destroy

To completely remove a guest domain from the control domain you may use the `gdom -c destroy` operation. The guest domain destroy executes the following steps:

- remove all node datasets from guest domain, if any
- stop the guest domain if the shutdown flag is used
- remove guest domain from control domain
- remove guest domain from VDCF

```
$ gdom -c destroy name=g0068 shutdown
```

```
You are about to completely destroy the following guest domain
```

```
General Guest Domain Information for: g0068 (s11.3 testing)
```

Name	C	cState	rState	cPool	Act-Date	Act-Time
g0068		ACTIVE	ACTIVE (RUNNING)	sol11	2016-03-03	19:19:33

Active Build	OS Patch-Level	Enabled Build	#V
s11u3-sru5	11 3.5.0.6.0 (U3.SRU5)	s11u3-sru5	0

Active BootEnv	Previous BootEnv
s11.3.5.0.6.0	-

```
AutoBoot
true
```

Guest Domain				Control Domain usage			
Cores	Max-Cores	VCPUs	RAM/MB	CPU%	RAM%	Control Domain	Ldm Version
0	0	2	2048	3	6	s0024 ORCL,SPARC-T4-1	3.3.0.1.4

```
Disk Devices
```

Type	GUID	Dev-Type	State	Serial	Size/MB	ID
root	6001438012599B6200011000245A0000	MPXIO	ACTIVATED	PAPCRA076	15360	0

```
Network Interfaces
```

Type	GDom->CDom	Hostname	Interface	IP	Netmask
management->MNGT	g0068-mngt	g0068	vnet0->igb0	192.168.20.68	255.255.255.0
public->PUBL	g0068	g0068	IPMP: vnet1,vnet2	192.168.100.68	255.255.255.0
public->PUBL	link-based	link-based	PROBE: vnet1->igb1		
public->PUBL	link-based	link-based	PROBE: vnet2->igb3		

```
Are you sure (yes/no) ? [no]: yes
```

```
shutdown command being issued
```

```
removing guest domain
```

```
removing resources
```

```
Guest domain g0068 (s11.3 testing) destroyed successfully.
```

Or you can execute these steps manually:

```
% dataset -c remove/commit name=webcache
% gdom -c shutdown name=g0001
% gdom -c remove name=g0001
% gdom -c commit name=g0001 remove
```

## 6.8 Virtual pools (vPools) - Permission to manage Guest domains

Virtual pools (vPools) are used to add an additional user authorization layer for the vServer and Guest domain related commands (vserver, gdom, dataset, rcdm). 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 *chapter 9* for details.

## 6.9 Runtime States (CDom and GDom)

### 6.9.1 Overview

The Runtime States (rState) of Control and Guest Domains are 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.

### 6.9.2 Cronjob

The Runtime States (rState) are updated in the VDCF configuration repository using a cronjob.

See *chapter 4.8.2* for more details.

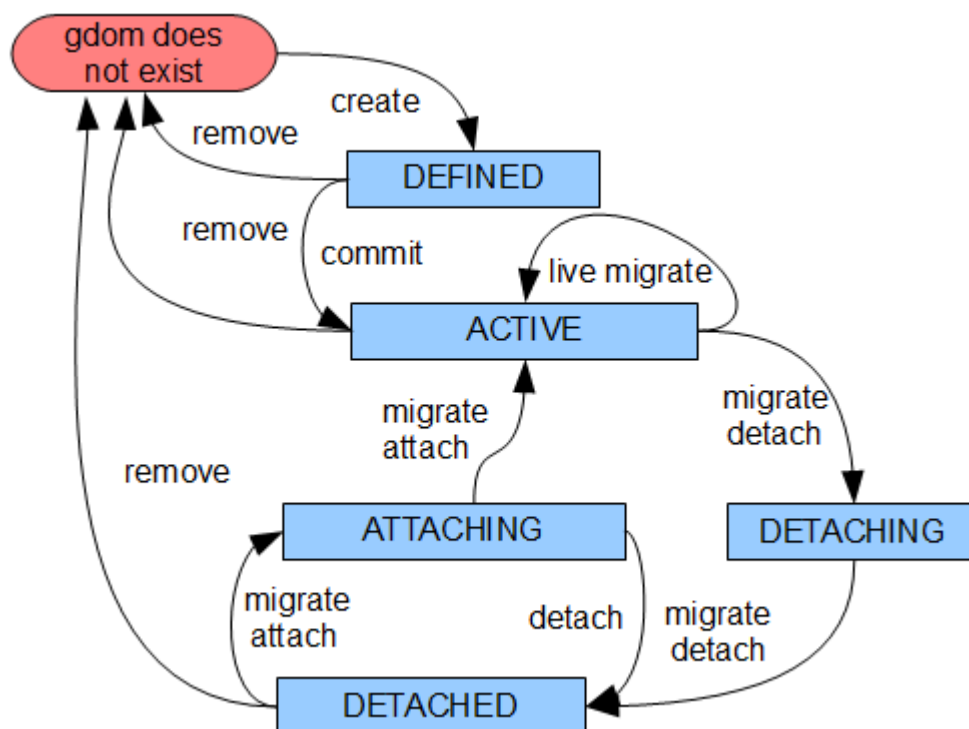
## 6.10 Configuration States (CDom and GDom)

### 6.10.1 Overview

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

See **chapter 7** for details about possible configuration states and the meaning of them.

### 6.10.2 GDom cState diagram



### 6.10.3 Supported GDom commands

Depending the cState 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 migrate (live)
- 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 migrate

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

- gdom -c attach
- gdom -c remove
- gdom -c migrate
- gdom -c modify

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

- gdom -c attach
- gdom -c detach
- gdom -c migrate

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

- gdom -c shutdown
- gdom -c boot
- gdom -c reboot
- gdom -c modify comment
- gdom -c modify readonly

## 7 Configuration States

### 7.1 Overview

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

### 7.2 Possible cState values

**Node cStates:** UNINSTALLED, INSTALLING, ACTIVE, INACTIVE, EVACUATING

**vServer cStates:** DEFINED, ACTIVATED, DETACHING, DETACHED, ATTACHING

**CDom states:** DEFINED, ACTIVE, PURGING

**GDom states:** DEFINED, ACTIVE, MODIFIED, DETACHING, DETACHED, ATTACHING, PURGING, READONLY

**Disk (LUN) states:** UNUSED, DEFINED, ACTIVATED, PURGING, REMOVING

**Dataset states:** DEFINED, ACTIVATED, DETACHED, PURGING

**Filesystem states:** DEFINED, ACTIVATED, PURGING

**Network states:** DEFINED, ACTIVATED, PURGING

### 7.3 cState 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 node exists in the VDCF repository and is installed and running.
INACTIVE	Indicates that the node is marked inactive manually or by the node evacuation feature.
EVACUATING	Indicates that the node is in an evacuation process now. All vServers are going to be detached from it.
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 GDom from a node.
ATTACHING	Indicates that VDCF is or was trying to attach the GDom to a node.
READONLY	Indicates that no changes are allowed for this GDom.

## 8 Security Management

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

### 8.1 Management Server RBAC

The VDCF base framework 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 the required VDCF commands.

#### Available RBAC profiles

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
VDCF serverconfig exec	serverconfig execution
VDCF pkg Module	package management
VDCF ldom Module	cdom and gdom management & operations
VDCF virtual Module	vServer management & operations
VDCF readonly	All read only modules (No update functions included)

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`.

Because the web server user (`webservd`) needs some rights when installing nodes, the VDCF framework adds the following mandatory entries to `/etc/user_attr`

```
webservd::::type=normal;profiles=VDCF Logger,VDCF install Module
```

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 ldom Module,VDCF vpool Manager,VDCF pkg 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 ldom Module,VDCF vpool Manager,VDCF pkg Module" vdcf
```

## 8.2 Remote Execution / SSH Environment

The 'Remote Execution' (`rexec`) feature allows start commands issued from the management server to be executed on the target node or vServer. The communication between the management server and the target systems runs over a SSH encrypted connection.

The SSH authentication is key-based using the private key of the root user stored in `/root/.ssh` on the management server. Ensure only root has read access to this directory. The public key of the root user is copied to the target systems during the installation process and is stored in `/export/home/vdcfexec/.ssh/AuthorizedKeys`.

On the target system RBAC entries for the `vdcfexec` user are created. The root user from the management server is allowed to execute the `rexec` command on the target system. The `rexec` command then transfers the command from the management server to the target system for execution.

## 9 Virtual pools (vPools) - Permission to manage Servers

### 9.0.1 Definition

Virtual pools (vPools) are used to implement user authorizations for vServer, Guest Domain and Physical Node related command (`vserver`, `rcadm`, `dataset`, `gdom`, `cdom`, `node`, `nodecfg`, `console`) actions. vPool authorization allow to restrict such command actions applied to the above servers.

This feature is disabled by default, which means everybody is allowed to manage all these servers. Set the configuration variable `VPOOL_ENABLED` to "TRUE" in your `customize.cfg` to activate it. For Physical Nodes to be activated, the variable `VPOOL_NODE` has to be set to "TRUE" in addition.

There is always one default vPool called "ALL". All servers belong at least to this default vPool. Assign users to this vPool, if they need to manage all servers.

You may add a server to as many vPools as you like. Only users assigned to a vPool may manipulate servers assigned to this vPool.

Users with assigned RBAC Profile "VDCF vpool Manager" are allowed to administer vPool definitions. I.e. assign servers and users to a vPool. Users with assigned RBAC Profile "VDCF vpool User" are only allowed to display vPool definitions.

## 9.0.2 Usage

USAGE: vpool [ -xhH ] -c <cmd>

Operations on Virtual pools (vpool):

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

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

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

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

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

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

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

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

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

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

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

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

The following format rules apply to the below listed parameters:

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

for more information about specific operations:  
vpool -H <operation>

## 10 Appendixes

### 10.1 Data File Overview

#### 10.1.1 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
<code>ai</code>	xml data and template files for Solaris Automated Installer (AI)

#### Configfiles

<code>customize.cfg</code>	Customer dependent customizing of predefined values
<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>system.cfg</code>	Additions to <code>/etc/system</code> for node installations <nodetype>_system.cfg is supported for GZONE, CDOM, GDOM
<code>ntp.cfg</code>	Additions to <code>/etc/inet/ntp.cfg</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
<code>disklocation.cfg</code>	Disk Location configuration
<code>datacenter.cfg</code>	Datacenter Location configuration

#### 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.

### 10.1.1.1 ntp.cfg

For NTP configuration template files may optionally be created in `/var/opt/jomasoft/vdcf/conf`

<code>ntp.cfg</code>	Default ntp.cfg Template
<code>ntp_10.cfg</code>	ntp.cfg Template for Solaris 10
<code>ntp_11.cfg</code>	ntp.cfg Template for Solaris 11
<code>&lt;node&gt;_ntp.cfg</code>	ntp.cfg Template for a specific Node <code>&lt;node&gt;</code>

The placeholders `{NTP_TIMESERVER_1}` and `{NTP_TIMESERVER_2}` will be replaced with the servers from the VDCF NTP Base Configuration.

#### Sample:

```
# Dont trust any strangers out there ...
restrict default ignore
# Trust these hosts for time
restrict {NTP_TIMESERVER_1} noquery
restrict {NTP_TIMESERVER_2} noquery
restrict 127.0.0.1
restrict -6 ::1
server {NTP_TIMESERVER_1} prefer
server {NTP_TIMESERVER_2}
multicastclient 224.0.1.1
driftfile /var/ntp/ntp.drift
statsdir /var/ntp/ntpstats/
filegen peerstats file peerstats type day enable
filegen loopstats file loopstats type day enable
filegen clockstats file clockstats type day enable
```

### 10.1.2 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`

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.

## 10.2 Customization of the 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
CONFIG_CONSOLE_USER	System Controller user
CONFIG_CONSOLE_POSTFIX	System Controller hostname postfix
CONFIG_CONSOLE_PRIVKEY	To use ssh keys to access the System Controllers
CONFIG_DEFAULTS	config values used as default when configuring new nodes with the command <code>nodecfg -c add</code> .
CONFIG_DISCOVER_SANDISK	Should nodecfg discover SAN LUNs (TRUE/FALSE)
CONFIG_NETMASK_DEFAULT	default netmasks used at <code>vserver -c addnet</code> , <code>nodecfg -c add</code>
CONFIG_IPMP_ALIASES	IPMP group name definitions
CONFIG_IPMP_LINK_BASED_ONLY	Enable if only link-based IPMP is used ( <code>nodecfg -c add</code> )
CONFIG_LINK_SPEED_AUTO_ONLY	Disable if you like to define speed at <code>nodecfg -c create</code>
DATASET_ALLOW_REMDISK	Enable or disable dataset <code>-c remdisk</code> for zpools (TRUE/FALSE)
DATASET_ATTACH_MIRROR_MAX_DISK	Define how many disk should be attached at once during a 'dataset -c attach_mirror/commit'
DATASET_ATTACH_MIRROR_MAX_DISK_MINSIZE_GB	Define a zpool size, when the 'conservative mirroring' should be used
DATASET_CHECK_LOCATIONS	Enable/Disable the Dataset Location Check (TRUE/FALSE)
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 ZVOL (VXVM: +DMP -MPXIO)
DISKS_ENABLE_MPXIO	Enable/Disable MPXIO
DISKS_ENABLE_SANBOOT	Enable/Disable Sanboot
FLASH_BOOT_METHOD	Flash install method (standard or wanboot)
FLASH_BOOTSERVER_IP	ip address where a jumpstart bootserver is running
FLASH_PUBLIC_WEBSERVER_URL	URL where the webserver is configured (public network). Format <code>http://&lt;ipaddr&gt;:&lt;port&gt;</code>
FLASH_WEBSERVER_URL	URL where the webserver is configured (default network). Format of <code>"http://&lt;ipaddr&gt;:&lt;port&gt;</code> .
FLASH_WEBSERVER_USER	User of the webserver
FLASHDIR	Directory for flash archives
FLASHPWD	default root password set after installing node or vserver
FS_VALID_OPTIONS	allowed filesystem options, when creating filesystems. See <code>vserver_adddfs(1M)</code> for details.
GDOM_DEFAULT_PROFILE	Default partitioning profile file used to setup Solaris 10 Gdoms (Filename located in the VDCF config directory)
GDOM_CANDIDATES_ATTR	Default is empty, allowed value is LOCATION
GDOM_CANDIDATES_LIST_FULL	By default only possible ('relevant') candidates are listed. Default is FALSE. Set to TRUE to list always all candidates
GDOM_CPU_ARGS	Define the allowed CPU arguments for gdom creation and modification allowed values: VCPU,CORES,MAXCORES
GDOM_ROOTDISK_MIRROR_REQUIRED	Defines if 2 root disks for a Host-Based Mirror is required for GDom (TRUE/FALSE)
GDOM_VIRTUAL_NET_VERBOSE	if TRUE, additional network details are displayed for a vServer/GDom Default is FALSE, allowed values are: TRUE, FALSE
HTTP_PROXY	HTTP Proxy in WGET(1) syntax
INSTALL_CONFIG_BASEDIR	VDCF server config directory
INSTALL_CONFIG_FILE	Directory where config files are stored on the management server
INSTALL_CONFIG_PKG	Directory where software packages are stored on the management server
INSTALL_CONFIG_SCRIPT	Directory where config scripts are stored on the management server
INSTALL_CONFIG_LOG	Installation logfiles on target
INSTALL_CONFIG_TMP	Temp vdcf installation directory on target
IPS_REPO_UPD_ASK	if YES, ipsadm prompt to confirm an update of the repository Default is YES, allowed values are: YES, NO



Variable name	Description
NOAUDIT_FOR_USER	Do not write audit log for these users (blank separated)
NODE_NET_ALIAS	Network type definition and default speed settings. Syntax: "MNGT:<speed>,PROBE:<speed>,PUBL:<speed>,BACK:<speed>"
NODE_NET_HOSTNAME_POSTFIX	Define DNS name postfix for network types (nodecfg -c add)
NODE_ROOT_DISK	Default Rootdisk layout
NODE_SCHEDULER_DEFAULT	Default scheduler that will be configured at node install
NODE_SHOW_ATTR	Default is 'empty', allowed values are: FSS, TS Used to activate additional attributes at node -c show Default is BUILD,GROUPPKG to display the build / Solaris11 group pkg if LOCATION is set, the node location is listed if BE is set, the BootEnvironment count (Active, Previous BootEnvironment) is displayed for Solaris 11 Nodes if TRUE, the 'force' flag is required to shutdown or reboot a node, if there are running vServers. allowed values are: TRUE, FLASE
NODE_SHUTDOWN_CHECK_VSERVER	Patch Spool directory
PATCH_SPOOL	Rexec check method (PING, NOPING)
REXEC_CHECK_METHOD	GID for the rexec group, 0 is default (auto id)
REXEC_GID	WebServer URL for rexec (public interface). Format http://<ipaddr>:<port>
REXEC_PUBLIC_WEBSERVER_URL	UID for the rexec user, 0 is default (auto id)
REXEC_UID	WebServer URL for rexec (default interface). Format http://<ipaddr>:<port>
REXEC_WEBSERVER_URL	If set to TRUE, the root user cannot execute VDCF commands set the retry count for the following zfs filesystem operations: vserver -c remfs/commit/mount/unmount and zfsadm -c set Default is 3
VDCF_DENY_ROOT_USER	default network stack used when adding networks to vServers on Solaris 10 nodes.
VIRTUAL_FS_RETRIES	default network stack used when adding networks to vServers on Solaris 11 nodes. allowed values are: SHARED, PRIVATE, EXCLUSIVE. see vserver_addnet(1M) for details.
VIRTUAL_NETSTACK_DEFAULT	Remove management network from vserver automatically after installation
VIRTUAL_NETSTACK_DEFAULT_S11	Sleep time before committing the auto remove of the management network
VIRTUAL_REMOVE_MNGT_NET	default server configuration group used when creating vservers. see vserver_create(1M) for details.
VIRTUAL_REMOVE_MNGT_NET_SLEEP	default vserver type: SPARSE, FULL
VIRTUAL_SGROUP_DEFAULT	if TRUE, vpool checks for vServers/GDoms are enabled, Default is FALSE
VIRTUAL_TYPE_DEFAULT	if TRUE, vpool checks for Physical Nodes are enabled Default is FALSE
VPOOL_ENABLED	If TRUE, dependency checking is enabled. Default is FALSE
VPOOL_NODE	WebServer URL for wanboot
VSERVER_CHECK_DEPEND	Rootdirectory of zones
WANBOOT_HTTP_URL	
ZONE_ROOTDIR	