

Oracle® VM Server for SPARC 2.1 Administration Guide

License Restrictions Warranty/Consequential Damages Disclaimer

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

Warranty Disclaimer

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Restricted Rights Notice

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

Hazardous Applications Notice

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group in the United States and other countries.

Third Party Content, Products, and Services Disclaimer

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf disposition de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, breveter, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Si ce logiciel, ou la documentation qui l'accompagne, est concédé sous licence au Gouvernement des Etats-Unis, ou à toute entité qui délivre la licence de ce logiciel ou l'utilise pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique :

U.S. GOVERNMENT RIGHTS. Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est pas conçu ni n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer des dommages corporels. Si vous utilisez ce logiciel ou matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour ce type d'applications.

Oracle et Java sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d'autres propriétaires qu'Oracle.

AMD, Opteron, le logo AMD et le logo AMD Opteron sont des marques ou des marques déposées d'Advanced Micro Devices. Intel et Intel Xeon sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. UNIX est une marque déposée concédée sous licence par X/Open Company, Ltd.

Contents

Preface	13
Part I Oracle VM Server for SPARC 2.1 Software	17
1 Overview of the Oracle VM Server for SPARC Software	19
Hypervisor and Logical Domains	19
Logical Domains Manager	22
Roles for Domains	22
Command-Line Interface	23
Virtual Input/Output	23
Resource Configuration	24
Persistent Configurations	24
Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool	25
Oracle VM Server for SPARC Configuration Assistant	25
Oracle VM Server for SPARC Management Information Base	25
2 Installing and Enabling Software	27
Installing Oracle VM Server for SPARC Software on a New System	28
Updating the Oracle Solaris OS	28
Upgrading the System Firmware	28
Downloading the Logical Domains Manager	31
Installing the Logical Domains Manager	31
Enabling the Logical Domains Manager Daemon	34
Upgrading a System Already Using Oracle VM Server for SPARC	34
Upgrading the Oracle Solaris OS	34
Upgrading the Logical Domains Manager and the System Firmware	36
Upgrading to Oracle VM Server for SPARC 2.1 Software	37

Factory Default Configuration and Disabling Logical Domains	38
▼ Remove All Guest Domains	38
▼ Remove All Logical Domains Configurations	38
▼ Restore the Factory Default Configuration	39
▼ Disable the Logical Domains Manager	39
▼ Removing the Logical Domains Manager	39
▼ Restore the Factory Default Configuration From the Service Processor	40
3 Security	41
Logical Domains Manager Authorization	41
Creating Authorizations and Profiles and Assigning Roles to User Accounts	42
Managing User Authorizations	43
Managing User Profiles	44
Assigning Roles to Users	45
Configuring RBAC for Guest Console Access	46
▼ Add an Authorization for a Domain Console	47
Enabling and Using Auditing	48
▼ Enable Auditing	48
▼ Disable Auditing	49
▼ Print Audit Output	49
▼ Rotate Audit Logs	49
4 Setting Up Services and the Control Domain	51
Output Messages	51
Creating Default Services	52
▼ Create Default Services	52
Initial Configuration of the Control Domain	53
▼ Set Up the Control Domain	53
Rebooting to Use Logical Domains	54
▼ Reboot	54
Enabling Networking Between the Control/Service Domain and Other Domains	55
▼ Configure the Virtual Switch as the Primary Interface	55
Enabling the Virtual Network Terminal Server Daemon	56
▼ Enable the Virtual Network Terminal Server Daemon	56

5	Setting Up Guest Domains	57
	Creating and Starting a Guest Domain	57
	▼ Create and Start a Guest Domain	57
	Installing Oracle Solaris OS on a Guest Domain	60
	▼ Install Oracle Solaris OS on a Guest Domain From a DVD	60
	▼ Install Oracle Solaris OS on a Guest Domain From a Oracle Solaris ISO File	61
	▼ Perform a JumpStart Operation on a Guest Domain	63
6	Setting Up I/O Domains	65
	I/O Domain Overview	65
	Assigning PCIe Buses	66
	▼ Create an I/O Domain by Assigning a PCIe Bus	67
	Assigning PCIe Endpoint Devices	70
	Direct I/O Hardware and Software Requirements	72
	Direct I/O Limitations	72
	Planning PCIe Endpoint Device Configuration	73
	Rebooting the primary Domain	74
	Making PCIe Hardware Changes	75
	▼ Create an I/O Domain by Assigning a PCIe Endpoint Device	76
7	Using Virtual Disks	81
	Introduction to Virtual Disks	81
	Managing Virtual Disks	82
	▼ Add a Virtual Disk	82
	▼ Export a Virtual Disk Back End Multiple Times	83
	▼ Change Virtual Disk Options	83
	▼ Change the Timeout Option	84
	▼ Remove a Virtual Disk	84
	Virtual Disk Identifier and Device Name	84
	Virtual Disk Appearance	85
	Full Disk	85
	Single-Slice Disk	85
	Virtual Disk Back End Options	86
	Read-only (ro) Option	86
	Exclusive (excl) Option	86

Slice (sLice) Option	87
Virtual Disk Back End	88
Physical Disk or Disk LUN	88
▼ Export a Physical Disk as a Virtual Disk	88
Physical Disk Slice	89
▼ Export a Physical Disk Slice as a Virtual Disk	89
▼ Export Slice 2	90
File and Volume	90
Configuring Virtual Disk Multipathing	93
▼ Configure Virtual Disk Multipathing	94
CD, DVD and ISO Images	95
▼ Export a CD or DVD From the Service Domain to the Guest Domain	96
▼ Export an ISO Image From the primary Domain to Install a Guest Domain	97
Virtual Disk Timeout	99
Virtual Disk and SCSI	99
Virtual Disk and the format Command	100
Using ZFS With Virtual Disks	100
Configuring a ZFS Pool in a Service Domain	100
Storing Disk Images With ZFS	101
Creating a Snapshot of a Disk Image	102
Using Clone to Provision a New Domain	103
Using Volume Managers in a Logical Domains Environment	104
Using Virtual Disks on Top of Volume Managers	104
Using Volume Managers on Top of Virtual Disks	106
 8 Using Virtual Networks	 109
Introduction to a Virtual Network	109
Virtual Switch	110
Virtual Network Device	110
Inter-Vnet LDC Channels	112
Virtual Device Identifier and Network Interface Name	114
▼ Find Oracle Solaris OS Network Interface Name	115
Assigning MAC Addresses Automatically or Manually	116
Range of MAC Addresses Assigned to Logical Domains	116
Automatic Assignment Algorithm	117

Duplicate MAC Address Detection	117
Freed MAC Addresses	118
Using Network Adapters With Logical Domains	119
▼ Determine If a Network Adapter Is GLDv3-Compliant	119
Configuring Virtual Switch and Service Domain for NAT and Routing	119
▼ Set Up the Virtual Switch to Provide External Connectivity to Domains	120
Configuring IPMP in a Logical Domains Environment	121
Configuring Virtual Network Devices Into an IPMP Group in a Domain	121
Configuring and Using IPMP in the Service Domain	122
Using Link-Based IPMP in Logical Domains Virtual Networking	123
Configuring and Using IPMP in Releases Prior to Logical Domains 1.3	126
Using VLAN Tagging	127
Port VLAN ID (PVID)	128
VLAN ID (VID)	129
▼ Assign VLANs to a Virtual Switch and Virtual Network Device	129
▼ Install a Guest Domain When the Install Server Is in a VLAN	130
Using NIU Hybrid I/O	130
▼ Configure a Virtual Switch With an NIU Network Device	133
▼ Enable Hybrid Mode	134
▼ Disable Hybrid Mode	134
Using Link Aggregation With a Virtual Switch	134
Configuring Jumbo Frames	135
▼ Configure Virtual Network and Virtual Switch Devices to Use Jumbo Frames	135
Compatibility With Older (Jumbo-Unaware) Versions of the vnet and vsw Drivers	138
9 Migrating Domains	139
Introduction to Domain Migration	140
Overview of a Migration Operation	140
Software Compatibility	141
Security for Migration Operations	141
Migrating a Domain	142
Performing a Dry Run	142
Performing Non-Interactive Migrations	142
Migrating an Active Domain	143
Migration Requirements for CPUs	143

Migration Requirements for Memory	144
Migration Requirements for Physical I/O Devices	145
Migration Requirements for Virtual I/O Devices	145
Migration Requirements for NIU Hybrid I/O	146
Migration Requirements Cryptographic Units	146
Delayed Reconfiguration in an Active Domain	146
Migrating While an Active Domain Is in Elastic Mode	147
Operations on Other Domains	147
Migrating a Domain That is Running in OpenBoot or in the Kernel Debugger	147
Migrating Bound or Inactive Domains	147
Migration Requirements for CPUs	148
Migration Requirements for Virtual I/O Devices	148
Migration Requirements for PCIe Endpoint Devices	148
Monitoring a Migration in Progress	148
Canceling a Migration in Progress	149
Recovering From a Failed Migration	149
Migration Examples	150
10 Managing Resources	153
Resource Reconfiguration	153
Dynamic Reconfiguration	153
Delayed Reconfiguration	154
Resource Allocation	155
CPU Allocation	155
Enabling the Whole-Core Constraint	155
Disabling the Whole-Core Constraint	157
Allocating CPUs to the Control Domain	157
Interactions Between the Whole-Core Constraint and Other Domain Features	158
Using Memory Dynamic Reconfiguration	159
Adding Memory	160
Removing Memory	160
Partial Memory DR Requests	160
Memory Reconfiguration of the Control Domain	161
Dynamic and Delayed Reconfiguration	161
Memory Alignment	161

Memory DR Examples	163
Using Power Management	166
Listing Power-Managed CPU Strands and Virtual CPUs	168
Using Dynamic Resource Management	171
Listing Domain Resources	174
Machine-Readable Output	174
Flag Definitions	174
Utilization Statistic Definition	175
Viewing Various Lists	176
Listing Constraints	179
11 Managing Configurations	181
Saving Domain Configurations for Future Rebuilding	181
▼ Save Domain Configurations	181
▼ Restore a Domain Configuration From an XML File (<code>ldm add-domain</code>)	182
▼ Restore a Domain Configuration From an XML File (<code>ldm init-system</code>)	183
Managing Logical Domains Configurations	184
▼ Modify the Autorecovery Policy	186
12 Performing Other Administration Tasks	189
Entering Names in the CLI	189
File Names (<i>file</i>) and Variable Names (<i>var-name</i>)	189
Virtual Disk Server <i>backend</i> and Virtual Switch Device Names	190
Configuration Name (<i>config-name</i>)	190
All Other Names	190
Connecting to a Guest Console Over a Network	190
Using Console Groups	191
▼ Combine Multiple Consoles Into One Group	191
Stopping a Heavily Loaded Domain Can Time Out	192
Operating the Oracle Solaris OS With Oracle VM Server for SPARC	192
OpenBoot Firmware Not Available After Oracle Solaris OS Has Started	192
Powercycling a Server	192
Do Not Use the <code>ps radm(1M)</code> Command on Active CPUs in a Power-Managed Domain	193
Result of Oracle Solaris OS Breaks	193
Results From Halting or Rebooting the Control Domain	193

Using Logical Domains With the Service Processor	194
▼ Reset the Domain Configuration to the Default or Another Configuration	194
Configuring Domain Dependencies	195
Domain Dependency Examples	196
Dependency Cycles	197
Determining Where Errors Occur by Mapping CPU and Memory Addresses	198
CPU Mapping	199
Memory Mapping	199
Examples of CPU and Memory Mapping	199
Using Universally Unique Identifiers	201
Virtual Domain Information Command and API	201
 Part II Optional Oracle VM Server for SPARC Software	203
 13 Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool	205
Oracle VM Server for SPARC P2V Tool Overview	205
Collection Phase	206
Preparation Phase	206
Conversion Phase	207
Back-End Devices	207
Installing the Oracle VM Server for SPARC P2V Tool	208
Prerequisites	208
Limitations	209
▼ Install the Oracle VM Server for SPARC P2V Tool	209
Using the ldmp2v Command	210
 14 Oracle VM Server for SPARC Configuration Assistant	217
Using the Configuration Assistant (ldmconfig)	217
Installing the Configuration Assistant	217
ldmconfig Features	218
 15 Using the Oracle VM Server for SPARC Management Information Base Software	221
Oracle VM Server for SPARC Management Information Base Overview	222
Software Components	222

System Management Agent	223
Logical Domains Manager and the Oracle VM Server for SPARC MIB	224
Oracle VM Server for SPARC MIB Object Tree	224
Installing and Configuring the Oracle VM Server for SPARC MIB Software	225
Installing and Configuring the Oracle VM Server for SPARC MIB Software (Task Map)	225
Managing Security	227
▼ Create the Initial snmpv3 User	228
Monitoring Domains	228
Setting Environment Variables	229
Querying the Oracle VM Server for SPARC MIB	229
Retrieving Oracle VM Server for SPARC MIB Information	231
Using SNMP Traps	249
Using Oracle VM Server for SPARC MIB Module Traps	249
Oracle VM Server for SPARC MIB Trap Descriptions	250
Starting and Stopping Domains	255
Starting and Stopping a Domain	256
 16 Logical Domains Manager Discovery	 259
Discovering Systems Running the Logical Domains Manager	259
Multicast Communication	259
Message Format	259
▼ Discover Logical Domains Managers Running on Your Subnet	260
 17 Using the XML Interface With the Logical Domains Manager	 263
XML Transport	263
XMPP Server	264
Local Connections	264
XML Protocol	264
Request and Response Messages	265
Event Messages	268
Registration and Unregistration	269
The <LDM_event> Messages	269
Event Types	270
Logical Domains Manager Actions	273
Logical Domains Manager Resources and Properties	274

Domain Information (ldom_info) Resource	274
CPU (cpu) Resource	275
MAU (mau) Resource	276
Memory (memory) Resource	276
Virtual Disk Server (vds) Resource	277
Virtual Disk Server Volume (vds_volume) Resource	277
Disk (disk) Resource	278
Virtual Switch (vsw) Resource	279
Network (network) Resource	280
Virtual Console Concentrator (vcc) Resource	281
Variable (var) Resource	281
Physical I/O Device (physio_device) Resource	282
SP Configuration (spconfig) Resource	282
DRM Policy Configuration (policy) Resource	283
Virtual Data Plane Channel Service (vdpcs) Resource	284
Virtual Data Plane Channel Client (vdpccl) Resource	284
Console (console) Resource	285
Domain Migration	285
XML Schemas	286
 Glossary	 289
 Index	 299

Preface

The *Oracle VM Server for SPARC 2.1 Administration Guide* provides detailed information and procedures that describe the overview, security considerations, installation, configuration, modification, and execution of common tasks for the Oracle VM Server for SPARC 2.1 software on supported servers, blades, and server modules. See “Supported Platforms” in *Oracle VM Server for SPARC 2.1 Release Notes*.

This guide is intended for the system administrators on these servers who have a working knowledge of UNIX systems and the Oracle Solaris operating system (Oracle Solaris OS).

Related Documentation

The following table shows the documentation that is available for the Oracle VM Server for SPARC 2.1 release. These documents are available in HTML and PDF formats unless indicated.

TABLE P-1 Related Documentation

Application	Title	Part Number
Oracle VM Server for SPARC 2.1 Software	Oracle VM Server for SPARC 2.1 Administration Guide	821-2854
	Oracle VM Server for SPARC 2.1 Reference Manual	821-2855
	Oracle VM Server for SPARC 2.1 Release Notes	821-2856
	Oracle Solaris 10 Reference Manual Documentation	
	<ul style="list-style-type: none">▪ drd(1M) man page▪ vntsd(1M) man page	
Oracle Solaris OS: Installation and Configuration	Oracle Solaris 10 9/10 Release and Installation Documentation	N/A

You can find documentation that relates to your server, software, or the Oracle Solaris OS at <http://www.oracle.com/technetwork/indexes/documentation/index.html>. Use the Search box to find the documents and the information that you need.

Documentation and Support

See the following web sites for additional resources:

- Documentation (<http://www.oracle.com/technetwork/indexes/documentation/index.html>)
- Support (<http://www.oracle.com/us/support/systems/index.html>)

Oracle Software Resources

Oracle Technology Network (<http://www.oracle.com/technetwork/index.html>) offers a range of resources related to Oracle software:

- Discuss technical problems and solutions on the Discussion Forums (<http://forums.oracle.com>).
- Get hands-on step-by-step tutorials with Oracle By Example (<http://www.oracle.com/technetwork/tutorials/index.html>).

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P-2 Typographic Conventions

Typeface	Meaning	Example
AaBbCc123	The names of commands, files, and directories, and onscreen computer output	Edit your <code>.login</code> file. Use <code>ls -a</code> to list all files. <code>machine_name%</code> you have mail.
AaBbCc123	What you type, contrasted with onscreen computer output	<code>machine_name%</code> su Password:
<i>aabbcc123</i>	Placeholder: replace with a real name or value	The command to remove a file is <i>rm filename</i> .
<i>AaBbCc123</i>	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the <i>User's Guide</i> . A <i>cache</i> is a copy that is stored locally. Do <i>not</i> save the file. Note: Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for shells that are included in the Oracle Solaris OS. Note that the default system prompt that is displayed in command examples varies, depending on the Oracle Solaris release.

TABLE P-3 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	\$
Bash shell, Korn shell, and Bourne shell for superuser	#
C shell	machine_name%
C shell for superuser	machine_name#

PART I

Oracle VM Server for SPARC 2.1 Software

This part introduces the Oracle VM Server for SPARC 2.1 software, which provides highly efficient, enterprise-class virtualization capabilities for Oracle's SPARC T-Series servers.

Overview of the Oracle VM Server for SPARC Software

This chapter provides an overview of the Oracle VM Server for SPARC software.

The Oracle VM Server for SPARC software depends on particular Oracle Solaris OS versions, required software patches, and particular versions of system firmware. For more information, see [“Required and Recommended Oracle Solaris OS”](#) in *Oracle VM Server for SPARC 2.1 Release Notes*.

Oracle VM Server for SPARC provides highly efficient, enterprise-class virtualization capabilities for Oracle's SPARC T-Series servers. Using the Oracle VM Server for SPARC software, you can create up to 128 virtual servers, called logical domains, on a single system. This kind of configuration enables you to take advantage of the massive thread scale offered by SPARC T-Series servers and the Oracle Solaris OS.

This chapter covers the following topics:

- [“Hypervisor and Logical Domains”](#) on page 19
- [“Logical Domains Manager”](#) on page 22
- [“Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool”](#) on page 25
- [“Oracle VM Server for SPARC Configuration Assistant”](#) on page 25
- [“Oracle VM Server for SPARC Management Information Base”](#) on page 25

Hypervisor and Logical Domains

This section provides an overview of the SPARC hypervisor, which supports logical domains.

The SPARC *hypervisor* is a small firmware layer that provides a stable virtualized machine architecture to which an operating system can be written. Oracle's Sun servers that use the hypervisor provide hardware features to support the hypervisor's control over a logical operating system's activities.

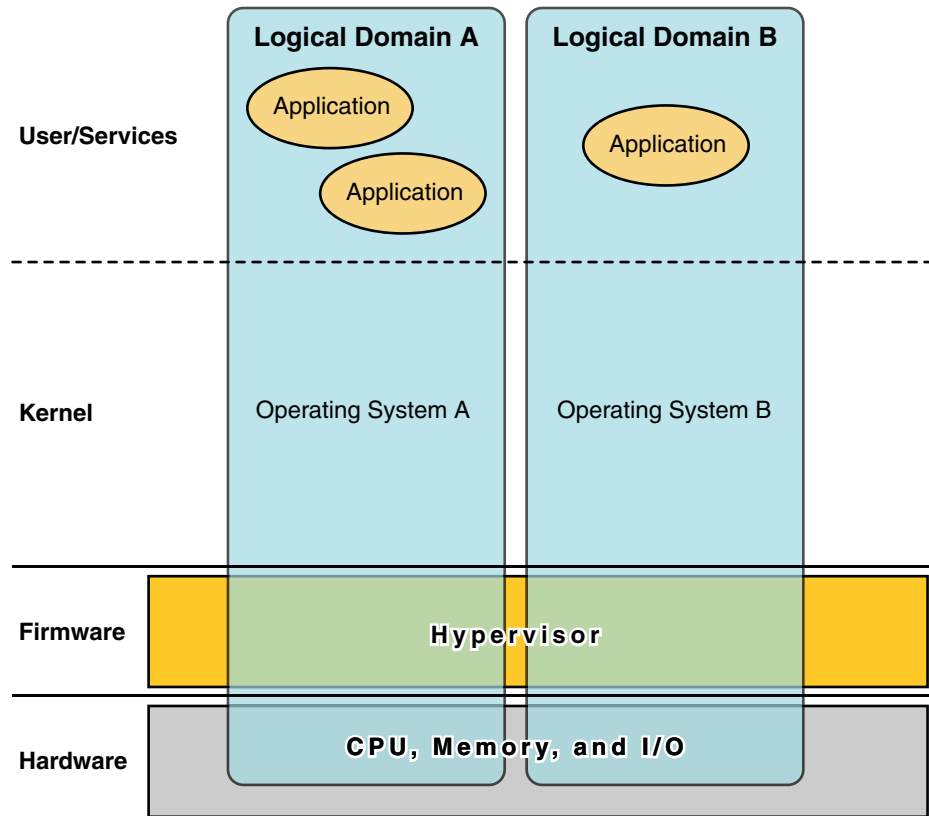
A *logical domain* is a virtual machine comprised of a discrete logical grouping of resources. A logical domain has its own operating system and identity within a single computer system. Each

logical domain can be created, destroyed, reconfigured, and rebooted independently, without requiring you to powercycle the server. You can run a variety of applications software in different logical domains and keep them independent for performance and security purposes.

Each logical domain is only permitted to observe and interact with those server resources that are made available to it by the hypervisor. The Logical Domains Manager enables you to specify what the hypervisor should do through the control domain. Thus, the hypervisor enforces the partitioning of the server's resources and provides limited subsets to multiple operating system environments. This partitioning and provisioning is the fundamental mechanism for creating logical domains. The following diagram shows the hypervisor supporting two logical domains. It also shows the following layers that make up the Logical Domains functionality:

- User/services, or applications
- Kernel, or operating systems
- Firmware, or hypervisor
- Hardware, including CPU, memory, and I/O

FIGURE 1-1 Hypervisor Supporting Two Domains



The number and capabilities of each logical domain that a specific SPARC hypervisor supports are server-dependent features. The hypervisor can allocate subsets of the overall CPU, memory, and I/O resources of a server to a given logical domain. This enables support of multiple operating systems simultaneously, each within its own logical domain. Resources can be rearranged between separate logical domains with an arbitrary granularity. For example, CPUs are assignable to a logical domain with the granularity of a CPU thread.

Each logical domain can be managed as an entirely independent machine with its own resources, such as:

- Kernel, patches, and tuning parameters
- User accounts and administrators
- Disks
- Network interfaces, MAC addresses, and IP addresses

Each logical domain can be stopped, started, and rebooted independently of each other without requiring a powercycle of the server.

The hypervisor software is responsible for maintaining the separation between logical domains. The hypervisor software also provides logical domain channels (LDCs) that enable logical domains to communicate with each other. LDCs enable domains to provide services to each other, such as networking or disk services.

The service processor (SP), also known as the system controller (SC), monitors and runs the physical machine, but it does not manage the logical domains. The Logical Domains Manager manages the logical domains.

Logical Domains Manager

The Logical Domains Manager is used to create and manage logical domains, as well as map logical domains to physical resources. Only one Logical Domains Manager can run on a server.

Roles for Domains

All logical domains are the same and can be distinguished from one another based on the roles that you specify for them. The following are the roles that logical domains can perform:

- **Control domain.** The Logical Domains Manager runs in this domain, which enables you to create and manage other logical domains, and to allocate virtual resources to other domains. You can have only one control domain per server. The control domain is the first domain created when you install the Oracle VM Server for SPARC software. The control domain is named `primary`.
- **Service domain.** A service domain provides virtual device services to other domains, such as a virtual switch, a virtual console concentrator, and a virtual disk server. Any domain can be configured as a service domain.
- **I/O domain.** An I/O domain has direct access to a physical I/O device, such as a network card in a PCI EXPRESS (PCIe) controller. An I/O domain can own a PCIe root complex, or it can own a PCIe slot or on-board PCIe device by using the direct I/O (DIO) feature. See [“Assigning PCIe Endpoint Devices” on page 70](#).
An I/O domain can share physical I/O devices with other domains in the form of virtual devices when the I/O domain is also used as a service domain.
- **Root domain.** A root domain has a PCIe root complex assigned to it. This domain owns the PCIe fabric and provides all fabric-related services, such as fabric error handling. A root domain is also an I/O domain, as it owns and has direct access to physical I/O devices.

The number of root domains that you can have depends on your platform architecture. For example, if you are using a Sun SPARC Enterprise T5440 server, you can have up to four root domains.

- **Guest domain.** A guest domain is a non-I/O domain that consumes virtual device services that are provided by one or more service domains. A guest domain does not have any physical I/O devices, but only has virtual I/O devices, such as virtual disks and virtual network interfaces.

You can install the Logical Domains Manager on an existing system that is not already configured with Logical Domains. In this case, the current instance of the OS becomes the control domain. Also, the system is configured with only one domain, the control domain. After configuring the control domain, you can balance the load of applications across other domains to make the most efficient use of the entire system. You do this by adding domains and moving those applications from the control domain to the new domains.

Command-Line Interface

The Logical Domains Manager uses a command-line interface (CLI) to create and configure logical domains. The CLI is a single command, `ldm`, that has multiple subcommands. See the [ldm\(1M\)](#) man page.

The Logical Domains Manager daemon, `ldmd`, must be running to use the Logical Domains Manager CLI.

Virtual Input/Output

In a Logical Domains environment, you can provision up to 128 domains on an UltraSPARC T2 Plus processor system and a SPARC T3 processor system. These systems have a limited number of I/O buses and physical I/O slots. As a result, you cannot provide exclusive access to a physical disk and network devices to all domains on these systems. You can assign a PCIe bus or endpoint device to a domain to provide it with access to a physical device. Note that this solution is insufficient to provide all domains with exclusive device access. See [Chapter 6, “Setting Up I/O Domains.”](#) This limitation on the number of physical I/O devices that can be directly accessed is addressed by implementing a virtualized I/O model.

Any logical domains that have no physical I/O access are configured with virtual I/O devices that communicate with a service domain. The service domain runs a virtual device service to provide access to a physical device or to its functions. In this client-server model, virtual I/O devices either communicate with each other or with a service counterpart through interdomain communication channels called logical domain channels (LDCs). The virtualized I/O functionality includes support for virtual networking, storage, and consoles.

Virtual Network

Logical Domains uses the virtual network device and virtual network switch device to implement virtual networking. The virtual network (`vnet`) device emulates an Ethernet device and communicates with other `vnet` devices in the system by using a point-to-point channel.

The virtual switch (vsw) device primarily functions as a multiplexor of all the virtual network's incoming and outgoing packets. The vsw device interfaces directly with a physical network adapter on a service domain, and sends and receives packets on behalf of a virtual network. The vsw device also functions as a simple layer-2 switch and switches packets between the vnet devices connected to it within the system.

Virtual Storage

The virtual storage infrastructure uses a client-server model to enable logical domains to access block-level storage that is not directly assigned to them. The model uses the following components:

- Virtual disk client (vdc) that exports a block device interface
- Virtual disk service (vds) that processes disk requests on behalf of the virtual disk client and submits them to the back-end storage that resides on the service domain

Although the virtual disks appear as regular disks on the client domain, most disk operations are forwarded to the virtual disk service and processed on the service domain.

Virtual Console

In a Logical Domains environment, console I/O from the primary domain is directed to the service processor. The console I/O from all other domains is redirected to the service domain that is running the virtual console concentrator (vcc). The domain that runs the vcc is typically the primary domain. The virtual console concentrator service functions as a concentrator for all domains' console traffic, and interfaces with the virtual network terminal server daemon (vntsd) to provide access to each console through a UNIX socket.

Resource Configuration

A system that runs the Oracle VM Server for SPARC software can configure resources, such as virtual CPUs, virtual I/O devices, cryptographic units, and memory. Some resources can be configured dynamically on a running domain, while others must be configured on a stopped domain. If a resource cannot be dynamically configured on the control domain, you must first initiate a delayed reconfiguration. The delayed reconfiguration postpones the configuration activities until after the control domain has been rebooted. For more information, see [“Resource Reconfiguration” on page 153](#).

Persistent Configurations

You can use the `ldm` command to store the current configuration of a logical domain on the service processor. You can add a configuration, specify a configuration to be used, remove a configuration, and list the configurations. See the `ldm(1M)` man page. You can also specify a configuration to boot from the SP. See [“Using Logical Domains With the Service Processor” on page 194](#).

For information about managing configurations, see [“Managing Logical Domains Configurations” on page 184](#).

Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool

The Oracle VM Server for SPARC Physical-to-Virtual (P2V) Conversion Tool automatically converts an existing physical system to a virtual system that runs in a logical domain on a chip multithreading (CMT) system. The source system can be any of the following:

- Any sun4u SPARC system that runs at least the Solaris 8 Operating System
- Any sun4v system that runs the Oracle Solaris 10 OS, but does not run the Oracle VM Server for SPARC software

For information about the tool and about installing it, see [Chapter 13, “Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool”](#). For information about the `ldmp2v` command, see the `ldmp2v(1M)` man page.

Oracle VM Server for SPARC Configuration Assistant

The Oracle VM Server for SPARC Configuration Assistant leads you through the configuration of a logical domain by setting basic properties. It can be used to configure any system where the Oracle VM Server for SPARC software is installed but not already configured.

After gathering the configuration data, the Configuration Assistant creates a configuration that is suitable for booting as a logical domain. You can also use the default values selected by the Configuration Assistant to create a usable system configuration.

The Configuration Assistant is a terminal-based tool.

For more information, see [Chapter 14, “Oracle VM Server for SPARC Configuration Assistant,”](#) and the `ldmconfig(1M)` man page.

Oracle VM Server for SPARC Management Information Base

The Oracle VM Server for SPARC Management Information Base (MIB) enables third-party system management applications to perform remote monitoring of domains, and to start and stop logical domains (domains) by using the Simple Network Management Protocol (SNMP). For more information, see [Chapter 15, “Using the Oracle VM Server for SPARC Management Information Base Software.”](#)

Installing and Enabling Software

This chapter describes how to install or upgrade the different software components required to enable the Oracle VM Server for SPARC 2.1 software. Using the Oracle VM Server for SPARC software requires the following components:

- Supported platform, refer to [“Supported Platforms” in Oracle VM Server for SPARC 2.1 Release Notes](#) for a list of supported platforms.
- Control domain running an operating system at least equivalent to the Oracle Solaris 10 9/10 OS with any patches recommended in [“Required Software and Patches” in Oracle VM Server for SPARC 2.1 Release Notes](#). See [“Upgrading the Oracle Solaris OS” on page 34](#).
- At least system firmware version 7.4.0 for your Sun UltraSPARC T2 or T2 Plus platform and version 8.1.0 for your SPARC T3 platform. See [“Upgrading the System Firmware” on page 28](#).
- Oracle VM Server for SPARC 2.1 software installed and enabled on the control domain. See [“Installing the Logical Domains Manager” on page 31](#).
- (Optional) the Oracle VM Server for SPARC Management Information Base (MIB) software package. See [Chapter 15, “Using the Oracle VM Server for SPARC Management Information Base Software,”](#) for more information about using the Oracle VM Server for SPARC MIB.

The Oracle Solaris OS and the system firmware must be installed or upgraded on your server before you install or upgrade the Logical Domains Manager. If your system is already using Oracle VM Server for SPARC software, see [“Upgrading a System Already Using Oracle VM Server for SPARC” on page 34](#). Otherwise, see [“Installing Oracle VM Server for SPARC Software on a New System” on page 28](#).

This chapter covers the following topics:

- [“Installing Oracle VM Server for SPARC Software on a New System” on page 28](#)
- [“Upgrading a System Already Using Oracle VM Server for SPARC” on page 34](#)
- [“Factory Default Configuration and Disabling Logical Domains” on page 38](#)

Note – The Solaris Security Toolkit (SST) software is no longer packaged with the Oracle VM Server for SPARC software. If you would like to use the most recent version of the SST software, see the [Oracle VM Server for SPARC 2.1 Release Notes](#).

Installing Oracle VM Server for SPARC Software on a New System

Oracle's Sun platforms that support the Oracle VM Server for SPARC software come preinstalled with the Oracle Solaris 10 OS. Initially, the platform appears as a single system hosting only one operating system. After the Oracle Solaris OS, system firmware, and Logical Domains Manager have been installed, the original system and instance of the Oracle Solaris OS become the control domain. That first domain of the platform is named `primary`, and you cannot change that name or destroy that domain. From there, the platform can be reconfigured to have multiple domains hosting different instances of the Oracle Solaris OS.

Updating the Oracle Solaris OS

On a brand new system, you might want to reinstall the factory-installed OS to conform to your installation policy. See “[Required and Recommended Oracle Solaris OS](#)” in [Oracle VM Server for SPARC 2.1 Release Notes](#). For complete Oracle Solaris OS installation instructions, see the [Oracle Solaris 10 9/10 Release and Installation documentation \(http://download.oracle.com/docs/cd/E18752_01/index.html\)](http://download.oracle.com/docs/cd/E18752_01/index.html). You can tailor the installation to the requirements of your system.

If your system is already installed with the Oracle Solaris OS, you must upgrade it to the OS version that is associated with the Oracle VM Server for SPARC 2.1 software. See “[Required Software and Patches](#)” in [Oracle VM Server for SPARC 2.1 Release Notes](#). For complete Oracle Solaris OS upgrade instructions, see the [Oracle Solaris 10 9/10 Release and Installation documentation \(http://download.oracle.com/docs/cd/E18752_01/index.html\)](http://download.oracle.com/docs/cd/E18752_01/index.html).

Upgrading the System Firmware

The following tasks describe how to upgrade system firmware by using the Integrated Lights Out Manager (ILOM) software.

For information about upgrading the system firmware by using the ILOM software, see “[Update the Firmware](#)” in [Sun SPARC Enterprise T5120 and T5220 Servers Topic Set](#) and “[Updating ILOM Firmware](#)” in the [Sun Integrated Lights Out Manager \(ILOM\) 3.0 CLI Procedures Guide](#).

▼ Upgrade System Firmware

You can find system firmware for your platform at <http://www.oracle.com/technetwork/systems/patches/firmware/index.html>.

For information about the required system firmware for the supported servers, see “Required and Recommended System Firmware Patches” in *Oracle VM Server for SPARC 2.1 Release Notes*.

To upgrade the system firmware from the control domain, refer to your system firmware release notes.

Refer to the administration guides or product notes for the supported servers for more information about installing and upgrading the system firmware for these servers.

You can also use the ILOM web interface to upgrade system firmware, see “Updating ILOM Firmware” in the *Sun Integrated Lights Out Manager (ILOM) 3.0 Web Interface Procedures Guide*.

1 Download the system firmware image to another system that is running the `tftp` service.

a. Ensure that the `tftp` service is online on the server.

```
# svcadm tftp/udp6
STATE      STIME      FMRI
online     Mar_26     svc:/network/tftp/udp6:default
```

b. Enable the `tftp` service if it is not in the online state.

```
# svcadm enable tftp/udp6
```

c. Download the system firmware image to the `/tftpboot` directory.

2 Ensure that the ILOM service processor network management port is configured.

This configuration is required to access the new flash image over the network. See “To Configure the Service Processor Network Management Port” in *Sun SPARC Enterprise T5120 and T5220 Servers Topic Set* and “Updating ILOM Firmware” in the *Sun Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide*.

3 Open an SSH session to connect to the service processor.

```
$ ssh root@system-name
...
Are you sure you want to continue connecting (yes/no)? yes
...
Password: password
...
->
```

4 Verify that the host is powered off.**a. Type the following command:**

```
-> show /SYS power_state
```

b. If the host is not powered off, type the following command:

```
-> stop /SYS
```

5 Verify that the `keyswitch_state` parameter is set to `normal`.**a. Type the following command:**

```
-> show /SYS keyswitch_state
```

b. If the value is other than `normal`, set it by using the following command:

```
-> set /SYS keyswitch_state=normal
```

6 Upgrade the service processor flash image and the host firmware.

```
-> load -source \
tftp://IP-addr/pathname/Sun_System_Firmware-x_x_x_build_nn-server-name.pkg
```

The `-source` option specifies the IP address and full path name (URL) to the system firmware flash image.

- *IP-addr* is the IP address of a `tftp` server on the network that can access the flash image.
- *pathname* is the full path name to the flash image on the `tftp` server.
- *x_x_x* is the version number of the system firmware
- *nn* is the build number that applies to this release.
- *server-name* is the name of your server.

For example, for the SPARC Enterprise T5440 server, *server-name* is `SPARC_Enterprise_T5440`.

For example, the `-source`

`tftp://192.168.1.1/Sun_System_Firmware-7_3_0-SPARC_Enterprise_T5440.pkg` option points to the `/tftpboot/Sun_System_Firmware-7_3_0-SPARC_Enterprise_T5440.pkg` file on the server with the `192.168.1.1` IP address.

After the flash image has been upgraded, the system automatically resets.

The service processor resets, runs diagnostics, and returns to the login prompt (on the serial console).

Downloading the Logical Domains Manager

▼ Download the Software

- 1 Download the zip file (`OVM_Server_SPARC-2_1.zip`).

You can find the software at <http://www.oracle.com/virtualization/index.html>.

- 2 Unzip the zip file.

```
$ unzip OVM_Server_SPARC-2_1.zip
```

See “Location of Oracle VM Server for SPARC 2.1 Software” in *Oracle VM Server for SPARC 2.1 Release Notes* for details about the structure of the file and what it includes.

Installing the Logical Domains Manager

There are three methods of installing the Logical Domains Manager software:

- Using the installation script to install the packages and patches. This automatically installs the Logical Domains Manager software. See “Installing the Logical Domains Manager Software Automatically” on page 31.
- Using JumpStart to install the packages as part of an Oracle Solaris network installation. See “Using JumpStart to Install the Oracle VM Server for SPARC 2.1 Software” on page 32.
- Installing the package manually. See “Installing the Logical Domains Manager Software Manually” on page 33.

Note – Remember that you must manually install the Oracle VM Server for SPARC MIB software package after you install the Oracle VM Server for SPARC packages. It is not automatically installed with the other packages. See [Chapter 15, “Using the Oracle VM Server for SPARC Management Information Base Software,”](#) for more information about installing and using the Oracle VM Server for SPARC MIB.

Installing the Logical Domains Manager Software Automatically

If you use the `install-ldm` installation script, you have several choices to specify how you want the script to run. Each choice is described in the procedures that follow.

- **Using the `install-ldm` script with no options does the following automatically:**
 - Checks that the Oracle Solaris OS release is Oracle Solaris 10 9/10 OS at a minimum
 - Verifies that the package subdirectories `SUNWldm/` and `SUNWldmp2v/` are present
 - Verifies that the prerequisite Logical Domains driver packages, `SUNWldomr` and `SUNWldomu`, are present

- Verifies that the `SUNWldm` and `SUNWldmp2v` packages have not been installed
- Installs the Oracle VM Server for SPARC 2.1 software
- Verifies that all packages are installed
- If the SST (`SUNWjass`) is already installed, you are prompted to harden the Oracle Solaris OS on the control domain.
- Determine whether to use the Oracle VM Server for SPARC Configuration Assistant (`ldmconfig`) to perform the installation.
- **Using the `install-ldm` script with the `-c` option automatically runs the Oracle VM Server for SPARC Configuration Assistant after the software is installed.**
- **Using the `install-ldm` script with the `-s` option skips the running of the Oracle VM Server for SPARC Configuration Assistant.**
- **Using the `install-ldm` script and the following options with the SST software enables you to do the following:**
 - `install-ldm -d`. Allows you to specify a SST driver other than a driver ending with `-secure.driver`. This option automatically performs all the functions listed in the preceding choice and hardens the Oracle Solaris OS on the control domain with the SST customized driver that you specify; for example, the `server-secure-myname.driver`.
 - `install-ldm -d none`. Specifies that you do *not* want to harden the Oracle Solaris OS running on your control domain by using the SST. This option automatically performs all the functions except hardening listed in the preceding choices. Bypassing the use of the SST is not suggested and should only be done when you intend to harden your control domain using an alternate process.
 - `install-ldm -p`. Specifies that you only want to perform the post-installation actions of enabling the Logical Domains Manager daemon (`ldmd`) and running the SST. For example, you would use this option if the `SUNWldm` and `SUNWjass` packages are preinstalled on your server.

Using JumpStart to Install the Oracle VM Server for SPARC 2.1 Software

See *JumpStart Technology: Effective Use in the Solaris Operating Environment* for complete information about using JumpStart.

▼ Set Up a JumpStart Server

If you have not already set up a JumpStart server, you must do so. See the [Oracle Solaris 10 9/10 Installation Guide: Custom JumpStart and Advanced Installations](#) for complete information about this procedure.

- 1 Refer to the *Oracle Solaris 10 9/10 Installation Guide: Custom JumpStart and Advanced Installations*.

Perform the following steps:

- a. See “**Task Map: Preparing Custom JumpStart Installations**” in *Oracle Solaris 10 9/10 Installation Guide: Custom JumpStart and Advanced Installations*.
 - b. Set up networked systems with the procedures in “Creating a Profile Server for Network Systems.”
 - c. Create a profile with the procedure in “Creating a Profile,” and add a line to the profile to install the `SUNWldm.v` package by using the package profile keyword.
For example, add the following line to your profile to install the `SUNWldm.v` package from the extra directory from the HTTP server 192.168.254.255.

```
package          SUNWldm.v http://192.168.254.255/extra timeout 5
```
 - d. Create the `rules` file with the procedure in “Creating the `rules` File.”
- 2 Validate the `rules` file with the procedure in “Validating the `rules` File.”

Installing the Logical Domains Manager Software Manually

▼ Install the Oracle VM Server for SPARC 2.1 Software Manually

Before You Begin Download the Oracle VM Server for SPARC 2.1 software (the `SUNWldm` and `SUNWldmp2v` packages). See “[Download the Software](#)” on page 31 for specific instructions.

- 1 Use the `pkgadd` command to install the `SUNWldm.v` and `SUNWldmp2v` packages.
For more information about the `pkgadd` command, see the `pkgadd(1M)` man page.
The `-G` option installs the package in the global zone only and the `-d` option specifies the path to the directory that contains the `SUNWldm.v` and `SUNWldmp2v` packages.

```
# pkgadd -Gd . SUNWldm.v SUNWldmp2v
```
- 2 Answer `y` for yes to all questions in the interactive prompts.
- 3 Use the `pkginfo` command to verify that the Oracle VM Server for SPARC 2.1 packages, `SUNWldm` and `SUNWldmp2v`, are installed.

For more information about the `pkginfo` command, see the `pkginfo(1)` man page.

The revision (REV) information shown below is an example.

```
# pkginfo -l SUNWldm | grep VERSION
VERSION=2.1,REV=2011.03.03.10.20
```

Enabling the Logical Domains Manager Daemon

The `install-ldm` installation script automatically enables the Logical Domains Manager daemon (`ldmd`). The `ldmd` daemon is also automatically enabled when the `SUNWldm` package is installed. When enabled, you can create, modify, and control the logical domains.

▼ Enable the Logical Domains Manager Daemon

Use this procedure to enable the `ldmd` daemon if it has been disabled.

- 1 Use the `svcadm` command to enable the Logical Domains Manager daemon, `ldmd`.

For more information about the `svcadm` command, see the [svcadm\(1M\)](#) man page.

```
# svcadm enable ldmd
```

- 2 Use the `ldm list` command to verify that the Logical Domains Manager is running.

The `ldm list` command should list all domains that are currently defined on the system. In particular, the primary domain should be listed and be in the active state. The following sample output shows that only the primary domain is defined on the system.

```
# /opt/SUNWldm/bin/ldm list
NAME          STATE  FLAGS  CONS  VCPU  MEMORY  UTIL  UPTIME
primary       active ---c-  SP    64    3264M   0.3%  19d 9m
```

Upgrading a System Already Using Oracle VM Server for SPARC

This section describes the process of upgrading the Oracle Solaris OS, firmware, and Logical Domains Manager components on a system that is already using the Oracle VM Server for SPARC software.

If your system is already configured with the Oracle VM Server for SPARC software, then the control domain has to be upgraded. The other existing domains also have to be upgraded if you want to be able to use all features of the Oracle VM Server for SPARC 2.1 software.

Upgrading the Oracle Solaris OS

Refer to “Required Software and Patches” in *Oracle VM Server for SPARC 2.1 Release Notes* to find the Oracle Solaris 10 OS that you should use for this version of the Oracle VM Server for SPARC software, and the required and recommended patches for the different domains. Refer to the Oracle Solaris 10 installation guide for complete instructions for upgrading the Oracle Solaris OS.

When reinstalling the Oracle Solaris OS in the control domain, you need to save and restore the Logical Domains autosave configuration data and the constraints database file, as described in this section.

Saving and Restoring Autosave Configuration Directories

You can save and restore autosave configuration directories prior to reinstalling the operating system on the control domain. Whenever you reinstall the operating system on the control domain, you must save and restore the Logical Domains autosave configuration data, which is found in the `/var/opt/SUNWldm/autosave-autosave-name` directories.

You can use the `tar` or `cpio` command to save and restore the entire contents of the directories.

Note – Each autosave directory includes a timestamp for the last SP configuration update for the related configuration. If you restore the autosave files, the timestamp might be out of sync. In this case, the restored autosave configurations are shown in their previous state, either [newer] or up to date.

For more information about autosave configurations, see [“Managing Logical Domains Configurations” on page 184](#).

▼ Save and Restore Autosave Directories

This procedure shows how to save and restore the autosave directories.

1 Save the autosave directories.

```
# cd /
# tar -cvpf autosave.tar var/opt/SUNWldm/autosave-*
```

2 (Optional) Remove the existing autosave directories to ensure a clean restore operation.

Sometimes an autosave directory might include extraneous files, perhaps left over from a previous configuration, that might corrupt the configuration that was downloaded to the SP. In such cases, clean the autosave directory prior to the restore operation as shown in this example:

```
# cd /
# rm -rf var/opt/SUNWldm/autosave-*
```

3 Restore the autosave directories.

These commands restore the files and directories in the `/var/opt/SUNWldm` directory.

```
# cd /
# tar -xvpf autosave.tar
```

Saving and Restoring the Logical Domains Constraints Database File

Whenever you upgrade the operating system on the control domain, you must save and restore the Logical Domains constraints database file that can be found in

`/var/opt/SUNWldm/ldom-db.xml`.

Note – Also, save and restore the `/var/opt/SUNWldm/ldom-db.xml` file when you perform any other operation that is destructive to the control domain's file data, such as a disk swap.

Preserving the Logical Domains Constraints Database File When Using Live Upgrade

If you are using live upgrade on the control domain, consider adding the following line to the `/etc/lu/syncList` file:

```
/var/opt/SUNWldm/ldom-db.xml      OVERWRITE
```

This causes the database to be copied automatically from the active boot environment to the new boot environment when switching boot environments. For more information about `/etc/lu/syncList` and synchronizing files between boot environments, refer to [“Synchronizing Files Between Boot Environments” in Oracle Solaris 10 9/10 Installation Guide: Solaris Live Upgrade and Upgrade Planning](#).

Upgrading From Oracle Solaris 10 OS Older Than Oracle Solaris 10 5/08 OS

If the control domain is upgraded from a Oracle Solaris 10 OS version older than Oracle Solaris 10 5/08 OS (or without patch 127127-11), and if volume manager volumes were exported as virtual disks, then the virtual disk back ends must be re-exported with `options=slice` after the Logical Domains Manager has been upgraded. See [“Exporting Volumes and Backward Compatibility” on page 92](#) for more information.

Upgrading the Logical Domains Manager and the System Firmware

This section shows how to upgrade to Oracle VM Server for SPARC 2.1 software.

First download the Logical Domains Manager to the control domain. See [“Downloading the Logical Domains Manager” on page 31](#).

Then stop all domains (except the control domain) running on the platform:

▼ Stop All Domains Running on the Platform, Except the Control Domain

- 1 Bring down each domain to the ok prompt.
- 2 Stop all domains by using the `-a` option.

```
primary# ldm stop-domain -a
```
- 3 Issue the `unbind-domain` subcommand from the control domain for each domain.

```
primary# ldm unbind-domain ldom
```

Upgrading to Oracle VM Server for SPARC 2.1 Software

This section explains how to upgrade to the Oracle VM Server for SPARC 2.1 software.

▼ Upgrade to the Oracle VM Server for SPARC 2.1 Software

- 1 Perform a flash upgrade of the system firmware.
 For the entire procedure, see [“Upgrade System Firmware” on page 29](#).
- 2 Disable the Logical Domains Manager daemon (`ldmd`).

```
# svcadm disable ldmd
```
- 3 Remove the old `SUNWldm` package.

```
# pkgrm SUNWldm
```
- 4 Add the new `SUNWldm` package.
 Specifying the `-d` option assumes that the package is in the current directory.

```
# pkgadd -Gd . SUNWldm
```
- 5 Use the `ldm list` command to verify that the Logical Domains Manager is running.
 The `ldm list` command should list all domains that are currently defined on the system. In particular, the primary domain should be listed and be in the active state. The following sample output shows that only the primary domain is defined on the system.

```
# ldm list
```

NAME	STATE	FLAGS	CONS	VCPU	MEMORY	UTIL	UPTIME
primary	active	---C-	SP	32	3264M	0.3%	19d 9m

Factory Default Configuration and Disabling Logical Domains

The initial configuration where the platform appears as a single system hosting only one operating system is called the factory default configuration. If you want to disable logical domains, you probably also want to restore this configuration so that the system regains access to all resources (CPUs, memory, I/O), which might have been assigned to other domains.

This section describes how to remove all guest domains, remove all Logical Domains configurations, and revert the configuration to the factory default.

▼ Remove All Guest Domains

- 1 Stop all domains by using the **-a** option.

```
primary# ldm stop-domain -a
```

- 2 Unbind all domains except for the **primary** domain.

```
primary# ldm unbind-domain ldom
```

Note – You might be unable to unbind an I/O domain if it is providing services required by the control domain. In this situation, skip this step.

- 3 Destroy all domains except for the **primary** domain.

```
primary# ldm remove-domain -a
```

▼ Remove All Logical Domains Configurations

- 1 List all the logical domain configurations that are stored on the service processor (SP).

```
primary# ldm list-config
```

- 2 Remove all configurations (*config-name*) previously saved to the SP except for the **factory-default** configuration.

Use the following command for each such configuration:

```
primary# ldm rm-config config-name
```

After you remove all the configurations previously saved to the SP, the **factory-default** domain is the next domain to use when the control domain (primary) is rebooted.

▼ Restore the Factory Default Configuration

- 1 Select the factory default configuration.

```
primary# ldm set-config factory-default
```

- 2 Stop the control domain.

```
primary# shutdown -i1 -g0 -y
```

- 3 Powercycle the system to load the factory default configuration.

```
-> stop /SYS  
-> start /SYS
```

▼ Disable the Logical Domains Manager

- Disable the Logical Domains Manager from the control domain.

```
primary# svcadm disable ldmd
```

Note – Disabling the Logical Domains Manager does not stop any running domains, but does disable the ability to create a new domains, change the configuration of existing domains, or monitor the state of the domains.



Caution – If you disable the Logical Domains Manager, this disables some services, such as error reporting or power management. In the case of error reporting, if you are in the `factory-default` configuration, you can reboot the control domain to restore error reporting. However, this is not the case with power management. In addition, some system management or monitoring tools rely on the Logical Domains Manager.

▼ Removing the Logical Domains Manager

After restoring the factory default configuration and disabling the Logical Domains Manager, you can remove the Logical Domains Manager software.

- Remove the Logical Domains Manager software.

```
primary# pkgrm SUNWldm SUNWldmp2v
```

Note – If you remove the Logical Domains Manager before restoring the factory default configuration, you can restore the factory default configuration from the service processor as shown in the following procedure.

▼ **Restore the Factory Default Configuration From the Service Processor**

If you remove the Logical Domains Manager before restoring the factory default configuration, you can restore the factory default configuration from the service processor.

- 1 **Restore the factory default configuration from the service processor.**

-> `set /HOST/bootmode config=factory-default`

- 2 **Powercycle the system to load the factory default configuration.**

-> `reset /SYS`

Security

This chapter describes some security features that you can enable on your Logical Domains system.

This chapter covers the following topics:

- “Logical Domains Manager Authorization” on page 41
- “Creating Authorizations and Profiles and Assigning Roles to User Accounts” on page 42
- “Configuring RBAC for Guest Console Access” on page 46
- “Enabling and Using Auditing” on page 48

Logical Domains Manager Authorization

Authorization for the Logical Domains Manager has two levels:

- Read – Allows you to view but not modify the configuration
- Read and write – Allows you to view and change the configuration

The following table lists the `ldm` subcommands with the corresponding user authorization that is needed to perform the commands.

TABLE 3-1 The `ldm` Subcommands and User Authorizations

ldm Subcommand ¹	User Authorization
<code>add-*</code>	<code>solaris.ldoms.write</code>
<code>bind-domain</code>	<code>solaris.ldoms.write</code>
<code>list</code>	<code>solaris.ldoms.read</code>
<code>list-*</code>	<code>solaris.ldoms.read</code>
<code>panic-domain</code>	<code>solaris.ldoms.write</code>

¹ Refers to all the resources you can add, list, remove, or set.

TABLE 3-1 The ldm Subcommands and User Authorizations (Continued)

ldm Subcommand ¹	User Authorization
remove-*	solaris.ldoms.write
set-*	solaris.ldoms.write
start-domain	solaris.ldoms.write
stop-domain	solaris.ldoms.write
unbind-domain	solaris.ldoms.write

¹ Refers to all the resources you can add, list, remove, or set.

Creating Authorizations and Profiles and Assigning Roles to User Accounts

You can manage authorizations and profiles and assign roles to user accounts by using the role-based access control (RBAC) feature of the Oracle Solaris OS. For more information about RBAC, see [System Administration Guide: Security Services](#).

Users, authorizations, profiles, and roles can be configured in the following ways:

- Locally on the system by using files
- Centrally in a naming service, such as LDAP

Installing the Logical Domains Manager adds the necessary authorizations and profiles to the local files. To configure users, authorizations, profiles, and roles in a naming service, see [System Administration Guide: Naming and Directory Services \(DNS, NIS, and LDAP\)](#).

Authorization for the Logical Domains Manager has two levels:

- Read – Allows you to view but not modify the configuration
- Read and write – Allows you to view and change the configuration

Following are the Logical Domains entries that are automatically added to the local Oracle Solaris OS `/etc/security/auth_attr` file:

- `solaris.ldoms:::LDom administration::`
- `solaris.ldoms.grant:::Delegate LDom configuration::`
- `solaris.ldoms.read:::View LDom configuration::`
- `solaris.ldoms.write:::Manage LDom configuration::`
- `solaris.smf.manage.ldoms:::Manage Start/Stop LDom::`

Managing User Authorizations

The following procedures show how to manage user authorizations on the system by using local files. To manage user authorizations in a naming service, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

▼ Assign an Authorization to a User

Use this procedure to assign authorizations to Logical Domains Manager users. This authorization assignment information is stored in the local `/etc/security/auth_attr` file.

Note – Superuser already has the `solaris.*` authorization, which includes the `solaris.ldoms.*` authorizations.

1 Become superuser or assume an equivalent role.

Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” in *System Administration Guide: Security Services*.

2 Assign the read or the read and write authorization to a user.

- Assign the read authorization to a user.

```
# usermod -A solaris.ldoms.read username
```

- Assign the read and write authorization to a user.

```
# usermod -A solaris.ldoms.write username
```

Note – Ensure that you include any existing authorizations for the user in the `usermod -A` command. The authorizations that you specify with this command replace any authorizations that have already been assigned to the user. See the `usermod(1M)` man page.

For the list of user authorizations that are required by the `ldm` subcommands, see [Table 3–1](#).

▼ Delete All Authorizations Assigned to a User

1 Become superuser or assume an equivalent role.

Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” in *System Administration Guide: Security Services*.

2 Delete all authorizations that are assigned to a local user.

```
# usermod -A "" username
```

Managing User Profiles

The following procedures show how to manage user profiles on the system by using local files. To manage user profiles in a naming service, see [System Administration Guide: Naming and Directory Services \(DNS, NIS, and LDAP\)](#).

The SUNWldm package adds two system-defined RBAC profiles to the local `/etc/security/prof_attr` file. The following profiles are used to authorize access to the Logical Domains Manager by unprivileged users:

- LDoms Review::`Review LDoms configuration:auths=solaris.ldoms.read`
- LDoms Management::`Manage LDoms domains:auths=solaris.ldoms.*`

The SUNWldm package also defines the following execution attribute that is associated with the LDoms Management profile:

```
LDoms_Management:user:cmd:::/usr/sbin/ldm:privs=file_dac_read,file_dac_search
```

▼ Assign a Profile to a User

Users who have been directly assigned the LDoms Management profile *must* invoke a profile shell to run the `ldm` command with security attributes. For more information, see [System Administration Guide: Security Services](#).

1 Become superuser or assume an equivalent role.

Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” in [System Administration Guide: Security Services](#).

2 Assign an administrative profile to a local user account.

You can assign either the LDoms Review profile or the LDoms Management profile to a user account.

```
# usermod -P "profile-name" username
```

The following command assigns the LDoms Management profile to user `sam`:

```
# usermod -P "LDoms_Management" sam
```

▼ Delete All Profiles Assigned to a User

1 Become superuser or assume an equivalent role.

Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” in [System Administration Guide: Security Services](#).

2 Delete all profiles that are assigned to a local user.

```
# usermod -P "" username
```

Assigning Roles to Users

The following procedure shows how to create a role and assign it to a user by using local files. To manage roles in a naming service, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

The advantage of using this procedure is that only a user who has been assigned a specific role can assume that role. When assuming a role, a password is required if the role has been assigned a password. These two layers of security prevent a user who has not been assigned a role, yet has the password, from assuming that role.

▼ Create a Role and Assign the Role to a User

1 Become superuser or assume an equivalent role.

Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” in *System Administration Guide: Security Services*.

2 Create a role.

```
# roleadd -P "profile-name" role-name
```

3 Assign a password to the role.

You will be prompted to specify and then verify a new password.

```
# passwd role-name
```

4 Assign the role to a user.

```
# useradd -R role-name username
```

5 Assign a password to the user.

You will be prompted to specify and then verify a new password.

```
# passwd username
```

6 Become the user and provide the password, if necessary.

```
# su username
```

7 Verify that the user has access to the assigned role.

```
$ id
uid=nn(username) gid=nn(group-name)
$ roles
role-name
```

8 Assume the role and provide the password, if necessary.

```
$ su role-name
```

9 Verify that the user has assumed the role.

```
$ id
uid=nn(role-name) gid=nn(group-name)
```

Example 3–1 Creating a Role and Assigning the Role to a User

This example shows how to create the `ldm_read` role, assign the role to the `user_1` user, become the `user_1` user, and assume the `ldm_read` role.

```
# roleadd -P "LDoms Review" ldm_read
# passwd ldm_read
New Password: ldm_read-password
Re-enter new Password: ldm_read-password
passwd: password successfully changed for ldm_read
# useradd -R ldm_read user_1
# passwd user_1
New Password: user_1-password
Re-enter new Password: user_1-password
passwd: password successfully changed for user_1
# su user_1
Password: user_1-password
$ id
uid=95555(user_1) gid=10(staff)
$ roles
ldm_read
$ su ldm_read
Password: ldm_read-password
$ id
uid=99667(ldm_read) gid=14(sysadmin)
```

Configuring RBAC for Guest Console Access

The `vntsd` daemon provides a Service Management Facility (SMF) property named `vntsd/authorization`. This property can be configured to enable the authorization checking of users and roles for a domain console or a console group. To enable authorization checking, use the `svccfg` command to set the value of this property to `true`. While this option is enabled, `vntsd` listens and accepts connections only on `localhost`. If the `listen_addr` property specifies an alternate IP address when `vntsd/authorization` is enabled, `vntsd` ignores the alternate IP address and continues to listen only on `localhost`.



Caution – Do *not* configure the `vntsd` service to use a host other than `localhost`.

If you specify a host other than `localhost`, you are no longer restricted from connecting to guest domain consoles from the control domain. If you use the `telnet` command to remotely connect to a guest domain, the login credentials are passed as clear text over the network.

By default, an authorization to access all guest consoles is present in the `local_auth_attr` database.

```
solaris.vntsd.consoles:::Access All LDoms Guest Consoles::
```

Use the `usermod` command to assign the required authorizations to other users or roles in local files. This permits only the user or role who has the required authorizations to access a given domain console or console group. To assign authorizations to other users or roles in a naming service, see [System Administration Guide: Naming and Directory Services \(DNS, NIS, and LDAP\)](#).

The following example gives user `terry` the authorization to access all domain consoles by updating local files:

```
# usermod -A "solaris.vntsd.consoles" terry
```

▼ Add an Authorization for a Domain Console

This procedure shows how to add a new authorization for a specific domain console and assign that authorization to a user by using local files. To manage authorizations and users in a naming service, see [System Administration Guide: Naming and Directory Services \(DNS, NIS, and LDAP\)](#).

For more information about authorizations and RBAC, see [System Administration Guide: Security Services](#).

1 Add an authorization entry to the local `auth_attr` file for a domain.

```
solaris.vntsd.console-domain-name:::Access domain-name Guest Console::
```

2 Assign the new authorization to a user.

```
# usermod -A "solaris.vntsd.console-domain-name" username
```

Example 3–2 Adding an Authorization for a Domain Console

The following example adds a new authorization for a specific domain console with the name `ldg1` and assigns that authorization to user `sam`.

The following authorization entry is added to the local `auth_attr` file for domain `ldg1`:

```
solaris.vntsd.console-ldg1:::Access ldg1 Guest Console::
```

The following command assigns the new authorization to user `sam`:

```
# usermod -A "solaris.vntsd.console-ldg1" sam
```

Enabling and Using Auditing

The Logical Domains Manager uses the Oracle Solaris OS auditing feature to examine the history of actions and events that have occurred on your control domain. The history is kept in a log of what was done, when it was done, by whom, and what was affected.

You can enable and disable the auditing feature based on the version of the Oracle Solaris OS that runs on your system, as follows:

- **Oracle Solaris 10 OS.** Use the `bsmconv` and `bsmunconv` commands. See the [bsmconv\(1M\)](#) and [bsmunconv\(1M\)](#) man pages, and the Oracle Solaris 10 version of *System Administration Guide: Security Services*.
- **Oracle Solaris 11 Express OS.** Use the `audit` command. See the [audit\(1M\)](#) man page and the Oracle Solaris 11 Express version of *System Administration Guide: Security Services*.

▼ Enable Auditing

- 1 **Add customizations to the `/etc/security/audit_event` and `/etc/security/audit_class` files.**

These customizations are preserved across Oracle Solaris upgrades, but should be re-added after a fresh Oracle Solaris installation.

- a. **Add the following entry to the `audit_event` file, if not already present:**

```
40700:AUE_ldoms:ldoms administration:vs
```

- b. **Add the following entry to the `audit_class` file, if not already present:**

```
0x10000000:vs:virtualization_software
```

- 2 **Enable the auditing feature.**

- **Enable the auditing feature on your Oracle Solaris 10 system.**

- a. **Run the `bsmconv` command.**

```
# /etc/security/bsmconv
```

- b. **Reboot the system.**

- **Enable the auditing feature on your Oracle Solaris 11 Express system.**

```
# audit -s
```

- 3 **Verify that the auditing software is running.**

```
# auditconfig -getcond
```

If the auditing software is running, `audit condition = auditing` appears in the output.

▼ Disable Auditing

- Disable the auditing feature.

- Disable the auditing feature on your Oracle Solaris 10 system.

- a. Run the `bsmunconv` command.

```
# /etc/security/bsmunconv
Are you sure you want to continue? [y/n] y
This script is used to disable the Basic Security Module (BSM).
Shall we continue the reversion to a non-BSM system now? [y/n] y
bsmunconv: INFO: removing c2audit:audit_load from /etc/system.
bsmunconv: INFO: stopping the cron daemon.
```

```
The Basic Security Module has been disabled.
Reboot this system now to come up without BSM.
```

- b. Reboot the system.

- Disable the auditing feature on your Oracle Solaris 11 Express system.

- a. Run the `audit -t` command.

```
# audit -t
```

- b. Verify that the auditing software is no longer running.

```
# auditconfig -getcond
audit condition = noaudit
```

▼ Print Audit Output

- Use one of the following to print audit output:

- Use the `auditreduce` and `praudit` commands to print audit output.

```
# auditreduce -c vs | praudit
# auditreduce -c vs -a 20060502000000 | praudit
```

- Use the `praudit -x` command to print XML output.

▼ Rotate Audit Logs

- Use the `audit -n` command to rotate audit logs.

Rotating the audit logs closes the current audit file and opens a new one in the current audit directory.

Setting Up Services and the Control Domain

This chapter describes the procedures necessary to set up default services and your control domain.

You can also use the Oracle VM Server for SPARC Configuration Assistant to configure logical domains and services. See [Chapter 14, “Oracle VM Server for SPARC Configuration Assistant.”](#)

This chapter covers the following topics:

- “Output Messages” on page 51
- “Creating Default Services” on page 52
- “Initial Configuration of the Control Domain” on page 53
- “Rebooting to Use Logical Domains” on page 54
- “Enabling Networking Between the Control/Service Domain and Other Domains” on page 55
- “Enabling the Virtual Network Terminal Server Daemon” on page 56

Output Messages

Starting with the Oracle VM Server for SPARC 2.0 release, if a resource cannot be dynamically configured on the control domain, it is best to first initiate a delayed reconfiguration. The delayed reconfiguration postpones the configuration activities until after the control domain has been rebooted.

You receive the following message when you initiate a delayed reconfiguration on the primary domain:

```
Initiating a delayed reconfiguration operation on the primary domain.  
All configuration changes for other domains are disabled until the  
primary domain reboots, at which time the new configuration for the  
primary domain also takes effect.
```

You receive the following notice after every subsequent operation on the primary domain until reboot:

Notice: The primary domain is in the process of a delayed reconfiguration. Any changes made to the primary domain will only take effect after it reboots.

Creating Default Services

The following virtual device services must be created to use the control domain as a service domain and to create virtual devices for other domains:

- vcc – Virtual console concentrator service
- vds – Virtual disk server
- vsw – Virtual switch service

▼ Create Default Services

- 1 **Create a virtual console concentrator (vcc) service for use by the virtual network terminal server daemon (vntsd) and as a concentrator for all logical domain consoles.**

For example, the following command would add a virtual console concentrator service (primary-vcc0) with a port range from 5000 to 5100 to the control domain (primary).

```
primary# ldm add-vcc port-range=5000-5100 primary-vcc0 primary
```

- 2 **Create a virtual disk server (vds) to allow importing virtual disks into a logical domain.**

For example, the following command adds a virtual disk server (primary-vds0) to the control domain (primary).

```
primary# ldm add-vds primary-vds0 primary
```

- 3 **Create a virtual switch service (vsw) to enable networking between virtual network (vnet) devices in logical domains.**

Assign a GLDv3-compliant network adapter to the virtual switch if each of the logical domains needs to communicate outside the box through the virtual switch.

For example, the following command would add a virtual switch service (primary-vsw0) on network adapter driver nxge0 to the control domain (primary).

```
primary# ldm add-vsw net-dev=nxge0 primary-vsw0 primary
```

This command automatically allocates a MAC address to the virtual switch. You can specify your own MAC address as an option to the `ldm add-vsw` command. However, in that case, it is your responsibility to ensure that the MAC address specified does not conflict with an already existing MAC address.

If the virtual switch being added replaces the underlying physical adapter as the primary network interface, it must be assigned the MAC address of the physical adapter, so that the

Dynamic Host Configuration Protocol (DHCP) server assigns the domain the same IP address. See [“Enabling Networking Between the Control/Service Domain and Other Domains”](#) on page 55.

```
primary# ldm add-vsw mac-addr=2:04:4f:fb:9f:0d net-dev=nxge0 primary-vsw0 primary
```

4 Verify the services have been created by using the `list-services` subcommand.

Your output should look similar to the following.

```
primary# ldm list-services primary
```

```
VDS
  NAME          VOLUME          OPTIONS          DEVICE
  primary-vds0

VCC
  NAME          PORT-RANGE
  primary-vcc0  5000-5100

VSW
  NAME          MAC          NET-DEV          DEVICE          MODE
  primary-vsw0  02:04:4f:fb:9f:0d nxge0  switch@0  prog,promisc
```

Initial Configuration of the Control Domain

Initially, all system resources are allocated to the control domain. To allow the creation of other logical domains, you must release some of these resources.

Do *not* attempt to use memory dynamic reconfiguration (DR) to perform the initial configuration of the control domain. Although you can use memory DR to perform this configuration without needing a reboot, it is *not* recommended. The memory DR approach might take a very long time (longer than a reboot) and could even potentially fail. Instead, use the `ldm start-reconf` command to place the control domain in delayed reconfiguration mode before you change the memory configuration. Then, you can reboot the control domain after you complete all the configuration steps.

▼ Set Up the Control Domain

Note – This procedure contains examples of resources to set for your control domain. These numbers are examples only, and the values used might not be appropriate for your control domain.

1 Determine whether you have cryptographic devices in the control domain.

```
primary# ldm list -o crypto primary
```

2 Assign cryptographic resources to the control domain.

The following example would assign one cryptographic resource to the control domain, `primary`. This leaves the remainder of the cryptographic resources available to a guest domain.

```
primary# ldm set-mau 1 primary
```

3 Assign virtual CPUs to the control domain.

For example, the following command would assign 8 virtual CPUs to the control domain, `primary`. This leaves the remainder of the virtual CPUs available to a guest domain.

```
primary# ldm set-vcpu 8 primary
```

4 Initiate a delayed reconfiguration on the control domain.

```
primary# ldm start-reconf primary
```

5 Assign memory to the control domain.

For example, the following command would assign 4 gigabytes of memory to the control domain, `primary`. This leaves the remainder of the memory available to a guest domain.

```
primary# ldm set-memory 4G primary
```

6 Add a logical domain machine configuration to the service processor (SP).

For example, the following command would add a configuration called `initial`.

```
primary# ldm add-config initial
```

7 Verify that the configuration is ready to be used at the next reboot.

```
primary# ldm list-config
factory-default
initial [next poweron]
```

This list subcommand shows the `initial` configuration set will be used once you powercycle.

Rebooting to Use Logical Domains

You must reboot the control domain for the configuration changes to take effect and for the resources to be released for other logical domains to use.

▼ Reboot

● Shut down and reboot the control domain.

```
primary# shutdown -y -g0 -i6
```

Note – Either a reboot or powercycle instantiates the new configuration. Only a powercycle actually boots the configuration saved to the service processor (SP), which is then reflected in the `list-config` output.

Enabling Networking Between the Control/Service Domain and Other Domains

By default, networking between the control domain and other domains in the system is disabled. To enable this, the virtual switch device should be configured as a network device. The virtual switch can either replace the underlying physical device (`nxge0` in this example) as the primary interface or be configured as an additional network interface in the domain.

Note – Perform the following procedure from the control domain's console, as the procedure could temporarily disrupt network connectivity to the domain.

▼ Configure the Virtual Switch as the Primary Interface

- 1 Print out the addressing information for all interfaces.

```
primary# ifconfig -a
```

- 2 Create the virtual switch.

In this example, `vsw0` is the virtual switch being configured.

```
primary# ifconfig vsw0 plumb
```

- 3 (Optional) To obtain the list of all virtual switch instances in a domain, you can list them.

```
primary# /usr/sbin/dladm show-link | grep vsw
vsw0          type: non-vlan  mtu: 1500      device: vsw0
```

- 4 Delete the physical network device assigned to the virtual switch (`net - dev`).

The physical network device is `nxge0` in this example.

```
primary# ifconfig nxge0 down unplumb
```

- 5 To migrate properties of the physical network device (`nxge0`) to the virtual switch (`vsw0`) device. Do one of the following:

- If networking is configured using a static IP address, reuse the IP address and netmask of `nxge0` for `vsw0`.

```
primary# ifconfig vsw0 IP-of-nxge0 netmask netmask-of-nxge0 broadcast + up
```

- If networking is configured using DHCP, enable DHCP for vsw0.

```
primary# ifconfig vsw0 dhcp start
```

- 6 Make the required configuration file modifications to make this change permanent.

```
primary# mv /etc/hostname.nxge0 /etc/hostname.vsw0
primary# mv /etc/dhcp.nxge0 /etc/dhcp.vsw0
```

Note – If necessary, you can also configure the virtual switch as well as the physical network device. In this case, create the virtual switch as in Step 2, and do not delete the physical device (skip Step 4). You must then configure the virtual switch with either a static IP address or a dynamic IP address. You can obtain a dynamic IP address from a DHCP server. For additional information and an example of this case, see [“Configuring Virtual Switch and Service Domain for NAT and Routing”](#) on page 119.

Enabling the Virtual Network Terminal Server Daemon

You must enable the virtual network terminal server daemon (vntsd) to provide access to the virtual console of each logical domain. Refer to the [vntsd\(1M\)](#) man page for information about how to use this daemon.

▼ Enable the Virtual Network Terminal Server Daemon

Note – Be sure that you have created the default service vconscon (vcc) on the control domain before you enable vntsd. See [“Creating Default Services”](#) on page 52 for more information.

- 1 Use the [svcadm\(1M\)](#) command to enable the virtual network terminal server daemon, vntsd(1M).

```
primary# svcadm enable vntsd
```

- 2 Use the [svcs\(1\)](#) command to verify that the vntsd daemon is enabled.

```
primary# svcs vntsd
STATE      STIME      FMRI
online     Oct_08     svc:/ldoms/vntsd:default
```

Setting Up Guest Domains

This chapter describes the procedures necessary to set up guest domains.

You can also use the Oracle VM Server for SPARC Configuration Assistant to configure logical domains and services. See [Chapter 14, “Oracle VM Server for SPARC Configuration Assistant.”](#)

This chapter covers the following topics:

- “Creating and Starting a Guest Domain” on page 57
- “Installing Oracle Solaris OS on a Guest Domain” on page 60

Creating and Starting a Guest Domain

The guest domain must run an operating system that understands both the sun4v platform and the virtual devices presented by the hypervisor. Currently, this means that you must run at least the Oracle Solaris 10 11/06 OS. Running the Oracle Solaris 10 9/10 OS provides you with all the Oracle VM Server for SPARC 2.1 features. See the [Oracle VM Server for SPARC 2.1 Release Notes](#) for any specific patches that might be necessary. Once you have created default services and reallocated resources from the control domain, you can create and start a guest domain.

▼ Create and Start a Guest Domain

1 Create a logical domain.

For example, the following command would create a guest domain named `ldg1`.

```
primary# ldm add-domain ldg1
```

2 Add CPUs to the guest domain.

For example, the following command would add eight virtual CPUs to guest domain `ldg1`.

```
primary# ldm add-vcpu 8 ldg1
```

3 Add memory to the guest domain.

For example, the following command would add 2 gigabytes of memory to guest domain `ldg1`.

```
primary# ldm add-memory 2G ldg1
```

4 Add a virtual network device to the guest domain.

For example, the following command would add a virtual network device with these specifics to the guest domain `ldg1`.

```
primary# ldm add-vnet vnet1 primary-vsw0 ldg1
```

Where:

- `vnet1` is a unique interface name to the logical domain, assigned to this virtual network device instance for reference on subsequent `set-vnet` or `remove-vnet` subcommands.
- `primary-vsw0` is the name of an existing network service (virtual switch) to which to connect.

Note – Steps 5 and 6 are simplified instructions for adding a virtual disk server device (`vdsdev`) to the primary domain and a virtual disk (`vdisk`) to the guest domain. To learn how ZFS volumes and file systems can be used as virtual disks, see [“Export a ZFS Volume as a Single-Slice Disk” on page 91](#) and [“Using ZFS With Virtual Disks” on page 100](#).

5 Specify the device to be exported by the virtual disk server as a virtual disk to the guest domain.

You can export a physical disk, disk slice, volumes, or file as a block device. The following examples show a physical disk and a file.

- **Physical Disk Example.** The first example adds a physical disk with these specifics.

```
primary# ldm add-vdsdev /dev/dsk/c2t1d0s2 vol1@primary-vds0
```

Where:

- `/dev/dsk/c2t1d0s2` is the path name of the actual physical device. When adding a device, the path name must be paired with the device name.
- `vol1` is a unique name you must specify for the device being added to the virtual disk server. The volume name must be unique to this virtual disk server instance, because this name is exported by this virtual disk server to the clients for adding. When adding a device, the volume name must be paired with the path name of the actual device.
- `primary-vds0` is the name of the virtual disk server to which to add this device.
- **File Example.** This second example is exporting a file as a block device.

```
primary# ldm add-vdsdev backend vol1@primary-vds0
```

Where:

- *backend* is the path name of the actual file exported as a block device. When adding a device, the back end must be paired with the device name.
- *vol1* is a unique name you must specify for the device being added to the virtual disk server. The volume name must be unique to this virtual disk server instance, because this name is exported by this virtual disk server to the clients for adding. When adding a device, the volume name must be paired with the path name of the actual device.
- *primary-vds0* is the name of the virtual disk server to which to add this device.

6 Add a virtual disk to the guest domain.

The following example adds a virtual disk to the guest domain `ldg1`.

```
primary# ldm add-vdisk vdisk1 vol1@primary-vds0 ldg1
```

Where:

- *vdisk1* is the name of the virtual disk.
- *vol1* is the name of the existing volume to which to connect.
- *primary-vds0* is the name of the existing virtual disk server to which to connect.

Note – The virtual disks are generic block devices that are associated with different types of physical devices, volumes, or files. A virtual disk is not synonymous with a SCSI disk and, therefore, excludes the target ID in the disk label. Virtual disks in a logical domain have the following format: *cNdNsN*, where *cN* is the virtual controller, *dN* is the virtual disk number, and *sN* is the slice.

7 Set `auto-boot?` and `boot-device` variables for the guest domain.

The first example command sets `auto-boot?` to `true` for guest domain `ldg1`.

```
primary# ldm set-var auto-boot\?=true ldg1
```

The second example command sets `boot-device` to `vdisk` for the guest domain `ldg1`.

```
primary# ldm set-var boot-device=vdisk1 ldg1
```

8 Bind resources to the guest domain `ldg1` and then list the domain to verify that it is bound.

```
primary# ldm bind-domain ldg1
```

```
primary# ldm list-domain ldg1
```

NAME	STATE	FLAGS	CONS	VCPU	MEMORY	UTIL	UPTIME
ldg1	bound	-----	5000	8	2G		

9 To find the console port of the guest domain, you can look at the output of the preceding `list-domain` subcommand.

You can see under the heading `CONS` that logical domain guest 1 (`ldg1`) has its console output bound to port `5000`.

- 10 Connect to the console of a guest domain from another terminal by logging into the control domain and connecting directly to the console port on the local host.
- ```
$ ssh hostname.domain-name
$ telnet localhost 5000
```
- 11 Start the guest domain `ldg1`.
- ```
primary# ldm start-domain ldg1
```

Installing Oracle Solaris OS on a Guest Domain

This section provides instructions for several different ways you can install the Oracle Solaris OS on a guest domain.



Caution – Do *not* disconnect from the virtual console during the installation of the Oracle Solaris OS.

▼ Install Oracle Solaris OS on a Guest Domain From a DVD

- 1 Insert the Oracle Solaris 10 OS DVD into the DVD drive.
- 2 Stop the volume management daemon, `vold(1M)`, on the primary domain.
- ```
primary# svcadm disable volfs
```
- 3 Stop and unbind the guest domain (`ldg1`).
- ```
primary# ldm stop ldg1
primary# ldm unbind ldg1
```
- 4 Add the DVD with the DVD-ROM media as a secondary volume and virtual disk.
- The following uses `c0t0d0s2` as the DVD drive in which the Oracle Solaris media resides, `dvd_vol@primary-vds0` as a secondary volume, and `vdisk_cd_media` as a virtual disk.
- ```
primary# ldm add-vdsdev /dev/dsk/c0t0d0s2 dvd_vol@primary-vds0
primary# ldm add-vdisk vdisk_cd_media dvd_vol@primary-vds0 ldg1
```
- 5 Check to see that the DVD is added as a secondary volume and virtual disk.

```
primary# ldm list-bindings
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -n-cv SP 4 4G 0.2% 22h 45m
...
VDS
NAME VOLUME OPTIONS DEVICE
primary-vds0 vol1 /dev/dsk/c2t1d0s2
```

```

 dvd_vol /dev/dsk/c0t0d0s2

NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
ldg1 inactive -----
...
DISK
 NAME VOLUME TOUT DEVICE SERVER
 vdisk1 voll@primary-vds0
 vdisk_cd_media dvd_vol@primary-vds0


```

## 6 Bind and start the guest domain (ldg1).

```

primary# ldm bind ldg1
primary# ldm start ldg1
LDom ldg1 started
primary# telnet localhost 5000
Trying 027.0.0.1...
Connected to localhost.
Escape character is '^]'.

Connecting to console "ldg1" in group "ldg1"
Press ~? for control options ..

```

## 7 Show the device aliases in the client OpenBoot PROM.

In this example, see the device aliases for `vdisk_cd_media`, which is the Oracle Solaris DVD, and `vdisk1`, which is a virtual disk on which you can install the Oracle Solaris OS.

```

ok devalias
vdisk_cd_media /virtual-devices@100/channel-devices@200/disk@1
vdisk1 /virtual-devices@100/channel-devices@200/disk@0
vnet1 /virtual-devices@100/channel-devices@200/network@0
virtual-console /virtual-devices/console@1
name aliases

```

## 8 On the guest domain's console, boot from `vdisk_cd_media` (disk@1) on slice `f`.

```

ok boot vdisk_cd_media:f
Boot device: /virtual-devices@100/channel-devices@200/disk@1:f File and args: -s
SunOS Release 5.10 Version Generic_139555-08 64-bit
Copyright (c), 1983-2010, Oracle and/or its affiliates. All rights reserved.

```

## 9 Continue with the Oracle Solaris OS installation menu.

# ▼ Install Oracle Solaris OS on a Guest Domain From a Oracle Solaris ISO File

## 1 Stop and unbind the guest domain (ldg1).

```

primary# ldm stop ldg1
primary# ldm unbind ldg1

```

**2 Add the Oracle Solaris ISO file as a secondary volume and virtual disk.**

The following uses `solarisdvd.iso` as the Oracle Solaris ISO file, `iso_vol@primary-vds0` as a secondary volume, and `vdisk_iso` as a virtual disk:

```
primary# ldm add-vdsdev /export/solarisdvd.iso iso_vol@primary-vds0
primary# ldm add-vdisk vdisk_iso iso_vol@primary-vds0 ldg1
```

**3 Check to see that the Oracle Solaris ISO file is added as a secondary volume and virtual disk.**

```
primary# ldm list-bindings
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -n-cv SP 4 4G 0.2% 22h 45m
...
VDS
 NAME VOLUME OPTIONS DEVICE
 primary-vds0 voll /dev/dsk/c2t1d0s2
 iso_vol /export/solarisdvd.iso
....

NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
ldg1 inactive ----- 60 6G
...
DISK
 NAME VOLUME TOUT ID DEVICE SERVER MPGROUP
 vdisk1 voll@primary-vds0
 vdisk_iso iso_vol@primary-vds0
....
```

**4 Bind and start the guest domain (ldg1).**

```
primary# ldm bind ldg1
primary# ldm start ldg1
LDom ldg1 started
primary# telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^J'.

Connecting to console "ldg1" in group "ldg1"
Press ~? for control options ..
```

**5 Show the device aliases in the client OpenBoot PROM.**

In this example, see the device aliases for `vdisk_iso`, which is the Oracle Solaris ISO image, and `vdisk_install`, which is the disk space.

```
ok devalias
vdisk iso /virtual-devices@100/channel-devices@200/disk@1
vdisk1 /virtual-devices@100/channel-devices@200/disk@0
vnet1 /virtual-devices@100/channel-devices@200/network@0
virtual-console /virtual-devices/console@1
name aliases
```

**6 On the guest domain's console, boot from vdisk\_iso (disk@1) on slice f.**

```
ok boot vdisk_iso:f
Boot device: /virtual-devices@100/channel-devices@200/disk@1:f File and args: -s
SunOS Release 5.10 Version Generic_139555-08 64-bit
Copyright (c) 1983-2010, Oracle and/or its affiliates. All rights reserved.
```

## 7 Continue with the Oracle Solaris OS installation menu.

### ▼ Perform a JumpStart Operation on a Guest Domain

This procedure describes how to perform a JumpStart operation on a guest domain. This procedure follows the usual JumpStart procedure, but it describes the different disk device name format to use in the JumpStart profile for the guest domain. See the [Oracle Solaris 10 9/10 Installation Guide: Custom JumpStart and Advanced Installations](#).

Virtual disk device names in a logical domain differ from physical disk device names. Virtual disk device names do not contain a target ID (tN).

Instead of the usual cNtNdNsN format, virtual disk device names use the cNdNsN format. cN is the virtual controller, dN is the virtual disk number, and sN is the slice number.

#### ● Modify your JumpStart profile to reflect this change.

A virtual disk can appear either as a full disk or as a single-slice disk. The Oracle Solaris OS can be installed on a full disk by using a regular JumpStart profile that specifies multiple partitions. A single-slice disk only has a single partition, s0, that uses the entire disk. To install the Oracle Solaris OS on a single disk, you must use a profile that has a single partition (/) that uses the entire disk. You cannot define any other partitions, such as swap. For more information about full disks and single-slice disks, see “[Virtual Disk Appearance](#)” on page 85.

#### ■ JumpStart profile for installing a UFS root file system.

See [Oracle Solaris 10 9/10 Installation Guide: Custom JumpStart and Advanced Installations](#).

##### Normal UFS Profile

```
filesys clt1d0s0 free /
filesys clt1d0s1 2048 swap
filesys clt1d0s5 120 /spare1
filesys clt1d0s6 120 /spare2
```

##### Actual UFS Profile for Installing a Domain on a Full Disk

```
filesys c0d0s0 free /
filesys c0d0s1 2048 swap
filesys c0d0s5 120 /spare1
filesys c0d0s6 120 /spare2
```

##### Actual UFS Profile for Installing a Domain on a Single-Slice Disk

```
filesys c0d0s0 free /
```

#### ■ JumpStart profile for installing a ZFS root file system.

See Chapter 9, “Installing a ZFS Root Pool With JumpStart,” in [Oracle Solaris 10 9/10 Installation Guide: Custom JumpStart and Advanced Installations](#).

### **Normal ZFS Profile**

```
pool rpool auto 2G 2G c1t1d0s0
```

### **Actual ZFS Profile for Installing a Domain**

```
pool rpool auto 2G 2G c0d0s0
```

# Setting Up I/O Domains

---

This chapter describes I/O domains and how to configure them in a Logical Domains environment.

This chapter covers the following topics:

- [“I/O Domain Overview” on page 65](#)
- [“Assigning PCIe Buses” on page 66](#)
- [“Assigning PCIe Endpoint Devices” on page 70](#)

## I/O Domain Overview

An I/O domain has direct ownership of and direct access to physical I/O devices. It can be created by assigning a PCI EXPRESS (PCIe) bus or a PCIe endpoint device to a domain. Use the `ldm add-io` command to assign a bus or device to a domain.

You might want to configure I/O domains for the following reasons:

- An I/O domain has direct access to a physical I/O device, which avoids the performance overhead that is associated with virtual I/O. As a result, the I/O performance on an I/O domain more closely matches the I/O performance on a bare-metal system.
- An I/O domain can host virtual I/O services to be used by other guest domains.

For information about configuring I/O domains, see the following:

- [“Assigning PCIe Buses” on page 66](#)
- [“Assigning PCIe Endpoint Devices” on page 70](#)

---

**Note** – You *cannot* migrate an I/O domain that is configured with PCIe endpoint devices. For information about other migration limitations, see [Chapter 9, “Migrating Domains.”](#)

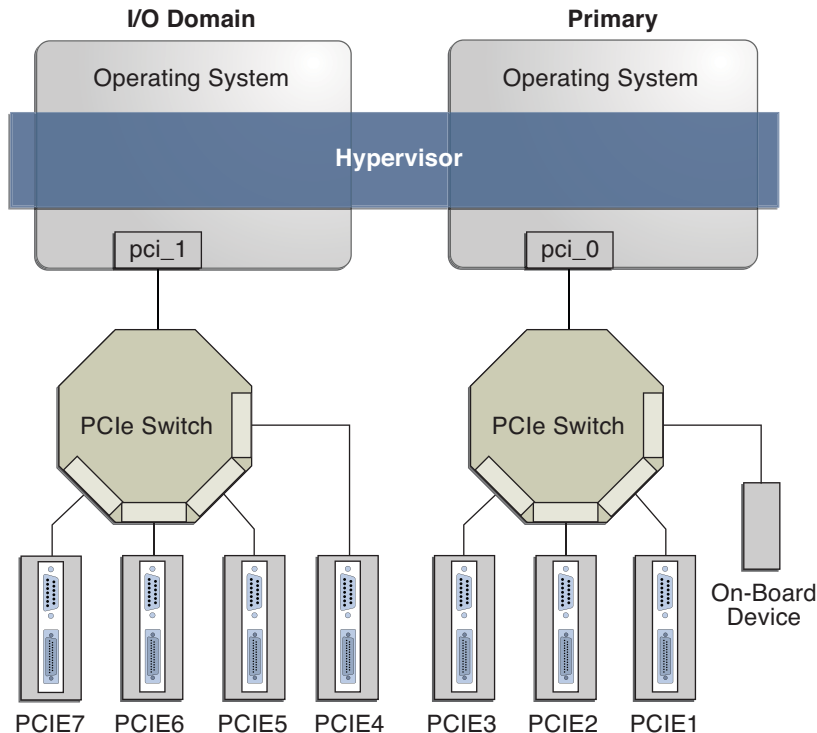
---

## Assigning PCIe Buses

You can use the Oracle VM Server for SPARC software to assign an entire PCIe bus (also known as a *root complex*) to a domain. An entire PCIe bus consists of the PCIe bus itself, and all of its PCI switches and devices. PCIe buses that are present on a server are identified with names such as `pci@400 (pci_0)`. An I/O domain that is configured with an entire PCIe bus is also known as a *root domain*.

The following diagram shows a system that has two PCIe buses (`pci_0` and `pci_1`). Each bus is assigned to a different domain. Thus, the system is configured with two I/O domains.

FIGURE 6-1 Assigning a PCIe Bus to an I/O Domain



The maximum number of I/O domains that you can create with PCIe buses depends on the number of PCIe buses that are available on the server. For example, if you are using a Sun SPARC Enterprise T5440 server, you can have up to four I/O domains.

---

**Note** – Some Sun UltraSPARC servers have only one PCIe bus. In such cases, you can create an I/O domain by assigning a PCIe endpoint (or direct I/O-assignable) device to a domain. See [“Assigning PCIe Endpoint Devices” on page 70](#). If the system has a Network Interface Unit (NIU), you can also assign an NIU to a domain to create an I/O domain.

---

When you assign a PCIe bus to an I/O domain, all devices on that bus are owned by that I/O domain. You are not permitted to assign any of the PCIe endpoint devices on that bus to other domains. Only the PCIe endpoint devices on the PCIe buses that are assigned to the primary domain can be assigned to other domains.

When a server is initially configured in a Logical Domains environment or is using the factory-default configuration, the primary domain has access to all the physical device resources. This means that the primary domain is the only I/O domain configured on the system and that it owns all the PCIe buses.

## ▼ Create an I/O Domain by Assigning a PCIe Bus

This example procedure shows how to create a new I/O domain from an initial configuration where several buses are owned by the primary domain. By default the primary domain owns all buses present on the system. This example is for a Sun SPARC Enterprise T5440 server. This procedure can also be used on other servers. The instructions for different servers might vary slightly from these, but you can obtain the basic principles from this example.

First, you must retain the bus that has the primary domain's boot disk. Then, remove another bus from the primary domain and assign it to another domain.



---

**Caution** – All internal disks on the supported servers might be connected to a single PCIe bus. If a domain is booted from an internal disk, do not remove that bus from the domain. Also, ensure that you are not removing a bus with devices (such as network ports) that are used by a domain. If you remove the wrong bus, a domain might not be able to access the required devices and could become unusable. To remove a bus that has devices that are used by a domain, reconfigure that domain to use devices from other buses. For example, you might have to reconfigure the domain to use a different onboard network port or a PCIe card from a different PCIe slot.

---

In this example, the primary domain only uses a ZFS pool (`rpool (c0t1d0s0)`) and network interface (`nxge0`). If the primary domain uses more devices, repeat Steps 2-4 for each device to ensure that none are located on the bus that will be removed.

**1 Verify that the primary domain owns more than one PCIe bus.**

```
primary# ldm list-io
IO PSEUDONYM DOMAIN

pci@400 pci_0 primary
pci@500 pci_1 primary
pci@600 pci_2 primary
pci@700 pci_3 primary
```

```
PCIE PSEUDONYM STATUS DOMAIN

pci@400/pci@0/pci@d MB/PCIE0 EMP -
pci@400/pci@0/pci@c MB/PCIE1 OCC primary
pci@400/pci@0/pci@l MB/HBA OCC primary
pci@500/pci@0/pci@d MB/PCIE4 EMP -
pci@500/pci@0/pci@9 MB/PCIE5 EMP -
pci@500/pci@0/pci@c MB/NET0 OCC primary
pci@600/pci@0/pci@c MB/PCIE2 OCC primary
pci@600/pci@0/pci@9 MB/PCIE3 OCC primary
pci@700/pci@0/pci@c MB/PCIE6 OCC primary
pci@700/pci@0/pci@9 MB/PCIE7 EMP -
```

**2 Determine the device path of the boot disk, which needs to be retained.**

- For UFS file systems, run the `df /` command to determine the device path of the boot disk.

```
primary# df /
/ (/dev/dsk/c0t1d0s0): 1309384 blocks 457028 files
```

- For ZFS file systems, first run the `df /` command to determine the pool name, and then run the `zpool status` command to determine the device path of the boot disk.

```
primary# df /
/ (rpool/ROOT/s10s_u8wos_08a):245176332 blocks 245176332 files
primary# zpool status rpool
zpool status rpool
 pool: rpool
 state: ONLINE
 scrub: none requested
config:
```

| NAME     | STATE  | READ | WRITE | CKSUM |
|----------|--------|------|-------|-------|
| rpool    | ONLINE | 0    | 0     | 0     |
| c0t1d0s0 | ONLINE | 0    | 0     | 0     |

**3 Determine the physical device to which the block device is linked.**

The following example uses block device `c1t0d0s0`:

```
primary# ls -l /dev/dsk/c0t1d0s0
lrwxrwxrwx 1 root root 49 Oct 1 10:39 /dev/dsk/c0t1d0s0 ->
../.. /devices/pci@400/pci@0/pci@1/scsi@0/sd@1,0:a
```

In this example, the physical device for the primary domain's boot disk is connected to bus `pci@400`, which corresponds to the earlier listing of `pci_0`. This means that you *cannot* assign `pci_0` (`pci@400`) to another domain.

#### 4 Determine the network interface that is used by the system.

```
primary# dladm show-dev
vsw0 link: up speed: 1000 Mbps duplex: full
nxge0 link: up speed: 1000 Mbps duplex: full
nxge1 link: unknown speed: 0 Mbps duplex: unknown
nxge2 link: unknown speed: 0 Mbps duplex: unknown
nxge3 link: unknown speed: 0 Mbps duplex: unknown
```

Interfaces that are in the unknown state are not configured, so they are not used. In this example, the nxge0 interface is used.

#### 5 Determine the physical device to which the network interface is linked.

The following command uses the nxge0 network interface:

```
primary# ls -l /dev/nxge0
lrwxrwxrwx 1 root root 46 Oct 1 10:39 /dev/nxge0 ->
../devices/pci@500/pci@0/pci@c/network@0:nxge0
```

In this example, the physical device for the network interface used by the primary domain is under bus pci@500, which corresponds to the earlier listing of pci\_1. So, the other two buses, pci\_2 (pci@600) and pci\_3 (pci@700), can safely be assigned to other domains because they are not used by the primary domain.

If the network interface used by the primary domain was on a bus that you want to assign to another domain, the primary domain would need to be reconfigured to use a different network interface.

#### 6 Remove the buses that do not contain the boot disk or the network interface from the primary domain.

In this example, bus pci\_2 and bus pci\_3 are being removed from the primary domain. You might see a message from the ldm command that the primary domain is entering delayed reconfiguration mode.

```
primary# ldm remove-io pci_2 primary
primary# ldm remove-io pci_3 primary
```

#### 7 Save this configuration to the service processor.

In this example, the configuration is io-domain.

```
primary# ldm add-config io-domain
```

This configuration, io-domain, is also set as the next configuration to be used after the reboot.

---

**Note** – Currently, there is a limit of 8 configurations that can be saved on the SP, not including the factory-default configuration.

---

#### 8 Reboot the primary domain so that the change takes effect.

```
primary# shutdown -i6 -g0 -y
```

**9 Stop the domain to which you want to add the PCIe bus.**

The following example stops the ldg1 domain:

```
primary# ldm stop ldg1
```

**10 Add the available bus to the domain that needs direct access.**

The available bus is pci\_2 and the domain is ldg1.

```
primary# ldm add-io pci_2 ldg1
```

**11 Restart the domain so that the change takes affect.**

The following commands restart the ldg1 domain:

```
primary# ldm start ldg1
```

**12 Confirm that the correct bus is still assigned to the primary domain and the correct bus is assigned to domain ldg1.**

```
primary# ldm list-io
IO PSEUDONYM DOMAIN
-- -
pci@400 pci_0 primary
pci@500 pci_1 primary
pci@600 pci_2 ldg1
pci@700 pci_3

PCIE

pci@400/pci@0/pci@d MB/PCIE0 EMP -
pci@400/pci@0/pci@c MB/PCIE1 OCC primary
pci@400/pci@0/pci@l MB/HBA OCC primary
pci@500/pci@0/pci@d MB/PCIE4 EMP -
pci@500/pci@0/pci@9 MB/PCIE5 EMP -
pci@500/pci@0/pci@c MB/NET0 OCC primary
pci@600/pci@0/pci@c MB/PCIE2 UNK -
pci@600/pci@0/pci@9 MB/PCIE3 UNK -
pci@700/pci@0/pci@c MB/PCIE6 UNK -
pci@700/pci@0/pci@9 MB/PCIE7 UNK -
```

This output confirms that the PCIe buses pci\_0 and pci\_1 and the devices below them are assigned to domain primary, and that pci\_2 and its devices are assigned to ldg1.

# Assigning PCIe Endpoint Devices

Starting with the Oracle VM Server for SPARC 2.0 release and the Oracle Solaris 10 9/10 OS, you can assign an individual PCIe endpoint (or direct I/O-assignable) device to a domain. This use of PCIe endpoint devices increases the granularity of the device assignment to I/O domains. This capability is delivered by means of the direct I/O (DIO) feature.

The DIO feature enables you to create more I/O domains than the number of PCIe buses in a system. The possible number of I/O domains is now limited only by the number of PCIe endpoint devices.

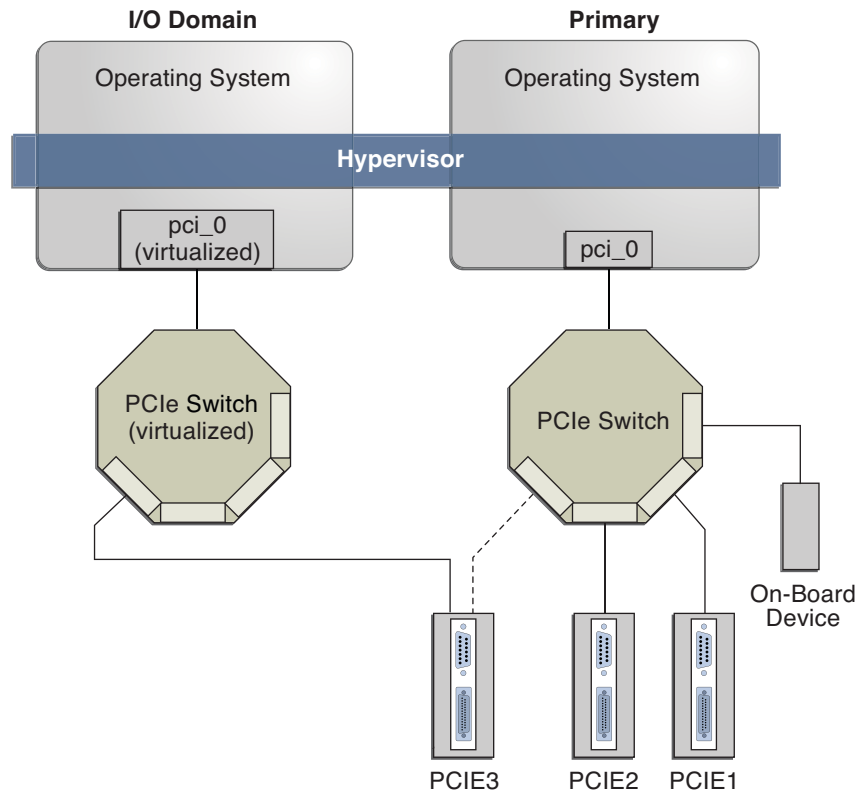
A PCIe endpoint device can be one of the following:

- A PCIe card in a slot
- An on-board PCIe device that is identified by the platform

The following diagram shows that the PCIe endpoint device, PCIE3, is assigned to an I/O domain. Both bus `pci_0` and the switch in the I/O domain are virtual. The PCIE3 endpoint device is no longer accessible in the primary domain.

In the I/O domain, the `pci_0` block and the switch are a virtual root complex and a virtual PCIe switch, respectively. This block and switch are very similar to the `pci_0` block and the switch in the primary domain. In the primary domain, the devices in slot PCIE3 are a shadow form of the original devices and are identified as `SUNW, assigned`.

FIGURE 6-2 Assigning a PCIe Endpoint Device to an I/O Domain



Use the `ldm list-io` command to list the PCIe endpoint devices.

Though the DIO feature permits any PCIe card in a slot to be assigned to an I/O domain, only certain PCIe cards are supported. See [“Direct I/O Hardware and Software Requirements” in Oracle VM Server for SPARC 2.1 Release Notes](#).

---

**Note** – PCIe cards that have a switch or bridge are not supported. PCIe function-level assignment is also not supported. Assigning an unsupported PCIe card to an I/O domain might result in unpredictable behavior.

---

The following are a few important details about the DIO feature:

- This feature is enabled only when all the software requirements are met. See [“Direct I/O Hardware and Software Requirements” in Oracle VM Server for SPARC 2.1 Release Notes](#).
- Only PCIe endpoints that are connected to a PCIe bus assigned to the primary domain can be assigned to another domain with the DIO feature.
- I/O domains using DIO have access to the PCIe endpoint devices only when the primary domain is running.
- Rebooting the primary domain affects I/O domains that have PCIe endpoint devices. See [“Rebooting the primary Domain” on page 74](#). The primary domain also has the following responsibilities:
  - Initializes the PCIe bus and manages the bus.
  - Handles all bus errors that are triggered by the PCIe endpoint devices that are assigned to I/O domains. Note that only the primary domain receives all PCIe bus-related errors.

## Direct I/O Hardware and Software Requirements

To successfully use the DIO feature, you must be running the appropriate software and assign only the PCIe cards that are supported by the DIO feature to I/O domains. For the hardware and software requirements, see [“Direct I/O Hardware and Software Requirements” in Oracle VM Server for SPARC 2.1 Release Notes](#).

---

**Note** – All PCIe cards that are supported on a platform are supported in the primary domain. See the documentation for your platform for the list of supported PCIe cards. However, *only* direct I/O-supported PCIe cards can be assigned to I/O domains.

---

## Direct I/O Limitations

For information about how to work around the following limitations, see [“Planning PCIe Endpoint Device Configuration” on page 73](#).

- A delayed reconfiguration is initiated when you assign or remove a PCIe endpoint device to or from the primary domain, which means that the changes are applied only after the primary domain reboots.

Rebooting the primary domain affects direct I/O, so carefully plan your direct I/O configuration changes to maximize the direct I/O-related changes to the primary domain and to minimize primary domain reboots.

- Assignment or removal of a PCIe endpoint device to any other domain is only permitted when that domain is either stopped or inactive.

## Planning PCIe Endpoint Device Configuration

Carefully plan ahead when you assign or remove PCIe endpoint devices to avoid primary domain downtime. The reboot of the primary domain not only affects the services that are available on the primary domain itself, but it also affects the I/O domains that have PCIe endpoint devices assigned. Though the changes to each I/O domain do not affect the other domains, planning ahead helps to minimize the consequences on the services that are provided by that domain.

The delayed reconfiguration is initiated the first time you assign or remove a device. As a result, you can continue to add or remove more devices and then reboot the primary domain only one time to make all the changes take effect.

For an example, see [“Create an I/O Domain by Assigning a PCIe Endpoint Device” on page 76](#).

The following describes the general steps you must take to plan and perform a DIO device configuration:

1. Understand and record your system hardware configuration.

Specifically, record information about the part numbers and other details of the PCIe cards in the system.

Use the `ldm list-io -l` and `prtdiag -v` commands to obtain the information and save it for future reference.

2. Determine which PCIe endpoint devices are required to be in the primary domain.

For example, determine the PCIe endpoint devices that provide access to the following:

- Boot disk device
- Network device
- Other devices that the primary domain offers as services

3. Remove all PCIe endpoint devices that you might use in I/O domains.

This step helps you to avoid performing subsequent reboot operations on the primary domain, as reboots affect I/O domains.

Use the `ldm rm-io` command to remove the PCIe endpoint devices. Use pseudonyms rather than device paths to specify the devices to the `rm-io` and `add-io` subcommands.

---

**Note** – Though the first removal of a PCIe endpoint device might initiate a delayed reconfiguration, you can continue to remove devices. After you have removed all the devices you want, you only need to reboot the primary domain one time to make all the changes take effect.

---

4. Save this configuration to the service processor (SP).

Use the `ldm add-config` command.

5. Reboot the primary domain to release the PCIe endpoint devices that you removed in Step 3.
6. Confirm that the PCIe endpoint devices you removed are no longer assigned to the primary domain.

Use the `ldm list-io -l` command to verify that the devices you removed appear as `SUNW,assigned-device` in the output.

7. Assign an available PCIe endpoint device to a guest domain to provide direct access to the physical device.

After you make this assignment, you can no longer migrate the guest domain to another physical system by means of the domain migration feature.

8. Add or remove a PCIe endpoint device to or from a guest domain.

Use the `ldm add-io` command.

Minimize the changes to I/O domains by reducing the reboot operations and by avoiding downtime of services offered by that domain.

9. (Optional) Make changes to the PCIe hardware.

See [“Making PCIe Hardware Changes” on page 75](#).

## Rebooting the primary Domain

The primary domain is the owner of the PCIe bus and is responsible for initializing and managing the bus. The primary domain must be active and running a version of the Oracle Solaris OS that supports the DIO feature. Shutting down, halting, or rebooting the primary domain interrupts access to the PCIe bus. When the PCIe bus is unavailable, the PCIe devices on that bus are affected and might become unavailable.

The behavior of I/O domains with PCIe endpoint devices is unpredictable when the primary domain is rebooted while those I/O domains are running. For instance, I/O domains with PCIe endpoint devices might panic during or after the reboot. Upon reboot of the primary domain, you would need to manually stop and start each domain.

To work around these issues, perform one of the following steps:

- Manually shut down any domains on the system that have PCIe endpoint devices assigned to them *before* you shut down the primary domain.

This step ensures that these domains are cleanly shut down before you shut down, halt, or reboot the primary domain.

To find all the domains that have PCIe endpoint devices assigned to them, run the `ldm list-io` command. This command enables you to list the PCIe endpoint devices that have been assigned to domains on the system. So, use this information to help you plan. For a detailed description of this command output, see the [ldm\(1M\)](#) man page.

For each domain found, stop the domain by running the `ldm stop` command.

- Configure a domain dependency relationship between the primary domain and the domains that have PCIe endpoint devices assigned to them.

This dependency relationship ensures that domains with PCIe endpoint devices are automatically restarted when the primary domain reboots for any reason.

Note that this dependency relationship forcibly resets those domains, and they cannot cleanly shut down. However, the dependency relationship does not affect any domains that were manually shut down.

```
ldm set-domain failure-policy=reset primary
ldm set-domain master=primary ldom
```

## Making PCIe Hardware Changes

The following steps help you avoid misconfiguring the PCIe endpoint assignments. For platform-specific information about installing and removing specific hardware, see the documentation for your platform.

- No action is required if you are installing a PCIe card into an empty slot. This PCIe card is automatically owned by the domain that owns the PCIe bus.  
To assign the new PCIe card to an I/O domain, use the `ldm rm-io` command to first remove the card from the primary domain. Then, use the `ldm add-io` command to assign the card to an I/O domain.
- No action is required if a PCIe card is removed from the system and assigned to the primary domain.
- To remove a PCIe card that is assigned to an I/O domain, first remove the device from the I/O domain. Then, add the device to the primary domain before you physically remove the device from the system.
- To replace a PCIe card that is assigned to an I/O domain, verify that the new card is supported by the DIO feature.

If so, no action is required to automatically assign the new card to the current I/O domain.

If not, first remove that PCIe card from the I/O domain by using the `ldm rm -io` command. Next, use the `ldm add -io` command to reassign that PCIe card to the primary domain. Then, physically replace the PCIe card you assigned to the primary domain with a different PCIe card. These steps enable you to avoid a configuration that is unsupported by the DIO feature.

## ▼ Create an I/O Domain by Assigning a PCIe Endpoint Device

Plan all DIO deployments ahead of time to minimize downtime.

For an example of adding a PCIe endpoint device to create an I/O domain, see [“Planning PCIe Endpoint Device Configuration” on page 73](#).

### 1 Identify and archive the devices that are currently installed on the system.

The output of the `ldm list -io -l` command shows how the I/O devices are currently configured. You can obtain more detailed information by using the `prtdiag -v` command.

---

**Note** – After the devices are assigned to I/O domains, the identity of the devices can only be determined in the I/O domains.

---

```
ldm list-io -l
IO PSEUDONYM DOMAIN
-- -
pci@400 pci_0 primary
pci@500 pci_1 primary

PCIIE PSEUDONYM STATUS DOMAIN

pci@400/pci@0/pci@c PCIIE1 EMP -
pci@400/pci@0/pci@9 PCIIE2 OCC primary
network@0
network@0,1
network@0,2
network@0,3
pci@400/pci@0/pci@d PCIIE3 OCC primary
SUNW,emlxs/fp/disk
SUNW,emlxs@0,1/fp/disk
SUNW,emlxs@0,1/fp@0,0
pci@400/pci@0/pci@8 MB/SASHBA OCC primary
scsi@0/tape
scsi@0/disk
scsi@0/sd@0,0
scsi@0/sd@1,0
pci@500/pci@0/pci@9 PCIIE0 EMP -
pci@500/pci@0/pci@d PCIIE4 OCC primary
network@0
network@0,1
```

```

pci@500/pci@0/pci@c PCIE5 OCC primary
 SUNW,qlc@0/fp/disk
 SUNW,qlc@0/fp@0,0
 SUNW,qlc@0,1/fp/disk
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c605dbab,0
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c6041434,0
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c6053652,0
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c6041b4f,0
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c605dbb3,0
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c60413bc,0
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c604167f,0
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c6041b3a,0
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c605dabf,0
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c60417a4,0
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c60416a7,0
 SUNW,qlc@0,1/fp@0,0/ssd@w21000011c60417e7,0
 SUNW,qlc@0,1/fp@0,0/ses@w215000c0ff082669,0
pci@500/pci@0/pci@8 MB/NET0 OCC primary
 network@0
 network@0,1
 network@0,2
 network@0,3

```

## 2 Determine the device path of the boot disk, which needs to be retained.

- For UFS file systems, run the `df /` command to determine the device path of the boot disk.

```

primary# df /
/ (/dev/dsk/c0t1d0s0): 1309384 blocks 457028 files

```

- For ZFS file systems, first run the `df /` command to determine the pool name, and then run the `zpool status` command to determine the device path of the boot disk.

```

primary# df /
/ (rpool/ROOT/s10s_u8wos_08a):245176332 blocks 245176332 files
primary# zpool status rpool
zpool status rpool
 pool: rpool
 state: ONLINE
 scrub: none requested
config:

```

| NAME     | STATE  | READ | WRITE | CKSUM |
|----------|--------|------|-------|-------|
| rpool    | ONLINE | 0    | 0     | 0     |
| c0t1d0s0 | ONLINE | 0    | 0     | 0     |

## 3 Determine the physical device to which the block device is linked.

The following example uses block device `c0t1d0s0`:

```

primary# ls -l /dev/dsk/c0t1d0s0
lrwxrwxrwx 1 root root 49 Jul 20 22:17 /dev/dsk/c0t1d0s0 ->
../../devices/pci@400/pci@0/pci@8/scsi@0/sd@0,0:a

```

In this example, the physical device for the primary domain's boot disk is connected to the PCIe endpoint device (`pci@400/pci@0/pci@8`), which corresponds to the listing of MB/SASHBA in Step 1. Removing this device will prevent the primary domain from booting, so do *not* remove this device from the primary domain.

#### 4 Determine the network interface that is used by the system.

```
ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
 inet 127.0.0.1 netmask ff000000
nxge0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 2
 inet 10.6.212.149 netmask fffffffe0 broadcast 10.6.213.255
 ether 0:21:28:4:27:cc
```

In this example, the nxge0 interface is used as the network interface for the primary domain.

#### 5 Determine the physical device to which the network interface is linked.

The following command uses the nxge0 network interface:

```
primary# ls -l /dev/nxge0
lrwxrwxrwx 1 root root 46 Jul 30 17:29 /dev/nxge0 ->
../devices/pci@500/pci@0/pci@8/network@0:nxge0
```

In this example, the physical device for the network interface used by the primary domain is connected to the PCIe endpoint device (pci@500/pci@0/pci@8), which corresponds to the listing of MB/NET0 in Step 1. So, you do *not* want to remove this device from the primary domain. You can safely assign all other PCIe devices to other domains because they are not used by the primary domain.

If the network interface used by the primary domain is on a bus that you want to assign to another domain, the primary domain would need to be reconfigured to use a different network interface.

#### 6 Remove the PCIe endpoint devices that you might use in I/O domains.

In this example, you can remove the PCIE2, PCIE3, PCIE4, and PCIE5 endpoint devices because they are not being used by the primary domain.

##### a. Remove the PCIe endpoint devices.




---

**Caution** – Do *not* remove the devices that are used in the primary domain.

If you mistakenly remove the wrong devices, use the `ldm cancel-op reconf primary` command to cancel the delayed reconfiguration on the primary domain.

---

You can remove multiple devices at one time to avoid multiple reboots.

```
ldm rm-io PCIE2 primary
```

```
Initiating a delayed reconfiguration operation on the primary domain.
All configuration changes for other domains are disabled until the primary
domain reboots, at which time the new configuration for the primary domain
will also take effect.
```

```
ldm rm-io PCIE3 primary
```

```

Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.

```

```
ldm rm-io PCIE4 primary
```

```

Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.

```

```
ldm rm-io PCIE5 primary
```

```

Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.

```

**b. Save the new configuration to the service processor (SP).**

The following command saves the configuration in a file called dio:

```
ldm add-config dio
```

**c. Reboot the system to reflect the removal of the PCIe endpoint devices.**

```
reboot -- -r
```

**7 Log in to the primary domain and verify that the PCIe endpoint devices are no longer assigned to the domain.**

```
ldm list-io
IO PSEUDONYM DOMAIN
-- -
pci@400 pci_0 primary
pci@500 pci_1 primary

PCIE

pci@400/pci@0/pci@c PCIE1 EMP -
pci@400/pci@0/pci@9 PCIE2 OCC -
pci@400/pci@0/pci@d PCIE3 OCC -
pci@400/pci@0/pci@8 MB/SASHBA OCC primary
pci@500/pci@0/pci@9 PCIE0 EMP -
pci@500/pci@0/pci@d PCIE4 OCC -
pci@500/pci@0/pci@c PCIE5 OCC -
pci@500/pci@0/pci@8 MB/NET0 OCC primary
```

---

**Note** – The `ldm list-io -l` output might show `SUNW,assigned-device` for the PCIe endpoint devices that were removed. Actual information is no longer available from the primary domain, but the domain to which the device is assigned has this information.

---

**8 Assign a PCIe endpoint device to a domain.**

**a. Add the PCIE2 device to the `ldg1` domain.**

```
ldm add-io PCIE2 ldg1
```

**b. Bind and start the `ldg1` domain.**

```
ldm bind ldg1
ldm start ldg1
LDom ldg1 started
```

**9 Log in to the `ldg1` domain and verify that the device is available for use.**

Use the `dladm show-dev` command to verify that the network device is available. Then, configure the network device for use in the domain.

```
dladm show-dev
```

|                    |                            |                       |                   |                              |
|--------------------|----------------------------|-----------------------|-------------------|------------------------------|
| <code>vnet0</code> | <code>link: up</code>      | <code>speed: 0</code> | <code>Mbps</code> | <code>duplex: unknown</code> |
| <code>nxge0</code> | <code>link: unknown</code> | <code>speed: 0</code> | <code>Mbps</code> | <code>duplex: unknown</code> |
| <code>nxge1</code> | <code>link: unknown</code> | <code>speed: 0</code> | <code>Mbps</code> | <code>duplex: unknown</code> |
| <code>nxge2</code> | <code>link: unknown</code> | <code>speed: 0</code> | <code>Mbps</code> | <code>duplex: unknown</code> |
| <code>nxge3</code> | <code>link: unknown</code> | <code>speed: 0</code> | <code>Mbps</code> | <code>duplex: unknown</code> |

# Using Virtual Disks

---

This chapter describes how to use virtual disks with Oracle VM Server for SPARC software.

This chapter covers the following topics:

- “Introduction to Virtual Disks” on page 81
- “Managing Virtual Disks” on page 82
- “Virtual Disk Identifier and Device Name” on page 84
- “Virtual Disk Appearance” on page 85
- “Virtual Disk Back End Options” on page 86
- “Virtual Disk Back End” on page 88
- “Configuring Virtual Disk Multipathing” on page 93
- “CD, DVD and ISO Images” on page 95
- “Virtual Disk Timeout” on page 99
- “Virtual Disk and SCSI” on page 99
- “Virtual Disk and the format Command” on page 100
- “Using ZFS With Virtual Disks” on page 100
- “Using Volume Managers in a Logical Domains Environment” on page 104

## Introduction to Virtual Disks

A virtual disk contains two components: the virtual disk itself as it appears in a guest domain, and the virtual disk back end, which is where data is stored and where virtual I/O ends up. The virtual disk back end is exported from a service domain by the virtual disk server (vds) driver. The vds driver communicates with the virtual disk client (vdc) driver in the guest domain through the hypervisor using a logical domain channel (LDC). Finally, a virtual disk appears as `/dev/[r]dsk/cXdYsZ` devices in the guest domain.

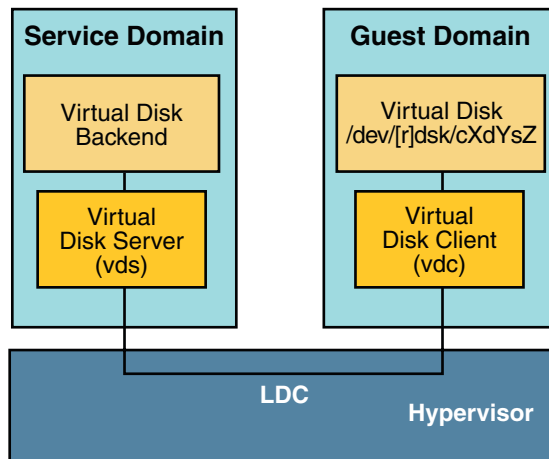
The virtual disk back end can be physical or logical. Physical devices can include the following:

- Physical disk or disk logical unit number (LUN)
- Physical disk slice

Logical devices can be any of the following:

- File on a file system, such as ZFS or UFS
- Logical volume from a volume manager, such as ZFS, VxVM, or Solaris Volume Manager
- Any disk pseudo device accessible from the service domain

FIGURE 7-1 Virtual Disks With Logical Domains



## Managing Virtual Disks

This section describes adding a virtual disk to a guest domain, changing virtual disk and timeout options, and removing a virtual disk from a guest domain. See “[Virtual Disk Back End Options](#)” on page 86 for a description of virtual disk options. See “[Virtual Disk Timeout](#)” on page 99 for a description of the virtual disk timeout.

### ▼ Add a Virtual Disk

- 1 Export the virtual disk back end from a service domain.

```
ldm add-vdsdev [-fq] [options={ro,slice,excl}] [mpgroup=mpgroup] \
 backend volume-name@service-name
```

- 2 Assign the back end to a guest domain.

```
ldm add-vdisk [timeout=seconds] [id=disk-id] disk-name volume-name@service-name ldom
```

You can specify an ID of a new virtual disk device by setting the `id` property. By default, ID values are automatically generated, so set this property if you need to match an existing device name in the OS. See “[Virtual Disk Identifier and Device Name](#)” on page 84.

---

**Note** – A back end is actually exported from the service domain and assigned to the guest domain when the guest domain (*ldom*) is bound.

---

## ▼ Export a Virtual Disk Back End Multiple Times

A virtual disk back end can be exported multiple times either through the same or different virtual disk servers. Each exported instance of the virtual disk back end can then be assigned to either the same or different guest domains.

When a virtual disk back end is exported multiple times, it should not be exported with the exclusive (`excl`) option. Specifying the `excl` option will only allow exporting the back end once. The back end can be safely exported multiple times as a read-only device with the `ro` option.




---

**Caution** – When a virtual disk back end is exported multiple times, applications running on guest domains and using that virtual disk are responsible for coordinating and synchronizing concurrent write access to ensure data coherency.

---

The following example describes how to add the same virtual disk to two different guest domains through the same virtual disk service.

- 1 **Export the virtual disk back end two times from a service domain by using the following commands.**

```
ldm add-vdsdev [options={ro,slice}] backend volume1@service-name
ldm add-vdsdev -f [options={ro,slice}] backend volume2@service-name
```

Note that the second `ldm add-vdsdev` command uses the `-f` option to force the second export of the back end. Use this option when using the same back-end path for both commands and when the virtual disk servers are located on the same service domain.

- 2 **Assign the exported back end to each guest domain by using the following commands.**

The *disk-name* can be different for `ldom1` and `ldom2`.

```
ldm add-vdisk [timeout=seconds] disk-name volume1@service-name ldom1
ldm add-vdisk [timeout=seconds] disk-name volume2@service-name ldom2
```

## ▼ Change Virtual Disk Options

For more information about virtual disk options, see [“Virtual Disk Back End Options” on page 86](#).

- **After a back end is exported from the service domain, you can change the virtual disk options by using the following command.**

```
ldm set-vdsdev options=[{ro,slice,excl}] volume-name@service-name
```

## ▼ Change the Timeout Option

For more information about virtual disk options, see [“Virtual Disk Back End Options”](#) on page 86.

- After a virtual disk is assigned to a guest domain, you can change the timeout of the virtual disk by using the following command.

```
ldm set-vdisk timeout=seconds disk-name ldom
```

## ▼ Remove a Virtual Disk

- 1 Remove a virtual disk from a guest domain by using the following command.

```
ldm rm-vdisk disk-name ldom
```

- 2 Stop exporting the corresponding back end from the service domain by using the following command.

```
ldm rm-vdsdev volume-name@service-name
```

# Virtual Disk Identifier and Device Name

When you use the `ldm add-vdisk` command to add a virtual disk to a domain, you can specify its device number by setting the `id` property.

```
ldm add-vdisk [id=disk-id] disk-name volume-name@service-name ldom
```

Each virtual disk of a domain has a unique device number that is assigned when the domain is bound. If a virtual disk is added with an explicit device number (by setting the `id` property), the specified device number is used. Otherwise, the system automatically assigns the lowest device number available. In that case, the device number assigned depends on how virtual disks were added to the domain. The device number eventually assigned to a virtual disk is visible in the output of the `ldm list-bindings` command when a domain is bound.

When a domain with virtual disks is running the Oracle Solaris OS, each virtual disk appears in the domain as a `c0dn` disk device, where *n* is the device number of the virtual disk.

In the following example, the `ldg1` domain has two virtual disks: `rootdisk` and `pdisk`. `rootdisk` has a device number of 0 (`disk@0`) and appears in the domain as the disk device `c0d0`. `pdisk` has a device number of 1 (`disk@1`) and appears in the domain as the disk device `c0d1`.

```
primary# ldm list-bindings ldg1
...
DISK
 NAME VOLUME TOUT DEVICE SERVER MPGROUP
```

|          |                         |        |         |
|----------|-------------------------|--------|---------|
| rootdisk | dsk_nevada@primary-vds0 | disk@0 | primary |
| pdisk    | c3t40d1@primary-vds0    | disk@1 | primary |
| ...      |                         |        |         |



**Caution** – If a device number is not explicitly assigned to a virtual disk, its device number can change when the domain is unbound and is later bound again. In that case, the device name assigned by the OS running in the domain can also change and break the existing configuration of the system. This might happen, for example, when a virtual disk is removed from the configuration of the domain.

## Virtual Disk Appearance

When a back end is exported as a virtual disk, it can appear in the guest domain either as a full disk or as a single-slice disk. The way it appears depends on the type of the back end and on the options used to export it.

### Full Disk

When a back end is exported to a domain as a full disk, it appears in that domain as a regular disk with 8 slices (`s0` to `s7`). Such a disk is visible with the `format(1M)` command. The disk's partition table can be changed using either the `fmthard(1M)` or `format(1M)` command.

A full disk is also visible to the OS installation software and can be selected as a disk onto which the OS can be installed.

Any back end can be exported as a full disk except physical disk slices that can only be exported as single-slice disks.

### Single-Slice Disk

When a back end is exported to a domain as a single-slice disk, it appears in that domain as a regular disk with 8 slices (`s0` to `s7`). However, only the first slice (`s0`) is usable. Such a disk is visible with the `format(1M)` command, but the disk's partition table cannot be changed.

A single-slice disk is also visible from the OS installation software and can be selected as a disk onto which you can install the OS. In that case, if you install the OS using the UNIX File System (UFS), then only the root partition (`/`) must be defined, and this partition must use all the disk space.

Any back end can be exported as a single-slice disk except physical disks that can only be exported as full disks.

---

**Note** – Prior to the Oracle Solaris 10 10/08 OS release, a single-slice disk appeared as a disk with a single partition (`s0`). Such a disk was not visible with the `format(1M)` command. The disk also was not visible from the OS installation software and could not be selected as a disk device onto which the OS could be installed.

---

## Virtual Disk Back End Options

Different options can be specified when exporting a virtual disk back end. These options are indicated in the `options=` argument of the `ldm add -vdsdev` command as a comma separated list. The valid options are: `ro`, `slice`, and `excl`.

### Read-only (ro) Option

The read-only (`ro`) option specifies that the back end is to be exported as a read-only device. In that case, the virtual disk assigned to the guest domain can only be accessed for read operations, and any write operation to the virtual disk will fail.

### Exclusive (excl) Option

The exclusive (`excl`) option specifies that the back end in the service domain has to be opened exclusively by the virtual disk server when it is exported as a virtual disk to another domain. When a back end is opened exclusively, it is not accessible by other applications in the service domain. This prevents the applications running in the service domain from inadvertently using a back end that is also being used by a guest domain.

---

**Note** – Some drivers do not honor the `excl` option and will disallow some virtual disk back ends from being opened exclusively. The `excl` option is known to work with physical disks and slices, but the option does not work with files. It may or may not work with pseudo devices, such as disk volumes. If the driver of the back end does not honor the exclusive open, the back end `excl` option is ignored, and the back end is not opened exclusively.

---

Because the `excl` option prevents applications running in the service domain from accessing a back end exported to a guest domain, do not set the `excl` option in the following situations:

- When guest domains are running, if you want to be able to use commands such as `format(1M)` or `luxadm(1M)` to manage physical disks, then do not export these disks with the `excl` option.
- When you export a Solaris Volume Manager volume, such as a RAID or a mirrored volume, do not set the `excl` option. Otherwise, this can prevent Solaris Volume Manager from starting some recovery operation in case a component of the RAID or mirrored volume fails. See [“Using Virtual Disks on Top of Solaris Volume Manager” on page 105](#) for more information.
- If the Veritas Volume Manager (VxVM) is installed in the service domain and Veritas Dynamic Multipathing (VxDMP) is enabled for physical disks, then physical disks have to be exported without the (non-default) `excl` option. Otherwise, the export fails, because the virtual disk server (`vds`) is unable to open the physical disk device. See [“Using Virtual Disks When VxVM Is Installed” on page 106](#) for more information.
- If you are exporting the same virtual disk back end multiple times from the same virtual disk service, see [“Export a Virtual Disk Back End Multiple Times” on page 83](#) for more information.

By default, the back end is opened non-exclusively. That way the back end still can be used by applications running in the service domain while it is exported to another domain. Note that this is a new behavior starting with the Oracle Solaris 10 5/08 OS release. Prior to the Oracle Solaris 10 5/08 OS release, disk back ends were always opened exclusively, and it was not possible to have a back end opened non-exclusively.

## Slice (slice) Option

A back end is normally exported either as a full disk or as a single-slice disk depending on its type. If the `slice` option is specified, then the back end is forcibly exported as a single-slice disk.

This option is useful when you want to export the raw content of a back end. For example, if you have a ZFS or Solaris Volume Manager volume where you have already stored data and you want your guest domain to access this data, then you should export the ZFS or Solaris Volume Manager volume using the `slice` option.

For more information about this option, see [“Virtual Disk Back End” on page 88](#).

# Virtual Disk Back End

The virtual disk back end is the location where data of a virtual disk are stored. The back end can be a disk, a disk slice, a file, or a volume, such as ZFS, Solaris Volume Manager, or VxVM. A back end appears in a guest domain either as a full disk or as single-slice disk, depending on whether the `slice` option is set when the back end is exported from the service domain. By default, a virtual disk back end is exported non-exclusively as a readable-writable full disk.

## Physical Disk or Disk LUN

A physical disk or disk LUN is always exported as a full disk. In that case, virtual disk drivers (`vds` and `vdc`) forward I/O from the virtual disk and act as a pass-through to the physical disk or disk LUN.

A physical disk or disk LUN is exported from a service domain by exporting the device that corresponds to the slice 2 (`s2`) of that disk without setting the `slice` option. If you export the slice 2 of a disk with the `slice` option, only this slice is exported and not the entire disk.

### ▼ Export a Physical Disk as a Virtual Disk

#### 1 Export a physical disk as a virtual disk.

For example, to export the physical disk `c1t48d0` as a virtual disk, you must export slice 2 of that disk (`c1t48d0s2`).

```
primary# ldm add-vdsdev /dev/dsk/c1t48d0s2 c1t48d0@primary-vds0
```

#### 2 Assign the disk to a guest domain.

For example, assign the disk (`pdisk`) to guest domain `ldg1`.

```
primary# ldm add-vdisk pdisk c1t48d0@primary-vds0 ldg1
```

#### 3 After the guest domain is started and running the Oracle Solaris OS, verify that the disk is accessible and is a full disk.

A full disk is a regular disk that has eight (8) slices.

For example, the disk being checked is `c0d1`.

```
ldg1# ls -l /dev/dsk/c0d1s*
/dev/dsk/c0d1s0
/dev/dsk/c0d1s1
/dev/dsk/c0d1s2
/dev/dsk/c0d1s3
/dev/dsk/c0d1s4
/dev/dsk/c0d1s5
/dev/dsk/c0d1s6
/dev/dsk/c0d1s7
```

## Physical Disk Slice

A physical disk slice is always exported as a single-slice disk. In that case, virtual disk drivers (vds and vdc) forward I/O from the virtual disk and act as a pass-through to the physical disk slice.

A physical disk slice is exported from a service domain by exporting the corresponding slice device. If the device is different from slice 2 then it is automatically exported as a single-slice disk whether or not you specify the `slice` option. If the device is the slice 2 of the disk, you must set the `slice` option to export only slice 2 as a single-slice disk; otherwise, the entire disk is exported as full disk.

### ▼ Export a Physical Disk Slice as a Virtual Disk

#### 1 Export a slice of a physical disk as a virtual disk.

For example, to export slice 0 of the physical disk `c1t57d0` as a virtual disk, you must export the device that corresponds to that slice (`c1t57d0s0`) as follows.

```
primary# ldm add-vdsdev /dev/dsk/c1t57d0s0 c1t57d0s0@primary-vds0
```

You do not need to specify the `slice` option, because a slice is always exported as a single-slice disk.

#### 2 Assign the disk to a guest domain.

For example, assign the disk (`pslice`) to guest domain `ldg1`.

```
primary# ldm add-vdisk pslice c1t57d0s0@primary-vds0 ldg1
```

#### 3 After the guest domain is started and running the Oracle Solaris OS, you can list the disk (`c0d13`, for example) and see that the disk is accessible.

```
ldg1# ls -l /dev/dsk/c0d13s*
/dev/dsk/c0d13s0
/dev/dsk/c0d13s1
/dev/dsk/c0d13s2
/dev/dsk/c0d13s3
/dev/dsk/c0d13s4
/dev/dsk/c0d13s5
/dev/dsk/c0d13s6
/dev/dsk/c0d13s7
```

Although there are 8 devices, because the disk is a single-slice disk, only the first slice (`s0`) is usable.

## ▼ Export Slice 2

- To export slice 2 (disk `c1t57d0s2`, for example) you must specify the `slice` option; otherwise, the entire disk is exported.

```
ldm add-vdsdev options=slice /dev/dsk/c1t57d0s2 c1t57d0s2@primary-vds0
```

## File and Volume

A file or volume (for example from ZFS or Solaris Volume Manager) is exported either as a full disk or as single-slice disk depending on whether or not the `slice` option is set.

### File or Volume Exported as a Full Disk

If you do not set the `slice` option, a file or volume is exported as a full disk. In that case, virtual disk drivers (`vds` and `vdc`) forward I/O from the virtual disk and manage the partitioning of the virtual disk. The file or volume eventually becomes a disk image containing data from all slices of the virtual disk and the metadata used to manage the partitioning and disk structure.

When a blank file or volume is exported as full disk, it appears in the guest domain as an unformatted disk; that is, a disk with no partition. Then you need to run the `format(1M)` command in the guest domain to define usable partitions and to write a valid disk label. Any I/O to the virtual disk fails while the disk is unformatted.

---

**Note** – Prior to the Oracle Solaris 5/08 OS release, when a blank file was exported as a virtual disk, the system wrote a default disk label and created default partitioning. This is no longer the case with the Oracle Solaris 5/08 OS release, and you must run `format(1M)` in the guest domain to create partitions.

---

## ▼ Export a File as a Full Disk

- 1 From the service domain, create a file (`fdisk0` for example) to use as the virtual disk.

```
service# mkfile 100m /ldoms/domain/test/fdisk0
```

The size of the file defines the size of the virtual disk. This example creates a 100-megabyte blank file to get a 100-megabyte virtual disk.

- 2 From the control domain, export the file as a virtual disk.

```
primary# ldm add-vdsdev /ldoms/domain/test/fdisk0 fdisk0@primary-vds0
```

In this example, the `slice` option is not set, so the file is exported as a full disk.

### 3 From the control domain, assign the disk to a guest domain.

For example, assign the disk (fdisk) to guest domain ldg1.

```
primary# ldm add-vdisk fdisk fdisk0@primary-vds0 ldg1
```

### 4 After the guest domain is started and running the Oracle Solaris OS, verify that the disk is accessible and is a full disk.

A full disk is a regular disk with 8 slices.

The following example shows how to list the disk, c0d5, and verify that it is accessible and is a full disk.

```
ldg1# ls -l /dev/dsk/c0d5*
/dev/dsk/c0d5s0
/dev/dsk/c0d5s1
/dev/dsk/c0d5s2
/dev/dsk/c0d5s3
/dev/dsk/c0d5s4
/dev/dsk/c0d5s5
/dev/dsk/c0d5s6
/dev/dsk/c0d5s7
```

## File or Volume Exported as a Single-Slice Disk

If the `slice` option is set, then the file or volume is exported as a single-slice disk. In that case, the virtual disk has only one partition (`s0`), which is directly mapped to the file or volume back end. The file or volume only contains data written to the virtual disk with no extra data like partitioning information or disk structure.

When a file or volume is exported as a single-slice disk, the system simulates a fake disk partitioning which makes that file or volume appear as a disk slice. Because the disk partitioning is simulated, you do not create partitioning for that disk.

## ▼ Export a ZFS Volume as a Single-Slice Disk

### 1 Create a ZFS volume to use as a single-slice disk.

The following example shows how to create a ZFS volume, `zdisk0`, to use as a single-slice disk.

```
service# zfs create -V 100m ldoms/domain/test/zdisk0
```

The size of the volume defines the size of the virtual disk. This example creates a 100-megabyte volume to get a 100-megabyte virtual disk.

### 2 From the control domain, export the corresponding device to that ZFS volume, and set the `slice` option so that the volume is exported as a single-slice disk.

```
primary# ldm add-vdsdev options=slice /dev/zvol/dsk/ldoms/domain/test/zdisk0 \
zdisk0@primary-vds0
```

**3 From the control domain, assign the volume to a guest domain.**

The following shows how to assign the volume, `zdisk0`, to guest domain `ldg1`.

```
primary# ldm add-vdisk zdisk0 zdisk0@primary-vds0 ldg1
```

**4 After the guest domain is started and running the Oracle Solaris OS, you can list the disk (`c0d9`, for example) and see that the disk is accessible and is a single-slice disk (`s0`).**

```
ldg1# ls -l /dev/dsk/c0d9s*
/dev/dsk/c0d9s0
/dev/dsk/c0d9s1
/dev/dsk/c0d9s2
/dev/dsk/c0d9s3
/dev/dsk/c0d9s4
/dev/dsk/c0d9s5
/dev/dsk/c0d9s6
/dev/dsk/c0d9s7
```

## Exporting Volumes and Backward Compatibility

Prior to the Oracle Solaris 10 5/08 OS release, the `slice` option did not exist, and volumes were exported as single-slice disks. If you have a configuration exporting volumes as virtual disks and if you upgrade the system to the Oracle Solaris 10 5/08 OS, volumes are now exported as full disks instead of single-slice disks. To preserve the old behavior and to have your volumes exported as single-slice disks, you need to do either of the following:

- Use the `ldm set -vdsdev` command in Oracle VM Server for SPARC 2.1 software, and set the `slice` option for all volumes you want to export as single-slice disks. Refer to the [ldm\(1M\)](#) man page for more information about this command.
- Add the following line to the `/etc/system` file on the service domain.

```
set vds:vd_volume_force_slice = 1
```

---

**Note** – Setting this tunable forces the export of all volumes as single-slice disks, and you cannot export any volume as a full disk.

---

## Summary of How Different Types of Back Ends Are Exported

| Back End                 | No Slice Option                | Slice Option Set               |
|--------------------------|--------------------------------|--------------------------------|
| Disk (disk slice 2)      | Full disk <sup>1</sup>         | Single-slice disk <sup>2</sup> |
| Disk slice (not slice 2) | Single-slice disk <sup>3</sup> | Single-slice disk              |
| File                     | Full disk                      | Single-slice disk              |

<sup>1</sup> Export the entire disk.

<sup>2</sup> Export only slice 2

<sup>3</sup> A slice is always exported as a single-slice disk.

| Back End                                               | No Slice Option | Slice Option Set  |
|--------------------------------------------------------|-----------------|-------------------|
| Volume, including ZFS, Solaris Volume Manager, or VxVM | Full disk       | Single-slice disk |

## Guidelines for Exporting Files and Disk Slices as Virtual Disks

This section includes guidelines for exporting a file and a disk slice as a virtual disk.

### Using the Loopback File (lofi) Driver

It is possible to use the loopback file (`lofi`) driver to export a file as a virtual disk. However, doing this adds an extra driver layer and impacts performance of the virtual disk. Instead, you can directly export a file as a full disk or as a single-slice disk. See [“File and Volume” on page 90](#).

### Directly or Indirectly Exporting a Disk Slice

To export a slice as a virtual disk either directly or indirectly (for example through a Solaris Volume Manager volume), ensure that the slice does not start on the first block (block 0) of the physical disk by using the `prtvtoc(1M)` command.

If you directly or indirectly export a disk slice which starts on the first block of a physical disk, you might overwrite the partition table of the physical disk and make all partitions of that disk inaccessible.

# Configuring Virtual Disk Multipathing

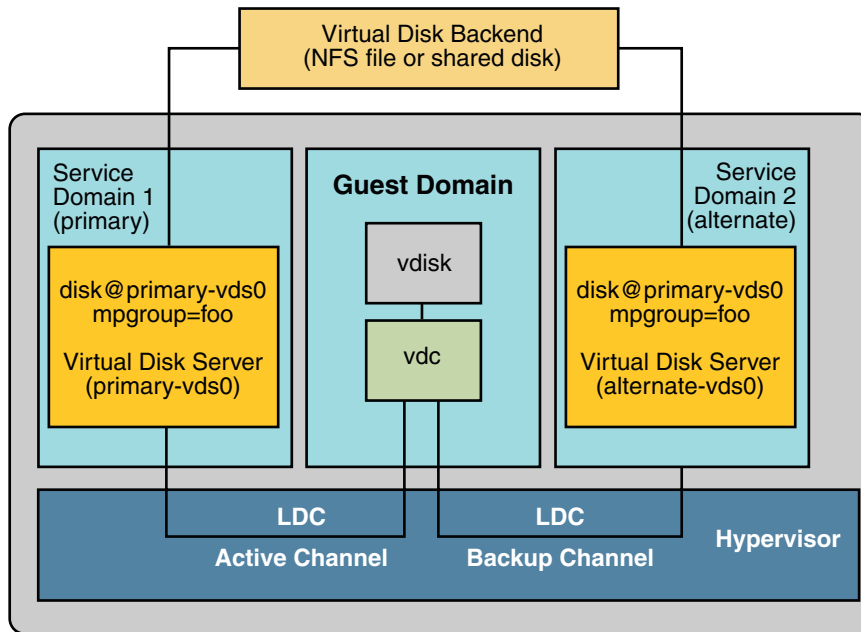
*Virtual disk multipathing* enables you to configure a virtual disk on a guest domain to access its back-end storage by more than one path. The paths lead through different service domains that provide access to the same back-end storage, such as a disk LUN. This feature enables a virtual disk in a guest domain to remain accessible even if one of the service domains goes down. For example, you might set up a virtual disk multipathing configuration to access a file on a network file system (NFS) server. Or, you can use this configuration to access a LUN from shared storage that is connected to more than one service domain. So, when the guest domain accesses the virtual disk, the virtual disk driver goes through one of the service domains to access the back-end storage. If the virtual disk driver cannot connect to the service domain, the virtual disk attempts to reach the back-end storage through a different service domain.

**Note** – Starting with the Oracle VM Server for SPARC 2.0 release, the virtual disk multipathing feature can detect when the service domain cannot access the back-end storage. In such an instance, the virtual disk driver attempts to access the back-end storage by another path.

To enable virtual disk multipathing, you must export a virtual disk back end from each service domain and add the virtual disk to the same multipathing group (mpgroup). The mpgroup is identified by a name and is configured when you export the virtual disk back end.

The following figure shows a virtual disk multipathing configuration, which is used as an example in the procedure “[Configure Virtual Disk Multipathing](#)” on page 94. In this example, a multipathing group named `foo` is used to create a virtual disk, whose back end is accessible from two service domains: `primary` and `alternate`.

FIGURE 7-2 Configuring Virtual Disk Multipathing



## ▼ Configure Virtual Disk Multipathing

- 1 Export the virtual disk back end from the **primary** service domain.

```
ldm add-vdsdev mpgroup=foo backend-path1 volume@primary-vds0
```

where *backend-path1* is the path to the virtual disk back end from the primary domain.

- 2 Export the same virtual disk back end from the **alternate** service domain.

```
ldm add-vdsdev mpgroup=foo backend-path2 volume@alternate-vds0
```

where *backend-path2* is the path to the virtual disk back end from the alternate domain.

---

**Note** – *backend-path1* and *backend-path2* are paths to the same virtual disk back end, but from two different domains (*primary* and *alternate*). These paths might be the same or different, depending on the configuration of the *primary* and *alternate* domains. The *volume* name is a user choice. It might be the same or different for both commands.

---

### 3 Export the virtual disk to the guest domain.

```
ldm add-vdisk disk-name volume@primary-vds0 ldom
```

---

**Note** – Although the virtual disk back end is exported several times through different service domains, you assign only one virtual disk to the guest domain and associate it with the virtual disk back end through any of the service domains.

---

## More Information Result of Virtual Disk Multipathing

After you configure the virtual disk with multipathing and start the guest domain, the virtual disk accesses its back end through the service domain it has been associated with (the *primary* domain in this example). If this service domain becomes unavailable, then the virtual disk tries to access its back end through a difference service domain that is part of the same multipathing group.




---

**Caution** – When defining a multipathing group (*mpgroup*), ensure that the virtual disk back ends that are part of the same *mpgroup* are effectively the same virtual disk back end. If you add different back ends into the same *mpgroup*, you might see some unexpected behavior, and you can potentially lose or corrupt data stored on the back ends.

---

## CD, DVD and ISO Images

You can export a compact disc (CD) or digital versatile disc (DVD) the same way you export any regular disk. To export a CD or DVD to a guest domain, export slice 2 of the CD or DVD device as a full disk; that is, without the *slice* option.

---

**Note** – You cannot export the CD or DVD drive itself; you only can export the CD or DVD that is inside the CD or DVD drive. Therefore, a CD or DVD must be present inside the drive before you can export it. Also, to be able to export a CD or DVD, that CD or DVD cannot be in use in the service domain. In particular, the Volume Management file system, *volfs(7FS)* service must not use the CD or DVD. See “[Export a CD or DVD From the Service Domain to the Guest Domain](#)” on page 96 for instructions on how to remove the device from use by *volfs*.

---

If you have an International Organization for Standardization (ISO) image of a CD or DVD stored in file or on a volume, and export that file or volume as a full disk then it appears as a CD or DVD in the guest domain.

When you export a CD, DVD, or an ISO image, it automatically appears as a read-only device in the guest domain. However, you cannot perform any CD control operations from the guest domain; that is, you cannot start, stop, or eject the CD from the guest domain. If the exported CD, DVD, or ISO image is bootable, the guest domain can be booted on the corresponding virtual disk.

For example, if you export a Oracle Solaris OS installation DVD, you can boot the guest domain on the virtual disk that corresponds to that DVD and install the guest domain from that DVD. To do so, when the guest domain reaches the ok prompt, use the following command.

```
ok boot /virtual-devices@100/channel-devices@200/disk@n:f
```

Where *n* is the index of the virtual disk representing the exported DVD.

---

**Note** – If you export a Oracle Solaris OS installation DVD and boot a guest domain on the virtual disk that corresponds to that DVD to install the guest domain, then you cannot change the DVD during the installation. So you might need to skip any step of the installation requesting a different CD/DVD, or you will need to provide an alternate path to access this requested media.

---

## ▼ Export a CD or DVD From the Service Domain to the Guest Domain

- 1 From the service domain, check whether the volume management daemon, **vol**d(1M), is running and online.

```
service# svcs volfs
STATE STIME FMRI
online 12:28:12 svc:/system/filesystem/volfs:default
```

- 2 Do one of the following.

- If the volume management daemon is not running or online, go to Step 3.
- If the volume management daemon is running and online, as in the example in Step 1, do the following:

- a. Edit the **/etc/vold.conf** file and comment out the line starting with the following words.

```
use cdrom drive....
```

See the **vold.conf(4)** man page.

- b. Insert the CD or DVD in the CD or DVD drive.

c. From the service domain, restart the volume management file system service.

```
service# svcadm refresh volfs
service# svcadm restart volfs
```

3 From the service domain, find the disk path for the CD-ROM device.

```
service# cdrw -l
Looking for CD devices...
Node Connected Device Device type
-----+-----+-----+-----+-----+-----+-----+-----
/dev/rdisk/clt0d0s2 | MATSHITA CD-RW CW-8124 DZ13 | CD Reader/Writer
```

4 Export the CD or DVD disk device as a full disk.

```
primary# ldm add-vdsdev /dev/dsk/clt0d0s2 cdrom@primary-vds0
```

5 Assign the exported CD or DVD to the guest domain.

The following shows how to assign the exported CD or DVD to domain ldg1:

```
primary# ldm add-vdisk cdrom cdrom@primary-vds0 ldg1
```

**More Information** Exporting a CD or DVD Multiple Times

A CD or DVD can be exported multiple times and assigned to different guest domains. See [“Export a Virtual Disk Back End Multiple Times” on page 83](#) for more information.

## ▼ Export an ISO Image From the primary Domain to Install a Guest Domain

This procedure shows how to export an ISO image from the primary domain and use it to install a guest domain. This procedure assumes that both the primary domain and the guest domain are configured.

For example, the following `ldm list` shows that both the primary and `ldom1` domains are configured:

```
ldm list
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -n-cv SP 4 4G 0.3% 15m
ldom1 active -t--- 5000 4 1G 25% 8m
```

1 Add a virtual disk server device to export the ISO image.

In this example, the ISO image is `/export/images/sol-10-u8-ga-sparc-dvd.iso`.

```
ldm add-vdsdev /export/images/sol-10-u8-ga-sparc-dvd.iso dvd-iso@primary-vds0
```

## 2 Stop the guest domain.

In this example, the logical domain is `ldom1`.

```
ldm stop-domain ldom1
LDom ldom1 stopped
```

## 3 Add the virtual disk for the ISO image to the logical domain.

In this example, the logical domain is `ldom1`.

```
ldm add-vdisk s10-dvd dvd-iso@primary-vds0 ldom1
```

## 4 Restart the guest domain.

In this example, the logical domain is `ldom1`.

```
ldm start-domain ldom1
LDom ldom1 started
ldm list
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -n-cv SP 4 4G 0.4% 25m
ldom1 active -t--- 5000 4 1G 0.0% 0s
```

In this example, the `ldm list` command shows that the `ldom1` domain has just been started.

## 5 Connect to the guest domain.

```
telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Connecting to console "ldom1" in group "ldom1"
Press ~? for control options ..
```

## 6 Verify the existence of the ISO image as a virtual disk.

```
{0} ok show-disks
a) /virtual-devices@100/channel-devices@200/disk@1
b) /virtual-devices@100/channel-devices@200/disk@0
q) NO SELECTION
Enter Selection, q to quit: q
```

In this example, the newly added device is

```
/virtual-devices@100/channel-devices@200/disk@1.
```

## 7 Boot the guest domain to install from the ISO image.

In this example, boot from the `f` slice of the

```
/virtual-devices@100/channel-devices@200/disk@1 disk.
```

```
{0} ok boot /virtual-devices@100/channel-devices@200/disk@1:f
```

## Virtual Disk Timeout

By default, if the service domain providing access to a virtual disk back end is down, all I/O from the guest domain to the corresponding virtual disk is blocked. The I/O automatically is resumed when the service domain is operational and is servicing I/O requests to the virtual disk back end.

However, there are some cases when file systems or applications might not want the I/O operation to block, but for it to fail and report an error if the service domain is down for too long. It is now possible to set a connection timeout period for each virtual disk, which can then be used to establish a connection between the virtual disk client on a guest domain and the virtual disk server on the service domain. When that timeout period is reached, any pending I/O and any new I/O will fail as long as the service domain is down and the connection between the virtual disk client and server is not reestablished.

This timeout can be set by doing one of the following:

- Using the `ldm add-vdisk` command.
 

```
ldm add-vdisk timeout=seconds disk-name volume-name@service-name ldom
```
- Using the `ldm set-vdisk` command.
 

```
ldm set-vdisk timeout=seconds disk-name ldom
```

Specify the timeout in seconds. If the timeout is set to 0, the timeout is disabled and I/O is blocked while the service domain is down (this is the default setting and behavior).

Alternatively, the timeout can be set by adding the following line to the `/etc/system` file on the guest domain.

```
set vdc:vdc_timeout=seconds
```

---

**Note** – If this tunable is set, it overwrites any timeout setting done using the `ldm` CLI. Also, the tunable sets the timeout for all virtual disks in the guest domain.

---

## Virtual Disk and SCSI

If a physical SCSI disk or LUN is exported as a full disk, the corresponding virtual disk supports the user SCSI command interface, [uscsi\(7I\)](#) and multihost disk control operations [mhd\(7i\)](#). Other virtual disks, such as virtual disks having a file or a volume as a back end, do not support these interfaces.

As a consequence, applications or product features using SCSI commands (such as Solaris Volume Manager `metaset`, or Oracle Solaris Cluster shared devices) can be used in guest domains only with virtual disks having a physical SCSI disk as a back end.

---

**Note** – SCSI operations are effectively executed by the service domain, which manages the physical SCSI disk or LUN used as a virtual disk back end. In particular, SCSI reservations are done by the service domain. Therefore, applications running in the service domain and in guest domains should not issue SCSI commands to the same physical SCSI disks; otherwise, this can lead to an unexpected disk state.

---

## Virtual Disk and the format Command

The `format(1M)` command recognizes all virtual disks that are present in a domain. However, for virtual disks that are exported as single-slice disks, the `format` command cannot change the partition table of the virtual disk. Commands such as `label` will fail unless you try to write a disk label similar to the one that is already associated with the virtual disk.

Virtual disks whose back ends are SCSI disks support all `format(1M)` subcommands. Virtual disks whose back ends are not SCSI disks do not support some `format(1M)` subcommands, such as `repair` and `defect`. In that case, the behavior of `format(1M)` is similar to the behavior of Integrated Drive Electronics (IDE) disks.

## Using ZFS With Virtual Disks

This section describes using the Zettabyte File System (ZFS) to store virtual disk back ends exported to guest domains. ZFS provides a convenient and powerful solution to create and manage virtual disk back ends. ZFS enables:

- Storing disk images in ZFS volumes or ZFS files
- Using snapshots to backup disk images
- Using clones to duplicate disk images and provision additional domains

Refer to the *Oracle Solaris ZFS Administration Guide* for more information about using the ZFS.

In the following descriptions and examples, the primary domain is also the service domain where disk images are stored.

## Configuring a ZFS Pool in a Service Domain

To store the disk images, first create a ZFS storage pool in the service domain. For example, this command creates the ZFS storage pool `ldmpool` containing the disk `c1t50d0` in the primary domain.

```
primary# zpool create ldmpool c1t50d0
```

## Storing Disk Images With ZFS

The following command creates a disk image for guest domain `ldg1`. A ZFS file system for this guest domain is created, and all disk images of this guest domain will be stored on that file system.

```
primary# zfs create ldmpool/ldg1
```

Disk images can be stored on ZFS volumes or ZFS files. Creating a ZFS volume, whatever its size, is quick using the `zfs create -V` command. On the other hand, ZFS files have to be created using the `mkfile` command. The command can take some time to complete, especially if the file to create is quite large, which is often the case when creating a disk image.

Both ZFS volumes and ZFS files can take advantage of ZFS features such as the snapshot and clone features, but a ZFS volume is a pseudo device while a ZFS file is a regular file.

If the disk image is to be used as a virtual disk onto which an OS is installed, the disk image must be large enough to accommodate the OS installation requirements. This size depends on the version of the OS and on the type of installation performed. If you install the Oracle Solaris OS, you can use a disk size of 20 Gbytes to accommodate any type of installation of any version of the Oracle Solaris OS.

## Examples of Storing Disk Images With ZFS

The following examples:

1. Create a 20-gigabyte image on a ZFS volume or file.
2. Export the ZFS volume or file as a virtual disk. The syntax to export a ZFS volume or file is the same, but the path to the back end is different.
3. Assign the exported ZFS volume or file to a guest domain.

When the guest domain is started, the ZFS volume or file appears as a virtual disk on which the Oracle Solaris OS can be installed.

### ▼ Create a Disk Image Using a ZFS Volume

- For example, create a 20-gigabyte disk image on a ZFS volume.

```
primary# zfs create -V 20gb ldmpool/ldg1/disk0
```

### ▼ Create a Disk Image Using a ZFS File

- For example, create a 20-gigabyte disk image on a ZFS volume.

```
primary# zfs create ldmpool/ldg1/disk0
primary# mkfile 20g /ldmpool/ldg1/disk0/file
```

## ▼ Export the ZFS Volume

- Export the ZFS volume as a virtual disk.

```
primary# ldm add-vdsdev /dev/zvol/dsk/ldmpool/ldg1/disk0 ldg1_disk0@primary-vds0
```

## ▼ Export the ZFS File

- Export the ZFS file as a virtual disk.

```
primary# ldm add-vdsdev /ldmpool/ldg1/disk0/file ldg1_disk0@primary-vds0
```

## ▼ Assign the ZFS Volume or File to a Guest Domain

- Assign the ZFS volume or file to a guest domain; in this example, `ldg1`.

```
primary# ldm add-vdisk disk0 ldg1_disk0@primary-vds0 ldg1
```

# Creating a Snapshot of a Disk Image

When your disk image is stored on a ZFS volume or on a ZFS file, you can create snapshots of this disk image by using the ZFS snapshot command.

Before you create a snapshot of the disk image, ensure that the disk is not currently in use in the guest domain to ensure that data currently stored on the disk image are coherent. There are several ways to ensure that a disk is not in use in a guest domain. You can either:

- Stop and unbind the guest domain. This is the safest solution, and this is the only solution available if you want to create a snapshot of a disk image used as the boot disk of a guest domain.
- Alternatively, you can unmount any slices of the disk you want to snapshot used in the guest domain, and ensure that no slice is in use the guest domain.

In this example, because of the ZFS layout, the command to create a snapshot of the disk image is the same whether the disk image is stored on a ZFS volume or on a ZFS file.

## ▼ Create a Snapshot of a Disk Image

- Create a snapshot of the disk image that was created for the `ldg1` domain, for example.

```
primary# zfs snapshot ldmpool/ldg1/disk0@version_1
```

## Using Clone to Provision a New Domain

Once you have created a snapshot of a disk image, you can duplicate this disk image by using the ZFS clone command. Then the cloned image can be assigned to another domain. Cloning a boot disk image quickly creates a boot disk for a new guest domain without having to perform the entire Oracle Solaris OS installation process.

For example, if the `disk0` created was the boot disk of domain `ldg1`, do the following to clone that disk to create a boot disk for domain `ldg2`.

```
primary# zfs create ldmpool/ldg2
primary# zfs clone ldmpool/ldg1/disk0@version_1 ldmpool/ldg2/disk0
```

Then `ldmpool/ldg2/disk0` can be exported as a virtual disk and assigned to the new `ldg2` domain. The domain `ldg2` can directly boot from that virtual disk without having to go through the OS installation process.

### Cloning a Boot Disk Image

When a boot disk image is cloned, the new image is exactly the same as the original boot disk, and it contains any information that has been stored on the boot disk before the image was cloned, such as the host name, the IP address, the mounted file system table, or any system configuration or tuning.

Because the mounted file system table is the same on the original boot disk image and on the cloned disk image, the cloned disk image has to be assigned to the new domain in the same order as it was on the original domain. For example, if the boot disk image was assigned as the first disk of the original domain, then the cloned disk image has to be assigned as the first disk of the new domain. Otherwise, the new domain is unable to boot.

If the original domain was configured with a static IP address, then a new domain using the cloned image starts with the same IP address. In that case, you can change the network configuration of the new domain by using the `sys-unconfig(1M)` command. To avoid this problem you can also create a snapshot of a disk image of an unconfigured system.

If the original domain was configured with the Dynamic Host Configuration Protocol (DHCP), then a new domain using the cloned image also uses DHCP. In that case, you do not need to change the network configuration of the new domain because it automatically receives an IP address and its network configuration as it boots.

---

**Note** – The host ID of a domain is not stored on the boot disk, but it is assigned by the Logical Domains Manager when you create a domain. Therefore, when you clone a disk image, the new domain does not keep the host ID of the original domain.

---

## ▼ Create a Snapshot of a Disk Image of an Unconfigured System

- 1 Bind and start the original domain.
- 2 Execute the `sys-unconfig` command.
- 3 After the `sys-unconfig` command completes, the domain halts.
- 4 Stop and unbind the domain; do *not* reboot it.
- 5 Take a snapshot of the domain boot disk image.

For example:

```
primary# zfs snapshot ldmpool/ldg1/disk0@unconfigured
```

At this point you have the snapshot of the boot disk image of an unconfigured system.

- 6 Clone this image to create a new domain which, when first booted, asks for the configuration of the system.

## Using Volume Managers in a Logical Domains Environment

This section describes using volume managers in a Logical Domains environment.

### Using Virtual Disks on Top of Volume Managers

Any Zettabyte File System (ZFS), Solaris Volume Manager, or Veritas Volume Manager (VxVM) volume can be exported from a service domain to a guest domain as a virtual disk. A volume can be exported either as a single-slice disk (if the `slice` option is specified with the `ldm add-vdsdev` command) or as a full disk.

---

**Note** – The remainder of this section uses a Solaris Volume Manager volume as an example. However, the discussion also applies to ZFS and VxVM volumes.

---

The following examples show how to export a volume as a single-slice disk.

The virtual disk in the guest domain (for example, `/dev/dsk/c0d2s0`) is directly mapped to the associated volume (for example, `/dev/md/dsk/d0`), and data stored onto the virtual disk from the guest domain are directly stored onto the associated volume with no extra metadata. So data stored on the virtual disk from the guest domain can also be directly accessed from the service domain through the associated volume.

### Examples

- If the Solaris Volume Manager volume `d0` is exported from the primary domain to `domain1`, then the configuration of `domain1` requires some extra steps.

```
primary# metainit d0 3 1 c2t70d0s6 1 c2t80d0s6 1 c2t90d0s6
primary# ldm add-vdsdev options=slice /dev/md/dsk/d0 vol3@primary-vds0
primary# ldm add-vdisk vdisk3 vol3@primary-vds0 domain1
```

- After `domain1` has been bound and started, the exported volume appears as `/dev/dsk/c0d2s0`, for example, and you can use it.

```
domain1# newfs /dev/rdisk/c0d2s0
domain1# mount /dev/dsk/c0d2s0 /mnt
domain1# echo test-domain1 > /mnt/file
```

- After `domain1` has been stopped and unbound, data stored on the virtual disk from `domain1` can be directly accessed from the primary domain through Solaris Volume Manager volume `d0`.

```
primary# mount /dev/md/dsk/d0 /mnt
primary# cat /mnt/file
test-domain1
```

### Using Virtual Disks on Top of Solaris Volume Manager

When a RAID or mirror Solaris Volume Manager volume is used as a virtual disk by another domain, then it has to be exported without setting the exclusive (`excl`) option. Otherwise, if there is a failure on one of the components of the Solaris Volume Manager volume, then the recovery of the Solaris Volume Manager volume using the `metareplace` command or using a hot spare does not start. The `metastat` command sees the volume as resynchronizing, but the resynchronization does not progress.

For example, `/dev/md/dsk/d0` is a RAID Solaris Volume Manager volume exported as a virtual disk with the `excl` option to another domain, and `d0` is configured with some hot-spare devices. If a component of `d0` fails, Solaris Volume Manager replaces the failing component with a hot spare and resynchronizes the Solaris Volume Manager volume. However, the resynchronization does not start. The volume is reported as resynchronizing, but the resynchronization does not progress.

```
metastat d0
d0: RAID
 State: Resyncing
 Hot spare pool: hsp000
 Interlace: 32 blocks
 Size: 20097600 blocks (9.6 GB)
Original device:
 Size: 20100992 blocks (9.6 GB)
Device Start Block Dbase State Reloc
c2t2d0s1 330 No Okay Yes
c4t12d0s1 330 No Okay Yes
/dev/dsk/c10t600C0FF00000000000015153295A4B10d0s1 330 No Resyncing Yes
```

In such a situation, the domain using the Solaris Volume Manager volume as a virtual disk has to be stopped and unbound to complete the resynchronization. Then the Solaris Volume Manager volume can be resynchronized using the `metasync` command.

```
metasync d0
```

## Using Virtual Disks When VxVM Is Installed

When the Veritas Volume Manager (VxVM) is installed on your system, and if Veritas Dynamic Multipathing (DMP) is enabled on a physical disk or partition you want to export as virtual disk, then you have to export that disk or partition without setting the (non-default) `excl` option. Otherwise, you receive an error in `/var/adm/messages` while binding a domain that uses such a disk.

```
vd_setup_vd(): ldi_open_by_name(/dev/dsk/c4t12d0s2) = errno 16
vds_add_vd(): Failed to add vdisk ID 0
```

You can check if Veritas DMP is enabled by checking multipathing information in the output of the command `vxdisk list`; for example:

```
vxdisk list Disk_3
Device: Disk_3
devicetag: Disk_3
type: auto
info: format=none
flags: online ready private autoconfig invalid
pubpaths: block=/dev/vx/dmp/Disk_3s2 char=/dev/vx/rdmp/Disk_3s2
guid: -
udid: SEAGATE%5FST336753LSUN36G%5FDISKS%5F3032333948303144304E0000
site: -
Multipathing information:
numpaths: 1
c4t12d0s2 state=enabled
```

Alternatively, if Veritas DMP is enabled on a disk or a slice that you want to export as a virtual disk with the `excl` option set, then you can disable DMP using the `vxddmpadm` command. For example:

```
vxddmpadm -f disable path=/dev/dsk/c4t12d0s2
```

## Using Volume Managers on Top of Virtual Disks

This section describes using volume managers on top of virtual disks.

### Using ZFS on Top of Virtual Disks

Any virtual disk can be used with ZFS. A ZFS storage pool (`zpool`) can be imported in any domain that sees all the storage devices that are part of this `zpool`, regardless of whether the domain sees all these devices as virtual devices or real devices.

## Using Solaris Volume Manager on Top of Virtual Disks

Any virtual disk can be used in the Solaris Volume Manager local disk set. For example, a virtual disk can be used for storing the Solaris Volume Manager metadvice state database, `metadb(1M)`, of the local disk set or for creating Solaris Volume Manager volumes in the local disk set.

Any virtual disk whose back end is a SCSI disk can be used in a Solaris Volume Manager shared disk set, `metaset(1M)`. Virtual disks whose back ends are not SCSI disks cannot be added into a Solaris Volume Manager share disk set. Trying to add a virtual disk whose back end is not a SCSI disk into a Solaris Volume Manager shared disk set fails with an error similar to the following.

```
metaset -s test -a c2d2
metaset: domain1: test: failed to reserve any drives
```

## Using VxVM on Top of Virtual Disks

For VxVM support in guest domains, refer to the VxVM documentation from Symantec.



# Using Virtual Networks

---

This chapter describes how to use a virtual network with Oracle VM Server for SPARC software, and covers the following topics:

- [“Introduction to a Virtual Network” on page 109](#)
- [“Virtual Switch” on page 110](#)
- [“Virtual Network Device” on page 110](#)
- [“Virtual Device Identifier and Network Interface Name” on page 114](#)
- [“Assigning MAC Addresses Automatically or Manually” on page 116](#)
- [“Using Network Adapters With Logical Domains” on page 119](#)
- [“Configuring Virtual Switch and Service Domain for NAT and Routing” on page 119](#)
- [“Configuring IPMP in a Logical Domains Environment” on page 121](#)
- [“Using VLAN Tagging” on page 127](#)
- [“Using NIU Hybrid I/O” on page 130](#)
- [“Using Link Aggregation With a Virtual Switch” on page 134](#)
- [“Configuring Jumbo Frames” on page 135](#)

## Introduction to a Virtual Network

A virtual network allows domains to communicate with each other without using any external physical networks. A virtual network also can allow domains to use the same physical network interface to access a physical network and communicate with remote systems. A virtual network is created by having a virtual switch to which you can connect virtual network devices.

## Virtual Switch

A virtual switch (vsw) is a component running in a service domain and managed by the virtual switch driver. A virtual switch can be connected to some guest domains to enable network communications between those domains. In addition, if the virtual switch is associated also with a physical network interface, then this allows network communications between guest domains and the physical network over the physical network interface. A virtual switch also has a network interface, `vswn`, which allows the service domain to communicate with the other domains connected to that virtual switch. It can be used like any regular network interface and configured with the `ifconfig` command.

---

**Note** – When a virtual switch is added to a service domain, its network interface is not created. So, by default, the service domain is unable to communicate with the guest domains connected to its virtual switch. To enable network communications between guest domains and the service domain, the network interface of the associated virtual switch must be created and configured in the service domain. See [“Enabling Networking Between the Control/Service Domain and Other Domains” on page 55](#) for instructions.

---

You can add a virtual switch to a domain, set options for a virtual switch, and remove a virtual switch by using the `ldm add-vsw`, `ldm set-vsw`, and `ldm rm-vsw` commands, respectively. For more information, see the [ldm\(1M\)](#) man page.

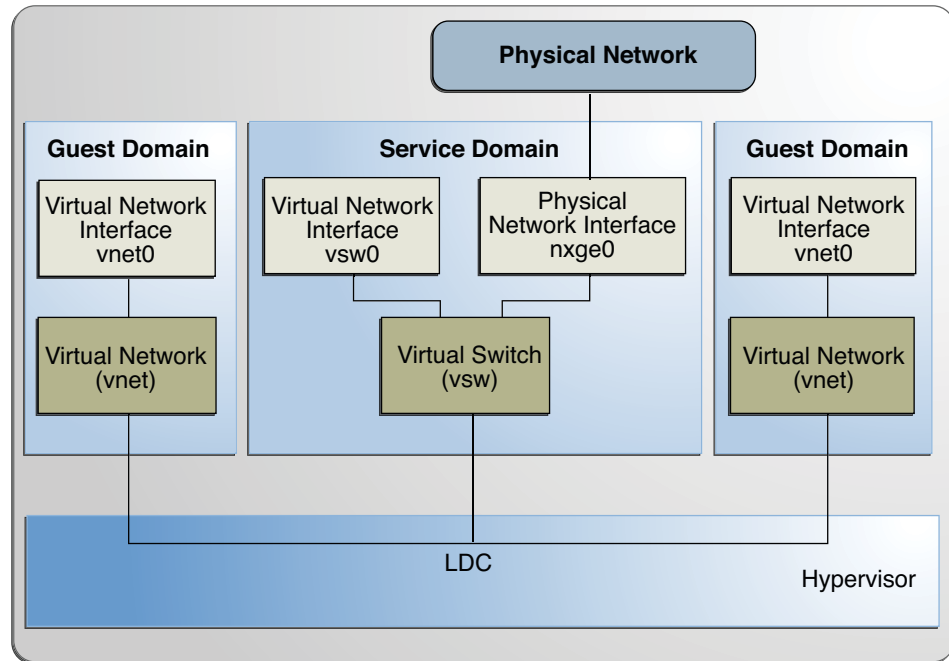
## Virtual Network Device

A virtual network (vnet) device is a virtual device that is defined in a domain connected to a virtual switch. A virtual network device is managed by the virtual network driver, and it is connected to a virtual network through the hypervisor using logical domain channels (LDCs).

A virtual network device can be used as a network interface with the name `vnetn`, which can be used like any regular network interface and configured with the `ifconfig` command.

You can add a virtual network device to a domain, set options for an existing virtual network device, and remove a virtual network device by using the `ldm add-vnet`, `ldm set-vnet`, and `ldm rm-vnet` commands, respectively. For more information, see the [ldm\(1M\)](#) man page.

FIGURE 8-1 Setting Up a Virtual Network



Following is an explanation for the example in [Figure 8-1](#).

- The virtual switch in the service domain is connected to the guest domains. This allows guest domains to communicate with each other.
- The virtual switch is also connected to the physical network interface `nxge0`. This allows guest domains to communicate with the physical network.
- The virtual switch network interface `vsw0` is created in the service domain, so this allows the two guest domains to communicate with the service domain.
- The virtual switch network interface `vsw0` in the service domain can be configured using the `ifconfig` command.
- The virtual network interfaces `vnet0` in the guest domains can be configured using the `ifconfig` command.

Basically the virtual switch behaves like a regular physical network switch and switches network packets between the different systems, such as guest domains, service domain, and physical network, to which it is connected.

## Inter-Vnet LDC Channels

Until the Oracle VM Server for SPARC 2.1 release, the Logical Domains Manager would assign LDC channels in the following manner:

- An LDC channel would be assigned between the virtual network devices and the virtual switch device.
- An LDC channel would be assigned between each pair of virtual network devices that are connected to the same virtual switch device (inter-vnet).

The inter-vnet LDC channels are configured so that virtual network devices can communicate directly to achieve high guest-to-guest communications performance. However, as the number of virtual network devices in a virtual switch device increases, the number of required LDC channels for inter-vnet communications increases exponentially.

Starting with the Oracle VM Server for SPARC 2.1 release, you can choose to enable or disable inter-vnet LDC channel allocation for all virtual network devices attached to a given virtual switch device. By disabling this allocation, you can reduce the consumption of LDC channels, which are limited in number.

Disabling this allocation is useful in the following situations:

- When guest-to-guest communications performance is not of primary importance
- When a large number of virtual network devices are required in a virtual switch device

By not assigning inter-vnet channels, more LDC channels are available for use to add more virtual I/O devices to a guest domain.

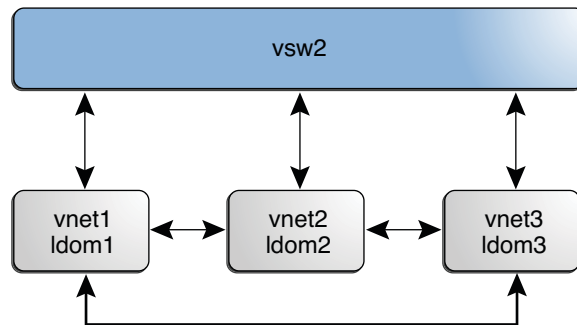
---

**Note** – If guest-to-guest performance is of higher importance than increasing the number of virtual network devices in the system, do not disable inter-vnet LDC channel allocation.

---

The following figure shows a typical virtual switch that has three virtual network devices. The `inter-vnet-link` property is set to `on`, which means that inter-vnet LDC channels are allocated. The guest-to-guest communications between `vnet1` and `vnet2` is performed directly without going through the virtual switch.

FIGURE 8-2 Virtual Switch Configuration That Uses Inter-Vnet Channels



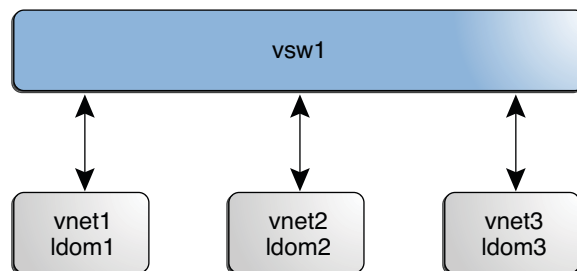
The following figure shows the same virtual switch configuration with the `inter-vnet-link` property set to off. That means that inter-vnet LDC channels are not allocated. You can see that fewer LDC channels are used than when the `inter-vnet-link` property is set to on. In this configuration, guest-to-guest communications between vnet1 and vnet2 must go through vsw1.

---

**Note** – Disabling the assignment of inter-vnet LDC channels does not prevent guest-to-guest communications. Instead, all guest-to-guest communications traffic goes through the virtual switch rather than directly from one guest domain to another guest domain.

---

FIGURE 8-3 Virtual Switch Configuration That Does Not Use Inter-Vnet Channels



## Virtual Device Identifier and Network Interface Name

When you add a virtual switch or virtual network device to a domain, you can specify its device number by setting the `id` property.

```
ldm add-vsw [id=switch-id] vswitch-name ldom
ldm add-vnet [id=network-id] if-name vswitch-name ldom
```

Each virtual switch and virtual network device of a domain has a unique device number that is assigned when the domain is bound. If a virtual switch or virtual network device was added with an explicit device number (by setting the `id` property), the specified device number is used. Otherwise, the system automatically assigns the lowest device number available. In that case, the device number assigned depends on how virtual switch or virtual network devices were added to the system. The device number eventually assigned to a virtual switch or virtual network device is visible in the output of the `ldm list-bindings` command when a domain is bound.

The following example shows that the `primary` domain has one virtual switch, `primary-vsw0`. This virtual switch has a device number of 0 (`switch@0`).

```
primary# ldm list-bindings primary
...
VSW
 NAME MAC NET-DEV DEVICE DEFAULT-VLAN-ID PVID VID MTU MODE
 primary-vsw0 00:14:4f:fb:54:f2 nxge0 switch@0 1 1 5,6 1500
...
```

The following example shows that the `ldg1` domain has two virtual network devices: `vnet` and `vnet1`. The `vnet` device has a device number of 0 (`network@0`) and the `vnet1` device has a device number of 1 (`network@1`).

```
primary# ldm list-bindings ldg1
...
NETWORK
 NAME SERVICE DEVICE MAC MODE PVID VID MTU
 vnet primary-vsw0@primary network@0 00:14:4f:fb:e0:4b hybrid 1 1500
 ...
 vnet1 primary-vsw0@primary network@1 00:14:4f:f8:e1:ea 1 1500
...
```

When a domain with a virtual switch is running the Oracle Solaris OS, the virtual switch has a network interface, `vswN`. However, the network interface number of the virtual switch, `N`, is not necessarily the same as the device number of the virtual switch, `n`.

Similarly, when a domain with a virtual network device is running the Oracle Solaris OS, the virtual network device has a network interface, `vnetN`. However, the network interface number of the virtual network device, `N`, is not necessarily the same as the device number of the virtual network device, `n`.



**Caution** – The Oracle Solaris OS preserves the mapping between the name of a network interface and a virtual switch or virtual network based on the device number. If a device number is not explicitly assigned to a virtual switch or virtual network device, its device number can change when the domain is unbound and is later bound again. In that case, the network interface name assigned by the OS running in the domain can also change and break the existing configuration of the system. This might happen, for example, when a virtual switch or a virtual network interface is removed from the configuration of the domain.

You cannot use the `ldm list -*` commands to directly determine the Oracle Solaris OS network interface name that corresponds to a virtual switch or virtual network device. However, you can obtain this information by using a combination of the output from `ldm list -l` command and from the entries under `/devices` on the Oracle Solaris OS.

## ▼ Find Oracle Solaris OS Network Interface Name

In this example procedure, guest domain `ldg1` contains two virtual network devices, `net -a` and `net -c`. To find the Oracle Solaris OS network interface name in `ldg1` that corresponds to `net -c`, do the following. This example also shows differences if you are looking for the network interface name of a virtual switch instead of a virtual network device.

### 1 Use the `ldm` command to find the virtual network device number for `net -c`.

```
ldm list -l ldg1
...
NETWORK
NAME SERVICE DEVICE MAC
net-a primary-vsw0@primary network@0 00:14:4f:f8:91:4f
net-c primary-vsw0@primary network@2 00:14:4f:f8:dd:68
...
```

The virtual network device number for `net -c` is 2 (`network@2`).

To determine the network interface name of a virtual switch, find the virtual switch device number, *n* as `switch@n`.

### 2 To find the corresponding network interface on `ldg1`, log into `ldg1` and find the entry for this device number under `/devices`.

```
uname -n
ldg1
find /devices/virtual-devices@100 -type c -name network@2*
/devices/virtual-devices@100/channel-devices@200/network@2:vnet1
```

The network interface name is the part of the entry after the colon; that is, `vnet1`.

To determine the network interface name of a virtual switch, replace the argument to the `-name` option with `virtual-network-switch@n\*`. Then, find the network interface with the name `vswN`.

- 3 Verify that `vnet1` has the MAC address `00:14:4f:f8:dd:68` as shown in the `ldm list -l` output for `net-c` in Step 1.

```
ifconfig vnet1
vnet1: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
 inet 0.0.0.0 netmask 0
 ether 0:14:4f:f8:dd:68
```

## Assigning MAC Addresses Automatically or Manually

You must have enough media access control (MAC) addresses to assign to the number of logical domains, virtual switches, and virtual networks you are going to use. You can have the Logical Domains Manager automatically assign MAC addresses to a logical domain, a virtual network (vnet), and a virtual switch (vsw), or you can manually assign MAC addresses from your own pool of assigned MAC addresses. The `ldm` subcommands that set MAC addresses are `add-domain`, `add-vsw`, `set-vsw`, `add-vnet`, and `set-vnet`. If you do not specify a MAC address in these subcommands, the Logical Domains Manager assigns one automatically.

The advantage to having the Logical Domains Manager assign the MAC addresses is that it utilizes the block of MAC addresses dedicated for use with logical domains. Also, the Logical Domains Manager detects and prevents MAC address collisions with other Logical Domains Manager instances on the same subnet. This frees you from having to manually manage your pool of MAC addresses.

MAC address assignment happens as soon as a logical domain is created or a network device is configured into a domain. In addition, the assignment is persistent until the device, or the logical domain itself, is removed.

## Range of MAC Addresses Assigned to Logical Domains

Logical Domains have been assigned the following block of 512K MAC addresses:

`00:14:4F:F8:00:00 ~ 00:14:4F:FF:FF:FF`

The lower 256K addresses are used by the Logical Domains Manager for *automatic MAC address allocation*, and you *cannot* manually request an address in this range:

`00:14:4F:F8:00:00 - 00:14:4F:FB:FF:FF`

You can use the upper half of this range for *manual MAC address allocation*:

`00:14:4F:FC:00:00 - 00:14:4F:FF:FF:FF`

## Automatic Assignment Algorithm

When you do not specify a MAC address in creating logical domain or a network device, the Logical Domains Manager automatically allocates and assigns a MAC address to that logical domain or network device. To obtain this MAC address, the Logical Domains Manager iteratively attempts to select an address and then checks for potential collisions.

Before selecting a potential address, the Logical Domains Manager first looks to see if it has a recently freed, automatically assigned address saved in a database for this purpose (see [“Freed MAC Addresses” on page 118](#)). If so, the Logical Domains Manager selects its candidate address from the database.

If no recently freed addresses are available, the MAC address is randomly selected from the 256K range of addresses set aside for this purpose. The MAC address is selected randomly to lessen the chance of a duplicate MAC address being selected as a candidate.

The address selected is then checked against other Logical Domains Managers on other systems to prevent duplicate MAC addresses from actually being assigned. The algorithm employed is described in [“Duplicate MAC Address Detection” on page 117](#). If the address is already assigned, the Logical Domains Manager iterates, choosing another address, and again checking for collisions. This continues until a MAC address is found that is not already allocated, or a time limit of 30 seconds has elapsed. If the time limit is reached, then the creation of the device fails, and an error message similar to the following is shown.

```
Automatic MAC allocation failed. Please set the vnet MAC address manually.
```

## Duplicate MAC Address Detection

To prevent the same MAC address from being allocated to different devices, one Logical Domains Manager checks with other Logical Domains Managers on other systems by sending a multicast message over the control domain's default network interface, including the address that the Logical Domains Manager wants to assign to the device. The Logical Domains Manager attempting to assign the MAC address waits for one second for a response back. If a different device on another Logical Domains-enabled system has already been assigned that MAC address, the Logical Domains Manager on that system sends back a response containing the MAC address in question. If the requesting Logical Domains Manager receives a response, it knows the chosen MAC address has already been allocated, chooses another, and iterates.

By default, these multicast messages are sent only to other managers on the same subnet; the default time-to-live (TTL) is 1. The TTL can be configured using the Service Management Facilities (SMF) property `ldmd/hops`.

Each Logical Domains Manager is responsible for:

- Listening for multicast messages
- Keeping track of MAC addresses assigned to its domains
- Looking for duplicates
- Responding so that duplicates do not occur

If the Logical Domains Manager on a system is shut down for any reason, duplicate MAC addresses could occur while the Logical Domains Manager is down.

Automatic MAC allocation occurs at the time the logical domain or network device is created and persists until the device or the logical domain is removed.

---

**Note** – A detection check for duplicate MAC addresses is performed when the logical domain or network device is created, and the logical domain is started.

---

## Freed MAC Addresses

When a logical domain or a device associated with an automatic MAC address is removed, that MAC address is saved in a database of recently freed MAC addresses for possible later use on that system. These MAC addresses are saved to prevent the exhaustion of Internet Protocol (IP) addresses from a Dynamic Host Configuration Protocol (DHCP) server. When DHCP servers allocate IP addresses, they do so for a period of time (the lease time). The lease duration is often configured to be quite long, generally hours or days. If network devices are created and removed at a high rate without the Logical Domains Manager reusing automatically allocated MAC addresses, the number of MAC addresses allocated could soon overwhelm a typically configured DHCP server.

When the Logical Domains Manager is requested to automatically obtain a MAC address for a logical domain or network device, it first looks to the freed MAC address database to see if there is a previously assigned MAC address it can reuse. If there is a MAC address available from this database, the duplicate MAC address detection algorithm is run. If the MAC address had not been assigned to someone else since it was previously freed, it will be reused and removed from the database. If a collision is detected, the address is simply removed from the database. The Logical Domains Manager then either tries the next address in the database or if none is available, randomly picks a new MAC address.

## Using Network Adapters With Logical Domains

In a logical domains environment, the virtual switch service running in a service domain can directly interact with GLDv3-compliant network adapters. Though non-GLDv3 compliant network adapters can be used in these systems, the virtual switch cannot interface with them directly. See [“Configuring Virtual Switch and Service Domain for NAT and Routing” on page 119](#) for information about how to use non-GLDv3 compliant network adapters.

For information about using link aggregations, see [“Using Link Aggregation With a Virtual Switch” on page 134](#).

### ▼ Determine If a Network Adapter Is GLDv3-Compliant

- 1 Use the Oracle Solaris OS `dladm(1M)` command, where, for example, `bge0` is the network device name.

```
dladm show-link bge0
bge0 type: non-vlan mtu: 1500 device: bge0
```

- 2 Look at **type**: in the output:

- GLDv3-compliant drivers will have a type of `non-vlan` or `vlan`.
- Non-GLDv3-compliant drivers will have a type of `legacy`.

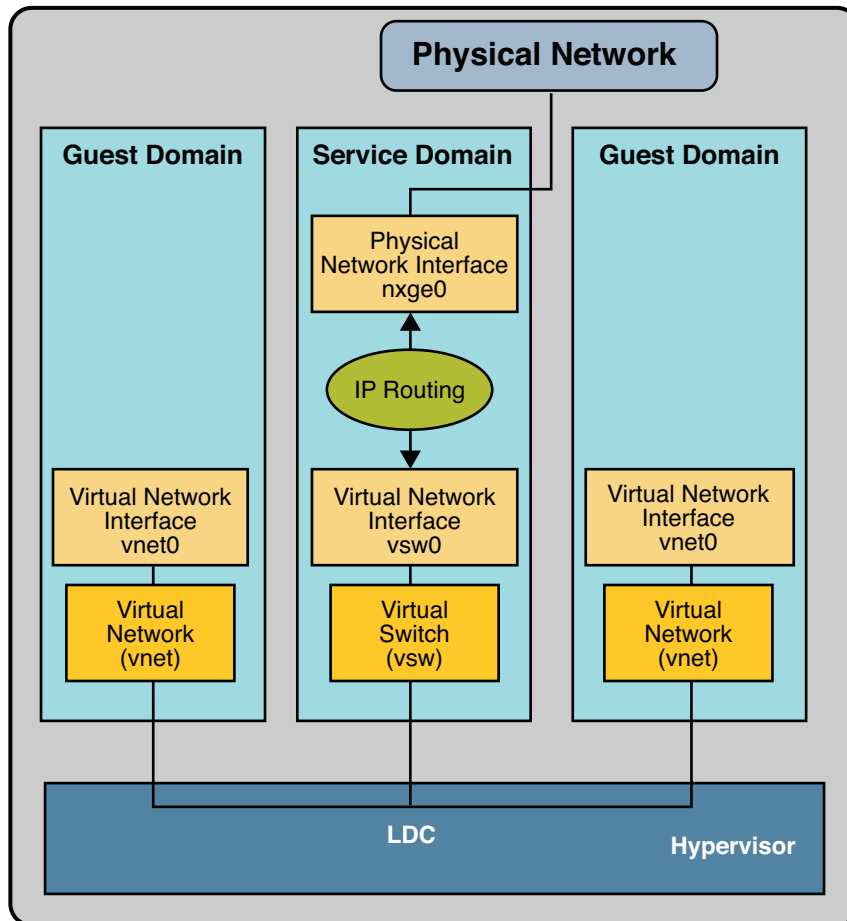
## Configuring Virtual Switch and Service Domain for NAT and Routing

The virtual switch (vsw) is a layer-2 switch, that also can be used as a network device in the service domain. The virtual switch can be configured to act only as a switch between the virtual network (vnet) devices in the various logical domains but with no connectivity to a network outside the box through a physical device. In this mode, creating the vsw as a network device and enabling IP routing in the service domain enables virtual networks to communicate outside the box using the service domain as a router. This mode of operation is very essential to provide external connectivity to the domains when the physical network adapter is not GLDv3-compliant.

The advantages of this configuration are:

- The virtual switch does not need to use a physical device directly and can provide external connectivity even when the underlying device is not GLDv3-compliant.
- The configuration can take advantage of the IP routing and filtering capabilities of the Oracle Solaris OS.

FIGURE 8-4 Virtual Network Routing



## ▼ Set Up the Virtual Switch to Provide External Connectivity to Domains

- 1 Create a virtual switch with no associated physical device.

If assigning an address, ensure that the virtual switch has a unique MAC address.

```
primary# ldm add-vsw [mac-addr=xxxxxxxxxxxx] primary-vsw0 primary
```

- 2 Create the virtual switch as a network device in addition to the physical network device being used by the domain.

See “Configure the Virtual Switch as the Primary Interface” on page 55 for more information about creating the virtual switch.

**3 Configure the virtual switch device for DHCP, if needed.**

See “[Configure the Virtual Switch as the Primary Interface](#)” on page 55 for more information about configuring the virtual switch device for DHCP.

**4 Create the `/etc/dhcp.vsw` file, if needed.**

**5 Configure IP routing in the service domain, and set up required routing tables in all the domains.**

For information about how to do this, refer to “[Packet Forwarding and Routing on IPv4 Networks](#)” in *System Administration Guide: IP Services*.

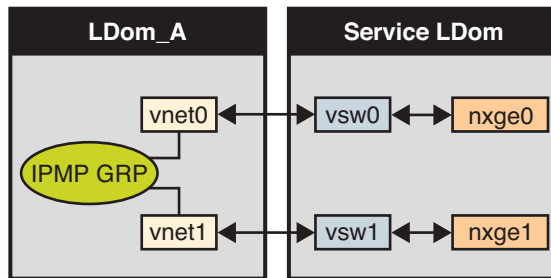
## Configuring IPMP in a Logical Domains Environment

The Logical Domains 1.3 release introduced support for link-based IPMP with virtual network devices. When configuring an IPMP group with virtual network devices, configure the group to use link-based detection. If using older versions of the Oracle VM Server for SPARC (Logical Domains) software, you can only configure probe-based detection with virtual network devices.

### Configuring Virtual Network Devices Into an IPMP Group in a Domain

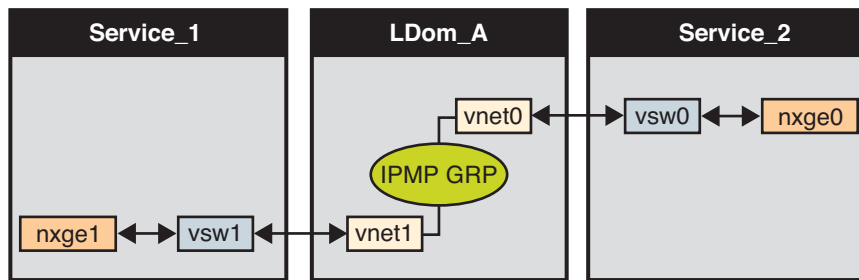
The following diagram shows two virtual networks (vnet1 and vnet2) connected to separate virtual switch instances (vsw0 and vsw1) in the service domain, which, in turn, use two different physical interfaces (nxge0 and nxge1). In the event of a physical link failure in the service domain, the virtual switch device that is bound to that physical device detects the link failure. Then, the virtual switch device propagates the failure to the corresponding virtual network device that is bound to this virtual switch. The virtual network device sends notification of this link event to the IP layer in the guest LDom\_A, which results in failover to the other virtual network device in the IPMP group.

FIGURE 8-5 Two Virtual Networks Connected to Separate Virtual Switch Instances



Further reliability can be achieved in the logical domain by connecting each virtual network device (vnet0 and vnet1) to virtual switch instances in different service domains (as shown in the following diagram). In this case, in addition to physical network failure, LDom\_A can detect virtual network failure and trigger a failover following a service domain crash or shutdown.

FIGURE 8-6 Each Virtual Network Device Connected to Different Service Domains

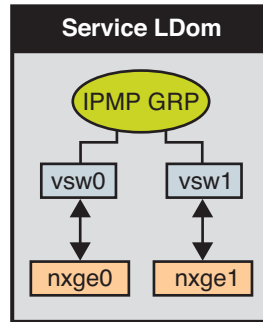


Refer to the Oracle Solaris 10 *System Administration Guide: IP Services* for more information about how to configure and use IPMP groups.

## Configuring and Using IPMP in the Service Domain

IPMP can be configured in the service domain by configuring virtual switch interfaces into a group. The following diagram shows two virtual switch instances (vsw0 and vsw1) that are bound to two different physical devices. The two virtual switch interfaces can then be created and configured into an IPMP group. In the event of a physical link failure, the virtual switch device that is bound to that physical device detects the link failure. Then, the virtual switch device sends notification of this link event to the IP layer in the service domain, which results in failover to the other virtual switch device in the IPMP group.

FIGURE 8-7 Two Virtual Switch Interfaces Configured as Part of IPMP Group



## Using Link-Based IPMP in Logical Domains Virtual Networking

The virtual network and virtual switch devices support link status updates to the network stack. By default, a virtual network device reports the status of its virtual link (its LDC to the virtual switch). This setup is enabled by default and does not require you to perform additional configuration steps.

Sometimes it might be necessary to detect physical network link state changes. For instance, if a physical device has been assigned to a virtual switch, even if the link from a virtual network device to its virtual switch device is up, the physical network link from the service domain to the external network might be down. In such a case, it might be necessary to obtain and report the physical link status to the virtual network device and its stack.

The `linkprop=phys-state` option can be used to configure physical link state tracking for virtual network devices as well as for virtual switch devices. When this option is enabled, the virtual device (virtual network or virtual switch) reports its link state based on the physical link state while it is created as an interface in the domain. You can use standard Oracle Solaris network administration commands such as `dladm` and `ifconfig` to check the link status. See the [`dladm\(1M\)`](#) and [`ifconfig\(1M\)`](#) man pages. In addition, the link status is also logged in the `/var/adm/messages` file.

---

**Note** – You can run both `link-state-unaware` and `link-state-aware` `vnet` and `vsw` drivers concurrently on a Logical Domains system. However, if you intend to configure link-based IPMP, you must install the `link-state-aware` driver. If you intend to enable physical link state updates, upgrade both the `vnet` and `vsw` drivers to the Oracle Solaris 10 9/10 OS, and run at least Version 1.3 of the Logical Domains Manager.

---

### ▼ Configure Physical Link Status Updates

This procedure shows how to enable physical link status updates for virtual network devices.

You can also enable physical link status updates for a virtual switch device by following similar steps and specifying the `linkprop=phys-state` option to the `ldm add-vsw` and `ldm set-vsw` commands.

---

**Note** – You need to use the `linkprop=phys-state` option only if the virtual switch device itself is created as an interface. If `linkprop=phys-state` is specified and the physical link is down, the virtual network device reports its link status as down, even if the connection to the virtual switch is up. This situation occurs because the Oracle Solaris OS does not currently provide interfaces to report two distinct link states, such as `virtual-link-state` and `physical-link-state`.

---

**1 Become superuser or assume an equivalent role.**

Roles contain authorizations and privileged commands. For more information about roles, see [“Configuring RBAC \(Task Map\)” in \*System Administration Guide: Security Services\*](#).

**2 Enable physical link status updates for the virtual device.**

You can enable physical link status updates for a virtual network device in the following ways:

- Create a virtual network device by specifying `linkprop=phys-state` when running the `ldm add-vnet` command.

Specifying the `linkprop=phys-state` option configures the virtual network device to obtain physical link state updates and report them to the stack.

---

**Note** – If `linkprop=phys-state` is specified and the physical link is down (even if the connection to the virtual switch is up), the virtual network device reports its link status as down. This situation occurs because the Oracle Solaris OS does not currently provide interfaces to report two distinct link states, such as `virtual-link-state` and `physical-link-state`.

---

```
ldm add-vnet linkprop=phys-state if-name vswitch-name ldom
```

The following example enables physical link status updates for `vnet0` connected to `primary-vsw0` on the logical domain `ldom1`:

```
ldm add-vnet linkprop=phys-state vnet0 primary-vsw0 ldom1
```

- Modify an existing virtual network device by specifying `linkprop=phys-state` when running the `ldm set-vnet` command.

```
ldm set-vnet linkprop=phys-state if-name ldom
```

The following example enables physical link status updates for `vnet0` on the logical domain `ldom1`:

```
ldm set-vnet linkprop=phys-state vnet0 ldom1
```

To disable physical link state updates, specify `linkprop=` by running the `ldm set-vnet` command.

The following example disables physical link status updates for `vnet0` on the logical domain `ldom1`:

```
ldm set-vnet linkprop= vnet0 ldom1
```

### Example 8–1 Configuring Link-Based IPMP

The following examples show how to configure link-based IPMP both with and without enabling physical link status updates:

- The following example configures two virtual network devices on a domain. Each virtual network device is connected to a separate virtual switch device on the service domain to use link-based IPMP.

---

**Note** – Test addresses are not configured on these virtual network devices. Also, you do not need to perform additional configuration when you use the `ldm add-vnet` command to create these virtual network devices.

---

The following commands add the virtual network devices to the domain. Note that because `linkprop=phys-state` is not specified, only the link to the virtual switch is monitored for state changes.

```
ldm add-vnet vnet0 primary-vsw0 ldom1
ldm add-vnet vnet1 primary-vsw1 ldom1
```

The following commands configure the virtual network devices on the guest domain and assign them to an IPMP group. Note that test addresses are not configured on these virtual network devices because link-based failure detection is being used.

```
ifconfig vnet0 plumb
ifconfig vnet1 plumb
ifconfig vnet0 192.168.1.1/24 up
ifconfig vnet1 192.168.1.2/24 up
ifconfig vnet0 group ipmp0
ifconfig vnet1 group ipmp0
```

- The following example configures two virtual network devices on a domain. Each domain is connected to a separate virtual switch device on the service domain to use link-based IPMP. The virtual network devices are also configured to obtain physical link state updates.

```
ldm add-vnet linkprop=phys-state vnet0 primary-vsw0 ldom1
ldm add-vnet linkprop=phys-state vnet1 primary-vsw1 ldom1
```

---

**Note** – The virtual switch must have a physical network device assigned for the domain to successfully bind. If the domain is already bound and the virtual switch does not have a physical network device assigned, the `ldm add-vnet` commands will fail.

---

The following commands create the virtual network devices and assign them to an IPMP group:

```
ifconfig vnet0 plumb
ifconfig vnet1 plumb
ifconfig vnet0 192.168.1.1/24 up
ifconfig vnet1 192.168.1.2/24 up
ifconfig vnet0 group ipmp0
ifconfig vnet1 group ipmp0
```

## Configuring and Using IPMP in Releases Prior to Logical Domains 1.3

In Logical Domains releases prior to 1.3, the virtual switch and the virtual network devices are not capable of performing link failure detection. In those releases, network failure detection and recovery can be set up by using probe-based IPMP.

### Configuring IPMP in the Guest Domain

The virtual network devices in a guest domain can be configured into an IPMP group as shown in [Figure 8–5](#) and [Figure 8–6](#). The only difference is that probe-based failure detection is used by configuring test addresses on the virtual network devices. See [System Administration Guide: IP Services](#) for more information about configuring probe-based IPMP.

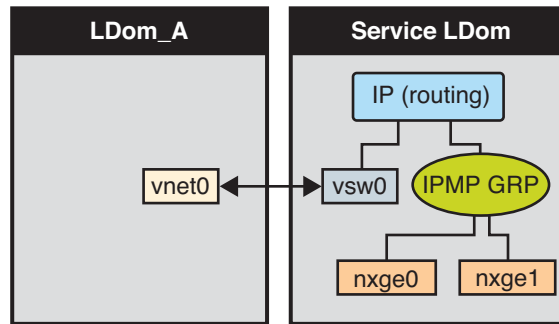
### Configuring IPMP in the Service Domain

In Logical Domains releases prior to 1.3, the virtual switch device is not capable of physical link failure detection. In such cases, network failure detection and recovery can be set up by configuring the physical interfaces in the service domain into an IPMP group. To do this, configure the virtual switch in the service domain without assigning a physical network device to it. Namely, do not specify a value for the `net-dev` (`net-dev=`) property while you use the `ldm add-vswitch` command to create the virtual switch. Create the virtual switch interface in the service domain and configure the service domain itself to act as an IP router. Refer to the Oracle Solaris 10 [System Administration Guide: IP Services](#) for information about setting up IP routing.

Once configured, the virtual switch sends all packets originating from virtual networks (and destined for an external machine) to its IP layer, instead of sending the packets directly by means of the physical device. In the event of a physical interface failure, the IP layer detects failure and automatically re-routes packets through the secondary interface.

Since the physical interfaces are directly being configured into an IPMP group, the group can be set up for either link-based or probe-based detection. The following diagram shows two network interfaces (`nxge0` and `nxge1`) configured as part of an IPMP group. The virtual switch instance (`vsw0`) has been created as a network device to send packets to its IP layer.

FIGURE 8-8 Two Network Interfaces Configured as Part of IPMP Group



### ▼ Configure a Host Route for Probe-Based IPMP

**Note** – This procedure only applies to guest domains and to releases prior to 1.3, where only probe-based IPMP is supported.

If no explicit route is configured for a router in the network corresponding to the IPMP interfaces, then one or more explicit host routes to target systems need to be configured for the IPMP probe-based detection to work as expected. Otherwise, probe detection can fail to detect the network failures.

- **Configure a host route.**

```
route add -host destination-IP gateway-IP -static
```

For example:

```
route add -host 192.168.102.1 192.168.102.1 -static
```

Refer to “Configuring Target Systems” in *System Administration Guide: IP Services* for more information.

## Using VLAN Tagging

As of the release of Oracle Solaris 10 10/08 OS and Logical Domains 1.1 software, 802.1Q VLAN-Tagging support is available in the Logical Domains network infrastructure.

**Note** – Tagged VLANs are not supported in any of the previous releases for Logical Domains networking components.

The virtual switch (vsw) and virtual network (vnet) devices support switching of Ethernet packets based on the virtual local area network (VLAN) identifier (ID) and handle the necessary tagging or untagging of Ethernet frames.

You can create multiple VLAN interfaces over a vnet device in a guest domain. You can use the Oracle Solaris OS `ifconfig` command to create a VLAN interface over a virtual network device, the same way it is used to configure a VLAN interface over any other physical network device. The additional requirement in the Logical Domains environment is that you must assign the vnet to the corresponding VLANs using the Logical Domains Manager CLI commands. Refer to the [ldm\(1M\)](#) for complete information about the Logical Domains Manager CLI commands.

Similarly, you can configure VLAN interfaces over a virtual switch device in the service domain. VLAN IDs 2 through 4094 are valid; VLAN ID 1 is reserved as the `default-vlan-id`.

When you create a vnet device on a guest domain, you must assign it to the required VLANs by specifying a port VLAN ID and zero or more VLAN IDs for this vnet, using the `pvid=` and `vid=` arguments to the `ldm add-vnet` command. This configures the virtual switch to support multiple VLANs in the Logical Domains network and switch packets using both MAC address and VLAN IDs in the network.

Similarly, any VLANs to which the vsw device itself should belong, when created as a network interface, must be configured in the vsw device using the `pvid=` and `vid=` arguments to the `ldm add-vsw` command.

You can change the VLANs to which a device belongs using `ldm set-vnet` or `ldm set-vsw` command.

## Port VLAN ID (PVID)

The PVID indicates a VLAN to which the virtual network device needs to be a member, in untagged mode. In this case, the vsw device provides the necessary tagging or untagging of frames for the vnet device over the VLAN specified by its PVID. Any outbound frames from the virtual network that are untagged are tagged with its PVID by the virtual switch. Inbound frames tagged with this PVID are untagged by the virtual switch, before sending it to the vnet device. Thus, assigning a PVID to a vnet implicitly means that the corresponding virtual network port on the virtual switch is marked untagged for the VLAN specified by the PVID. You can have only one PVID for a vnet device.

The corresponding virtual network interface, when configured without a VLAN ID and using only its device instance, results in the interface being implicitly assigned to the VLAN specified by the virtual network's PVID.

For example, if you were to create vnet instance 0, using the following command, and if the `pvid=` argument for the vnet has been specified as 10, the `vnet0` interface would be implicitly assigned to belong to the VLAN 10.

```
ifconfig vnet0 plumb
```

## VLAN ID (VID)

The VID indicates the VLAN to which a virtual network device or virtual switch needs to be a member, in tagged mode. The virtual network device sends and receives tagged frames over the VLANs specified by its VIDs. The virtual switch passes any frames that are tagged with the specified VID between the virtual network device and the external network.

## ▼ Assign VLANs to a Virtual Switch and Virtual Network Device

### 1 Assign the virtual switch (vsw) to two VLANs.

For example, configure VLAN 21 as untagged and VLAN 20 as tagged. Assign the virtual network (vnet) to three VLANs. Configure VLAN 20 as untagged and VLAN 21 and 22 as tagged.

```
ldm add-vsw net-dev=nxge0 pvid=21 vid=20 primary-vsw0 primary
ldm add-vnet pvid=20 vid=21,22 vnet01 primary-vsw0 ldom1
```

### 2 Create the VLAN interfaces.

This example assumes that the instance number of these devices is 0 in the domains and the VLANs are mapped to these subnets:

| VLAN | Subnet                               |
|------|--------------------------------------|
| 20   | 192.168.1.0 (netmask: 255.255.255.0) |
| 21   | 192.168.2.0 (netmask: 255.255.255.0) |
| 22   | 192.168.3.0 (netmask: 255.255.255.0) |

#### a. Create the VLAN interface in the service (primary) domain.

```
primary# ifconfig vsw0 plumb
primary# ifconfig vsw0 192.168.2.100 netmask 0xffffffff00 broadcast + up
primary# ifconfig vsw20000 plumb
primary# ifconfig vsw20000 192.168.1.100 netmask 0xffffffff00 broadcast + up
```

#### b. Create the VLAN interface in the guest (ldom1) domain.

```
ldom1# ifconfig vnet0 plumb
ldom1# ifconfig vnet0 192.168.1.101 netmask 0xffffffff00 broadcast + up
ldom1# ifconfig vnet21000 plumb
ldom1# ifconfig vnet21000 192.168.2.101 netmask 0xffffffff00 broadcast + up
ldom1# ifconfig vnet22000 plumb
ldom1# ifconfig vnet22000 192.168.3.101 netmask 0xffffffff00 broadcast + up
```

For more information about how to configure VLAN interfaces in the Oracle Solaris OS, refer to “[Administering Virtual Local Area Networks](#)” in *System Administration Guide: IP Services*.

## ▼ Install a Guest Domain When the Install Server Is in a VLAN

Be careful when installing a guest domain over the network (JumpStart) and the installation server is in a VLAN. Specify the VLAN ID that is associated with the installation server as the PVID of the virtual network device, and do not configure any tagged VLANs (`vid`) for that virtual network device. You must do this because OBP is not aware of VLANs and cannot handle VLAN-tagged network packets. The virtual switch handles the untagging and tagging of packets to and from the guest domain during network installation. After the network installation completes and the Oracle Solaris OS boots, you can configure the virtual network device to be tagged in that VLAN. You can then add the virtual network device to additional VLANs in tagged mode.

For information about using JumpStart to install a guest domain, see “[Perform a JumpStart Operation on a Guest Domain](#)” on page 63.

### 1 Initially configure the network device in untagged mode.

For example, if the install server is in VLAN 21, configure the virtual network initially as follows:

```
primary# ldm add-vnet pvid=21 vnet01 primary-vsw0 ldom1
```

### 2 After the installation is complete and the Oracle Solaris OS boots, configure the virtual network in tagged mode.

```
primary# ldm set-vnet pvid= vid=21, 22, 23 vnet01 primary-vsw0 ldom1
```

## Using NIU Hybrid I/O

The virtual I/O framework implements a *hybrid* I/O model for improved functionality and performance. The hybrid I/O model combines direct and virtualized I/O to allow flexible deployment of I/O resources to virtual machines. It is particularly useful when direct I/O does not provide full capability for the virtual machine, or direct I/O is not persistently or consistently available to the virtual machine. This could be because of resource availability or virtual machine migration. The hybrid I/O architecture is well-suited for the Network Interface Unit (NIU) on Sun UltraSPARC T2 and SPARC T3 platforms. An NIU is a network I/O interface that is integrated on chip. This architecture enables the dynamic assignment of Direct Memory Access (DMA) resources to virtual networking devices and, thereby, provides consistent performance to applications in the domain.

NIU hybrid I/O is available for Sun UltraSPARC T2 and SPARC T3 platforms. This feature is enabled by an optional hybrid mode that provides for a virtual network (vnet) device where the DMA hardware resources are loaned to a vnet device in a guest domain for improved performance. In the hybrid mode, a vnet device in a guest domain can send and receive unicast traffic from an external network directly into the guest domain using the DMA hardware resources. The broadcast or multicast traffic and unicast traffic to the other guest domains in the same system continue to be sent using the virtual I/O communication mechanism.

---

**Note** – NIU hybrid I/O is not available on UltraSPARC T2 Plus platforms.

---

[illegible]

Following are points you need to be aware of when using this feature:

- Hybrid mode option for a vnet device is treated as a suggestion only. That means the DMA resources are assigned only when they are available and the device is capable of using them.
- Logical Domains Manager CLI commands do not validate the hybrid mode option; that is, it is possible to set the hybrid mode on any vnet or any number of vnet devices.
- Guest domains and the service domain need to run Oracle Solaris 10 10/08 OS at a minimum.
- Up to a maximum of only three vnet devices per vsw can have DMA hardware resources loaned at a given time. As there are two NIU network devices, there can be a total of six vnet devices with DMA hardware resources loaned.

---

**Note** – Set the hybrid mode only for three vnet devices per vsw so that they are guaranteed to have DMA hardware resources assigned.

---

- Hybrid mode is disabled by default for a vnet device. It needs to be explicitly enabled with Logical Domains Manager CLI commands. See [“Enable Hybrid Mode” on page 134](#). (Refer to the `ldm(1M)` man page for more details.)
- The hybrid mode option cannot be changed dynamically while the guest domain is active.
- The DMA hardware resources are assigned only when a vnet device is active that is created in the guest domain.
- The NIU 10-gigabit Ethernet driver (nxge) is used for the NIU card. The same driver is also used for other 10-gigabit network cards. However, the NIU hybrid I/O feature is available for NIU network devices only.

## ▼ Configure a Virtual Switch With an NIU Network Device

### 1 Determine an NIU network device.

The following example shows the output on an UltraSPARC T2 server:

```
grep nxge /etc/path_to_inst
"/niu@80/network@0" 0 "nxge"
"/niu@80/network@1" 1 "nxge"
```

The following example shows the output on a SPARC T3-1 server:

```
grep nxge /etc/path_to_inst
"/niu@480/network@0" 0 "nxge"
"/niu@480/network@1" 1 "nxge"
```

### 2 Configure a virtual switch.

```
ldm add-vsw net-dev=nxge0 primary-vsw0 primary
```

## ▼ Enable Hybrid Mode

- For example, enable a hybrid mode for a vnet device while it is being created.

```
ldm add-vnet mode=hybrid vnet01 primary-vsw0 ldom01
```

## ▼ Disable Hybrid Mode

- For example, disable hybrid mode for a vnet device.

```
ldm set-vnet mode= vnet01 ldom01
```

# Using Link Aggregation With a Virtual Switch

As of the release of the Oracle Solaris 10 10/08 OS and the Logical Domains 1.1 software, the virtual switch can be configured to use a link aggregation. A link aggregation is used as the virtual switch's network device to connect to the physical network. This configuration enables the virtual switch to leverage the features provided by the IEEE 802.3ad Link Aggregation Standard. Such features include increased bandwidth, load balancing, and failover. For information about how to configure link aggregation, see the [System Administration Guide: IP Services](#).

After you create a link aggregation, you can assign it to the virtual switch. Making this assignment is similar to assigning a physical network device to a virtual switch. Use the `ldm add-vswitch` or `ldm set-vswitch` command to set the `net-dev` property.

When the link aggregation is assigned to the virtual switch, traffic to and from the physical network flows through the aggregation. Any necessary load balancing or failover is handled transparently by the underlying aggregation framework. Link aggregation is completely transparent to the virtual network (vnet) devices that are on the guest domains and that are bound to a virtual switch that uses an aggregation.

---

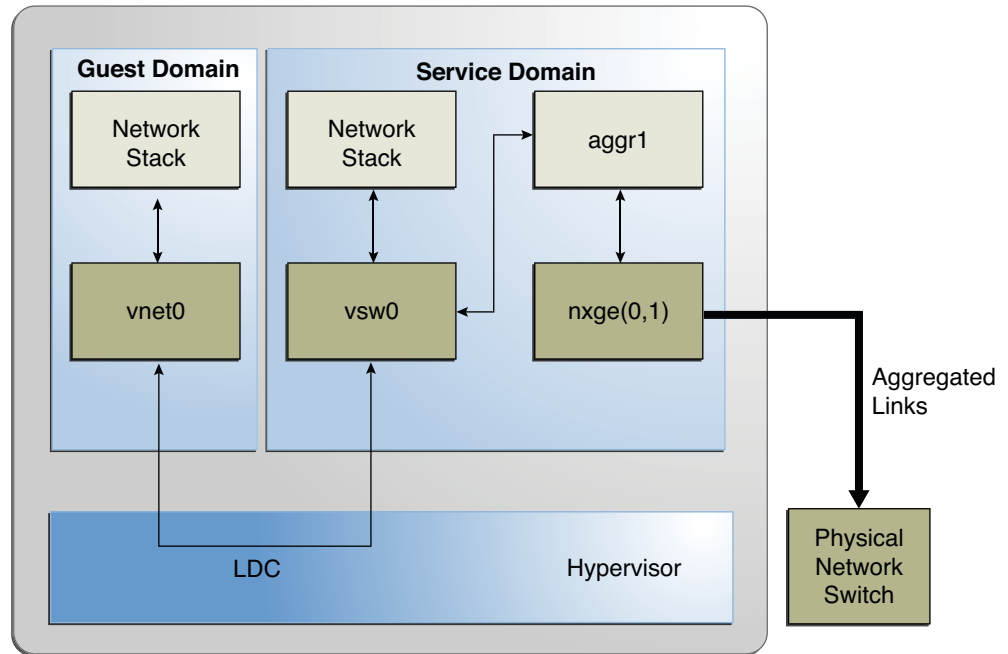
**Note** – You cannot group the virtual network devices (vnet and vsw) into a link aggregation.

---

You can create and use the virtual switch that is configured to use a link aggregation in the service domain. See “[Configure the Virtual Switch as the Primary Interface](#)” on page 55.

The following figure illustrates a virtual switch configured to use an aggregation, `aggr1`, over physical interfaces `nxge0` and `nxge1`.

FIGURE 8-10 Configuring a Virtual Switch to Use a Link Aggregation



## Configuring Jumbo Frames

The Logical Domains virtual switch (vsw) and virtual network (vnet) devices can now support Ethernet frames with payload sizes larger than 1500 bytes. This change results in these drivers being able to increase network throughput.

### ▼ Configure Virtual Network and Virtual Switch Devices to Use Jumbo Frames

You enable jumbo frames by specifying the maximum transmission unit (MTU) for the virtual switch device. In such cases, the virtual switch device and all virtual network devices that are bound to the virtual switch device use the specified MTU value.

In certain circumstances, you can specify an MTU value directly on a virtual network device. You might do this if the required MTU value for the virtual network device should be less than that supported by the virtual switch.

---

**Note** – On the Oracle Solaris 10 5/09 OS, the MTU of a physical device must be configured to match the MTU of the virtual switch. For more information about configuring particular drivers, see the man page that corresponds to that driver in Section 7D of the Oracle Solaris reference manual. For example, to obtain information about the `nxge` driver, see the [nxge\(7D\)](#) man page.

---

**1 Log in to the control domain.**

**2 Become superuser or assume an equivalent role.**

Roles contain authorizations and privileged commands. For more information about roles, see “[Configuring RBAC \(Task Map\)](#)” in *System Administration Guide: Security Services*.

**3 Determine the value of MTU that you want to use for the virtual network.**

You can specify an MTU value from 1500 to 16000 bytes. The specified MTU must match the MTU of the physical network device that is assigned to the virtual switch.

**4 Specify the MTU value of a virtual switch device or virtual network device.**

Do one of the following:

- Enable jumbo frames on a new virtual switch device in the service domain by specifying its MTU as a value of the `mtu` property.

```
ldm add-vsw mtu=value vswitch-name ldom
```

In addition to configuring the virtual switch, this command updates the MTU value of each virtual network device that will be bound to this virtual switch.

- Enable jumbo frames on an existing virtual switch device in the service domain by specifying its MTU as a value of the `mtu` property.

```
ldm set-vsw mtu=value vswitch-name
```

In addition to configuring the virtual switch, this command updates the MTU value of each virtual network device that will be bound to this virtual switch.

In rare circumstances, you might need to use the `ldm add-vnet` or `ldm set-vnet` command to specify an MTU value for a virtual network device that differs from the MTU value of the virtual switch. For example, you might change the virtual network device's MTU value if you configure VLANs over a virtual network device and the largest VLAN MTU is less than the MTU value on the virtual switch. A `vnet` driver that supports jumbo frames might not be required for domains where only the default MTU value is used. However, if the domains have virtual network devices bound to a virtual switch that uses jumbo frames, ensure that the `vnet` driver supports jumbo frames.

If you use the `ldm set-vnet` command to specify an `mtu` value on a virtual network device, future updates to the MTU value of the virtual switch device are not propagated to that virtual

network device. To reenable the virtual network device to obtain the MTU value from the virtual switch device, run the following command:

```
ldm set-vnet mtu= vnet-name ldom
```

Note that enabling jumbo frames for a virtual network device automatically enables jumbo frames for any HybridIO resource that is assigned to that virtual network device.

On the control domain, the Logical Domains Manager updates the MTU values that are initiated by the `ldm set -vsw` and `ldm set -vnet` commands as delayed reconfiguration operations. To make MTU updates to domains other than the control domain, you must stop a domain prior to running the `ldm set -vsw` or `ldm set -vnet` command to modify the MTU value.

## Example 8–2 Configuring Jumbo Frames on Virtual Switch and Virtual Network Devices

- The following example shows how to add a new virtual switch device that uses an MTU value of 9000. This MTU value is propagated from the virtual switch device to all of the client virtual network devices.

First, the `ldm add -vsw` command creates the virtual switch device, `primary-vsw0`, with an MTU value of 9000. Note that instance 0 of the network device `nxge0` is specified as a value of the `net - dev` property.

```
ldm add-vsw net-dev=nxge0 mtu=9000 primary-vsw0 primary
```

Next, the `ldm add -vnet` command adds a client virtual network device to this virtual switch, `primary-vsw0`. Note that the MTU of the virtual network device is implicitly assigned from the virtual switch to which it is bound. As a result, the `ldm add -vnet` command does not require that you specify a value for the `mtu` property.

```
ldm add-vnet vnet01 primary-vsw0 ldom1
```

The `ifconfig` command creates the virtual switch interface in the service domain, `primary`. The `ifconfig vsw0` command output shows that the value of the `mtu` property is 9000.

```
ifconfig vsw0 plumb
ifconfig vsw0 192.168.1.100/24 up
ifconfig vsw0
vsw0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 9000 index 5
 inet 192.168.1.100 netmask fffffff0 broadcast 192.168.1.255
 ether 0:14:4f:fa:0:99
```

The `ifconfig` command creates the virtual network interface in the guest domain, `ldom1`. The `ifconfig vnet0` command output shows that the value of the `mtu` property is 9000.

```
ifconfig vnet0 plumb
ifconfig vnet0 192.168.1.101/24 up
ifconfig vnet0
vnet0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 9000 index 4
 inet 192.168.1.101 netmask fffffff0 broadcast 192.168.1.255
 ether 0:14:4f:f9:c4:13
```

- The following example shows how to use the `ifconfig` command to change the MTU of the interface to 4000.

Note that the MTU of an interface can only be changed to a value that is less than the MTU of the device that is assigned by the Logical Domains Manager. This method is useful when VLANs are configured and each VLAN interface needs a different MTU.

```
ifconfig vnet0 mtu 4000
ifconfig vnet0
vnet0: flags=1201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS,FIXEDMTU>
mtu 4000 index 4
 inet 192.168.1.101 netmask ffffffff broadcast 192.168.1.255
 ether 0:14:4f:f9:c4:13
```

## Compatibility With Older (Jumbo-Unaware) Versions of the vnet and vsw Drivers

Drivers that support jumbo frames can interoperate with drivers that do not support jumbo frames on the same system. This interoperability is possible as long as jumbo frame support is not enabled when you create the virtual switch.

---

**Note** – Do not set the `mtu` property if any guest or service domains that are associated with the virtual switch do not use Logical Domains drivers that support jumbo frames.

---

Jumbo frames can be enabled by changing the `mtu` property of a virtual switch from the default value of 1500. In this instance, older driver versions ignore the `mtu` setting and continue to use the default value. Note that the `ldm list` output will show the MTU value you specified and not the default value. Any frames larger than the default MTU are not sent to those devices and are dropped by the new drivers. This situation might result in inconsistent network behavior with those guests that still use the older drivers. This applies to both client guest domains and the service domain.

So, while jumbo frames are enabled, ensure that all virtual devices in the Logical Domains network are upgraded to use the new drivers that support jumbo frames. Also ensure that you upgrade to at least Logical Domains 1.2 so that you can configure jumbo frames.

# Migrating Domains

---

This chapter describes how to migrate domains from one host machine to another host machine.

This chapter covers the following topics:

- “Introduction to Domain Migration” on page 140
- “Overview of a Migration Operation” on page 140
- “Software Compatibility” on page 141
- “Security for Migration Operations” on page 141
- “Migrating a Domain” on page 142
- “Migrating an Active Domain” on page 143
- “Migrating Bound or Inactive Domains” on page 147
- “Performing a Dry Run” on page 142
- “Monitoring a Migration in Progress” on page 148
- “Canceling a Migration in Progress” on page 149
- “Recovering From a Failed Migration” on page 149
- “Performing Non-Interactive Migrations” on page 142
- “Migration Examples” on page 150

---

**Note** – To use the migration features described in this chapter, you must be running the most recent versions of the Logical Domains Manager, system firmware, and Oracle Solaris OS. For information about migration using previous versions of Oracle VM Server for SPARC, see *Oracle VM Server for SPARC 2.1 Release Notes* and related versions of the administration guide.

---

# Introduction to Domain Migration

Domain migration enables you to migrate a guest domain from one host machine to another host machine. The machine on which the migration is initiated is the *source machine*. The machine to which the domain is migrated is the *target machine*.

While a migration operation is in progress, the *domain to be migrated* is transferred from the source machine to the *migrated domain* on the target machine.

The Oracle VM Server for SPARC 2.1 release introduces *live migration*, which provides performance improvements that enable an active domain to be migrated while it continues to run. In addition to live migration, you can migrate bound or inactive domains, which is called *cold migration*.

You can use domain migration to perform tasks such as the following:

- Balancing the load between machines
- Performing hardware maintenance while a guest domain continues to run

## Overview of a Migration Operation

The Logical Domains Manager on the source machine accepts the request to migrate a domain and establishes a secure network connection with the Logical Domains Manager that runs on the target machine. The migration occurs after this connection has been established. The migration operation is performed in the following phases:

**Phase 1:** After the source machine connects with the Logical Domains Manager that runs in the target machine, information about the source machine and the domain to be migrated are transferred to the target machine. This information is used to perform a series of checks to determine whether a migration is possible. The checks to perform are based on the state of the domain to be migrated. For example, if the domain to be migrated is active, a different set of checks are performed than if that domain is bound or inactive.

**Phase 2:** When all checks in Phase 1 have passed, the source and target machines prepare for the migration. On the target machine, a domain is created to receive the domain to be migrated. If the domain to be migrated is inactive or bound, the migration operation proceeds to Phase 5.

**Phase 3:** If the domain to be migrated is active, its runtime state information is transferred to the target machine. The domain to be migrated continues to run, and the Logical Domains Manager simultaneously tracks the modifications being made by the OS to this domain. This information is retrieved from the hypervisor on the source machine and installed in the hypervisor on the target machine.

**Phase 4:** The domain to be migrated is suspended. At this time, all of the remaining modified state information is re-copied to the target machine. In this way, there should be little or no perceivable interruption to the domain. The amount of interruption depends on the workload.

**Phase 5:** A handoff occurs from the Logical Domains Manager on the source machine to the Logical Domains Manager on the target machine. The handoff occurs when the migrated domain resumes execution (if the domain to be migrated was active), and the domain on the source machine is destroyed. From this point forward, the migrated domain is the sole version of the domain running.

## Software Compatibility

For a migration to occur, both the source and target machines must be running compatible software, as follows:

- Version 2.1 of the Logical Domains Manager must be running on both machines.
- Both the source and target machines must have a compatible version of firmware installed to support live migration. See [“Required Software to Enable Oracle VM Server for SPARC 2.1 Features”](#) in *Oracle VM Server for SPARC 2.1 Release Notes*.

## Security for Migration Operations

Oracle VM Server for SPARC provides the following security features for migration operations:

- **Authentication.** Because the migration operation executes on two machines, a user must be authenticated on both the source and target machines. In particular, a user other than superuser must have the `solaris.ldoms.read` and `solaris.ldoms.write` authorizations.

The `ldm migrate-domain` command permits you to optionally specify an alternate user name for authentication on the target machine. If this alternate user name is not specified, the user name of the user who is executing the migration command is used. See [Example 9-2](#). In either case, the user is prompted for a password for the target machine, unless the `-p` option is used to initiate a non-interactive migration. See [“Performing Non-Interactive Migrations”](#) on page 142.

- **Encryption.** Oracle VM Server for SPARC uses SSL to encrypt migration traffic to protect sensitive data from exploitation and to eliminate the requirement for additional hardware and dedicated networks.

The speed of the migration operation increases when the primary domain on the source and target machines has cryptographic units assigned. This increase in speed occurs because the SSL operations can be off-loaded to the cryptographic units.

# Migrating a Domain

You can use the `ldm migrate-domain` command to initiate the migration of a domain from one host machine to another host machine.

For information about migrating an active domain while it continues to run, see [“Migrating an Active Domain” on page 143](#). For information about migrating a bound or inactive domain, see [“Migrating Bound or Inactive Domains” on page 147](#).

For information about the migration options and operands, see the `ldm(1M)` man page.

## Performing a Dry Run

When you provide the `-n` option to the `ldm migrate-domain` command, migration checks are performed, but the domain is not migrated. Any requirement that is not satisfied is reported as an error. The dry run results enable you to correct any configuration errors before you attempt an actual migration.

---

**Note** – Because of the dynamic nature of logical domains, it is possible for a dry run to succeed and an actual migration to fail, and vice-versa.

---

## Performing Non-Interactive Migrations

You can use the `ldm migrate-domain -p filename` command to initiate a non-interactive migration operation.

The file name you specify as an argument to the `-p` option must have the following characteristics:

- The first line of the file must contain the password
- The password must be plain text
- The password must not exceed 256 characters in length

A newline character at the end of the password and all lines that follow the first line are ignored.

The file in which you store the target machine's password must be properly secured. If you plan to store passwords in this manner, ensure that the file permissions are set so that only the root owner, or a privileged user, can read or write the file (400 or 600).

# Migrating an Active Domain

Certain requirements and restrictions are imposed on the domain to be migrated, the source machine, and the target machine when you attempt to migrate an active domain. For more information, see [“Domain Migration Restrictions” in Oracle VM Server for SPARC 2.1 Release Notes](#).

---

**Tip** – You can reduce the overall migration time by adding more virtual CPUs to the primary domain on both the source and target machines. It is best, but not required, to have at least 16 CPUs in the each primary domain.

---

A domain “loses time” during the migration process. To mitigate this time-loss issue, synchronize the domain to be migrated with an external time source, such as a Network Time Protocol (NTP) server. When you configure a domain as an NTP client, the domain's date and time are corrected shortly after the migration completes.

To configure a domain as an NTP client, see [“Managing Network Time Protocol \(Tasks\)” in System Administration Guide: Network Services](#).

## Migration Requirements for CPUs

Following are the requirements and restrictions on CPUs when you perform a migration:

- The source and target machines must have the same processor type.  
Use the `psrinfo -pv` command to determine the processor type, as follows:

```
psrinfo -pv
The physical processor has 8 virtual processors (0-7)
SPARC-T3 (chipid 0, clock 1649 MHz)
```

- The source and target machines must have the processor running at the same frequency (in MHz) and have identical STICK register values.

Use the `prtconf -pv` command to determine the STICK frequency, as follows:

```
prtconf -pv | grep stick-frequency
stick-frequency: 05f4bc08
```

---

**Note** – The frequency at which the STICK register increments is derived from the full-speed CPU frequency. However, even though the CPU frequency on both machines might be identical, the exact STICK register frequency might differ slightly and thus block a migration.

---

- The target machine must have sufficient free strands to accommodate the number of strands in use by the domain to be migrated.

## Migration Requirements for Memory

There must be sufficient free memory on the target machine to accommodate the migration of a domain. In addition, following are a few properties that must be maintained across the migration:

- It must be possible to create the same number of identically sized memory blocks.
- The physical addresses of the memory blocks do not need to match, but the same real addresses must be maintained across the migration.

In addition, the layout of the available memory on the target machine must be compatible with the memory layout of the domain to be migrated or the migration will fail. In particular, if the memory on the target machine is fragmented into multiple small address ranges, but the domain to be migrated requires a single large address range, the migration will fail. The following example illustrates this scenario. The target machine has 2 Gbytes of free memory in two memory blocks:

```
ldm list-devices memory
MEMORY
 PA SIZE
 0x108000000 1G
 0x188000000 1G
```

The domain to be migrated, `ldg-src`, also has 2 Gbytes of free memory, but it is laid out in a single memory block:

```
ldm list -o memory ldg-src
NAME
ldg-src

MEMORY
 RA PA SIZE
 0x8000000 0x208000000 2G
```

Given this memory layout situation, the migration fails:

```
ldm migrate-domain ldg-src t5440-sys-2
Target Password:
Unable to bind 2G memory region at real address 0x8000000
Domain Migration of LDom ldg-src failed
```

---

**Note** – After a migration, memory dynamic reconfiguration (DR) is disabled for the migrated domain until it has been rebooted. After the reboot completes, memory DR is re-enabled for the migrated domain.

---

## Migration Requirements for Physical I/O Devices

Domains that have direct access to physical devices cannot be migrated. For example, you cannot migrate I/O domains. However, virtual devices that are associated with physical devices can be migrated.

## Migration Requirements for Virtual I/O Devices

All virtual I/O services that are used by the domain to be migrated must be available on the target machine. In other words, the following conditions must exist:

- Each virtual disk back end that is used in the domain to be migrated must be defined on the target machine. The virtual disk back end you define must have the same volume and service names as on the source machine. Paths to the back end might be different on the source and target machines, but they *must* refer to the same back end.



---

**Caution** – A migration will succeed even if the paths to a virtual disk back end on the source and target machines do not refer to the same storage. However, the behavior of the domain on the target machine will be unpredictable, and the domain is likely to be unusable. To remedy the situation, stop the domain, correct the configuration issue, and then restart the domain. If you do not perform these steps, the domain might be left in an inconsistent state.

---

- Each virtual network device in the domain to be migrated must have a corresponding virtual network switch on the target machine. Each virtual network switch must have the same name as the virtual network switch to which the device is attached on the source machine.

For example, if `vnet0` in the domain to be migrated is attached to a virtual switch service named `switch-y`, a domain on the target machine must provide a virtual switch service named `switch-y`.

---

**Note** – The physical network on the target machine must be correctly configured so that the migrated domain can access the network resources it requires. Otherwise, some network services might become unavailable on the domain after the migration completes.

For example, you might want to ensure that the domain can access the correct network subnet. Also, you might want to ensure that gateways, routers, or firewalls are properly configured so that the domain can reach the required remote systems from the target machine.

---

MAC addresses used by the domain to be migrated that are in the automatically allocated range must be available for use on the target machine.

- A virtual console concentrator (vcc) service must exist on the target machine and have at least one free port. Explicit console constraints are ignored during the migration. The console for the migrated domain is created by using the migrated domain name as the console group and by using any available port on the first vcc device in the control domain. The migration fails if there is a conflict with the default group name.

## Migration Requirements for NIU Hybrid I/O

You can migrate a domain that uses NIU Hybrid I/O resources. A constraint that specifies NIU Hybrid I/O resources is not a hard requirement of a domain. If such a domain is migrated to a machine that does not have available NIU resources, the constraint is preserved, but not fulfilled.

## Migration Requirements Cryptographic Units

You can migrate a guest domain that has bound cryptographic units if it runs an operating system that supports cryptographic unit dynamic reconfiguration (DR).

The following Oracle Solaris OS versions support cryptographic unit DR:

- At least the Solaris 10 10/09 OS
- At least the Solaris 10 5/08 OS plus patch ID 142245-01

At the start of the migration, the Logical Domains Manager determines whether the domain to be migrated supports cryptographic unit DR. If supported, the Logical Domains Manager attempts to remove any cryptographic units from the domain. After the migration completes, the cryptographic units are re-added to the migrated domain.

---

**Note** – If the constraints for cryptographic units cannot be met on the target machine, the migration operation will not be blocked. In such a case, the migrated domain might have fewer cryptographic units than it had prior to the migration operation.

---

## Delayed Reconfiguration in an Active Domain

Any active delayed reconfiguration operations on the source or target machine prevent a migration from starting. Delayed reconfiguration operations are blocked while a migration is in progress.

## Migrating While an Active Domain Is in Elastic Mode

Domain migrations are not supported for a source or target machine that is in elastic mode. If the PM policy on the source or target machine is switched from performance mode to elastic mode while a migration is in progress, the policy switch is deferred until the migration completes. The migration command returns an error if a domain migration is attempted while either the source or target machine is in elastic mode.

## Operations on Other Domains

While a migration is in progress on a machine, any operation that might result in the modification of the state or configuration of the domain being migrated, is blocked. All operations on the domain itself, as well as operations such as bind and stop on other domains on the machine, are blocked.

## Migrating a Domain That is Running in OpenBoot or in the Kernel Debugger

Performing a domain migration requires coordination between the Logical Domains Manager and the OS that is running in the domain to be migrated. When a domain to be migrated is running in OpenBoot or in the kernel debugger (kldb), this coordination is not possible. As a result, a migration attempt fails unless the domain to be migrated has only a single CPU. When the domain to be migrated has a single CPU, the migration proceeds when certain requirements and restrictions are met. See [“Domain Migration Restrictions” in Oracle VM Server for SPARC 2.1 Release Notes](#).

## Migrating Bound or Inactive Domains

Only a few domain migration restrictions apply to a bound or inactive domain because such domains are not executing at the time of the migration.

The migration of a bound domain requires that the target machine is able to satisfy the CPU, memory, and I/O constraints of the domain to be migrated. If these constraints cannot be met, the migration will fail.

The migration of an inactive domain does not have such requirements. However, the target machine must satisfy the migrated domain's constraints when a bind is later attempted, or the domain binding will fail.

## Migration Requirements for CPUs

You can migrate a bound or inactive domain between machines that run different processor types and machines that run at different frequencies.

The Oracle Solaris OS image in the domain to be migrated must support the processor type on the target machine.

## Migration Requirements for Virtual I/O Devices

For an inactive domain, no checks are performed of the virtual I/O (VIO) constraints. So, the VIO servers do not need to exist for the migration to succeed. As with any inactive domain, the VIO servers must exist and be available at the time the domain is bound.

## Migration Requirements for PCIe Endpoint Devices

You cannot perform a domain migration on an I/O domain that is configured with PCIe endpoint devices.

For information about the direct I/O (DIO) feature, see [“Assigning PCIe Endpoint Devices” on page 70](#).

## Monitoring a Migration in Progress

When a migration is in progress, the domain being migrated and the migrated domain are shown differently in the status output. The output of the `ldm list` command indicates the state of the migrating domain.

The sixth column in the **FLAGS** field shows one of the following values:

- The domain being migrated shows an **s** to indicate that it is the source of the migration.
- The migrated domain shows a **t** to indicate that it is the target of a migration.
- If an error occurs that requires user intervention, an **e** is shown.

The following command shows that the `ldg-src` domain is the source of the migration:

```
ldm list ldg-src
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
ldg-src suspended -n---s 1 1G 0.0% 2h 7m
```

The following command shows that the `ldg-tgt` domain is the target of the migration:

```
ldm list ldg-tgt
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
ldg-tgt bound -----t 5000 1 1G
```

The long form of the status output shows additional information about the migration. On the source machine, the status output shows the completion percentage of the operation as well as the names of the target machine and the migrated domain. Similarly, on the target machine, the status output shows the completion percentage of the operation as well as the names of the source machine and the domain being migrated.

The following command shows the progress of a migration operation for the `ldg-src` domain:

```
ldm list -o status ldg-src
NAME
ldg-src

STATUS
 OPERATION PROGRESS TARGET
 migration 17% t5440-sys-2
```

## Canceling a Migration in Progress

After a migration starts, the migration operation is terminated if the `ldm` command is interrupted by a KILL signal. When the migration operation is terminated, the migrated domain is destroyed, and the domain to be migrated is resumed if it was active. If the controlling shell of the `ldm` command is lost, the migration continues in the background.

A migration operation can also be canceled externally by using the `ldm cancel-operation` command. This command terminates the migration in progress, and the domain being migrated resumes as the active domain. The `ldm cancel-operation` command must be initiated from the source machine. On a given machine, any migration-related command affects the migration operation that was started from that machine. A target machine *cannot* control a migration operation.

---

**Note** – After a migration has been initiated, suspending the `ldm` process does not pause the operation. The operation is not paused because the Logical Domains Manager daemon (`ldmd`) on the source and target machines, and not the `ldm` process, affects the migration. The `ldm` process awaits a signal from `ldmd` that the migration has completed before returning.

---

## Recovering From a Failed Migration

The migration operation terminates if the network connection is lost in the following manner:

- *After* the domain being migrated has completed sending all the runtime state information to the migrated domain
- But *before* the migrated domain can acknowledge that the domain has been resumed

You must determine whether the migration completed successfully by taking the following steps:

1. Determine whether the migrated domain has successfully resumed operations. The migrated domain will be in one of two states:
  - If the migration completed successfully, the migrated domain is in the normal state.
  - If the migration failed, the target machine cleans up and destroys the migrated domain.
2. If the migrated domain successfully resumed operations, you can safely destroy the domain on the source machine that is in the error state. However, if the migrated domain is not present, the domain on the source machine is still the master version of the domain and must be recovered. To recover this domain, run the `ldm cancel-operation` command on the source machine. This command clears the error state and restores the domain to its original condition.

## Migration Examples

### EXAMPLE 9-1 Migrating a Guest Domain

This example shows how to migrate the `ldg1` domain to a machine called `t5440-sys-2`.

```
ldm migrate-domain ldg1 t5440-sys-2
Target Password:
```

To perform this migration without being prompted for the target machine password, use the following command:

```
ldm migrate-domain -p pfile ldg1 t5440-sys-2
```

The `-p` option takes a file name as an argument. The specified file contains the superuser password for the target machine. In this example, `pfile` contains the password for the target machine, `t5440-sys-2`.

### EXAMPLE 9-2 Migrating and Renaming a Guest Domain

This example shows how to rename a domain as part of the migration operation. The `ldg-src` domain on the source machine is renamed to `ldg-tgt` on the target machine (`t5440-sys-2`) as part of the migration. In addition, the `ldm-admin` user is used for authentication on the target machine.

```
ldm migrate ldg-src ldm-admin@t5440-sys-2:ldg-tgt
Target Password:
```

### EXAMPLE 9-3 Migration Failure Message

This example shows the error message that you might see if the target machine does not support the latest migration functionality.

```
ldm migrate ldg1 dt212-346
Target Password:
```

**EXAMPLE 9-3** Migration Failure Message (Continued)

The target machine is running an older version of the domain manager that does not support the latest migration functionality.

Upgrading to the latest software will remove restrictions on a migrated domain that are in effect until it is rebooted. Consult the product documentation for a full description of these restrictions.

The target machine is running an older version of the domain manager that is not compatible with the version running on the source machine.

Domain Migration of LDom ldg1 failed

**EXAMPLE 9-4** Obtaining the Migration Status for the Domain on the Target Machine

This example shows how to obtain the status on a migrated domain while a migration is in progress. In this example, the source machine is t5440-sys-1.

```
ldm list -o status ldg-tgt
NAME
ldg-tgt

STATUS
 OPERATION PROGRESS SOURCE
 migration 55% t5440-sys-1
```

**EXAMPLE 9-5** Obtaining the Parseable Migration Status for the Domain on the Source Machine

This example shows how to obtain the parseable status on the domain being migrated while a migration is in progress. In this example, the target machine is t5440-sys-2.

```
ldm list -o status -p ldg-src
VERSION 1.5
DOMAIN|name=ldg-src|
STATUS
|op=migration|progress=42|error=no|target=t5440-sys-2
```



# Managing Resources

---

This chapter contains information about performing resource management on Oracle VM Server for SPARC systems.

This chapter covers the following topics:

- [“Resource Reconfiguration” on page 153](#)
- [“Resource Allocation” on page 155](#)
- [“CPU Allocation” on page 155](#)
- [“Using Memory Dynamic Reconfiguration” on page 159](#)
- [“Using Power Management” on page 166](#)
- [“Using Dynamic Resource Management” on page 171](#)
- [“Listing Domain Resources” on page 174](#)

## Resource Reconfiguration

A system that runs the Oracle VM Server for SPARC software is able to configure resources, such as virtual CPUs, virtual I/O devices, cryptographic units, and memory. Some resources can be configured dynamically on a running domain, while others must be configured on a stopped domain. If a resource cannot be dynamically configured on the control domain, you must first initiate a delayed reconfiguration. The delayed reconfiguration postpones the configuration activities until after the control domain has been rebooted.

## Dynamic Reconfiguration

Dynamic reconfiguration (DR) enables resources to be added or removed while the operating system (OS) is running. The capability to perform DR of a particular resource type is dependent on having support in the OS running in the logical domain.

Dynamic reconfiguration is supported for the following resources:

- **Virtual CPUs** – Supported in all versions of the Oracle Solaris 10 OS
- **Virtual I/O devices** – Supported in at least the Solaris 10 10/08 OS
- **Cryptographic units** – Supported in at least the Oracle Solaris 10 9/10 OS
- **Memory** – Supported starting with the Oracle VM Server for SPARC 2.0 release (see [“Using Memory Dynamic Reconfiguration” on page 159](#))
- **Physical I/O devices** – Not supported

To use the DR capability, the Logical Domains DR daemon, `drd`, must be running in the domain that you want to change. See the [`drd\(1M\)`](#) man page.

## Delayed Reconfiguration

In contrast to DR operations that take place immediately, delayed reconfiguration operations take effect in the following circumstances:

- After the next reboot of the OS
- After a stop and start of a logical domain

Delayed reconfiguration operations are restricted to the control domain. For all other domains, you must stop the domain to modify the configuration unless the resource can be dynamically reconfigured.

When a delayed reconfiguration is in progress on the control domain, other reconfiguration requests for the control domain are deferred until it is rebooted, or stopped and started.

The `ldm cancel-operation reconf` command cancels delayed reconfiguration operations on the control domain. For more information about how to use the delayed reconfiguration feature, see the [`ldm\(1M\)`](#) man page.

---

**Note** – You cannot use the `ldm cancel-operation reconf` command if any other `ldm remove-*` commands have already performed a delayed reconfiguration operation on virtual I/O devices. The `ldm cancel-operation reconf` command fails in these circumstances.

---

You can use delayed reconfiguration to decrease resources on the control domain. To remove a large number of CPUs from the control domain, see [“Removing a Large Number of CPUs From the Control Domain” in \*Oracle VM Server for SPARC 2.1 Release Notes\*](#). To remove large amounts of memory from the control domain, see [“Decrease the Control Domain's Memory” on page 161](#).

---

## Resource Allocation

Starting with the Oracle VM Server for SPARC 2.0 release, the resource allocation mechanism uses resource allocation constraints and hints to assign resources to a domain at bind time.

A *resource allocation constraint* is a hard requirement that the system *must* meet when you assign a resource to a domain. If the constraint cannot be met, both the resource allocation and the binding of the domain fail.

A *resource allocation hint* is a soft requirement that the system *attempts* to meet when you assign a resource to a domain. The allocation of a resource can still succeed, and the domain can be bound even if the hint cannot be fully met. This situation can happen if the system can assign the resource in a way that does not necessarily meet the requirement.

## CPU Allocation

The CPU allocation mechanism uses the following constraints and hint for CPU resources:

- **Whole-core constraint.** This constraint specifies that virtual CPUs are allocated to a domain based on a specified number of CPU cores. The system must be able to allocate the specified number of cores and must also assign all the virtual CPUs of those allocated cores to the domain. If the system cannot allocate the specified number of cores, the domain fails to bind.
- **Maximum number of cores constraint.** This constraint specifies the maximum number of cores that can be assigned to a bound or active domain. This constraint is automatically enabled when the whole-core constraint is set on a domain. In that case, the maximum number of cores is automatically set to the number of cores configured when the domain is inactive. Currently, this constraint cannot be enabled independently of the whole-core constraint, and the maximum number of cores cannot be set manually.
- **Core affinity hint.** This hint requests that virtual CPUs allocated to a domain come from the same CPU cores or from the fewest number of CPU cores. The system makes its best effort to honor this request. The domain fails to bind *only* if insufficient free virtual CPUs are available on the system.

The core affinity hint is enabled by default and cannot be disabled.

---

**Note** – The whole-core constraint and the core affinity hint only address the location of a virtual CPU on cores. They do not address the location of a core on chips or of a chip on sockets.

---

## Enabling the Whole-Core Constraint

The whole-core constraint is automatically enabled when you specify the number of cores to assign to a domain. By default, you specify the virtual CPUs to assign to a domain. You can only

enable the whole-core constraint on an inactive domain, not on a domain that is bound or active. Before you enable the whole-core constraint on the control domain, you must first initiate a delayed reconfiguration.

Use the `ldm add-vcpu -c number`, `ldm set-vcpu -c number`, or `ldm remove-vcpu -c number` command to assign or remove CPU cores to and from a domain. *number* specifies the number of CPU cores and enables the whole-core constraint. For more information, see the [ldm\(1M\)](#) man page.

You can use the `ldm add-vcpu -c number` or `ldm remove-vcpu -c number` command on a domain that was previously configured with virtual CPUs. In that case, the existing number of virtual CPUs is automatically converted to the corresponding number of cores. This conversion is possible only if the existing number of virtual CPUs is a multiple of the number of virtual CPUs per core. If not, the conversion cannot be performed, and the command fails.

---

**Note** – If you use these commands to enable the whole-core constraint on an inactive domain or on the control domain in delayed reconfiguration mode, the maximum number of cores is also set. The maximum number of cores is not affected when you use these commands on a bound or active domain.

---

For example, a core is comprised of eight virtual CPUs. If a domain has seven virtual CPUs assigned, an `ldm add-vcpu -c` or `ldm remove-vcpu -c` command could not meet the whole-core constraint. Instead, you could use the `set-vcpu -c` command to specify the number of cores and to enable the whole-core constraint.

The following example enables the whole-core constraint on the inactive `ldg1` domain. The `ldm list` command verifies that the whole-core constraint is enabled.

```
primary# ldm add-vcpu -c 1 ldg1
primary# ldm list -o resmgmt ldg1
NAME
ldg1
CONSTRAINT
 whole-core
 max-cores=1
```

---

**Note** – When the whole-core constraint is enabled on a domain, the cryptographic units that are associated with those cores are unaffected by core additions. So, the system does not automatically add the associated cryptographic units to the domain. However, a cryptographic unit is automatically removed *only* when the last virtual CPU of the core is being removed. This action prevents a cryptographic unit from being unassociated with a virtual CPU.

---

## Disabling the Whole-Core Constraint

When a domain is assigned virtual CPUs instead of cores, the whole-core constraint is disabled. You can only disable the whole-core constraint on an inactive domain, not on a domain that is bound or active. Before you disable the whole-core constraint on the control domain, you must first initiate a delayed reconfiguration.

Use the `ldm add-vcpu number`, `ldm set-vcpu number`, or `ldm remove-vcpu number` command to assign or remove virtual CPUs to and from a domain. *number* specifies the number of virtual CPUs and disables the whole-core constraint. For more information, see the [ldm\(1M\)](#) man page.

You can use the `ldm add-vcpu number` or `ldm rm-vcpu number` command on a domain that was previously configured with CPU cores. In that case, the existing number of CPU cores is automatically converted to the corresponding number of virtual CPUs.

---

**Note** – When you disable the whole-core constraint, the maximum core constraint is also automatically disabled.

---

The following example disables the whole-core constraint on the inactive `ldg1` domain:

```
primary# ldm set-vcpu 1 ldg1
```

## Allocating CPUs to the Control Domain

To enable the whole-core constraint on the control domain, the control domain must be in delayed reconfiguration mode. Enabling the whole-core constraint on the control domain only succeeds if sufficient CPU cores are available to meet the requested constraint. That is, unused cores, cores that are already used by the control domain, or cores that are partially used by the control domain must be available. Otherwise, the CPU allocation on the control domain remains unchanged.

---

**Note** – When the control domain is in delayed reconfiguration mode, the whole-core constraint and the setting of the number of cores also specifies the maximum number of cores.

---

The following example enables the whole-core constraint on the control domain (`primary`). First, initiate a delayed reconfiguration on the control domain. Next, assign one whole core to the control domain, and then reboot the domain to make the changes take effect.

```
primary# ldm start-reconf primary
Initiating a delayed reconfiguration operation on the primary domain.
All configuration changes for other domains are disabled until the
```

primary domain reboots, at which time the new configuration for the primary domain also takes effect.

```
primary# ldm add-vcpu -c 1 primary
primary# reboot
```

## Interactions Between the Whole-Core Constraint and Other Domain Features

This section describes the interactions between the whole-core constraint and the following features:

- [“CPU Dynamic Reconfiguration” on page 158](#)
- [“Dynamic Resource Management” on page 158](#)
- [“Domain Migration” on page 159](#)
- [“Power Management” on page 159](#)

### CPU Dynamic Reconfiguration

The whole-core constraint is fully compatible with CPU dynamic reconfiguration (DR). When a domain is defined with the whole-core constraint, you can use the `ldm add-vcpu -c`, `ldm set-vcpu -c`, or `remove-vcpu -c` command to change the number of cores on an active domain.

However, if a bound or active domain is not in delayed reconfiguration mode, its number of cores cannot exceed the maximum number of cores. This maximum is set with the maximum core constraint, which is automatically enabled when the whole-core constraint is enabled. Any CPU DR operation that does not satisfy the maximum core constraint fails.

### Dynamic Resource Management

The whole-core constraint is not compatible with dynamic resource management (DRM). When a DRM policy is enabled on a domain that uses the whole-core constraint, the policy is automatically disabled. The whole-core constraint remains enabled.

Even though a DRM policy cannot be enabled when the whole-core constraint is in effect, you can still define a DRM policy for the domain. Note that when a policy is automatically disabled, it still remains active. The policy is automatically re-enabled if the domain is restarted without the whole-core constraint.

The following are the expected interactions between the whole-core constraint and DRM:

- If the whole-core constraint is set on a domain, a warning message is issued when you attempt to enable a DRM policy on that domain.
- If a DRM policy is in effect on an inactive domain, you are permitted to enable the whole-core constraint on the domain. When the domain becomes active and the policy is enabled, the system automatically disables the DRM policy for the domain.
- If a DRM policy is enabled on an active or bound domain, you are not permitted to enable the whole-core constraint.

## Domain Migration

CPU whole-core configuration is incompatible with domain migration. However, you can still migrate a domain that is configured with CPU whole cores. To restore the whole-core constraint after such a migration, stop the domain and reconfigure it for whole-core allocation.

## Power Management

The whole-core constraint is fully compatible with the power management (PM) performance and elastic modes. When elastic mode is enabled, the PM subsystem can add or remove CPU cores to or from domains that are configured with the whole-core constraint. In that case, the whole-core constraint continues to be honored, and domains that use that constraint remain configured only with whole cores.

# Using Memory Dynamic Reconfiguration

The Oracle VM Server for SPARC 2.0 release introduces memory dynamic reconfiguration (DR). This feature is capacity-based and enables you to add or remove an arbitrary amount of memory to or from an active logical domain.

The following are the requirements and restrictions for using the memory DR feature:

- You can perform memory DR operations on any domain. However, only a single memory DR operation can be in progress on a domain at a given time.
- The memory DR feature enforces 256-Mbyte alignment on the address and size of the memory involved in a given operation. See [“Memory Alignment” on page 161](#).
- Unaligned memory in the free memory pool *cannot* be assigned to a domain by using the memory DR feature. See [“Adding Unaligned Memory” on page 162](#).

If the memory of a domain cannot be reconfigured by using a memory DR operation, the domain must be stopped before the memory can be reconfigured. If the domain is the control domain, you must first initiate a delayed reconfiguration.

## Adding Memory

If a domain is active, you can use the `ldm add-memory` command to dynamically add memory to the domain. The `ldm set-memory` command can also dynamically add memory if the specified memory size is greater than the current memory size of the domain.

## Removing Memory

If a domain is active, you can use the `ldm remove-memory` command to dynamically remove memory from the domain. The `ldm set-memory` command can also dynamically remove memory if the specified memory size is smaller than the current memory size of the domain.

Memory removal can be a long-running operation. You can track the progress of an operation or cancel an ongoing memory DR request.

### Tracking the Progress of a Memory DR Request

You can track the progress of an `ldm remove-memory` command by running the `ldm list -l` command for the specified domain.

### Canceling a Memory DR Request

You can cancel a removal request that is in progress by interrupting the `ldm remove-memory` command (by pressing Control-C) or by issuing the `ldm cancel-operation memdr` command. If you cancel a memory removal request, only the outstanding portion of the removal request is affected, namely, the amount of memory still to be removed from the domain.

## Partial Memory DR Requests

A memory addition request is rejected if there is insufficient free memory to fulfill the entire request. However, a memory addition request can be partially fulfilled if the target domain fails to add any of the memory requested by the Logical Domains Manager.

A memory removal request is rejected if there is insufficient memory in the domain to fulfill the entire request. However, a memory removal request can be partially fulfilled if the target domain fails to remove any of the memory requested by the Logical Domains Manager.

---

**Note** – Memory is cleared after it is removed from a domain and before it is added to another domain.

---

## Memory Reconfiguration of the Control Domain

The memory DR feature can be used to reconfigure the memory of the control domain. If a memory DR request cannot be performed on the control domain, you must first initiate a delayed reconfiguration.

Using memory DR might not be appropriate for removing large amounts of memory from an active domain because memory DR operations might be long running. In particular, during the initial configuration of the system, you should use delayed reconfiguration to decrease the memory in the control domain.

### Decrease the Control Domain's Memory

Use a delayed reconfiguration instead of a memory DR to decrease the control domain's memory from an initial factory default configuration. In such a case, the control domain owns all of the host system's memory. The memory DR feature is not well suited for this purpose because an active domain is not guaranteed to add or more typically give up all of the requested memory. Rather, the OS running in that domain makes a best effort to fulfill the request. In addition, memory removal can be a long-running operation. These issues are amplified when large memory operations are involved, as is the case for the initial decrease of the control domain's memory.

For these reasons, use a delayed reconfiguration by following these steps:

1. Use the `ldm start -reconf primary` command to put the control domain in delayed reconfiguration mode.
2. Partition the host system's resources that are owned by the control domain, as necessary.
3. Use the `ldm cancel -reconf` command to undo the operations in Step 2, if necessary, and start over.
4. Reboot the control domain to make the reconfiguration changes take effect.

## Dynamic and Delayed Reconfiguration

If a delayed reconfiguration is pending in the control domain, a memory reconfiguration request is rejected for any other domain. If a delayed reconfiguration is not pending in the control domain, a memory reconfiguration request is rejected for any domain that does not support memory DR. A memory reconfiguration request on a control domain that does not support memory DR is converted to a delayed reconfiguration request.

## Memory Alignment

Memory reconfiguration requests have different alignment requirements that depend on the state of the domain to which the request is applied.

## Memory Alignment for Active Domains

- **Dynamic addition and removal.** The address and size of a memory block are 256-Mbyte-aligned for dynamic addition and dynamic removal. The minimum operation size is 256 Mbytes.

A nonaligned request or a removal request that is larger than the bound size is rejected.

Use the following commands to adjust memory allocations:

- `ldm add-memory`. If you specify the `--auto-adj` option with this command, the amount of memory to be added is 256-Mbyte-aligned, which might increase the amount of memory actually added to the domain.
- `ldm remove-memory`. If you specify the `--auto-adj` option with this command, the amount of memory to be removed is 256-Mbyte-aligned, which might decrease the amount of memory actually removed from the domain.
- `ldm set-memory`. This command is treated as an addition or a removal operation. If you specify the `--auto-adj` option, the amount of memory to be added or removed is 256-Mbyte-aligned as previously described. Note that this alignment might increase the resulting memory size of the domain.
- **Delayed reconfiguration.** The address and size of a memory block are 4-Mbyte-aligned. If you make a nonaligned request, the request is rounded up to be 4-Mbyte-aligned.

## Memory Alignment for Bound Domains

The address and size of a memory block are 4-Mbyte-aligned for bound domains. If you make a nonaligned request, the request is rounded up to be 4-Mbyte-aligned. This means that the resulting memory size of the domain might be more than you specified.

For the `ldm add-memory`, `ldm set-memory`, and `ldm remove-memory` commands, the `--auto-adj` option rounds up the size of the resulting memory to be 256-Mbyte-aligned. This means that the resulting memory might be more than you specified.

## Memory Alignment for Inactive Domains

For the `ldm add-memory`, `ldm set-memory`, and `ldm remove-memory` commands, the `--auto-adj` option rounds up the size of the resulting memory to be 256-Mbyte-aligned. There is no alignment requirement for an inactive domain. The restrictions described in [“Memory Alignment for Bound Domains” on page 162](#) take effect after such a domain is bound.

## Adding Unaligned Memory

The memory DR feature enforces 256-Mbyte memory alignment on the address and size of the memory that is dynamically added to or removed from an active domain. This means that any unaligned memory in an active domain cannot be removed by using memory DR.

This also means than any unaligned memory in the free memory pool cannot be added to an active domain by using memory DR.

After all the aligned memory has been allocated, you can use the `ldm add-memory` command to add the remaining unaligned memory to a bound or inactive domain. You can also use this command to add the remaining unaligned memory to the control domain by means of a delayed reconfiguration operation.

The following example shows how to add the two remaining 128-Mbyte memory blocks to the primary and `ldom1` domains. The `ldom1` domain is in the bound state. The following commands add the two remaining memory blocks. The first command initiates a delayed reconfiguration operation on the control domain. The second command adds one of the 128-Mbyte memory blocks to the control domain. The fifth command adds the other 128-Mbyte memory block to the `ldom1` domain.

```
ldm start-reconf primary
Initiating a delayed reconfiguration operation on the primary domain.
All configuration changes for other domains are disabled until the
primary domain reboots, at which time the new configuration for the
primary domain also takes effect.

ldm add-memory 128M primary

Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.

ldm list
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -ndcv- SP 8 2688M 0.1% 23d 8h 8m

ldm list
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -n-cv- SP 8 2560M 0.5% 23d 8h 9m
ldom1 bound - - - - - 5000 1 524M

ldm add-mem 128M ldom1
ldm list
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -n-cv- SP 8 2560M 0.1% 23d 8h 9m
ldom1 bound - - - - - 5000 1 652M
```

## Memory DR Examples

The following examples show how to perform memory DR operations. For information about the related CLI commands, see the `ldm(1M)` man page.

### EXAMPLE 10-1 Memory DR Operations on Active Domains

This example shows how to dynamically add memory to and remove it from an active domain, `ldom1`.

**EXAMPLE 10-1** Memory DR Operations on Active Domains *(Continued)*

The `ldm list` output shows the memory for each domain in the Memory field. The first `ldm add-mem` command exits with an error because you must specify memory in multiples of 256 Mbytes. The next `ldm add-mem` command uses the `--auto-adj` option so that even though you specify 200M as the amount of memory to add, the amount is rounded up to 256 Mbytes.

The `ldm rm-mem` command exits with an error because you must specify memory in multiples of 256 Mbytes. When you add the `--auto-adj` option to the same command, the memory removal succeeds, as the amount of memory is rounded down to the next 256-Mbyte boundary.

```
ldm list
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -n-cv- SP 4 27392M 0.4% 1d 22h 53m
ldom1 active -n---- 5000 2 2G 0.4% 1d 1h 23m
ldom2 bound ------ 5001 2 200M
```

```
ldm add-mem 200M ldom1
The size of memory must be a multiple of 256MB.
```

```
ldm add-mem --auto-adj 200M ldom1
Adjusting request size to 256M.
The ldom1 domain has been allocated 56M more memory
than requested because of memory alignment constraints.
```

```
ldm list
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -n-cv- SP 4 27392M 5.0% 8m
ldom1 active -n---- 5000 2 2304M 0.5% 1m
ldom2 bound ------ 5001 2 200M
```

```
ldm rm-mem --auto-adj 300M ldom1
Adjusting requested size to 256M.
The ldom1 domain has been allocated 44M more memory
than requested because of memory alignment constraints.
```

```
ldm list
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -n-cv- SP 4 27392M 0.3% 8m
ldom1 active -n---- 5000 2 2G 0.2% 2m
ldom2 bound ------ 5001 2 200M
```

**EXAMPLE 10-2** Memory DR Operations on Bound Domains

This example shows how to add memory to and remove it from a bound domain, `ldom2`.

The `ldm list` output shows the memory for each domain in the Memory field. The first `ldm add-mem` command adds 100 Mbytes of memory to the `ldom2` domain. The next `ldm add-mem` command specifies the `--auto-adj` option, which causes an additional 112 Mbytes of memory to be dynamically added to `ldom2`.

The `ldm rm-mem` command dynamically removes 100 Mbytes from the `ldom2` domain. If you specify the `--auto-adj` option to the same command to remove 300 Mbytes of memory, the amount of memory is rounded down to the next 256-Mbyte boundary.

**EXAMPLE 10-2** Memory DR Operations on Bound Domains (Continued)

```
ldm list
```

| NAME    | STATE  | FLAGS  | CONS | VCPU | MEMORY | UTIL | UPTIME     |
|---------|--------|--------|------|------|--------|------|------------|
| primary | active | -n-cv- | SP   | 4    | 27392M | 0.4% | 1d 22h 53m |
| ldom1   | active | -n---- | 5000 | 2    | 2G     | 0.4% | 1d 1h 23m  |
| ldom2   | bound  | -----  | 5001 | 2    | 200M   |      |            |

```
ldm add-mem 100M ldom2
```

```
ldm list
```

| NAME    | STATE  | FLAGS  | CONS | VCPU | MEMORY | UTIL | UPTIME     |
|---------|--------|--------|------|------|--------|------|------------|
| primary | active | -n-cv- | SP   | 4    | 27392M | 0.5% | 1d 22h 54m |
| ldom1   | active | -n---- | 5000 | 2    | 2G     | 0.2% | 1d 1h 25m  |
| ldom2   | bound  | -----  | 5001 | 2    | 300M   |      |            |

```
ldm add-mem --auto-adj 100M ldom2
```

Adjusting request size to 256M.  
The ldom2 domain has been allocated 112M more memory  
than requested because of memory alignment constraints.

```
ldm list
```

| NAME    | STATE  | FLAGS  | CONS | VCPU | MEMORY | UTIL | UPTIME     |
|---------|--------|--------|------|------|--------|------|------------|
| primary | active | -n-cv- | SP   | 4    | 27392M | 0.4% | 1d 22h 55m |
| ldom1   | active | -n---- | 5000 | 2    | 2G     | 0.5% | 1d 1h 25m  |
| ldom2   | bound  | -----  | 5001 | 2    | 512M   |      |            |

```
ldm rm-mem 100M ldom2
```

```
ldm list
```

| NAME    | STATE  | FLAGS  | CONS | VCPU | MEMORY | UTIL | UPTIME     |
|---------|--------|--------|------|------|--------|------|------------|
| primary | active | -n-cv- | SP   | 4    | 27392M | 3.3% | 1d 22h 55m |
| ldom1   | active | -n---- | 5000 | 2    | 2G     | 0.2% | 1d 1h 25m  |
| ldom2   | bound  | -----  | 5001 | 2    | 412M   |      |            |

```
ldm rm-mem --auto-adj 300M ldom2
```

Adjusting request size to 256M.  
The ldom2 domain has been allocated 144M more memory  
than requested because of memory alignment constraints.

```
ldm list
```

| NAME    | STATE  | FLAGS  | CONS | VCPU | MEMORY | UTIL | UPTIME     |
|---------|--------|--------|------|------|--------|------|------------|
| primary | active | -n-cv- | SP   | 4    | 27392M | 0.5% | 1d 22h 55m |
| ldom1   | active | -n---- | 5000 | 2    | 2G     | 0.2% | 1d 1h 26m  |
| ldom2   | bound  | -----  | 5001 | 2    | 256M   |      |            |

**EXAMPLE 10-3** Setting Domain Memory Sizes

This example shows how to use the `ldm set-memory` command to add memory to and remove it from a domain.

The `ldm list` output shows the memory for each domain in the Memory field. The first `ldm set-mem` command attempts to set the primary domain's size to 3400 Mbytes. The resulting error states that the specified value is not on a 256-Mbyte boundary. Adding the `--auto-adj` option to the same command enables you to successfully remove some memory and stay on the 256-Mbyte boundary. This command also issues a warning to state that not all of the requested memory could be removed as the domain is using that memory.

**EXAMPLE 10-3** Setting Domain Memory Sizes (Continued)

The next `ldm set-mem` command sets the memory size of the `ldom2` domain, which is in the bound state, to 690 Mbytes. If you add the `--auto-adj` option to the same command, an additional 78 Mbytes of memory is dynamically added to `ldom2` to stay on a 256-Mbyte boundary.

```
ldm list
```

| NAME    | STATE  | FLAGS  | CONS | VCPU | MEMORY | UTIL | UPTIME     |
|---------|--------|--------|------|------|--------|------|------------|
| primary | active | -n-cv- | SP   | 4    | 27392M | 0.5% | 1d 22h 55m |
| ldom1   | active | -n---- | 5000 | 2    | 2G     | 0.2% | 1d 1h 26m  |
| ldom2   | bound  | -----  | 5001 | 2    | 256M   |      |            |

```
ldm set-mem 3400M primary
```

An `ldm set-mem 3400M` command would remove 23992MB, which is not a multiple of 256MB. Instead, run `ldm rm-mem 23808MB` to ensure a 256MB alignment.

```
ldm set-mem --auto-adj 3400M primary
```

Adjusting request size to 3.4G.

The primary domain has been allocated 184M more memory than requested because of memory alignment constraints. Only 9472M of memory could be removed from the primary domain because the rest of the memory is in use.

```
ldm set-mem 690M ldom2
```

```
ldm list
```

| NAME    | STATE  | FLAGS  | CONS | VCPU | MEMORY | UTIL | UPTIME     |
|---------|--------|--------|------|------|--------|------|------------|
| primary | active | -n-cv- | SP   | 4    | 17920M | 0.5% | 1d 22h 56m |
| ldom1   | active | -n---- | 5000 | 2    | 2G     | 0.6% | 1d 1h 27m  |
| ldom2   | bound  | -----  | 5001 | 2    | 690M   |      |            |

```
ldm set-mem --auto-adj 690M ldom2
```

Adjusting request size to 256M.

The `ldom2` domain has been allocated 78M more memory than requested because of memory alignment constraints.

```
ldm list
```

| NAME    | STATE  | FLAGS  | CONS | VCPU | MEMORY | UTIL | UPTIME     |
|---------|--------|--------|------|------|--------|------|------------|
| primary | active | -n-cv- | SP   | 4    | 17920M | 2.1% | 1d 22h 57m |
| ldom1   | active | -n---- | 5000 | 2    | 2G     | 0.2% | 1d 1h 27m  |
| ldom2   | bound  | -----  | 5001 | 2    | 768M   |      |            |

## Using Power Management

To use Power Management (PM), you first need to set the PM mode in the Oracle Integrated Lights Out Manager (ILOM) 3.0 firmware. This section summarizes the information that you need to be able to use PM with the Oracle VM Server for SPARC software.

For more information about ILOM, see the following:

- “Monitoring Power Consumption” in the *Sun Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide*
- *Oracle Integrated Lights Out Manager (ILOM) 3.0 Feature Updates and Release Notes*

The power mode is the setting that governs system power usage at any point in time. The following power modes are supported, assuming that the underlying platform has implemented PM features:

- **Performance mode.** The system is allowed to use all the power that is available.
- **Elastic mode.** The system power usage is adapted to the current utilization level. For example, the power state of resources is reduced as utilization decreases.

Following are the PM features:

- **CPU core auto-disabling.** PM automatically disables a CPU core when all the strands on that core have been disabled.
- **CPU clock cycle skip.** Starting with the Oracle VM Server for SPARC 2.0 release, PM can automatically adjust the CPU clock cycle skip on SPARC T3 platforms. The adjustment can increase or decrease the number of clock cycles that are skipped to keep all domains within the power utilization thresholds. PM determines whether to make such adjustments based on CPU utilization. When the system goes in to performance mode, the number of clock cycles that are skipped is automatically adjusted to being none.
- **Memory operations in deep idle mode.** Starting with the Oracle VM Server for SPARC 2.0 release, when SPARC T3 platforms are in elastic mode, they automatically configure under-utilized memory to operate in a deeper idle mode to save power.
- **Power limit.** Starting with the Oracle VM Server for SPARC 2.1 release, you can set a *power limit* on SPARC T3 platforms to restrict the power draw of a system. If the power draw is greater than the power limit, PM techniques are used to reduce power. You can use the ILOM service processor (SP) to set the power limit.

See the following documents:

- *Sun Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide*
- *Oracle Integrated Lights Out Manager (ILOM) 3.0 Feature Updates and Release Notes*

You can use the ILOM interface to set a power limit, grace period, and violation action. If the power limit is exceeded for more than the grace period, the violation action is performed.

If the current power draw exceeds the power limit, an attempt is made to reduce the power state of resources that can be power-managed. If the power draw drops below the power limit, the power state of those resources is permitted to increase. If the system is in elastic mode, an increase in the power state of resources is driven by the utilization level.

When a system is in elastic mode, some domain configuration modifications are first validated to confirm that the power limit is not exceeded. If the power limit is exceeded, only some of the resources might be modified or added as requested. If the power limit is later increased, you can then add any resources that were unsuccessfully modified.

If a domain's load causes resources to consume more power, only the resources that maintain the power draw being under the power limit are successfully powered up.

For instructions on configuring the power mode by using the ILOM 3.0 firmware CLI, see “Monitoring Power Consumption” in the *Sun Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide*.

# Listing Power-Managed CPU Strands and Virtual CPUs

This section shows how to list power-managed strands and virtual CPUs.

## ▼ List Power-Managed CPU Strands

- List power-managed strands by using one of the following commands:

- Use the `list -l` subcommand.

```
ldm list -l primary
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -n-cv- UART 64 16G 1.0% 21h 33m

SOFTSTATE
Solaris running

UUID
b9288150-327f-44f7-8c64-d4d57b92e524

MAC
00:21:28:8f:8f:34

HOSTID
0x858f8f34

CONTROL
failure-policy=ignore

DEPENDENCY
master=

CORE
CID CPUSSET
0 (0, 1, 2, 3, 4, 5, 6, 7)
1 (8, 9, 10, 11, 12, 13, 14, 15)
2 (16, 17, 18, 19, 20, 21, 22, 23)
3 (24, 25, 26, 27, 28, 29, 30, 31)
4 (32, 33, 34, 35, 36, 37, 38, 39)
```

```

5 (40, 41, 42, 43, 44, 45, 46, 47)
6 (48, 49, 50, 51, 52, 53, 54, 55)
7 (56, 57, 58, 59, 60, 61, 62, 63)

```

```

VCPU
VID PID CID UTIL STRAND
0 0 0 1.2% 100%
1 1 0 1.5% 100%
2 2 0 0.1% 100%
3 3 0 0.2% 100%

```

```

.
.
.

```

The following output shows dashes ( - - - ) in the UTIL column for the CPU, which means that the strand is power-managed. The dashes are shown only for domains other than the primary domain.

```

ldm list -l ldg1
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
ldg1 active -n--v- 5000 64 16G 1.1% 20h 55m

```

```

SOFTSTATE
Solaris running

```

```

UUID
98d86371-24f6-4792-c631-eb14e81ad4a0

```

```

MAC
00:14:4f:f9:02:f2

```

```

HOSTID
0x84f902f2

```

```

CONTROL
failure-policy=ignore

```

```

DEPENDENCY
master=

```

```

CORE
CID CPuset
8 (64, 65, 66, 67, 68, 69, 70, 71)
9 (72, 73, 74, 75, 76, 77, 78, 79)
10 (80, 81, 82, 83, 84, 85, 86, 87)
11 (88, 89, 90, 91, 92, 93, 94, 95)
12 (96, 97, 98, 99, 100, 101, 102, 103)
13 (104, 105, 106, 107, 108, 109, 110, 111)
14 (112, 113, 114, 115, 116, 117, 118, 119)
15 (120, 121, 122, 123, 124, 125, 126, 127)

```

```

VCPU
VID PID CID UTIL STRAND
0 64 8 0.8% 100%
1 65 8 2.0% 100%
2 66 8 ---- 100%
3 67 8 ---- 100%
4 68 8 ---- 100%

```

```

.

```

- Use the parseable option (-p) to the `list -l` subcommand.

In the output, a blank after `util=` means that the strand is power-managed.

```
ldm list -l -p
```

```
VCPU
|vid=0|pid=0|util=0.7%|strand=100
|vid=1|pid=1|util=|strand=100
|vid=2|pid=2|util=|strand=100
|vid=3|pid=3|util=|strand=100
|vid=4|pid=4|util=0.7%|strand=100
|vid=5|pid=5|util=|strand=100
|vid=6|pid=6|util=|strand=100
|vid=7|pid=7|util=|strand=100
```

## ▼ List Power-Managed CPUs

- List power-managed CPUs by using one of the following commands:

- a. Use the `list-devices -a cpu` command.

In the PM column of the output, a yes means that the CPU is power-managed, and a no means that the CPU is powered on. It is assumed that 100 percent free CPUs are power-managed by default, hence the dashes (---) under the PM column for them.

```
ldm list-devices -a cpu
```

```
VCPU
 PID %FREE PM
 0 0 no
 1 0 yes
 2 0 yes
 3 0 yes
 4 100 ---
 5 100 ---
 6 100 ---
 7 100 ---
```

- b. Use the parseable option (-p) to the `list-devices -a cpu` subcommand.

In the `pm=` field in the output, a yes means that the CPU is power-managed, and a no means that the CPU is powered on. It is assumed that 100 percent free CPUs are power-managed by default, hence the blank in that field for them.

```
ldm list-devices -a -p cpu
```

```
VERSION 1.4
VCPU
|pid=0|free=0|pm=no
|pid=1|free=0|pm=yes
|pid=2|free=0|pm=yes
|pid=3|free=0|pm=yes
|pid=4|free=0|pm=no
|pid=5|free=0|pm=yes
```

```
|pid=6|free=0|pm=yes
|pid=7|free=0|pm=yes
|pid=8|free=100|pm=
|pid=9|free=100|pm=
|pid=10|free=100|pm=
```

## Using Dynamic Resource Management

You can use policies to determine how to automatically perform DR activities. At this time, you can *only* create policies to govern the dynamic resource management of virtual CPUs.



**Caution** – The following restrictions affect CPU dynamic resource management (DRM):

- When PM is in elastic mode, DRM cannot be enabled.
- Any change from performance mode to elastic mode is delayed while DRM is enabled.
- Ensure that you disable CPU DRM prior to performing a domain migration operation.
- DRM policies do not apply to domains that are configured with the whole-core constraint.

A *resource management policy* specifies under what conditions virtual CPUs can be automatically added to and removed from a logical domain. A policy is managed by using the `ldm add-policy`, `ldm set-policy`, and `ldm remove-policy` commands:

```
ldm add-policy [enable=yes|no] [priority=value] [attack=value] [decay=value]
[elastic-margin=value] [sample-rate=value] [tod-begin=hh:mm[:ss]]
[tod-end=hh:mm[:ss]] [util-lower=percent] [util-upper=percent] [vcpu-min=value]
[vcpu-max=value] name=policy-name ldom...
ldm set-policy [enable=[yes|no]] [priority=[value]] [attack=[value]] [decay=[value]]
[elastic-margin=[value]] [sample-rate=[value]] [tod-begin=[hh:mm:ss]]
[tod-end=[hh:mm:ss]] [util-lower=[percent]] [util-upper=[percent]] [vcpu-min=[value]]
[vcpu-max=[value]] name=policy-name ldom...
ldm remove-policy [name=]policy-name... ldom
```

For information about these commands and about creating resource management policies, see the [ldm\(1M\)](#) man page.

A policy is in effect during the times specified by the `tod-begin` and `tod-end` properties. The time specified by `tod-begin` must be earlier than the time specified by `tod-end` in 24-hour period. By default, values for the `tod-begin` and `tod-end` properties are 00:00:00 and 23:59:59, respectively. When the default values are used, the policy is always in effect.

The policy uses the value of the `priority` property to specify a priority for a dynamic resource management (DRM) policy. Priority values are used to determine the relationship between DRM policies on a single domain and between DRM-enabled domains on a single system. Lower numerical values represent higher (better) priorities. Valid values are between 1 and 9999. The default value is 99.

The behavior of the `priority` property depends on the availability of a pool of free CPU resources, as follows:

- **Free CPU resources are available in the pool.** In this case, the `priority` property determines which DRM policy will be in effect when more than one overlapping policy is defined for a single domain.
- **No free CPU resources are available in the pool.** In this case, the `priority` property specifies whether a resource can be dynamically moved from a lower-priority domain to a higher-priority domain on the same system. The priority of a domain is the priority specified by the DRM policy that is in effect for that domain.

For example, a higher-priority domain can acquire CPU resources from another domain that has a DRM policy with a lower priority. This resource-acquisition capability pertains *only* to domains that have DRM policies enabled. Domains that have equal priority values are unaffected by this capability. So, if the default priority is used for all policies, domains cannot obtain resources from lower-priority domains. To take advantage of this capability, adjust the `priority` property values so that they have unequal values.

For example, the `ldg1` and `ldg2` domains both have DRM policies in effect. The `priority` property for the `ldg1` domain is 1, which is more favorable than the `priority` property value of the `ldg2` domain (2). The `ldg1` domain can dynamically remove a CPU resource from the `ldg2` domain and assign it to itself in the following circumstances:

- The `ldg1` domain requires another CPU resource
- The pool of free CPU resources has been exhausted

The policy uses the `util-high` and `util-low` property values to specify the high and low thresholds for CPU utilization. If the utilization exceeds the value of `util-high`, virtual CPUs are added to the domain until the number is between the `vcpu-min` and `vcpu-max` values. If the utilization drops below the `util-low` value, virtual CPUs are removed from the domain until the number is between the `vcpu-min` and `vcpu-max` values. If `vcpu-min` is reached, no more virtual CPUs can be dynamically removed. If the `vcpu-max` is reached, no more virtual CPUs can be dynamically added.

#### EXAMPLE 10-4 Adding Resource Management Policies

For example, after observing the typical utilization of your systems over several weeks, you might set up policies to optimize resource usage. The highest usage is daily from 9:00 a.m. to 6:00 p.m. Pacific, and the low usage is daily from 6:00 p.m. to 9:00 a.m. Pacific.

Based on this system utilization observation, you decide to create the following high and low policies based on overall system utilization:

- **High:** Daily from 9:00 a.m. to 6:00 p.m. Pacific
- **Low:** Daily from 6:00 p.m. to 9:00 a.m. Pacific

The following `ldm add-policy` command creates the high-usage policy to be used during the high utilization period on the `ldom1` domain.

**EXAMPLE 10-4** Adding Resource Management Policies (Continued)

The following high-usage policy does the following:

- Specifies that the beginning and ending times are 9:00 a.m. and 6:00 p.m. by setting the `tod-begin` and `tod-end` properties, respectively.
- Specifies that the lower and upper limits at which to perform policy analysis are 25 percent and 75 percent by setting the `util-lower` and `util-upper` properties, respectively.
- Specifies that the minimum and maximum number of virtual CPUs is 2 and 16 by setting the `vcpu-min` and `vcpu-max` properties, respectively.
- Specifies that the maximum number of virtual CPUs to be added during any one resource control cycle is 1 by setting the `attack` property.
- Specifies that the maximum number of virtual CPUs to be removed during any one resource control cycle is 1 by setting the `decay` property.
- Specifies that the priority of this policy is 1 by setting the `priority` property. A priority of 1 means that this policy will be enforced even if another policy can take effect.
- Specifies that the name of the policy file is `high-usage` by setting the `name` property.
- Uses the default values for those properties that are not specified, such as `enable` and `sample-rate`. See the `ldm(1M)` man page.

```
ldm add-policy tod-begin=09:00 tod-end=18:00 util-lower=25 util-upper=75 \
vcpu-min=2 vcpu-max=16 attack=1 decay=1 priority=1 name=high-usage ldom1
```

The following `ldm add-policy` command creates the `med-usage` policy to be used during the low utilization period on the `ldom1` domain.

The following `med-usage` policy does the following:

- Specifies that the beginning and ending times are 6:00 p.m. and 9:00 a.m. by setting the `tod-begin` and `tod-end` properties, respectively.
- Specifies that the lower and upper limits at which to perform policy analysis are 10 percent and 50 percent by setting the `util-lower` and `util-upper` properties, respectively.
- Specifies that the minimum and maximum number of virtual CPUs is 2 and 16 by setting the `vcpu-min` and `vcpu-max` properties, respectively.
- Specifies that the maximum number of virtual CPUs to be added during any one resource control cycle is 1 by setting the `attack` property.
- Specifies that the maximum number of virtual CPUs to be removed during any one resource control cycle is 1 by setting the `decay` property.
- Specifies that the priority of this policy is 1 by setting the `priority` property. A priority of 1 means that this policy will be enforced even if another policy can take effect.
- Specifies that the name of the policy file is `high-usage` by setting the `name` property.

**EXAMPLE 10-4** Adding Resource Management Policies (Continued)

- Uses the default values for those properties that are not specified, such as `enable` and `sample-rate`. See the [ldm\(1M\)](#) man page.

```
ldm add-policy tod-begin=18:00 tod-end=09:00 util-lower=10 util-upper=50 \
vcpu-min=2 vcpu-max=16 attack=1 decay=1 priority=1 name=med-usage ldom1
```

## Listing Domain Resources

This section shows the syntax usage for the `ldm` subcommands, defines some output terms, such as flags and utilization statistics, and provides examples that are similar to what you actually see as output.

## Machine-Readable Output

If you are creating scripts that use `ldm list` command output, *always* use the `-p` option to produce the machine-readable form of the output. See “[Generate a Parseable, Machine-Readable List \(-p\)](#)” on page 176 for more information.

### ▼ Show Syntax Usage for ldm Subcommands

- Look at syntax usage for all `ldm` subcommands.

```
primary# ldm --help
```

For more information about the `ldm` subcommands, see the [ldm\(1M\)](#) man page.

## Flag Definitions

The following flags can be shown in the output for a domain (`ldm list`). If you use the long, parseable options (`-l -p`) for the command, the flags are spelled out; for example, `flags=normal`, `control`, `vio-service`. If not, you see the letter abbreviation; for example `-n-cv-`. The list flag values are position dependent. Following are the values that can appear in each of the six columns from left to right.

### Column 1

- `s` starting or stopping
- `-` placeholder

**Column 2**

- n normal
- t transition

**Column 3**

- d delayed reconfiguration
- r memory dynamic reconfiguration (DR)
- - placeholder

**Column 4**

- c control domain
- - placeholder

**Column 5**

- v virtual I/O service domain
- - placeholder

**Column 6**

- s source domain in a migration
- t target domain in a migration
- e error occurred during a migration
- - placeholder

## Utilization Statistic Definition

The per virtual CPU utilization statistic (UTIL) is shown on the long (-l) option of the `ldm list` command. The statistic is the percentage of time that the virtual CPU spent executing on behalf of the guest operating system. A virtual CPU is considered to be executing on behalf of the guest operating system except when it has been yielded to the hypervisor. If the guest operating system does not yield virtual CPUs to the hypervisor, the utilization of CPUs in the guest operating system will always show as 100%.

The utilization statistic reported for a logical domain is the average of the virtual CPU utilizations for the virtual CPUs in the domain. A dash (- - -) in the UTIL column means that the strand is power-managed.

## Viewing Various Lists

### ▼ Show Software Versions (-V)

- View the current software versions installed.

```
primary# ldm -V
```

### ▼ Generate a Short List

- Generate a short list for all domains.

```
primary# ldm list
```

### ▼ Generate a Long List (-l)

- Generate a long list for all domains.

```
primary# ldm list -l
```

### ▼ Generate an Extended List (-e)

- Generate an extended list of all domains.

```
primary# ldm list -e
```

### ▼ Generate a Parseable, Machine-Readable List (-p)

- Generate a parseable, machine-readable list of all domains.

```
primary# ldm list -p
```

### ▼ Generate a Subset of a Long List (-o format)

- Generate output as a subset of resources by entering one or more of the following *format* options. If you specify more than one format, delimit the items by a comma with no spaces.

```
primary# ldm list -o resource[,resource...] ldom
```

- console – Output contains virtual console (vcons) and virtual console concentrator (vcc) service
- core – Output contains information about domains that have whole cores allocated
- cpu – Output contains information about the virtual CPU (vcpu), physical CPU (pcpu), and core ID
- crypto – Cryptographic unit output contains Modular Arithmetic Unit (mau) and any other supported cryptographic unit, such as the Control Word Queue (CWQ)

- **disk** – Output contains virtual disk (**vdisk**) and virtual disk server (**vds**)
- **domain** – Output contains variables (**var**), host ID (**hostid**), domain state, flags, UUID, and software state
- **memory** – Output contains memory
- **network** – Output contains media access control (**mac**) address, virtual network switch (**vsw**), and virtual network (**vnet**) device
- **physio** – Physical input/output contains peripheral component interconnect (**pci**) and network interface unit (**niu**)
- **resmgmt** – Output contains dynamic resource management (DRM) policy information, indicates which policy is currently running, and lists constraints related to whole-core configuration
- **serial** – Output contains virtual logical domain channel (**vlcdc**) service, virtual logical domain channel client (**vldcc**), virtual data plane channel client (**vdpc**), virtual data plane channel service (**vdpcs**)
- **stats** – Output contains statistics that are related to resource management policies
- **status** – Output contains status about a domain migration in progress

The following examples show various subsets of output that you can specify:

- List CPU information for the control domain
 

```
ldm list -o cpu primary
```
- List domain information for a guest domain
 

```
ldm list -o domain ldm2
```
- List memory and network information for a guest domain
 

```
ldm list -o network,memory ldm1
```
- List DRM policy information for a guest domain
 

```
ldm list -o resmgmt,stats ldm1
```

## ▼ List a Variable

- **Show a variable and its value for a domain.**

```
primary# ldm list-variable variable-name ldom
```

For example, the following command shows the value for the **boot-device** variable on the **ldg1** domain:

```
primary# ldm list-variable boot-device ldg1
boot-device=/virtual-devices@100/channel-devices@200/disk@0:a
```

## ▼ List Bindings

- List the resources that are bound to a domain.

```
primary# ldm list-bindings ldom
```

## ▼ List Configurations

- List logical domain configurations that have been stored on the SP.

### Example 10–5 Configurations List

The `ldm list-config` command lists the logical domain configurations that are stored on the service processor. When used with the `-r` option, this command lists those configurations for which autosave files exist on the control domain.

For more information about configurations, see [“Managing Logical Domains Configurations” on page 184](#). For more examples, see the `ldm(1M)` man page.

```
primary# ldm list-config
factory-default
3guests
foo [next poweron]
primary
reconfig-primary
```

### More Information Meaning of Labels

The labels to the right of the configuration name mean the following:

- [current] – Last booted configuration, only as long as it matches the currently running configuration; that is, until you initiate a reconfiguration. After the reconfiguration, the annotation changes to [next poweron].
- [next poweron] – Configuration to be used at the next powercycle.

## ▼ List Devices

- List all server resources, bound and unbound.

```
primary# ldm list-devices -a
```

## ▼ List Available Memory

- List the amount of memory available to be allocated.

```
primary# ldm list-devices mem
MEMORY
 PA SIZE
 0x14e000000 2848M
```

## ▼ List Services

- List the services that are available.

```
primary# ldm list-services
```

## Listing Constraints

To the Logical Domains Manager, constraints are one or more resources you want to have assigned to a particular domain. You either receive all the resources you ask to be added to a domain or you get none of them, depending upon the available resources. The `list-constraints` subcommand lists those resources you requested assigned to the domain.

## ▼ List Constraints for One Domain

- List constraints for one domain.

```
primary# ldm list-constraints ldom
```

## ▼ List Constraints in XML Format

- List constraints in XML format for a particular domain.

```
primary# ldm list-constraints -x ldom
```

## ▼ List Constraints in a Machine-Readable Format

- List constraints for all domains in a parseable format.

```
primary# ldm list-constraints -p
```



# Managing Configurations

---

This chapter contains information about managing domain configurations.

This chapter covers the following topics:

- [“Saving Domain Configurations for Future Rebuilding” on page 181](#)
- [“Managing Logical Domains Configurations” on page 184](#)

## Saving Domain Configurations for Future Rebuilding

The basic process is to save the resource constraints information for each domain into an XML file, which can then be re-issued to the Logical Domains Manager, for example, after a hardware failure to rebuild a desired configuration.

[“Restore a Domain Configuration From an XML File \(`ldm add-domain`\)” on page 182](#) works for guest domains, but not for the control (primary) domain. You can save the primary domain's constraints to an XML file, but you cannot feed the file back into the `ldm add-domain -i` command. However, you can use the `ldm init-system` command and the resource constraints from the XML file to reconfigure your primary domain. You can also use the `ldm init-system` command to reconfigure other domains that are described in the XML file, but those domains are left inactive when the configuration is complete.

The method that follows does not preserve actual bindings, only the constraints used to create those bindings. This means that, after this procedure, the domains will have the same virtual resources, but will not necessarily be bound to the same physical resources.

### ▼ Save Domain Configurations

This procedure shows how to save a domain configuration for a single domain or for all the domains on a system.

- **Save a domain configuration for one or more domains.**

- **To save the configuration for a single domain, create an XML file containing the domain's constraints.**

```
ldm list-constraints -x ldom >ldom.xml
```

The following example shows how to create an XML file, `ldg1.xml`, which contains the `ldg1` domain's constraints:

```
ldm list-constraints -x ldg1 >ldg1.xml
```

- **To save the configurations for all the domains on a system, create an XML file containing the constraints for all domains.**

```
ldm list-constraints -x >file.xml
```

The following example shows how to create an XML file, `config.xml`, which contains the constraints for all the domains on a system:

```
ldm list-constraints -x >config.xml
```

## ▼ Restore a Domain Configuration From an XML File (`ldm add-domain`)

Instead of this procedure, you can use the `ldm init-system` command to restore domain configurations from an XML file. See [“Restore a Domain Configuration From an XML File \(`ldm init-system`\)” on page 183](#).

- 1 **Create the domain by using the XML file that you created as input.**

```
ldm add-domain -i ldom.xml
```

- 2 **Bind the domain.**

```
ldm bind-domain [-fq] ldom
```

The `-f` option forces the binding of the domain even if invalid back-end devices are detected. The `-q` option disables the validation of back-end devices so that the command runs more quickly.

- 3 **Start the domain.**

```
ldm start-domain ldom
```

### Example 11–1 Restoring a Single Domain From an XML File

The following example shows how to restore a single domain. First, you restore the `ldg1` domain from the XML file. Then, you bind and restart the `ldg1` domain that you restored.

```
ldm add-domain -i ldg1.xml
ldm bind ldg1
ldm start ldg1
```

## ▼ Restore a Domain Configuration From an XML File (ldm init-system)

This procedure explains how to use the `ldm init-system` command with an XML file to re-create a previously saved configuration. The XML file describes one or more domain configurations. The XML file can be created by running the `ldm ls-constraints -x` command. The `ldm init-system` command is expected to be run in the `factory-default` configuration, but it can restore any configuration from an XML file. The primary domain is reconfigured as specified in the file, and any non-primary domains that have configurations in the XML file are reconfigured but left inactive.

Instead of this procedure, you can use the `ldm add-domain` command to restore a single domain configuration from an XML file. See [“Restore a Domain Configuration From an XML File \(ldm add-domain\)”](#) on page 182.

- 1 Log in to the primary domain.
- 2 Verify that the system is in the `factory-default` configuration.

```
primary# ldm list-config | grep "factory-default"
factory-default [current]
```

If the system is not in the `factory-default` configuration, see [“Restore the Factory Default Configuration”](#) on page 39.

- 3 Become superuser or assume an equivalent role.

Roles contain authorizations and privileged commands. For more information about roles, see [“Configuring RBAC \(Task Map\)”](#) in *System Administration Guide: Security Services*.

- 4 Restore the domain configuration or configurations from the XML file.

```
ldm init-system [-frs] -i filename.xml
```

The primary domain *must* be rebooted for the configuration to take effect. The `-r` option reboots the primary domain after the configuration. If you do not specify the `-r` option, you *must* perform the reboot manually.

The `-s` option restores only the virtual services configuration (`vds`, `vcc`, and `vsw`) and might be able to be performed without having to reboot.

The `-f` option skips the `factory-default` configuration check and continues irrespective of what was already configured on the system. Use the `-f` option with caution. The `ldm init-system` command assumes that the system is in the `factory-default` configuration and so *directly* applies the changes that are specified by the XML file. Using `-f` when the system is in a configuration

other than the factory default will likely result in a system that is not configured as specified by the XML file. One or more changes might fail to be applied to the system, depending on the combination of changes in the XML file and the initial configuration.

### Example 11–2 Restoring Domains From XML Configuration Files

The following examples show how to use the `ldm init-system` command to restore the primary domain and all the domains on a system from the factory-default configuration.

- **Restore the primary domain.** The `-r` option is used to reboot the primary domain after the configuration completes. The `primary.xml` file contains the XML domain configuration that you saved at an earlier time.

```
primary# ldm init-system -r -i primary.xml
```

- **Restore all the domains on a system.** Restore the domains on the system to the configurations in the `config.xml` XML file. The `config.xml` file contains the XML domain configurations that you saved at an earlier time. The primary domain is restarted automatically by the `ldm init-system` command. Any other domains are restored, but not bound and restarted.

```
ldm init-system -r -i config.xml
```

After the system reboots, the following commands bind and restart the `ldg1` and `ldg2` domains:

```
ldm bind ldg1
ldm start ldg1
ldm bind ldg2
ldm start ldg2
```

## Managing Logical Domains Configurations

A Logical Domains *configuration* is a complete description of all the domains and their resource allocations within a single system. You can save and store configurations on the service processor (SP) for later use.

When you power up a system, the SP boots the selected configuration. By booting a configuration, the system runs the same set of domains, and uses the same virtualization and partitioning resource allocations that are specified in the configuration. The default configuration is the one that is most recently saved.

A copy of the current configuration is automatically saved on the control domain whenever the Logical Domains configuration is changed.

The autosave operation occurs immediately, even in the following situations:

- When the new configuration is not explicitly saved on the SP
- When the actual configuration change is not made until after the affected domain reboots

This autosave operation enables you to recover a configuration when the configurations that are saved on the SP are lost. This operation also enables you to recover a configuration when the current configuration was not explicitly saved to the SP when the system powercycled. In these circumstances, the Logical Domains Manager can restore that configuration on restart if it is newer than the configuration marked for the next boot.

---

**Note** – Power management, FMA, ASR, and PRI update events do not cause an update to the autosave files.

---

You can automatically or manually restore autosave files to new or existing configurations. By default, when an autosave configuration is newer than the corresponding running configuration, a message is written to the Logical Domains log. Thus, you must use the `ldm add-spconfig -r` command to manually update an existing configuration or create a new one based on the autosave data.

---

**Note** – When a delayed reconfiguration is pending, the configuration changes are immediately autosaved. As a result, if you run the `ldm list-config -r` command, the autosave configuration is shown as being newer than the current configuration.

---

For information about how to use the `ldm *-spconfig` commands to manage configurations and to manually recover autosave files, see the [ldm\(1M\)](#) man page.

For information about how to select a configuration to boot, see [“Using Logical Domains With the Service Processor”](#) on page 194.

## ▼ Modify the Autorecovery Policy

The autorecovery policy specifies how to handle the recovery of a configuration when one configuration that is automatically saved on the control domain is newer than the corresponding running configuration. The autorecovery policy is specified by setting the `autorecovery_policy` property of the `ldmd` SMF service. The `autorecovery_policy` property can have the following values:

- `autorecovery_policy=1` – Logs warning messages when an autosave configuration is newer than the corresponding running configuration. These messages are logged in the `ldmd` SMF log file. The user must manually perform any configuration recovery. This is the default policy.
- `autorecovery_policy=2` – Displays a notification message if an autosave configuration is newer than the corresponding running configuration. This notification message is printed in the output of any `ldm` command the first time an `ldm` command is issued after each restart of the Logical Domains Manager. The user must manually perform any configuration recovery.
- `autorecovery_policy=3` – Automatically updates the configuration if an autosave configuration is newer than the corresponding running configuration. This action overwrites the SP configuration that will be used during the next powercycle. This configuration is updated with the newer configuration that is saved on the control domain. This action does not impact the currently running configuration. It only impacts the configuration that will be used during the next powercycle. A message is also logged, which states that a newer configuration has been saved on the SP and that it will be booted the next time the system is powercycled. These messages are logged in the `ldmd` SMF log file.

### 1 Log in to the control domain.

### 2 Become superuser or assume an equivalent role.

Roles contain authorizations and privileged commands. For more information about roles, see [“Configuring RBAC \(Task Map\)” in \*System Administration Guide: Security Services\*](#).

### 3 View the `autorecovery_policy` property value.

```
svccfg -s ldmd listprop ldmd/autorecovery_policy
```

### 4 Stop the `ldmd` service.

```
svcadm disable ldmd
```

### 5 Change the `autorecovery_policy` property value.

```
svccfg -s ldmd setprop ldmd/autorecovery_policy=value
```

For example, to set the policy to perform autorecovery, set the property value to 3:

```
svccfg -s ldmd setprop ldmd/autorecovery_policy=3
```

**6 Refresh and restart the ldmd service.**

```
svcadm refresh ldmd
svcadm enable ldmd
```

**Example 11–3 Modifying the Autorecovery Policy From Log to Autorecovery**

The following example shows how to view the current value of the `autorecovery_policy` property and change it to a new value. The original value of this property is 1, which means that autosave changes are logged. The `svcadm` command is used to stop and restart the `ldmd` service, and the `svccfg` command is used to view and set the property value.

```
svccfg -s ldmd listprop ldmd/autorecovery_policy
ldmd/autorecovery_policy integer 1
svcadm disable ldmd
svccfg -s ldmd setprop ldmd/autorecovery_policy=3
svcadm refresh ldmd
svcadm enable ldmd
```



## Performing Other Administration Tasks

---

This chapter contains information and tasks about using the Oracle VM Server for SPARC software that are not described in the preceding chapters.

This chapter covers the following topics:

- “Entering Names in the CLI” on page 189
- “Connecting to a Guest Console Over a Network” on page 190
- “Using Console Groups” on page 191
- “Stopping a Heavily Loaded Domain Can Time Out” on page 192
- “Operating the Oracle Solaris OS With Oracle VM Server for SPARC” on page 192
- “Using Logical Domains With the Service Processor” on page 194
- “Configuring Domain Dependencies” on page 195
- “Determining Where Errors Occur by Mapping CPU and Memory Addresses” on page 198
- “Using Universally Unique Identifiers” on page 201
- “Virtual Domain Information Command and API” on page 201

### Entering Names in the CLI

The following sections describe the restrictions on entering names in the Logical Domains Manager CLI.

#### File Names (file) and Variable Names (var-name)

- First character must be a letter, a number, or a forward slash (/).
- Subsequent letters must be letters, numbers, or punctuation.

## Virtual Disk Server backend and Virtual Switch Device Names

The names must contain letters, numbers, or punctuation.

## Configuration Name (config-name)

The logical domain configuration name (*config-name*) that you assign to a configuration stored on the service processor (SP) must have no more than 64 characters.

## All Other Names

The remainder of the names, such as the logical domain name (*ldom*), service names (*vswitch-name*, *service-name*, *vdpcs-service-name*, and *vcc-name*), virtual network name (*if-name*), and virtual disk name (*disk-name*), must be in the following format:

- First character must be a letter or number.
- Subsequent characters must be letters, numbers, or any of the following characters  
- \_ + # . : ; ~ ( ) .

## Connecting to a Guest Console Over a Network

You can connect to a guest console over a network if the `listen_addr` property is set to the IP address of the control domain in the `vntsd(1M)` SMF manifest. For example:

```
$ telnet host-name 5001
```

---

**Note** – Enabling network access to a console has security implications. Any user can connect to a console and for this reason it is disabled by default.

---

A Service Management Facility manifest is an XML file that describes a service. For more information about creating an SMF manifest, refer to the [Oracle Solaris 10 System Administrator Documentation \(http://download.oracle.com/docs/cd/E18752\\_01/index.html\)](http://download.oracle.com/docs/cd/E18752_01/index.html).

---

**Note** – To access a non-English OS in a guest domain through the console, the terminal for the console must be in the locale required by the OS.

---

## Using Console Groups

The virtual network terminal server daemon, [vntsd\(1M\)](#), enables you to provide access for multiple domain consoles using a single TCP port. At the time of domain creation, the Logical Domains Manager assigns a unique TCP port to each console by creating a new default group for that domain's console. The TCP port is then assigned to the console group as opposed to the console itself. The console can be bound to an existing group using the `set - vcons` subcommand.

### ▼ Combine Multiple Consoles Into One Group

#### 1 Bind the consoles for the domains into one group.

The following example shows binding the console for three different domains (`ldg1`, `ldg2`, and `ldg3`) to the same console group (`group1`).

```
primary# ldm set-vcons group=group1 service=primary-vcc0 ldg1
primary# ldm set-vcons group=group1 service=primary-vcc0 ldg2
primary# ldm set-vcons group=group1 service=primary-vcc0 ldg3
```

#### 2 Connect to the associated TCP port (localhost at port 5000 in this example).

```
telnet localhost 5000
primary-vnts-group1: h, l, c{id}, n{name}, q:
```

You are prompted to select one of the domain consoles.

#### 3 List the domains within the group by selecting `l` (list).

```
primary-vnts-group1: h, l, c{id}, n{name}, q: l
DOMAIN ID DOMAIN NAME DOMAIN STATE
0 ldg1 online
1 ldg2 online
2 ldg3 online
```

---

**Note** – To re-assign the console to a different group or vcc instance, the domain must be unbound; that is, it has to be in the inactive state. Refer to the Oracle Solaris 10 OS [vntsd\(1M\)](#) man page for more information about configuring and using SMF to manage `vntsd` and using console groups.

---

# Stopping a Heavily Loaded Domain Can Time Out

An `ldm stop-domain` command can time out before the domain completes shutting down. When this happens, an error similar to the following is returned by the Logical Domains Manager.

```
LDom ldg8 stop notification failed
```

However, the domain could still be processing the shutdown request. Use the `ldm list-domain` command to verify the status of the domain. For example:

```
ldm list-domain ldg8
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
ldg8 active s---- 5000 22 3328M 0.3% 1d 14h 31m
```

The preceding list shows the domain as active, but the `s` flag indicates that the domain is in the process of stopping. This should be a transitory state.

The following example shows the domain has now stopped.

```
ldm list-domain ldg8
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
ldg8 bound ----- 5000 22 3328M
```

# Operating the Oracle Solaris OS With Oracle VM Server for SPARC

This section describes the changes in behavior in using the Oracle Solaris OS that occur once a configuration created by the Logical Domains Manager is instantiated.

## OpenBoot Firmware Not Available After Oracle Solaris OS Has Started

The OpenBoot firmware is not available after the Oracle Solaris OS has started because it is removed from memory.

To reach the `ok` prompt from the Oracle Solaris OS, you must halt the domain. You can use the Oracle Solaris OS `halt` command to halt the domain.

## Powercycling a Server

Whenever performing any maintenance on a system running Oracle VM Server for SPARC software that requires powercycling the server, you must save your current logical domain configurations to the SP first.

▼ **Save Your Current Domain Configurations to the SP**

- Use the following command.

```
ldm add-config config-name
```

**Do Not Use the `psradm(1M)` Command on Active CPUs in a Power-Managed Domain**

Do not attempt to change an active CPU's operational status in a power-managed domain by using the `psradm(1M)` command.

**Result of Oracle Solaris OS Breaks**

The behavior described in this section is seen when you do the following:

1. Press the L1-A key sequence when the input device is set to keyboard.
2. Enter the send break command when the virtual console is at the `telnet` prompt.

After these types of breaks, you receive the following prompt:

```
c)ontinue, s)ync, r)eset, h)alt?
```

Type the letter that represents what you want the system to do after these types of breaks.

**Results From Halting or Rebooting the Control Domain**

The following table shows the expected behavior of halting or rebooting the control (primary) domain.

TABLE 12-1 Expected Behavior of Halting or Rebooting the Control (primary) Domain

| Command | Other Domain Configured? | Behavior                                                                                                                                                   |
|---------|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| halt    | Not Configured           | Host powered off and stays off until powered on at the SP.                                                                                                 |
|         | Configured               | Soft resets and boots up if the variable <code>auto-boot?=true</code> . Soft resets and halts at ok prompt if the variable <code>auto-boot?=false</code> . |
| reboot  | Not Configured           | Reboots the host, no power off.                                                                                                                            |
|         | Configured               | Reboots the host, no power off.                                                                                                                            |

TABLE 12-1 Expected Behavior of Halting or Rebooting the Control (primary) Domain (Continued)

| Command       | Other Domain Configured? | Behavior                                                |
|---------------|--------------------------|---------------------------------------------------------|
| shutdown -i 5 | Not Configured           | Host powered off, stays off until powered on at the SP. |
|               | Configured               | Soft resets and reboots.                                |

For information about the consequences of rebooting a control domain that has the root domain role, see “Rebooting the primary Domain” on page 74.

## Using Logical Domains With the Service Processor

The section describes information to be aware of in using the Integrated Lights Out Manager (ILOM) service processor (SP) with the Logical Domains Manager. For more information about using the ILOM software, see the documents for your specific platform, such as [Sun SPARC Enterprise T5120 and T5220 Servers Topic Set](#) for the Sun SPARC Enterprise T5120 and T5220 servers.

An additional option is available to the existing ILOM command.

```
-> set /HOST/bootmode config=config-name
```

The `config=config-name` option enables you to set the configuration on the next power on to another configuration, including the `factory-default` shipping configuration.

You can invoke the command whether the host is powered on or off. It takes effect on the next host reset or power on.

### ▼ Reset the Domain Configuration to the Default or Another Configuration

- Reset the logical domain configuration on the next power on to the default shipping configuration by executing this command:

```
-> set /HOST/bootmode config=factory-default
```

You also can select other configurations that have been created with the Logical Domains Manager using the `ldm add-config` command and stored on the service processor (SP). The name you specify in the Logical Domains Manager `ldm add-config` command can be used to select that configuration with the ILOM `bootmode` command. For example, assume you stored the configuration with the name `ldm-config1`.

```
-> set /HOST/bootmode config=ldm-config1
```

Now, you must powercycle the system to load the new configuration.

See the [ldm\(1M\)](#) man page for more information about the `ldm add-config` command.

## Configuring Domain Dependencies

You can use the Logical Domains Manager to establish dependency relationships between domains. A domain that has one or more domains that depend on it is called a *master domain*. A domain that depends on another domain is called a *slave domain*.

Each slave domain can specify up to four master domains by setting the `master` property. For example, the `pine` slave domain specifies its four master domains in the following comma-separated list:

```
ldm add-domain master=apple,lemon,orange,peach pine
```

Each master domain can specify what happens to its slave domains in the event that the master domain fails. For instance, if a master domain fails, it might require its slave domains to panic. If a slave domain has more than one master domain, the first master domain to fail triggers its defined failure policy on all of its slave domains.

---

**Note** – If more than one master domain fails simultaneously, only one of the specified failure policies will be enforced on all the affected slave domains. For example, if the failed master domains have failure policies of `stop` and `panic`, all slave domains will be either stopped or panicked.

---

The master domain's failure policy is controlled by setting one of the following values to the `failure-policy` property:

- `ignore` ignores any slave domains when the master domain fails.
- `panic` panics any slave domains when the master domain fails.
- `reset` resets any slave domains when the master domain fails.
- `stop` stops any slave domains when the master domain fails.

In this example, the master domains specify their failure policy as follows:

```
ldm set-domain failure-policy=ignore apple
ldm set-domain failure-policy=panic lemon
ldm set-domain failure-policy=reset orange
ldm set-domain failure-policy=stop peach
```

You can use this mechanism to create explicit dependencies between domains. For example, a guest domain implicitly depends on the service domain to provide its virtual devices. A guest domain's I/O is blocked when the service domain on which it depends is not up and running. By defining a guest domain as a slave of its service domain, you can specify the behavior of the

guest domain when its service domain goes down. When no such dependency is established, a guest domain just waits for its service domain to return to service.

---

**Note** – The Logical Domains Manager does not permit you to create domain relationships that create a dependency cycle. For more information, see [“Dependency Cycles” on page 197](#).

---

For domain dependency XML examples, see [Example 17–6](#).

## Domain Dependency Examples

The following examples show how to configure domain dependencies.

- The first command creates a master domain called `twizzle`. This command uses `failure-policy=reset` to specify that slave domains reset if the `twizzle` domain fails. The second command modifies a master domain called `primary`. This command uses `failure-policy=panic` to specify that slave domains panic if the `primary` domain fails. The third command creates a slave domain called `chocktaw` that depends on two master domains, `twizzle` and `primary`. The slave domain uses `master=twizzle,primary` to specify its master domains. In the event either the `twizzle` or `primary` domain fails, the `chocktaw` domain will reset or panic. The first master domain to fail is the one responsible for determining the behavior of the slave domains.

```
ldm add-domain failure-policy=reset twizzle
ldm set-domain failure-policy=panic primary
ldm add-domain master=twizzle,primary chocktaw
```

- This example shows how to use the `ldm set-domain` command to modify the `orange` domain to assign `primary` as the master domain. The second command uses the `ldm set-domain` command to assign `orange` and `primary` as master domains for the `tangerine` domain. The third command lists information about all of these domains.

```
ldm set-domain master=primary orange
ldm set-domain master=orange,primary tangerine
ldm list -o domain
```

| NAME    | STATE  | FLAGS  | UTIL |
|---------|--------|--------|------|
| primary | active | -n-cv- | 0.2% |

```
SOFTSTATE
Solaris running
```

```
HOSTID
0x83d8b31c
```

```
CONTROL
failure-policy=ignore
```

```
DEPENDENCY
master=
```

```

NAME STATE FLAGS UTIL
orange bound -----
HOSTID
 0x84fb28ef
CONTROL
 failure-policy=stop
DEPENDENCY
 master=primary

NAME STATE FLAGS UTIL
tangerine bound -----
HOSTID
 0x84f948e9
CONTROL
 failure-policy=ignore
DEPENDENCY
 master=orange,primary
```

- The following shows an example listing with parseable output:

```
ldm list -o domain -p
```

## Dependency Cycles

The Logical Domains Manager does not permit you to create domain relationships that create a dependency cycle. A *dependency cycle* is a relationship between two or more domains that lead to a situation where a slave domain depends on itself, or a master domain depends on one of its slave domains.

The Logical Domains Manager determines whether a dependency cycle exists before adding a dependency. The Logical Domains Manager starts at the slave domain and searches along all paths that are specified by the master array until the end of the path is reached. Any dependency cycles found along the way are reported as errors.

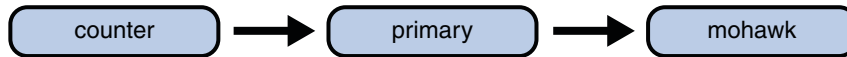
The following example shows how a dependency cycle might be created. The first command creates a slave domain called *mohawk* that specifies its master domain as *primary*. So, *mohawk* depends on *primary* in the following dependency chain:

FIGURE 12-1 Single Domain Dependency



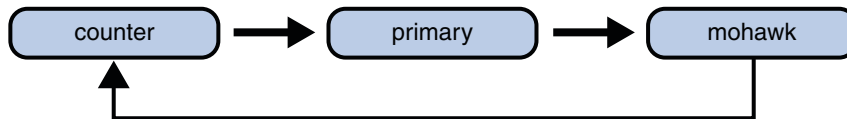
The second command creates a slave domain called `primary` that specifies its master domain as `counter`. So, `mohawk` depends on `primary`, which depends on `counter` in the following dependency chain:

FIGURE 12-2 Multiple Domain Dependency



The third command attempts to create a dependency between the `counter` and `mohawk` domains, which would produce the following dependency cycle:

FIGURE 12-3 Domain Dependency Cycle



The `ldm set-domain` command will fail with the following error message:

```
ldm add-domain master=primary mohawk
ldm set-domain master=counter primary
ldm set-domain master=mohawk counter
Dependency cycle detected: LDom "counter" indicates "primary" as its master
```

## Determining Where Errors Occur by Mapping CPU and Memory Addresses

This section describes how you can correlate the information that is reported by the Oracle Solaris Fault Management Architecture (FMA) with the logical domain resources that are marked as being faulty.

FMA reports CPU errors in terms of physical CPU numbers and memory errors in terms of physical memory addresses.

If you want to determine within which logical domain an error occurred and the corresponding virtual CPU number or real memory address within the domain, then you must perform a mapping.

## CPU Mapping

The domain and the virtual CPU number within the domain, which correspond to a given physical CPU number, can be determined with the following procedures.

### ▼ Determine the CPU Number

- 1 Generate a long parseable list for all domains.

```
primary# ldm list -l -p
```

- 2 Look for the entry in the list's VCPU sections that has a `pid` field equal to the physical CPU number.

- If you find such an entry, the CPU is in the domain the entry is listed under, and the virtual CPU number within the domain is given by the entry's `vid` field.
- If you do not find such an entry, the CPU is not in any domain.

## Memory Mapping

The domain and the real memory address within the domain, which correspond to a given physical memory address (PA), can be determined as follows.

### ▼ Determine the Real Memory Address

- 1 Generate a long parseable list for all domains.

```
primary# ldm list -l -p
```

- 2 Look for the line in the list's MEMORY sections where the PA falls within the inclusive range  $pa$  to  $(pa + size - 1)$ ; that is,  $pa \leq PA \leq (pa + size - 1)$ .

Here  $pa$  and  $size$  refer to the values in the corresponding fields of the line.

- If you find such an entry, the PA is in the domain the entry is listed under and the corresponding real address within the domain is given by  $ra + (PA - pa)$ .
- If you do not find such an entry, the PA is not in any domain.

## Examples of CPU and Memory Mapping

Suppose you have a logical domain configuration as shown in [Example 12–1](#), and you want to determine the domain and the virtual CPU corresponding to physical CPU number 5, and the domain and the real address corresponding to physical address `0x7e816000`.

Looking through the VCPU entries in the list for the one with the `pid` field equal to 5, you can find the following entry under logical domain `ldg1`.

```
|vid=1|pid=5|util=29|strand=100
```

Hence, the physical CPU number 5 is in domain `ldg1` and within the domain it has virtual CPU number 1.

Looking through the MEMORY entries in the list, you can find the following entry under domain `ldg2`.

```
ra=0x80000000|pa=0x78000000|size=1073741824
```

Where  $0x78000000 \leq 0x7e816000 \leq (0x78000000 + 1073741824 - 1)$ ; that is,  $pa \leq PA \leq (pa + size - 1)$ . Hence, the PA is in domain `ldg2` and the corresponding real address is  $0x80000000 + (0x7e816000 - 0x78000000) = 0xe816000$ .

#### EXAMPLE 12-1 Long Parseable List of Logical Domains Configurations

```
primary# ldm list -l -p
VERSION 1.0
DOMAIN|name=primary|state=active|flags=normal,control,vio-service|cons=SP|ncpu=4|mem=1073741824|util=0.6|
uptime=64801|softstate=Solaris running
VCPU
|vid=0|pid=0|util=0.9|strand=100
|vid=1|pid=1|util=0.5|strand=100
|vid=2|pid=2|util=0.6|strand=100
|vid=3|pid=3|util=0.6|strand=100
MEMORY
|ra=0x80000000|pa=0x80000000|size=1073741824
IO
|dev=pci@780|alias=bus_a
|dev=pci@7c0|alias=bus_b
...
DOMAIN|name=ldg1|state=active|flags=normal|cons=5000|ncpu=2|mem=805306368|util=29|uptime=903|
softstate=Solaris running
VCPU
|vid=0|pid=4|util=29|strand=100
|vid=1|pid=5|util=29|strand=100
MEMORY
|ra=0x80000000|pa=0x48000000|size=805306368
...
DOMAIN|name=ldg2|state=active|flags=normal|cons=5001|ncpu=3|mem=1073741824|util=35|uptime=775|
softstate=Solaris running
VCPU
|vid=0|pid=6|util=35|strand=100
|vid=1|pid=7|util=34|strand=100
|vid=2|pid=8|util=35|strand=100
MEMORY
|ra=0x80000000|pa=0x78000000|size=1073741824
...
```

## Using Universally Unique Identifiers

Starting with the Oracle VM Server for SPARC 2.0 release, each domain is assigned a universally unique identifier (UUID). The UUID is assigned when a domain is created. For legacy domains, the UUID is assigned when the `ldmd` daemon initializes.

---

**Note** – The UUID is lost if you use the `ldm migrate-domain -f` command to migrate a domain to a target machine that runs an older version of the Logical Domains Manager. When you migrate a domain from a source machine that runs an older version of the Logical Domains Manager, the domain is assigned a new UUID as part of the migration. Otherwise, the UUID is migrated.

---

You can obtain the UUID for a domain by running the `ldm list -l`, `ldm list-bindings`, or `ldm list -o domain` command. The following examples show the UUID for the `ldg1` domain:

```
primary# ldm create ldg1
primary# ldm ls -l ldg1
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
ldg1 inactive -----
UUID
6c908858-12ef-e520-9eb3-f1cd3dbc3a59

primary# ldm ls -l -p ldg1
VERSION 1.4
DOMAIN|name=ldg1|state=inactive|flags=|cons=|ncpu=|mem=|util=|uptime=
UUID|uuid=6c908858-12ef-e520-9eb3-f1cd3dbc3a59
```

## Virtual Domain Information Command and API

The `virtinfo` command enables you to gather information about a running virtual domain. You can also use the Virtual Domain Information API to create programs to gather information related to virtual domains.

The following list shows some of the information that you can gather about a virtual domain by using the command or API:

- Domain type (implementation, control, guest, I/O, service, root)
- Domain name determined by the Virtual Domain Manager
- Universally unique identifier (UUID) of the domain
- Network node name of the domain's control domain
- Chassis serial number on which the domain is running

For information about the `virtinfo` command, see the [virtinfo\(1M\)](#) man page. For information about the API, see the [libv12n\(3LIB\)](#) and [v12n\(3EXT\)](#) man pages.



## PART II

# Optional Oracle VM Server for SPARC Software

This part introduces optional software and features that you can use with the Oracle VM Server for SPARC 2.1 software.



# Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool

---

This chapter covers the following topics:

- “Oracle VM Server for SPARC P2V Tool Overview” on page 205
- “Back-End Devices” on page 207
- “Installing the Oracle VM Server for SPARC P2V Tool” on page 208
- “Using the `ldmp2v` Command” on page 210

## Oracle VM Server for SPARC P2V Tool Overview

The Oracle VM Server for SPARC P2V Tool automatically converts an existing physical system to a virtual system that runs in a logical domain on a chip multithreading (CMT) system. The source system can be any of the following:

- Any sun4u SPARC based system that runs at least the Solaris 8 OS
- Any sun4v system that runs the Oracle Solaris 10 OS, but does not run in a logical domain

The conversion from a physical system to a virtual system is performed in the following phases:

- **Collection phase.** Runs on the physical source system. In the `collect` phase, a file system image of the source system is created based on the configuration information that it collects about the source system.
- **Preparation phase.** Runs on the control domain of the target system. In the `prepare` phase, a logical domain is created on the target system based on the configuration information collected in the `collect` phase. The file system image is restored to one or more virtual disks. You can use the P2V tool to create virtual disks on plain files or ZFS volumes. You can also create virtual disks on physical disks or LUNs, or on volume manager volumes that you created. The image is modified to enable it to run as a logical domain.
- **Conversion phase.** Runs on the control domain of the target system. In the `convert` phase, the created logical domain is converted into a logical domain that runs the Oracle Solaris 10 OS by using the standard Solaris upgrade process.

For information about the P2V tool, see the [ldmp2v\(1M\)](#) man page.

The following sections describe how the conversion from a physical system to a virtual system is performed in phases.

## Collection Phase

The Collection phase runs on the system to be converted. To create a consistent file system image, ensure that the system is as quiet as possible and that all applications are stopped. The `ldmp2v` command creates a backup of all mounted UFS file systems, so ensure that any file systems to be moved to a logical domain are mounted. You can exclude mounted file systems that you do not want to move, such as file systems on SAN storage or file systems that will be moved by other means. Use the `-x` option to exclude such file systems. File systems that are excluded by the `-x` option are not re-created on the guest domain. You can use the `-O` option to exclude files and directories.

No changes are required on the source system. The only requirement is the `ldmp2v` script that was installed on the control domain. Ensure that the `flarc` utility is present on the source system.

## Preparation Phase

The preparation phase uses the data collected during the collection phase to create a logical domain that is comparable to the source system.

You can use the `ldmp2v prepare` command in one of the following ways:

- **Automatic mode.** This mode automatically creates virtual disks and restores file system data.
  - Creates the logical domain and the required virtual disks of the same size as on the source system.
  - Partitions the disks and restores the file systems.

If the combined size of the `/`, `/usr`, and `/var` file systems is less than 10 Gbytes, the sizes of these file systems are automatically adjusted to allow for the larger disk space requirements of the Oracle Solaris 10 OS. Automatic resize can be disabled by using the `-x no-auto-adjust-fs` option or by using the `-m` option to manually resize a file system.
  - Modifies the OS image of the logical domain to replace all references to physical hardware with versions that are appropriate for a logical domain. This enables you to upgrade the system to the Oracle Solaris 10 OS by using the normal Solaris upgrade process. Modifications include updating the `/etc/vfstab` file to account for new disk names. Any Solaris Volume Manager or Veritas Volume Manager (VxVM) encapsulated boot disks are automatically unencapsulated during this process. When a

disk is unencapsulated, it is converted into plain disk slices. If VxVM is installed on the source system, the P2V process disables VxVM on the created guest domain.

- **Non-automatic mode.** You must create the virtual disks and restore the file system data manually. This mode enables you to change the size and number of disks, the partitioning, and the file system layout. The preparation phase in this mode only runs the logical domain creation and the OS image modification steps on the file system.
- **Cleanup mode.** Removes a logical domain and all of the underlying back-end devices that are created by `ldmp2v`.

## Conversion Phase

In the conversion phase, the logical domain uses the Solaris upgrade process to upgrade to the Oracle Solaris 10 OS. The upgrade operation removes all existing packages and installs the Oracle Solaris 10 sun4v packages, which automatically performs a sun4u-to-sun4v conversion. The convert phase can use an Oracle Solaris DVD ISO image or a network installation image. You can also use Custom JumpStart to perform a fully automated hands-off upgrade operation.

## Back-End Devices

You can create virtual disks for a guest domain on a number of back-end types: files (`file`), ZFS volumes (`zvol`), physical disks or LUNs (`disk`), or volume manager volumes (`disk`). The `ldmp2v` command automatically creates files or ZFS volumes of the appropriate size if you specify `file` or `zvol` as the back-end type in one of the following ways:

- By using the `-b` option
- By specifying the value of the `BACKEND_TYPE` parameter in the `/etc/ldmp2v.conf` file

The `disk` back-end type enables you to use a physical disk, LUN, or volume manager volume (Solaris Volume Manager and Veritas Volume Manager (VxVM)) as a back-end device for virtual disks. You must create the disk or volume with an appropriate size prior to beginning the prepare phase. For a physical disk or LUN, specify the back-end device as slice 2 of the block or character device of the disk, such as `/dev/dsk/c0t3d0s2`. For a volume manager volume, specify the block or character device for the volume, such as `/dev/md/dsk/d100` for Solaris Volume Manager or `/dev/vx/dsk/ldomdg/vol1` for VxVM.

Unless you specify the volume and virtual disk names with the `-B backend:volume:vdisk` option, the volumes and virtual disks that you create for the guest are given default names.

- *backend* specifies the name of the back end to use. You must specify *backend* for the disk back-end type. *backend* is optional for the `file` and `zvol` back-end types, and can be used to set a non-default name for the file or ZFS volume that `ldmp2v` creates. The default name is `$BACKEND_PREFIX/guest-name/diskN`.
- *volume* is optional for all back-end types and specifies the name of the virtual disk server volume to create for the guest domain. If not specified, *volume* is `guest-name-volN`.
- *vdisk* is optional for all back-end types and specifies the name of the volume in the guest domain. If not specified, *vdisk* is `diskN`.

---

**Note** – During the conversion process, the virtual disk is temporarily named `guest-name-diskN` to ensure that the name in the control domain is unique.

---

To specify a blank value for *backend*, *volume*, or *vdisk*, include only the colon separator. For example, specifying `-B : :vdisk001` sets the name of the virtual disk to `vdisk001` and uses the default names for the back end and volume. If you do not specify *vdisk*, you can omit the trailing colon separator. For example, `-B /ldoms/ldom1/vol001:vol001` specifies the name of the back-end file as `/ldoms/ldom1/vol001` and the volume name as `vol001`. The default virtual disk name is `disk0`.

## Installing the Oracle VM Server for SPARC P2V Tool

The Oracle VM Server for SPARC P2V Tool package must be installed and configured *only* on the control domain of the target system. You do not need to install the package on the source system. Instead, you can simply copy the `/usr/sbin/ldmp2v` script from the target system to the source system.

### Prerequisites

Before you can run the Oracle VM Server for SPARC P2V Tool, ensure that the following conditions are met:

- The following Flash utility patches are installed on the source system:
  - **For the Solaris 8 OS:** At least patch ID 109318-34
  - **For the Solaris 9 OS:** At least patch ID 113343-06
- Target system runs at least Logical Domains 1.1 on the following:
  - Solaris 10 10/08 OS
  - Solaris 10 5/08 OS with the appropriate Logical Domains 1.1 patches

- Guest domains run at least the Solaris 10 5/08 OS
- Source system runs at least the Solaris 8 OS

In addition to these prerequisites, configure an NFS file system to be shared by both the source and target systems. This file system should be writable by root. However, if a shared file system is not available, use a local file system that is large enough to hold a file system dump of the source system on both the source and target systems.

## Limitations

The Oracle VM Server for SPARC P2V Tool has the following limitations:

- Only UFS file systems are supported.
- Only plain disks (`/dev/dsk/c0t0d0s0`), Solaris Volume Manager metadevices (`/dev/md/dsk/dNNN`), and VxVM encapsulated boot disks are supported on the source system.
- During the P2V process, each guest domain can have only a single virtual switch and virtual disk server. You can add more virtual switches and virtual disk servers to the domain after the P2V conversion.
- Support for VxVM volumes is limited to the following volumes on an encapsulated boot disk: `rootvol`, `swapvol`, `usr`, `var`, `opt`, and `home`. The original slices for these volumes must still be present on the boot disk. The P2V tool supports Veritas Volume Manager 5.x on the Solaris 10 OS. However, you can also use the P2V tool to convert Solaris 8 and Solaris 9 operating systems that use VxVM.
- You cannot convert Oracle Solaris 10 systems that are configured with zones.

## ▼ Install the Oracle VM Server for SPARC P2V Tool

- 1 Go to the Oracle VM Server for SPARC download page at <http://www.oracle.com/virtualization/index.html>.

- 2 Download the P2V software package, `SUNWldmp2v`.

The `SUNWldmp2v` package is included in the Oracle VM Server for SPARC zip file.

- 3 Become superuser or assume an equivalent role.

Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” in *System Administration Guide: Security Services*.

- 4 Use the `pkgadd` command to install the `SUNWldmp2v` package.

```
pkgadd -d . SUNWldmp2v
```

## 5 Create the `/etc/ldmp2v.conf` file and configure the following default properties:

- VDS – Name of the virtual disk service, such as `VDS="primary-vds0"`
- VSW – Name of the virtual switch, such as `VSW="primary-vsw0"`
- VCC – Name of the virtual console concentrator, such as `VCC="primary-vcc0"`
- BACKEND\_TYPE – Back-end type of zvol, file, or disk
- BACKEND\_SPARSE – Whether to create back-end devices as sparse volumes or files  
`BACKEND_SPARSE="yes"`, or non-sparse volumes or files `BACKEND_SPARSE="no"`
- BACKEND\_PREFIX – Location to create virtual disk back-end devices

When `BACKEND_TYPE="zvol"`, specify the `BACKEND_PREFIX` value as a ZFS dataset name.

When `BACKEND_TYPE="files"`, the `BACKEND_PREFIX` value is interpreted as a path name of a directory that is relative to `/`.

For example, `BACKEND_PREFIX="tank/ldoms"` would result in having ZVOLs created in the `tank/ldoms/domain-name` dataset, and files created in the `/tank/ldoms/domain-name` subdirectory.

The `BACKEND_PREFIX` property is not applicable to the disk back end.

- BOOT\_TIMEOUT – Timeout for Oracle Solaris OS boot in seconds

For more information, see the `ldmp2v.conf.sample` configuration file that is part of the downloadable bundle.

## Using the ldmp2v Command

This section includes examples for the three phases.

### EXAMPLE 13-1 Collection Phase Examples

The following examples show how you might use the `ldmp2v collect` command.

- **Sharing an NFS-mounted file system.** The following example shows the simplest way to perform the `collect` step where the source and target systems share an NFS-mounted file system.

As superuser, ensure that all required UFS file systems are mounted.

```
volumia# df -k
Filesystem kbytes used avail capacity Mounted on
/dev/dsk/clt1d0s0 16516485 463289 15888032 3% /
/proc 0 0 0 0% /proc
fd 0 0 0 0% /dev/fd
mnttab 0 0 0 0% /etc/mnttab
/dev/dsk/clt1d0s3 8258597 4304 8171708 1% /var
swap 4487448 16 4487432 1% /var/run
swap 4487448 16 4487432 1% /tmp
/dev/dsk/clt0d0s0 1016122 9 955146 1% /u01
vandikhout:/u1/home/dana
6230996752 1051158977 5179837775 17% /home/dana
```

**EXAMPLE 13-1** Collection Phase Examples (Continued)

The following shows how to run the collection tool when the source and target systems share an NFS-mounted file system:

```
volumia# ldmp2v collect -d home/dana/volumia
Collecting system configuration ...
Archiving file systems ...
Determining which filesystems will be included in the archive...
Creating the archive...
895080 blocks
Archive creation complete.
```

- **Not sharing an NFS-mounted file system.** When the source and target systems do not share an NFS-mounted file system, the file system image can be written to local storage and later copied to the control domain. The Flash utility automatically excludes the archive that it creates.

```
volumia# ldmp2v collect -d /var/tmp/volumia
Collecting system configuration ...
Archiving file systems ...
Determining which filesystems will be included in the archive...
Creating the archive...
895080 blocks
Archive creation complete.
```

Copy the Flash archive and the manifest file from the /var/tmp/volumia directory to the target system.

---

**Tip** – In some cases, ldmp2v might show cpio command errors. Most commonly, these are messages such as File size of etc/mnttab has increased by 435. You can ignore messages that pertain to log files or to files that reflect system state. Be sure to review all error messages thoroughly.

---

- **Skip file-system backup step.** If backups of the system are already available using a third-party backup tool such as NetBackup, you can skip the file system backup step by using the none archiving method. When you use this option, only the system configuration manifest is created.

```
volumia# ldmp2v collect -d /home/dana/p2v/volumia -a none
Collecting system configuration ...
The following file system(s) must be archived manually: / /u01 /var
```

Note that if the directory specified by -d is not shared by the source and target systems, you must copy the contents of that directory to the control domain. The directory contents must be copied to the control domain prior to the preparation phase.

**EXAMPLE 13-2** Preparation Phase Examples

The following examples show how you might use the `ldmp2v prepare` command.

- The following example creates a logical domain called `volumia` by using the defaults configured in `/etc/ldmp2v.conf` while keeping the MAC addresses of the physical system:

```
ldmp2v prepare -d /home/dana/p2v/volumia -o keep-mac volumia
Creating vdisks ...
Creating file systems ...
Populating file systems ...
Modifying guest domain OS image ...
Removing SVM configuration ...
Unmounting guest file systems ...
Creating domain volumia ...
Attaching vdisks to domain volumia ...
```

- The following command shows information about the `volumia` logical domain:

```
ldm list -l volumia
```

| NAME    | STATE    | FLAGS | CONS | VCPU | MEMORY | UTIL | UPTIME |
|---------|----------|-------|------|------|--------|------|--------|
| volumia | inactive | ----- |      | 2    | 4G     |      |        |

```
NETWORK
```

| NAME  | SERVICE      | DEVICE | MAC               | MODE | PVID | VID |
|-------|--------------|--------|-------------------|------|------|-----|
| vnet0 | primary-vsw0 |        | 00:03:ba:1d:7a:5a |      | 1    |     |

```
DISK
```

| NAME  | DEVICE | TOUT | MPGROUP | VOLUME | SERVER                    |
|-------|--------|------|---------|--------|---------------------------|
| disk0 |        |      |         |        | volumia-vol0@primary-vds0 |
| disk1 |        |      |         |        | volumia-vol1@primary-vds0 |

- The following shows that you can completely remove a domain and its back-end devices by using the `-C` option:

```
ldmp2v prepare -C volumia
Cleaning up domain volumia ...
Removing vdisk disk0 ...
Removing vdisk disk1 ...
Removing domain volumia ...
Removing volume volumia-vol0@primary-vds0 ...
Removing ZFS volume tank/ldoms/volumia/disk0 ...
Removing volume volumia-vol1@primary-vds0 ...
Removing ZFS volume tank/ldoms/volumia/disk1 ...
```

- The following shows that you can resize one or more file systems during P2V by specifying the mount point and the new size with the `-m` option:

```
ldmp2v prepare -d /home/dana/p2v/volumia -m /:8g volumia
Resizing file systems ...
Creating vdisks ...
Creating file systems ...
Populating file systems ...
Modifying guest domain OS image ...
Removing SVM configuration ...
Modifying file systems on SVM devices ...
Unmounting guest file systems ...
Creating domain volumia ...
Attaching vdisks to domain volumia ...
```

**EXAMPLE 13-3** Conversion Phase Examples

The following examples show how you might use the `ldmp2v convert` command.

- **Using a network installation server.** The `ldmp2v convert` command boots the domain over the network by using the specified virtual network interface. You must run the `setup_install_server` and `add_install_client` scripts on the installation server.

You can use the Custom JumpStart feature to perform a completely hands-off conversion. This feature requires that you create and configure the appropriate `sysidcfg` and profile files for the client on the JumpStart server. The profile should consist of the following lines:

```
install_type upgrade
root_device c0d0s0
```

The `sysidcfg` file is only used for the upgrade operation, so a configuration such as the following should be sufficient:

```
name_service=NONE
root_password=uQkoXlMLCsZhI
system_locale=C
timeserver=localhost
timezone=Europe/Amsterdam
terminal=vt100
security_policy=NONE
nfs4_domain=dynamic
auto_reg=disable
network_interface=PRIMARY {netmask=255.255.255.192
 default_route=none protocol_ipv6=no}
```

For more information about using Custom JumpStart, see [Oracle Solaris 10 9/10 Installation Guide: Custom JumpStart and Advanced Installations](#).

---

**Note** – The example `sysidcfg` file includes the `auto_reg` keyword, which was introduced in the Oracle Solaris 10 9/10 release. This keyword is *only* required if you are running at least the Oracle Solaris 10 9/10 release.

---

```
ldmp2v convert -j -n vnet0 -d /p2v/volumia volumia
LDom volumia started
Waiting for Solaris to come up ...
Using Custom JumpStart
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.

Connecting to console "volumia" in group "volumia"
Press ~? for control options ..
SunOS Release 5.10 Version Generic_137137-09 64-bit
Copyright (c) 1983-2010, Oracle and/or its affiliates. All rights reserved.
Configuring devices.
Using RPC Bootparams for network configuration information.
Attempting to configure interface vnet0...
Configured interface vnet0
Reading ZFS config: done.
Setting up Java. Please wait...
```

**EXAMPLE 13-3** Conversion Phase Examples (Continued)

```
Serial console, reverting to text install
Beginning system identification...
Searching for configuration file(s)...
Using sysid configuration file
 129.159.206.54:/opt/SUNWjet/Clients/volumia/sysidcfg
Search complete.
Discovering additional network configuration...
Completing system identification...
Starting remote procedure call (RPC) services: done.
System identification complete.
Starting Solaris installation program...
Searching for JumpStart directory...
Using rules.ok from 129.159.206.54:/opt/SUNWjet.
Checking rules.ok file...
Using begin script: Clients/volumia/begin
Using profile: Clients/volumia/profile
Using finish script: Clients/volumia/finish
Executing JumpStart preinstall phase...
Executing begin script "Clients/volumia/begin"...
Begin script Clients/volumia/begin execution completed.
Searching for SolStart directory...
Checking rules.ok file...
Using begin script: install_begin
Using finish script: patch_finish
Executing SolStart preinstall phase...
Executing begin script "install_begin"...
Begin script install_begin execution completed.
WARNING: Backup media not specified. A backup media (backup_media)
keyword must be specified if an upgrade with disk space reallocation
is required
```

Processing profile

Loading local environment and services

Generating upgrade actions  
Checking file system space: 100% completed  
Space check complete.

Building upgrade script

Preparing system for Solaris upgrade

Upgrading Solaris: 10% completed  
[...]

- **Using an ISO image.** The `ldmp2v convert` command attaches the Oracle Solaris DVD ISO image to the logical domain and boots from it. To upgrade, answer all `sysid` prompts and select Upgrade.

## EXAMPLE 13-3 Conversion Phase Examples (Continued)



**Caution** – A safety check is performed prior to converting the guest domain. This check ensures that none of the original system's IP addresses are active so as to prevent duplicate active IP addresses on the network. You can use the `-x skip-ping-test` option to skip this safety check. Skipping this check speeds up the conversion process. Use this option *only* if you are certain that no duplicate IP addresses exist, such as when the original host is not active.

**Note** – The answers to the `sysid` questions are *only* used for the duration of the upgrade process. This data is not applied to the existing OS image on disk. The fastest and simplest way to run the conversion is to select Non-networked. The root password that you specify does not need to match the root password of the source system. The system's original identity is preserved by the upgrade and takes effect after the post-upgrade reboot. The time required to perform the upgrade depends on the Oracle Solaris Cluster that is installed on the original system.

```
ldmp2v convert -i /tank/iso/s10s_u5.iso -d /home/dana/p2v/volumia volumia
Testing original system status ...
LDom volumia started
Waiting for Solaris to come up ...
```

```
 Select 'Upgrade' (F2) when prompted for the installation type.
 Disconnect from the console after the Upgrade has finished.
```

```
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.
```

```
Connecting to console "volumia" in group "volumia"
Press ~? for control options ..
Configuring devices.
Using RPC Bootparams for network configuration information.
Attempting to configure interface vnet0...
Extracting windowing system. Please wait...
Beginning system identification...
Searching for configuration file(s)...
Search complete.
Discovering additional network configuration...
Configured interface vnet0
Setting up Java. Please wait...
```

```
Select a Language
```

- 0. English
- 1. French
- 2. German
- 3. Italian
- 4. Japanese

**EXAMPLE 13-3** Conversion Phase Examples      *(Continued)*

- 5. Korean
- 6. Simplified Chinese
- 7. Spanish
- 8. Swedish
- 9. Traditional Chinese

Please make a choice (0 - 9), or press h or ? for help:

[...]

- Solaris Interactive Installation -----

This system is upgradable, so there are two ways to install the Solaris software.

The Upgrade option updates the Solaris software to the new release, saving as many modifications to the previous version of Solaris software as possible. Back up the system before using the Upgrade option.

The Initial option overwrites the system disks with the new version of Solaris software. This option allows you to preserve any existing file systems. Back up any modifications made to the previous version of Solaris software before starting the Initial option.

After you select an option and complete the tasks that follow, a summary of your actions will be displayed.

-----  
F2\_Upgrade    F3\_Go Back    F4\_Initial    F5\_Exit    F6\_Help

# Oracle VM Server for SPARC Configuration Assistant

---

The Oracle VM Server for SPARC Configuration Assistant leads you through the configuration of a logical domain by setting basic properties. It runs on chip multithreading (CMT)-based systems.

After gathering the configuration data, the Configuration Assistant creates a configuration that is suitable for booting as a logical domain. You can also use the default values selected by the Configuration Assistant to create a usable system configuration.

In addition to this chapter, see the [ldmconfig\(1M\)](#) man page.

## Using the Configuration Assistant (ldmconfig)

The `ldmconfig` utility works through a series of operations that correspond to user interface screens. The end result is the creation of a configuration that you can deploy to a logical domain.

The following sections describe how to install the `ldmconfig` command and some features of the Configuration Assistant tool.

## Installing the Configuration Assistant

The Configuration Assistant is delivered as part of the `SUNWldm` package.

After you install the `SUNWldm` package, you can find the `ldmconfig` command in the `/usr/sbin` directory. The command is also installed in the `/opt/SUNWldm/bin` directory for legacy purposes.

## Prerequisites

Before you install and run the Configuration Assistant, ensure that the following conditions are met:

- The target system must be running at least the Logical Domains 1.2 software.
- Your terminal window must be at least 80 characters wide by 24 lines long.

## Limitations and Known Issues

The Configuration Assistant has the following limitations:

- Resizing the terminal while using `ldmconfig` might cause garbled output
- Support for UFS disk files as virtual disks only
- Only works with systems where no existing logical domains configurations are present
- Virtual console concentrator ports are from 5000 to 5100
- Default names that are used for guest domains, services, and devices cannot be changed

## ldmconfig Features

The `ldmconfig` utility works through a series of operations that correspond to user interface screens. You can navigate backward (previous) and forward (next) through these screens until you reach the final step. The final step produces the configuration. At any time you can quit the Configuration Assistant or reset the configuration to use the defaults. From the final screen, you can deploy the configuration to a logical domain.

First, the Configuration Assistant automatically inspects the system to determine the most suitable default property values based on best practices, and then shows those properties that are required to control a deployment. Note that this is not an exhaustive list. You can set other properties to further customize the configuration.

For information about the using the `ldmconfig` tool, see the [ldmconfig\(1M\)](#) man page.

You can adjust the following properties:

- **Number of guest domains.** Specify the number of guest domains for the application to create. The minimum is one guest domain. The maximum value is determined by the availability of VCPU resources. For example, you could create up to 60 guest domains with a single thread each on a 64-thread CMT system, and four threads reserved for the control domain. If best practices are selected, the minimum number of VCPU resources per guest domain is a single core. So, on an 8-core, 8-thread-per-core system with best practices selected, you could create up to seven guest domains with one core each. Also, one core is assigned to the control domain.

The Configuration Assistant shows the maximum number of domains that can be configured for this system.

The Configuration Assistant performs the following tasks to create domains:

- **For all domains.**
  - Creates a virtual terminal service on ports from 5000 to 5100
  - Creates a virtual disk service
  - Creates a virtual network switch on the network adapter nominated
  - Enables the virtual terminal server daemon
- **For each domain.**
  - Creates the logical domain
  - Configures VCPUs assigned to the domain
  - Configures memory assigned to the domain
  - Creates a UFS disk file to use as a virtual disk
  - Creates a virtual disk server device (`vdsdev`) for the disk file
  - Assigns the disk file as virtual disk `vdisk0` for the domain
  - Adds a virtual network adapter attached to the virtual switch on the network adapter nominated
  - Sets the OBP property `auto-boot?=true`
  - Sets the OBP property `boot-device=vdisk0`
  - Binds the domain
  - Starts the domain
- **Default network.** Specify the network adapter that the new domains will use for virtual networking. The adapter must be present in the system. The Configuration Assistant highlights those adapters that are currently in use by the system as default adapters, and those that have active link status (cabled adapters).
- **Virtual disk size.** Create virtual disks for each of the new domains. These virtual disks are created based on the disk files that are located in the local file systems. This property controls the size of each virtual disk in Gbytes. The minimum size, 8 Gbytes, is based on the approximate size required to contain a Oracle Solaris 10 OS, and the maximum size is 100 Gbytes.
 

If the Configuration Assistant cannot find file systems that have adequate space to contain the disk files for all domains, an error screen is shown. In this case, you might need to do the following before rerunning the application:

  - Reduce the size of the virtual disks
  - Reduce the number of domains
  - Add more higher-capacity file systems
- **Virtual disk directory.** Specify a file system that has sufficient capacity on which to store the files to be created as virtual disks for the new domains. The directory is based on the number of domains that are selected and the size of the virtual disks. The value must be recalculated and destination directories selected any time that these property values are changed. The

Configuration Assistant presents you with a list of file systems that have sufficient space. After you specify the file system name, the Configuration Assistant creates a directory in this file system called `/ldoms/disks` in which to create the disk images.

- **Best practice.** Specify whether to use best practice for property values.
  - When the value is yes, the Configuration Assistant uses best practice for several configuration property values. It forces the minimum of one core per domain, including the system domains. As a result, this limits the maximum number of guest domains to the total number of cores present in the system minus one core for the system domains. For example, in the case of a two-socket SPARC Enterprise T5140 with eight cores each, the maximum number of guest domains is 15 plus the system domain.
  - When the value is no, the Configuration Assistant permits the creation of domains that have a minimum of one thread, but maintain at least four threads for the system domain.

Next, the Configuration Assistant summarizes the deployment configuration to be created, which includes the following information:

- Number of domains
- CPU assigned to each guest domain
- Memory assigned to each guest domain
- Size and location of the virtual disks
- Network adapter to be used for virtual network services for guest domains
- Amount of CPU and memory to be used by the system for services
- If a valid Oracle Solaris OS DVD was identified, it will be used to create a shared virtual CD-ROM device to permit guest domains to install the Oracle Solaris OS

Finally, the Configuration Assistant configures the system to create the specified Logical Domains deployment. It also describes the actions to be taken and shows the commands to be run to configure the system. This information can assist you in learning how to use the `ldm` commands that are needed to configure the system.



**Caution** – Do *not* interact with this configuration step and do *not* interrupt this process as it might result in a partially configured system.

---

After the commands have been completed successfully, reboot the system for the changes to take effect.

# Using the Oracle VM Server for SPARC Management Information Base Software

---

The Oracle VM Server for SPARC Management Information Base (MIB) enables third-party system management applications to perform remote monitoring of domains, and to start and stop logical domains (domains) by using the Simple Network Management Protocol (SNMP).

You can run only one instance of the Oracle VM Server for SPARC MIB software on the control domain. The control domain should run at least the Solaris 10 11/06 OS and at least the Oracle VM Server for SPARC 2.1 software.

This chapter covers the following topics:

- [“Oracle VM Server for SPARC Management Information Base Overview” on page 222](#)
- [“Installing and Configuring the Oracle VM Server for SPARC MIB Software” on page 225](#)
- [“Managing Security” on page 227](#)
- [“Monitoring Domains” on page 228](#)
- [“Using SNMP Traps” on page 249](#)
- [“Starting and Stopping Domains” on page 255](#)

To successfully use the Oracle VM Server for SPARC MIB, you must understand how to use the following software products and features:

- Oracle Solaris OS
- Oracle VM Server for SPARC software
- Simple Network Management Protocol (SNMP)
- SNMP Management Information Base (MIB)
- System Management Agent (SMA)
- SNMP version 1 (SNMPv1), SNMP version 2 (SNMPv2c), and SNMP version 3 (SNMPv3) protocols
- Structure of Management Information (SMI) version 1 and version 2
- Management Information Base (MIB) structure
- Abstract Syntax Notation (ASN.1)

# Oracle VM Server for SPARC Management Information Base Overview

This section covers the following topics:

- [“Software Components” on page 222](#)
- [“System Management Agent” on page 223](#)
- [“Logical Domains Manager and the Oracle VM Server for SPARC MIB” on page 224](#)
- [“Oracle VM Server for SPARC MIB Object Tree” on page 224](#)

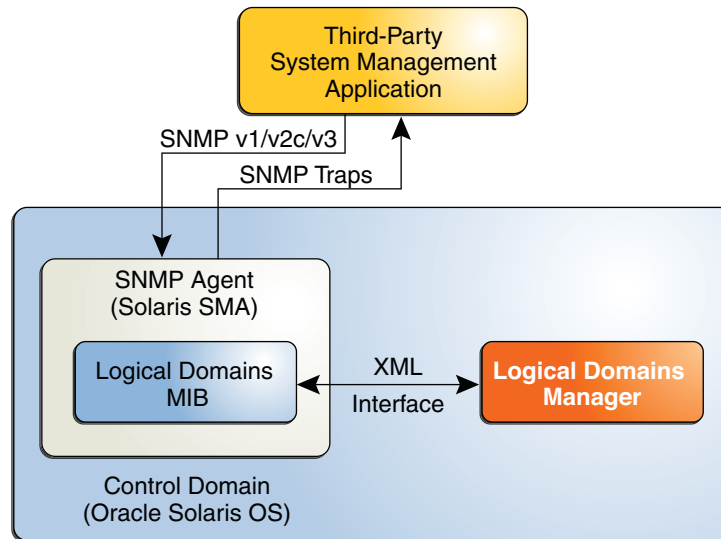
## Software Components

The Oracle VM Server for SPARC MIB package, `SUNWldmib.v`, contains the following software components:

- `SUN-LDOM-MIB.mib` is an SNMP MIB in the form of a text file. This file defines the objects in the Oracle VM Server for SPARC MIB.
- `ldomMIB.so` is a System Management Agent extension module in the form of a shared library. This module enables the SMA to respond to requests for information that are specified in the Oracle VM Server for SPARC MIB and to generate traps.

The following figure shows the interaction between the Oracle VM Server for SPARC MIB, the SMA, the Logical Domains Manager, and a third-party system management application. The interaction shown in this figure is described in [“System Management Agent” on page 223](#) and [“Logical Domains Manager and the Oracle VM Server for SPARC MIB” on page 224](#).

**FIGURE 15-1** Oracle VM Server for SPARC MIB Interaction With SMA, Logical Domains Manager, and Third-Party System Management Application



## System Management Agent

The Solaris SNMP agent (SMA) performs the following functions:

- Listens for requests from a third-party system management application to get or set data offered by the Oracle VM Server for SPARC MIB. The agent listens on the standard SNMP port, 161.
- Issues traps to the configured system management application by using the standard port for SNMP notifications, 162.

The Oracle VM Server for SPARC MIB is exported by the Oracle Solaris OS default SMA on the control domain.

The SMA supports the get, set, and trap functions of SNMP versions v1, v2c, and v3. Most Oracle VM Server for SPARC MIB objects are read-only for monitoring purposes. However, to start or stop a domain, you must write a value to the `ldomAdminState` property of the `ldomTable` table. See [Table 15-1](#).

## Logical Domains Manager and the Oracle VM Server for SPARC MIB

A *domain* is a container that consists of a set of virtual resources for a guest operating system. The Logical Domains Manager provides the command-line interface (CLI) for creating, configuring, and managing the domains. The Logical Domains Manager and the Oracle VM Server for SPARC MIB support the following virtual resources:

- CPUs
- Memory
- Disk, network, and console I/O
- Cryptographic units

### Parsing the XML-Based Control Interface

The Logical Domains Manager exports an XML-based control interface to the Oracle VM Server for SPARC MIB. The Oracle VM Server for SPARC MIB parses the XML interface and populates the MIB. The Oracle VM Server for SPARC MIB only provides support for the control domain.

### Providing SNMP Traps

The Oracle VM Server for SPARC MIB polls the Logical Domains Manager periodically for updates or status changes, and then issues SNMP traps to the system management applications.

### Providing Fault and Recovery Information

If the Oracle VM Server for SPARC MIB can no longer allocate a needed resource, the MIB returns a general error to the system management application through the SNMP agent. The SNMP trap-delivery mechanism does not confirm the error. No specific state or checkpointing is implemented in the Oracle VM Server for SPARC MIB. The SMA with the Oracle VM Server for SPARC MIB is started and monitored by the `init` process and the Service Management Facility (SMF). If the SMA fails and exits, SMF restarts the process automatically, and then the new process dynamically restarts the Oracle VM Server for SPARC MIB module.

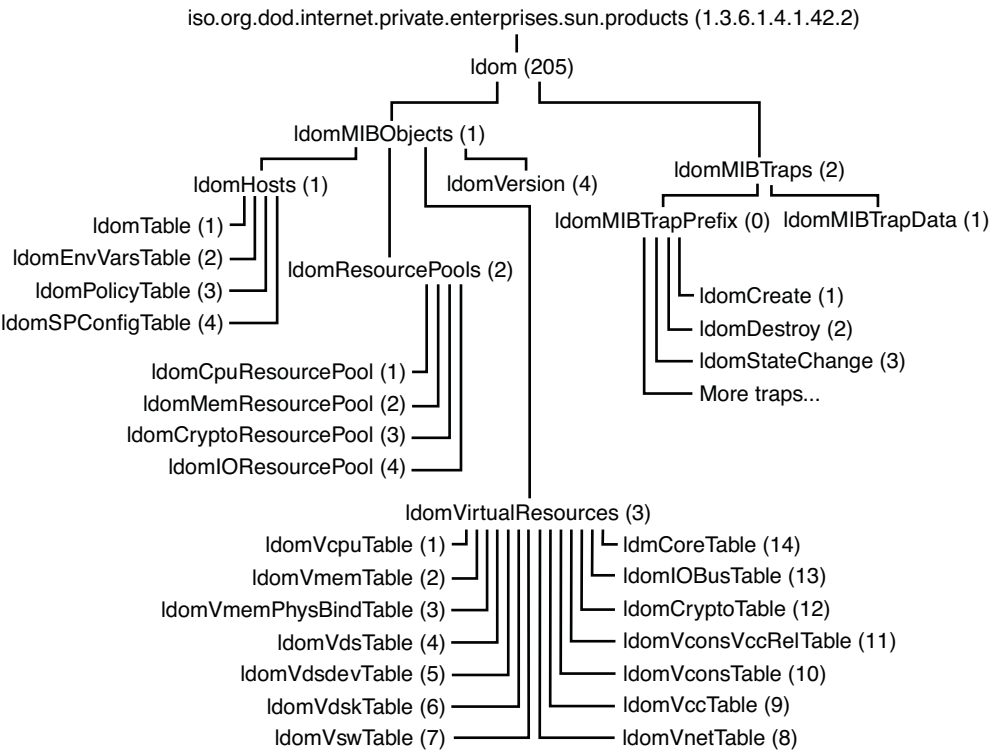
## Oracle VM Server for SPARC MIB Object Tree

SNMP-managed objects are organized into a tree-like hierarchy. An object identifier (OID) consists of a series of integers based on the nodes in the tree, separated by dots. Each managed object has a numerical OID and an associated textual name. The Oracle VM Server for SPARC MIB is registered as the `ldom` (205) branch in this part of the object tree:

```
iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).sun(42).products(2)
```

The following figure shows the major subtrees under the Oracle VM Server for SPARC MIB.

FIGURE 15-2 Oracle VM Server for SPARC MIB Tree



# Installing and Configuring the Oracle VM Server for SPARC MIB Software

This section covers the installation and configuration of the Oracle VM Server for SPARC MIB software. For information about administering SNMP, see the [snmpd.conf\(4\)](#) man page.

## Installing and Configuring the Oracle VM Server for SPARC MIB Software (Task Map)

The following table points to the tasks that you can use to install and configure the Oracle VM Server for SPARC MIB software.

| Task                                                                                | Description                                                                                                         | For Instructions                                                                          |
|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| Install the Oracle VM Server for SPARC MIB software package on the primary domain.  | Use the <code>pkgadd</code> command to install the <code>SUNWldmib.v</code> package.                                | <a href="#">“Install the Oracle VM Server for SPARC MIB Software Package” on page 226</a> |
| Load the Oracle VM Server for SPARC MIB module into the SMA.                        | Load the <code>ldomMIB.so</code> module into the Oracle Solaris OS SMA to query the Oracle VM Server for SPARC MIB. | <a href="#">“Load the Oracle VM Server for SPARC MIB Module Into the SMA” on page 226</a> |
| Remove the Oracle VM Server for SPARC MIB software package from the primary domain. | Use the <code>pkgrm</code> command to remove the <code>SUNWldmib.v</code> package.                                  | <a href="#">“Remove the Oracle VM Server for SPARC MIB Software Package” on page 227</a>  |

## ▼ Install the Oracle VM Server for SPARC MIB Software Package

This procedure describes how to install the Oracle VM Server for SPARC MIB software package, `SUNWldmib.v`, which is included as part of the Oracle VM Server for SPARC 2.1 software.

After you install this package, you can configure your system to dynamically load the Oracle VM Server for SPARC MIB module. See [“Load the Oracle VM Server for SPARC MIB Module Into the SMA” on page 226](#).

**Before You Begin** Download and install the Oracle VM Server for SPARC 2.1 software. See [Chapter 2, “Installing and Enabling Software.”](#)

### ● Add the `SUNWldmib.v` package to the primary domain.

```
pkgadd -d . SUNWldmib.v
```

This command installs the following files:

- `/opt/SUNWldmib/lib/mibs/SUN-LDOM-MIB.mib`
- `/opt/SUNWldmib/lib/ldomMIB.so`

## ▼ Load the Oracle VM Server for SPARC MIB Module Into the SMA

The Oracle VM Server for SPARC MIB module, `ldomMIB.so`, must be loaded into the Oracle Solaris OS SMA to query the Oracle VM Server for SPARC MIB. The Oracle VM Server for SPARC MIB module is dynamically loaded so that the module is included within the SNMP agent without requiring you to recompile and relink the agent binary.

This procedure describes how to configure your system to dynamically load the Oracle VM Server for SPARC MIB module. Instructions for dynamically loading the module without restarting the SMA are provided in *Solaris System Management Agent Developer's Guide*. For more information about the SMA, see *Solaris System Management Administration Guide*.

### 1 Update the SMA SNMP configuration file.

Append the following line to the `/etc/sma/snmp/snmpd.conf` configuration file:

```
dlmod ldومMIB /opt/SUNWldمib/lib/ldومMIB.so
```

### 2 Restart the SMA.

```
svcadm restart svc:/application/management/sma:default
```

## ▼ Remove the Oracle VM Server for SPARC MIB Software Package

This procedure describes how to remove the Oracle VM Server for SPARC MIB software package, `SUNWldمib.v`, and unload the Oracle VM Server for SPARC MIB module from the SMA.

### 1 Stop the SMA.

```
svcadm disable svc:/application/management/sma:default
```

### 2 Remove the Oracle VM Server for SPARC MIB software package from the primary domain.

```
pkgrm SUNWldمib
```

### 3 Update the SMA SNMP configuration file.

Remove the line that you added to the `/etc/sma/snmp/snmpd.conf` file during installation.

```
dlmod ldومMIB /opt/SUNWldمib/lib/ldومMIB.so
```

### 4 Restart the SMA.

```
svcadm restart svc:/application/management/sma:default
```

## Managing Security

This section describes how to create new Simple Network Management Protocol (SNMP) version 3 (v3) users to provide secure access to the System Management Agent (SMA). For SNMP version 1 (v1) and version 2 (v2c), the access control mechanism is the *community string*, which defines the relationship between an SNMP server and its clients. This string controls the client access to the server similar to a password controlling a user's access to a system. See *Solaris System Management Agent Administration Guide*.

---

**Note** – Creating `snmpv3` users enables you to use the SMA in SNMP with the Oracle VM Server for SPARC MIB. This type of user in no way interacts with or conflicts with users that you might have configured by using the role-based access control (RBAC) feature of Oracle Solaris for the Logical Domains Manager.

---

## ▼ Create the Initial snmpv3 User

This procedure describes how to create the initial snmpv3 user.

You can create additional users by cloning this initial user. Cloning enables subsequent users to inherit the initial user's authentication and security types. You can change these types later.

When you clone the initial user, you set secret key data for the new user. You must know the passwords for the initial user and for the subsequent users that you configure. You can only clone one user at a time from the initial user. See “To Create Additional SNMPv3 Users with Security” in *Solaris System Management Agent Administration Guide* for your version of the Oracle Solaris OS.

### 1 Stop the SMA.

```
svcadm disable -t svc:/application/management/sma:default
```

### 2 Create the initial user.

```
/usr/sfw/bin/net-snmp-config --create-snmpv3-user -a my-password initial-user
```

This command creates user *initial-user* with a password that you choose, *my-password*, and adds an entry to the `/etc/sma/snmp/snmpd.conf` file. This entry gives the initial user read and write access to the agent.

---

**Note** – Passwords must contain at least eight characters.

---

### 3 Start the SMA.

```
svcadm enable svc:/application/management/sma:default
```

### 4 Verify that the initial user has been created.

```
snmpget -v 3 -u initial-user -l authNoPriv -a MD5 -A my-password localhost sysUpTime.0
```

## Monitoring Domains

This section describes how to monitor logical domains (domains) by querying the Oracle VM Server for SPARC MIB. This section also provides descriptions of the various types of MIB output.

This section covers the following topics:

- “Setting Environment Variables” on page 229
- “Querying the Oracle VM Server for SPARC MIB” on page 229
- “Retrieving Oracle VM Server for SPARC MIB Information” on page 231

# Setting Environment Variables

## ▼ Set Environment Variables

Before you can query the Oracle VM Server for SPARC MIB, you must set environment variables for the shell that you use. This procedure describes how to set these variables for the C shell, Bourne shell, and Korn shell.

### ● Set the PATH, MIBDIRS, and MIBS environment variables.

#### ■ For C shell users:

```
% setenv PATH /usr/sfw/bin:$PATH
% setenv MIBDIRS /opt/SUNWldmib/lib/mibs:/etc/sma/snmp/mibs
% setenv MIBS +SUN-LDOM-MIB
```

#### ■ For Bourne and Korn shell users:

```
$ PATH=/usr/sfw/bin:$PATH; export PATH
$ MIBDIRS=/opt/SUNWldmib/lib/mibs:/etc/sma/snmp/mibs; export MIBDIRS
$ MIBS=+SUN-LDOM-MIB; export MIBS
```

# Querying the Oracle VM Server for SPARC MIB

## ▼ Retrieve Oracle VM Server for SPARC MIB Objects

When a system has large number of domains, the SNMP agent might time out before being able to respond to an SNMP request. To increase the timeout value, use the `-t` option to specify a longer timeout value. For example, the following `snmpwalk` command sets the timeout value to 20 seconds:

```
snmpwalk -t 20 -v1 -c public localhost SUN-LDOM-MIB::ldomTable
```

You can also use the `-t` option to specify the timeout value for the `snmpget` and `snmptable` commands.

### ● Retrieve one or more MIB objects.

#### ■ Retrieve a single MIB object.

```
snmpget -v version -c community-string host MIB-object
```

#### ■ Retrieve an array of MIB objects.

Use the `snmpwalk` or `snmptable` command.

```
snmpwalk -v version -c community-string host MIB-object
snmptable -v version -c community-string host MIB-object
```

**Example 15-1** Retrieving a Single Oracle VM Server for SPARC MIB Object (snmpget)

The following `snmpget` command queries the value of the `ldomVersionMajor` object. The command specifies `snmpv1` (`-v1`) and a community string (`-c public`) for the `localhost` host.

```
snmpget -v1 -c public localhost SUN-LDOM-MIB::ldomVersionMajor.0
SUN-LDOM-MIB::ldomVersionMajor.0 = INTEGER: 1
```

**Example 15-2** Retrieving Object Values From `ldomTable` (snmpwalk)

The following examples show how to use the `snmpwalk` command to retrieve object values from `ldomTable`.

- The following `snmpwalk -v1` command returns the values for all objects in the `ldomTable` table:

```
snmpwalk -v1 -c public localhost SUN-LDOM-MIB::ldomTable
SUN-LDOM-MIB::ldomName.1 = STRING: primary
SUN-LDOM-MIB::ldomName.2 = STRING: LdomMibTest_1
SUN-LDOM-MIB::ldomAdminState.1 = INTEGER: 0
SUN-LDOM-MIB::ldomAdminState.2 = INTEGER: 0
SUN-LDOM-MIB::ldomOperState.1 = INTEGER: active(1)
SUN-LDOM-MIB::ldomOperState.2 = INTEGER: bound(6)
SUN-LDOM-MIB::ldomNumVCpu.1 = INTEGER: 32
SUN-LDOM-MIB::ldomNumVCpu.2 = INTEGER: 2
SUN-LDOM-MIB::ldomMemSize.1 = INTEGER: 3968
SUN-LDOM-MIB::ldomMemSize.2 = INTEGER: 256
SUN-LDOM-MIB::ldomMemUnit.1 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomMemUnit.2 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomNumCrypto.1 = INTEGER: 8
SUN-LDOM-MIB::ldomNumCrypto.2 = INTEGER: 0
SUN-LDOM-MIB::ldomNumIOBus.1 = INTEGER: 2
SUN-LDOM-MIB::ldomNumIOBus.2 = INTEGER: 0
SUN-LDOM-MIB::ldomUUID.1 = STRING: c2c3d93b-a3f9-60f6-a45e-f35d55c05fb6
SUN-LDOM-MIB::ldomUUID.2 = STRING: af0b05f0-d262-e633-af32-a6c4e81fb81c
SUN-LDOM-MIB::ldomMacAddress.1 = STRING: 00:14:4f:86:63:2a
SUN-LDOM-MIB::ldomMacAddress.2 = STRING: 00:14:4f:fa:78:b9
SUN-LDOM-MIB::ldomHostID.1 = STRING: 0x8486632a
SUN-LDOM-MIB::ldomHostID.2 = STRING: 0x84fa78b9
SUN-LDOM-MIB::ldomFailurePolicy.1 = STRING: ignore
SUN-LDOM-MIB::ldomFailurePolicy.2 = STRING: ignore
SUN-LDOM-MIB::ldomMaster.1 = STRING:
SUN-LDOM-MIB::ldomMaster.2 = STRING:
```

- The following `snmpwalk` commands use `snmpv2c` and `snmpv3` to retrieve the contents of `ldomTable`:

```
snmpwalk -v2c -c public localhost SUN-LDOM-MIB::ldomTable
snmpwalk -v 3 -u test -l authNoPriv -a MD5 -A testpassword localhost \
SUN-LDOM-MIB::ldomTable
```

**Example 15–3 Retrieving Object Values From ldomTable in Tabular Form (snmptable)**

The following examples show how to use the `snmptable` command to retrieve object values from `ldomTable` in tabular form.

- The following `snmptable -v1` command shows the contents of `ldomTable` in tabular form:

```
snmptable -v1 -c public localhost SUN-LDOM-MIB::ldomTable
```

- The following `snmptable` command shows the contents of `ldomTable` in tabular form by using `snmpv2c`.

Note that for the `v2c` or `v3` `snmptable` command, use the `-CB` option to specify only `GETNEXT`, not `GETBULK`, requests to retrieve data.

```
snmptable -v2c -CB -c public localhost SUN-LDOM-MIB::ldomTable
```

## Retrieving Oracle VM Server for SPARC MIB Information

This section describes the information that you can retrieve from the Oracle VM Server for SPARC MIB in the form of tables or scalar objects.

### Domain Table (ldomTable)

`ldomTable` is used to represent each domain in the system. Information includes resource constraints for virtual CPUs, memory, cryptographic units, and I/O buses. The table also includes other domain information, such as the universally unique identifier (UUID), MAC address, host ID, failure policy, and master domain.

**TABLE 15–1** Domain Table (`ldomTable`)

| Name                        | Data Type      | Access         | Description                                                                                                                                                               |
|-----------------------------|----------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ldomIndex</code>      | Integer        | Not accessible | Integer that is used as an index of this table                                                                                                                            |
| <code>ldomName</code>       | Display string | Read-only      | Name of the domain                                                                                                                                                        |
| <code>ldomAdminState</code> | Integer        | Read/Write     | Starts or stops the domain for active management: <ul style="list-style-type: none"> <li>■ Value of 1 starts the domain</li> <li>■ Value of 2 stops the domain</li> </ul> |

TABLE 15-1 Domain Table (ldomTable) (Continued)

| Name              | Data Type      | Access    | Description                                                                                                                                                                                                                                                                                                                                             |
|-------------------|----------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomOperState     | Integer        | Read-only | Current state of the domain, which can be one of the following values: <ul style="list-style-type: none"><li>■ 1 is the Active state</li><li>■ 2 is the Stopping state</li><li>■ 3 is the Inactive state</li><li>■ 4 is the Binding state</li><li>■ 5 is the Unbinding state</li><li>■ 6 is the Bound state</li><li>■ 7 is the Starting state</li></ul> |
| ldomNumVCPU       | Integer        | Read-only | Number of virtual CPUs used. If the domain is in an inactive state, this value is the requested number of virtual CPUs.                                                                                                                                                                                                                                 |
| ldomMemSize       | Integer        | Read-only | Amount of virtual memory used. If the domain is in an inactive state, this value is the requested memory size.                                                                                                                                                                                                                                          |
| ldomMemUnit       | Integer        | Read-only | One of the following memory units: <ul style="list-style-type: none"><li>■ 1 is KB</li><li>■ 2 is MB</li><li>■ 3 is GB</li><li>■ 4 is bytes</li></ul> If not specified, the unit value is bytes.                                                                                                                                                        |
| ldomNumCrypto     | Integer        | Read-only | Number of cryptographic units used. If the domain is in an inactive state, this value is the requested number of cryptographic units.                                                                                                                                                                                                                   |
| ldomNumIOBus      | Integer        | Read-only | Number of physical I/O devices used                                                                                                                                                                                                                                                                                                                     |
| ldomUUID          | Display string | Read-only | UUID of the domain                                                                                                                                                                                                                                                                                                                                      |
| ldomMacAddress    | Display string | Read-only | MAC address of the domain                                                                                                                                                                                                                                                                                                                               |
| ldomHostID        | Display string | Read-only | Host ID of the domain                                                                                                                                                                                                                                                                                                                                   |
| ldomFailurePolicy | Display string | Read-only | Master domain's failure policy, which can be one of ignore, panic, reset, or stop                                                                                                                                                                                                                                                                       |
| ldomMaster        | Display string | Read-only | Name of up to four master domains for a slave domain                                                                                                                                                                                                                                                                                                    |

Environment Variables Table (ldomEnvVarsTable)

ldomEnvVarsTable describes the OpenBoot PROM environment variables that all domains use.

**TABLE 15–2** Environment Variables Table (ldomEnvVarsTable)

| Name                 | Data Type      | Access    | Description                                                                                                                      |
|----------------------|----------------|-----------|----------------------------------------------------------------------------------------------------------------------------------|
| ldomEnvVarsLdomIndex | Integer        | Read-only | Integer that is used as an index into ldomTable that represents the domain that contains the OpenBoot PROM environment variables |
| ldomEnvVarsIndex     | Integer        | Read-only | Integer that is used to index the OpenBoot PROM environment variables in this table                                              |
| ldomEnvVarsName      | Display string | Read-only | Name of the OpenBoot PROM variable                                                                                               |
| ldomEnvVarsValue     | Display string | Read-only | Value of the OpenBoot PROM variable                                                                                              |

## Domain Policy Table (ldomPolicyTable)

ldomPolicyTable describes the dynamic resource management (DRM) policies that apply to all domains.

**TABLE 15–3** Domain Policy Table (ldomPolicyTable)

| Name                | Data Type      | Access         | Description                                                                                                     |
|---------------------|----------------|----------------|-----------------------------------------------------------------------------------------------------------------|
| ldomPolicyLdomIndex | Integer        | Read-only      | Integer that is used as an index into ldomTable that represents the domain that contains the DRM policy         |
| ldomPolicyIndex     | Integer        | Not accessible | Integer that is used to index the DRM policy in this table                                                      |
| ldomPolicyName      | Display string | Read-only      | Policy name                                                                                                     |
| ldomPolicyStatus    | Display string | Read-only      | Policy status                                                                                                   |
| ldomPolicyPriority  | Integer        | Read-only      | Priority that is used to determine which DRM policy is selected when policies overlap                           |
| ldomPolicyVcpuMin   | Integer        | Read-only      | Minimum number of virtual CPUs for a domain                                                                     |
| ldomPolicyVcpuMax   | Integer        | Read-only      | Maximum number of virtual CPUs for a domain. A value of unlimited uses the maximum integer value of 2147483647. |
| ldomPolicyUtilLower | Integer        | Read-only      | Lower utilization level at which policy analysis is triggered                                                   |
| ldomPolicyUtilUpper | Integer        | Read-only      | Upper utilization level at which policy analysis is triggered                                                   |

**TABLE 15–3** Domain Policy Table (ldomPolicyTable) *(Continued)*

| Name                    | Data Type      | Access    | Description                                                                                                                                               |
|-------------------------|----------------|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomPolicyTodBegin      | Display string | Read-only | Effective start time of a policy with a format of <i>hh:mm:ss</i>                                                                                         |
| ldomPolicyTodEnd        | Display string | Read-only | Effective stop time of a policy with a format of <i>hh:mm:ss</i>                                                                                          |
| ldomPolicySampleRate    | Integer        | Read-only | Resource cycle time in seconds                                                                                                                            |
| ldomPolicyElasticMargin | Integer        | Read-only | Amount of buffer between util - lower property (ldomPolicyUtilLower) and the number of free virtual CPUs to avoid oscillations at low virtual CPU counts  |
| ldomPolicyAttack        | Integer        | Read-only | Maximum amount of a resource to be added during any one resource-control cycle. A value of <i>unlimited</i> uses the maximum integer value of 2147483647. |
| ldomPolicyDecay         | Integer        | Read-only | Maximum amount of a resource to be removed during any one resource-control cycle                                                                          |

## Service Processor Configuration Table (ldomSPConfigTable)

ldomSPConfigTable describes the service processor (SP) configurations for all domains.

**TABLE 15–4** Service Processor Configuration Table (ldomSPConfigTable)

| Name               | Data Type      | Access         | Description                                                     |
|--------------------|----------------|----------------|-----------------------------------------------------------------|
| ldomSPConfigIndex  | Integer        | Not accessible | Integer that is used to index an SP configuration in this table |
| ldomSPConfigName   | Display string | Read-only      | SP configuration name                                           |
| ldomSPConfigStatus | Display string | Read-only      | SP configuration status                                         |

## Domain Resource Pool and Scalar Variables

The following resources can be assigned to domains:

- Virtual CPU (vcpu)
- Memory (mem)
- Cryptographic unit (mau)
- Virtual switch (vsw)
- Virtual network (vnet)
- Virtual disk server (vds)
- Virtual disk server device (vdsdev)
- Virtual disk (vdisk)

- Virtual console concentrator (vcc)
- Virtual console (vcons)
- Physical I/O device (io)

The following scalar MIB variables are used to represent resource pools and their properties.

**TABLE 15–5** Scalar Variables for CPU Resource Pool

| Name                                                  | Data Type | Access    | Description                                                                                                                                                  |
|-------------------------------------------------------|-----------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomCpuRpCapacity                                     | Integer   | Read-only | Maximum reservation allowed by the resource pool in ldomCpuRpCapacityUnits                                                                                   |
| ldomCpuRpReserved                                     | Integer   | Read-only | Accumulated processor clock speed of the CPU, in MHz, that is currently reserved from the resource pool                                                      |
| ldomCpuRpCapacityUnit<br>and<br>ldomCpuRpReservedUnit | Integer   | Read-only | One of the following CPU allocation units: <ul style="list-style-type: none"> <li>■ 1 is MHz</li> <li>■ 2 is GHz</li> </ul> <p>The default value is MHz.</p> |

**TABLE 15–6** Scalar Variables for Memory Resource Pool

| Name                                                  | Data Type | Access    | Description                                                                                                                                                                                                             |
|-------------------------------------------------------|-----------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomMemRpCapacity                                     | Integer   | Read-only | Maximum reservation allowed by the resource pool in MemRpCapacityUnits                                                                                                                                                  |
| ldomMemRpReserved                                     | Integer   | Read-only | Amount of memory, in MemRpReservedUnits, that is currently reserved from the resource pool                                                                                                                              |
| ldomMemRpCapacityUnit<br>and<br>ldomMemRpReservedUnit | Integer   | Read-only | One of the following memory allocation units: <ul style="list-style-type: none"> <li>■ 1 is KB</li> <li>■ 2 is MB</li> <li>■ 3 is GB</li> <li>■ 4 is bytes</li> </ul> <p>If not specified, the unit value is bytes.</p> |

**TABLE 15–7** Scalar Variables for Cryptographic Resource Pool

| Name                 | Data Type | Access    | Description                                      |
|----------------------|-----------|-----------|--------------------------------------------------|
| ldomCryptoRpCapacity | Integer   | Read-only | Maximum reservation allowed by the resource pool |

**TABLE 15–7** Scalar Variables for Cryptographic Resource Pool *(Continued)*

| Name                              | Data Type | Access    | Description                                                                     |
|-----------------------------------|-----------|-----------|---------------------------------------------------------------------------------|
| <code>ldomCryptoRpReserved</code> | Integer   | Read-only | Number of cryptographic units that is currently reserved from the resource pool |

**TABLE 15–8** Scalar Variables for I/O Bus Resource Pool

| Name                             | Data Type | Access    | Description                                                           |
|----------------------------------|-----------|-----------|-----------------------------------------------------------------------|
| <code>ldomIOBusRpCapacity</code> | Integer   | Read-only | Maximum reservation allowed by the pool                               |
| <code>ldomIOBusRpReserved</code> | Integer   | Read-only | Number of I/O buses that is currently reserved from the resource pool |

## Virtual CPU Table (`ldomVcpuTable`)

`ldomVcpuTable` describes the virtual CPUs that all domains use.

**TABLE 15–9** Virtual CPU Table (`ldomVcpuTable`)

| Name                           | Data Type      | Access         | Description                                                                                                           |
|--------------------------------|----------------|----------------|-----------------------------------------------------------------------------------------------------------------------|
| <code>ldomVcpuLdomIndex</code> | Integer        | Read-only      | Integer that is used as an index into <code>ldomTable</code> that represents the domain that contains the virtual CPU |
| <code>ldomVcpuIndex</code>     | Integer        | Not accessible | Integer that is used to index the virtual CPU in this table                                                           |
| <code>ldomVcpuDeviceID</code>  | Display string | Read-only      | Identifier of the virtual CPU (VID)                                                                                   |

TABLE 15–9 Virtual CPU Table (ldomVcpuTable) (Continued)

| Name                      | Data Type      | Access    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------|----------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomVcpuOperationalStatus | Integer        | Read-only | <p>One of the following CPU statuses:</p> <p>1=Unknown</p> <p>2=Other</p> <p>3=OK</p> <p>4=Degraded</p> <p>5=Stressed</p> <p>6=Predictive failure</p> <p>7=Error</p> <p>8=Nonrecoverable error</p> <p>9=Starting</p> <p>10=Stopping</p> <p>11=Stopped</p> <p>12=In service</p> <p>13=No contact</p> <p>14=Lost communication</p> <p>15=Aborted</p> <p>16=Dormant</p> <p>17=Supporting entity in error</p> <p>18=Completed</p> <p>19=Power mode</p> <p>The default value is 1 (Unknown) because the Logical Domains Manager does not provide the CPU state.</p> |
| ldomVcpuPhysBind          | Display string | Read-only | Physical binding (PID). Contains the identifier of a strand (hardware thread) that is assigned to this virtual CPU. This identifier uniquely identifies the core and chip.                                                                                                                                                                                                                                                                                                                                                                                     |

TABLE 15–9 Virtual CPU Table (ldomVcpuTable) (Continued)

| Name                  | Data Type      | Access    | Description                                                                                                                                                                                                                                                               |
|-----------------------|----------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomVcpuPhysBindUsage | Integer        | Read-only | Indicates how much (in MHz) of the total capacity of the strand is used by this virtual CPU. For example, assume a thread can run at a maximum of one GHz. If only half of that capacity is allocated to this virtual CPU (50% of the strand), the property value is 500. |
| ldomVcpuCoreID        | Display string | Read-only | Identifier of the core (core ID)                                                                                                                                                                                                                                          |
| ldomVcpuUtilPercent   | Display string | Read-only | Indicates the utilization percentage of the virtual CPU                                                                                                                                                                                                                   |

## Virtual Memory Tables

A domain's memory space is referred to as *real memory*, that is, *virtual memory*. Host platform memory space that is detected by the hypervisor is referred to as *physical memory*. The hypervisor maps blocks of physical memory to form a block of real memory that is used by a domain.

The following example shows that the requested memory size can be split between two memory blocks instead of being assigned to a single large memory block. Assume that a domain requests 521 Mbytes of real memory. The memory can be assigned two 256-Mbyte blocks on the host system as physical memory by using the *{physical-address, real-address, size}* format.

```
{0x1000000, 0x1000000, 256}, {0x2000000, 0x2000000, 256}
```

A domain can have up to 64 physical memory segments assigned to a guest domain. So, an auxiliary table, instead of a display string, is used to hold each memory segment. A display string has a 255-character limit.

## Virtual Memory Table (ldomVmemTable)

ldomVmemTable describes the properties of virtual memory that domains use.

TABLE 15–10 Virtual Memory Table (ldomVmemTable)

| Name              | Data Type | Access         | Description                                                                                                 |
|-------------------|-----------|----------------|-------------------------------------------------------------------------------------------------------------|
| ldomVmemLdomIndex | Integer   | Read-only      | Integer that is used as an index into ldomTable that represents the domain that contains the virtual memory |
| ldomVmemIndex     | Integer   | Not accessible | Integer that is used to index the virtual memory in this table                                              |

**TABLE 15–10** Virtual Memory Table (ldomVmemTable) *(Continued)*

| Name                   | Data Type | Access    | Description                        |
|------------------------|-----------|-----------|------------------------------------|
| ldomVmemNumberOfBlocks | Integer   | Read-only | Number of blocks of virtual memory |

## Virtual Memory Physical Binding Table (ldomVmemPhysBindTable)

ldomVmemPhysBindTable is an auxiliary table that contains physical memory segments for all domains.

**TABLE 15–11** Virtual Memory Physical Binding Table (ldomVmemPhysBindTable)

| Name                      | Data Type      | Access    | Description                                                                                                                                                  |
|---------------------------|----------------|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomVmemPhysBindLdomIndex | Integer        | Read-only | Integer that is used as an index into ldomTable that represents the domain that contains the physical memory segments                                        |
| ldomVmemPhysBind          | Display string | Read-only | List of physical memory that is mapped to this virtual memory block in the following format: { <i>physical-address</i> , <i>real-address</i> , <i>size</i> } |

## Virtual Disk Tables

A virtual disk service (vds) and the physical device to which it maps (vdsdev) provide the virtual disk capability to the Oracle VM Server for SPARC technology. A virtual disk service exports a number of local volumes (physical disks or file systems). When a virtual disk service is specified, the following are included:

- Complete /dev path of the backing device (vdsdev)
- Unique name (volume name) for the device being added to the service

One or more disks, disk slices, and file systems can be bound to a single disk service. Each disk has a unique name and volume name. The volume name is used when the disk is bound to the service. The Logical Domains Manager creates virtual disk clients (vdisk) from the virtual disk service and its logical volumes.

## Virtual Disk Service Table (ldomVdsTable)

ldomVdsTable describes the virtual disk services for all domains.

**TABLE 15–12** Virtual Disk Service Table (ldomVdsTable)

| Name             | Data Type | Access    | Description                                                                                                       |
|------------------|-----------|-----------|-------------------------------------------------------------------------------------------------------------------|
| ldomVdsLdomIndex | Integer   | Read-only | Integer that is used as an index into ldomTable that represents the domain that contains the virtual disk service |

**TABLE 15–12** Virtual Disk Service Table (ldomVdsTable) *(Continued)*

| Name                    | Data Type      | Access         | Description                                                                                                                                 |
|-------------------------|----------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| ldomVdsIndex            | Integer        | Not accessible | Integer that is used to index the virtual disk service in this table                                                                        |
| ldomVdsServiceName      | Display string | Read-only      | Service name for the virtual disk service. The property value is the <i>service-name</i> specified by the <code>ldm add-vds</code> command. |
| ldomVdsNumofAvailVolume | Integer        | Read-only      | Number of logical volumes exported by this virtual disk service                                                                             |
| ldomVdsNumofUsedVolume  | Integer        | Read-only      | Number of logical volumes used (bound) to this virtual disk service                                                                         |

## Virtual Disk Service Device Table (ldomVdsdevTable)

ldomVdsdevTable describes the virtual disk service devices that all virtual disk services use.

**TABLE 15–13** Virtual Disk Service Device Table (ldomVdsdevTable)

| Name                 | Data Type      | Access         | Description                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|----------------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomVdsdevVdsIndex   | Integer        | Read-only      | Integer that is used to index into ldomVdsTable that represents the virtual disk service that contains the virtual disk device                                                                                                                                                                                                                                    |
| ldomVdsdevIndex      | Integer        | Not accessible | Integer that is used to index the virtual disk service device in this table                                                                                                                                                                                                                                                                                       |
| ldomVdsdevVolumeName | Display string | Read-only      | Volume name for the virtual disk service device. This property specifies a unique name for the device that is being added to the virtual disk service. This name is exported by the virtual disk service to the clients for the purpose of adding this device. The property value is the <i>volume-name</i> specified by the <code>ldm add-vdsdev</code> command. |
| ldomVdsdevDevPath    | Display string | Read-only      | Path name of the physical disk device. The property value is the <i>backend</i> specified by the <code>ldm add-vdsdev</code> command.                                                                                                                                                                                                                             |
| ldomVdsdevOptions    | Display string | Read-only      | One or more of the options for the disk device, which are <code>ro</code> , <code>slice</code> , or <code>excl</code>                                                                                                                                                                                                                                             |
| ldomVdsdevMPGroup    | Display string | Read-only      | Multipath group name for the disk device                                                                                                                                                                                                                                                                                                                          |

## Virtual Disk Table (ldomVdiskTable)

ldomVdiskTable describes the virtual disks for all domains.

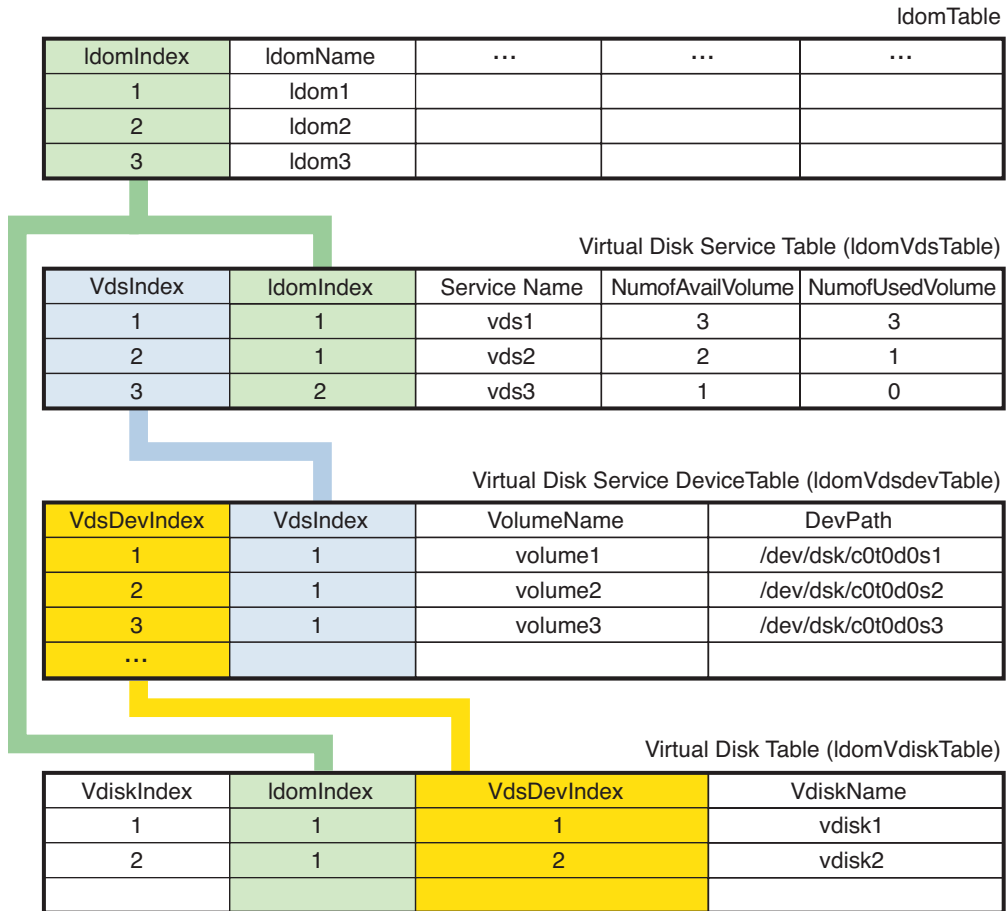
**TABLE 15–14** Virtual Disk Table (ldomVdiskTable)

| Name                 | Data Type      | Access         | Description                                                                                                               |
|----------------------|----------------|----------------|---------------------------------------------------------------------------------------------------------------------------|
| ldomVdiskLdomIndex   | Integer        | Read-only      | Integer that is used as an index into ldomTable that represents the domain that contains the virtual disk device          |
| ldomVdiskVdsDevIndex | Integer        | Read-only      | Integer that is used to index into ldomVdsdevTable that represents the virtual disk service device                        |
| ldomVdiskIndex       | Integer        | Not accessible | Integer that is used to index the virtual disk in this table                                                              |
| ldomVdiskName        | Display string | Read-only      | Name of the virtual disk. The property value is the <i>disk-name</i> specified by the <code>ldm add-vdisk</code> command. |
| ldomVdiskTimeout     | Integer        | Read-only      | Timeout, in seconds, for establishing a connection between a virtual disk client and a virtual disk server                |
| ldomVdiskID          | Display string | Read-only      | Identifier of the virtual disk                                                                                            |

The following figure shows how indexes are used to define relationships among the virtual disk tables and the domain table. The indexes are used as follows:

- ldomIndex in ldomVdsTable and ldomVdiskTable points to ldomTable.
- VdsIndex in ldomVdsdevTable points to ldomVdsTable.
- VdsDevIndex in ldomVdiskTable points to ldomVdsdevTable.

FIGURE 15-3 Relationship Among Virtual Disk Tables and the Domain Table



## Virtual Network Tables

Oracle VM Server for SPARC virtual network support enables guest domains to communicate with each other and with external hosts through a physical Ethernet device. The virtual network contains the following main components:

- Virtual switch (vsw)
- Virtual network device (vnet)

After you create a virtual switch on a service domain, you can bind a physical network device to the virtual switch. After that, you can create a virtual network device for a domain that uses the virtual switch service for communication. The virtual switch service communicates with other domains by connecting to the same virtual switch. The virtual switch service communicates with external hosts if a physical device is bound to the virtual switch.

## Virtual Switch Service Table (ldomVswTable)

ldomVswTable describes the virtual switch services for all domains.

TABLE 15–15 Virtual Switch Service Table (ldomVswTable)

| Name                 | Data Type      | Access         | Description                                                                                                                             |
|----------------------|----------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| ldomVswLdomIndex     | Integer        | Read-only      | Integer that is used as an index into ldomTable that represents the domain that contains the virtual switch service                     |
| ldomVswIndex         | Integer        | Not accessible | Integer that is used to index the virtual switch device in this table                                                                   |
| ldomVswServiceName   | Display string | Read-only      | Virtual switch service name                                                                                                             |
| ldomVswMacAddress    | Display string | Read-only      | MAC address used by the virtual switch                                                                                                  |
| ldomVswPhysDevPath   | Display string | Read-only      | Physical device path for the virtual network switch. The property value is null when no physical device is bound to the virtual switch. |
| ldomVswMode          | Display string | Read-only      | Value is mode=sc for running cluster nodes                                                                                              |
| ldomVswDefaultVlanID | Display string | Read-only      | Default VLAN ID for the virtual switch                                                                                                  |
| ldomVswPortVlanID    | Display string | Read-only      | Port VLAN ID for the virtual switch                                                                                                     |
| ldomVswVlanID        | Display string | Read-only      | VLAN ID for the virtual switch                                                                                                          |
| ldomVswLinkprop      | Display string | Read-only      | Value is linkprop=phys-state to report the link status based on the physical network device                                             |
| ldomVswMtu           | Integer        | Read-only      | Maximum transmission unit (MTU) for a virtual switch device                                                                             |
| ldomVswID            | Display string | Read-only      | Identifier of the virtual switch device                                                                                                 |
| ldomVswInterVnetLink | Display string | Read-only      | State of LDC channel assignment for inter-vnet communications. Value is either on or off.                                               |

## Virtual Network Device Table (ldomVnetTable)

ldomVnetTable describes the virtual network devices for all domains.

**TABLE 15–16** Virtual Network Device Table (ldomVnetTable)

| Name                  | Data Type      | Access         | Description                                                                                                             |
|-----------------------|----------------|----------------|-------------------------------------------------------------------------------------------------------------------------|
| ldomVnetLdomIndex     | Integer        | Read-only      | Integer that is used as an index into ldomTable that represents the domain that contains the virtual network device     |
| ldomVnetVswIndex      | Integer        | Read-only      | Integer that is used to index into the virtual switch service table                                                     |
| ldomVnetIndex         | Integer        | Not accessible | Integer that is used to index the virtual network device in this table                                                  |
| ldomVnetDevName       | Display string | Read-only      | Virtual network device name. The property value is the net-dev property specified by the ldm add-vnet command.          |
| ldomVnetDevMacAddress | Display string | Read-only      | MAC address for this network device. The property value is the mac-addr property specified by the ldm add-vnet command. |
| ldomVnetMode          | Display string | Read-only      | Value is mode=hybrid to use NIU hybrid I/O on the virtual network device                                                |
| ldomVnetPortVlanID    | Display string | Read-only      | Port VLAN ID for the virtual network device                                                                             |
| ldomVnetVlanID        | Display string | Read-only      | VLAN ID for the virtual network device                                                                                  |
| ldomVnetLinkprop      | Display string | Read-only      | Value is linkprop=phys-state to report the link status based on the physical network device                             |
| ldomVnetMtu           | Integer        | Read-only      | MTU for a virtual network device                                                                                        |
| ldomVnetID            | Display string | Read-only      | Identifier of the virtual network device                                                                                |

## Virtual Console Tables

The Oracle VM Server for SPARC service domain provides a virtual network terminal service (vNTS). vNTS provides a virtual console service, called a virtual console concentrator (vcc), with a range of port numbers. Each virtual console concentrator has multiple console groups (vcons), and each group is assigned a port number. Each group can contain multiple domains.

## Virtual Console Concentrator Table (ldomVccTable)

ldomVccTable describes the virtual console concentrators for all domains.

TABLE 15–17 Virtual Console Concentrator Table (ldomVccTable)

| Name                 | Data Type      | Access         | Description                                                                                                                                                                                           |
|----------------------|----------------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomVccLdomIndex     | Integer        | Read-only      | Integer that is used as an index into ldomTable that represents the domain that contains the virtual console service                                                                                  |
| ldomVccIndex         | Integer        | Not accessible | Integer that is used to index the virtual console concentrator in this table                                                                                                                          |
| ldomVccName          | Display string | Read-only      | Virtual console concentrator name. The property value is the <i>vcc-name</i> specified by the <code>ldm add-vcc</code> command.                                                                       |
| ldomVccPortRangeLow  | Integer        | Read-only      | Low number for the range of TCP ports to be used by the virtual console concentrator. The property value is the <i>x</i> part of the port - range specified by the <code>ldm add-vcc</code> command.  |
| ldomVccPortRangeHigh | Integer        | Read-only      | High number for the range of TCP ports to be used by the virtual console concentrator. The property value is the <i>y</i> part of the port - range specified by the <code>ldm add-vcc</code> command. |

## Virtual Console Group Table (ldomVconsTable)

ldomVconsTable describes the virtual console groups for all virtual console services.

TABLE 15–18 Virtual Console Group Table (ldomVconsTable)

| Name                | Data Type      | Access         | Description                                                                                                                             |
|---------------------|----------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| ldomVconsIndex      | Integer        | Not accessible | Integer that is used to index a virtual group in this table                                                                             |
| ldomVconsGroupName  | Display string | Read-only      | Group name to which to attach the virtual console. The property value is the group specified by the <code>ldm set-vcons</code> command. |
| ldomVconsPortNumber | Integer        | Read-only      | Port number assigned to this group. The property value is the <i>port</i> specified by the <code>ldm set-vcons</code> command.          |

## Virtual Console Relationship Table (ldomVconsVccRelTable)

ldomVconsVccRelTable contains index values to show the inter-table relationships among a domain, a virtual console concentrator, and console groups.

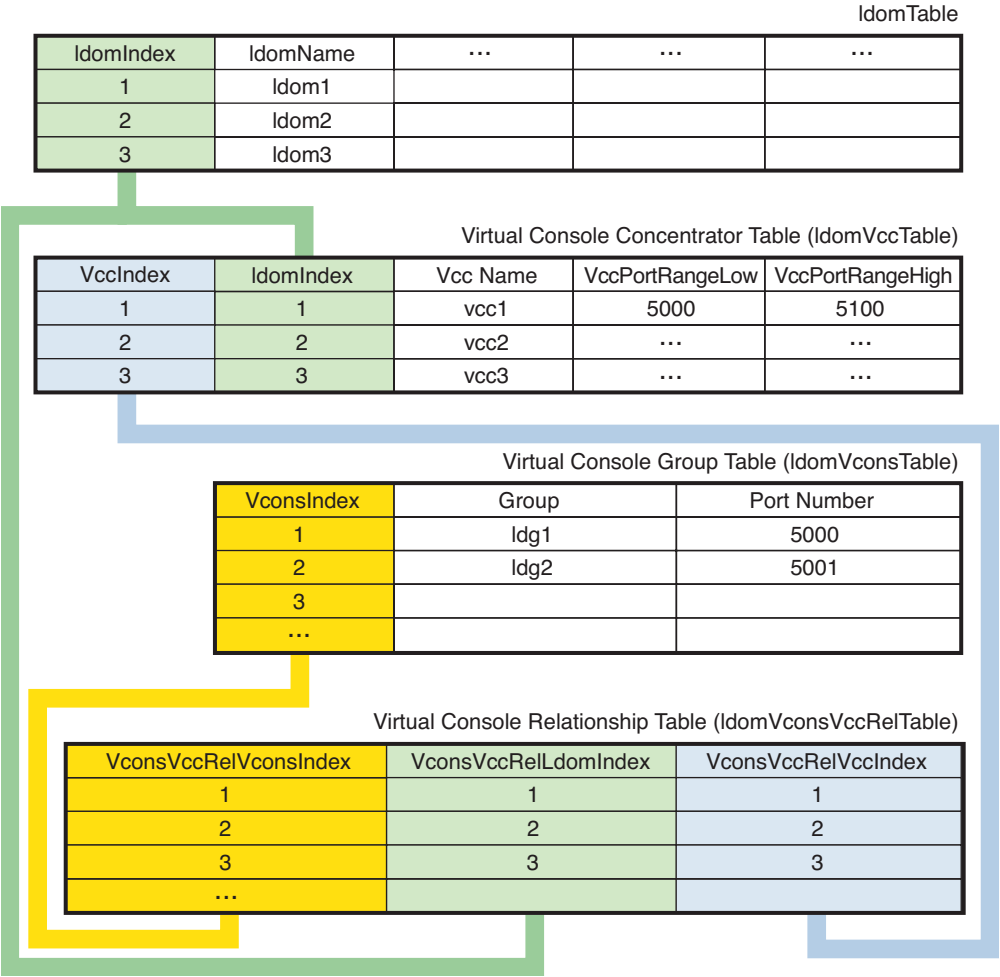
TABLE 15–19 Virtual Console Relationship Table (ldomVconsVccRelTable)

| Name                      | Data Type | Access    | Description                               |
|---------------------------|-----------|-----------|-------------------------------------------|
| ldomVconsVccRelVconsIndex | Integer   | Read-only | Value of ldomVconsIndex in ldomVconsTable |
| ldomVconsVccRelLdomIndex  | Integer   | Read-only | Value of ldomIndex in ldomTable           |
| ldomVconsVccRelVccIndex   | Integer   | Read-only | Value of ldomVccIndex in ldomVccTable     |

The following figure shows how indexes are used to define relationships among the virtual console tables and the domain table. The indexes are used as follows:

- ldomIndex in ldomVccTable and ldomVconsVccRelTable points to ldomTable.
- VccIndex in ldomVconsVccRelTable points to ldomVccTable.
- VconsIndex in ldomVconsVccRelTable points to ldomVconsTable.

FIGURE 15-4 Relationship Among Virtual Console Tables and the Domain Table



**Cryptographic Units Table (IdomCryptoTable)**

`IdomCryptoTable` describes the cryptographic units that all domains use. A cryptographic unit is sometimes referred to as a modular arithmetic unit (MAU).

TABLE 15-20 Cryptographic Units Table (IdomCryptoTable)

| Name                | Data Type | Access    | Description                                                                                                                  |
|---------------------|-----------|-----------|------------------------------------------------------------------------------------------------------------------------------|
| IdomCryptoLdomIndex | Integer   | Read-only | Integer that is used as an index into <code>IdomTable</code> that represents the domain that contains the cryptographic unit |

**TABLE 15–20** Cryptographic Units Table (ldomCryptoTable) *(Continued)*

| Name             | Data Type      | Access         | Description                                                                |
|------------------|----------------|----------------|----------------------------------------------------------------------------|
| ldomCryptoIndex  | Integer        | Not accessible | Integer that is used to index the cryptographic unit in this table         |
| ldomCryptoCpuSet | Display string | Read-only      | List of CPUs that is mapped to MAU-unit cpuset. For example, {0, 1, 2, 3}. |

## I/O Bus Table (ldomIOBusTable)

ldomIOBusTable describes the physical I/O devices and PCI buses that all domains use.

**TABLE 15–21** I/O Bus Table (ldomIOBusTable)

| Name               | Data Type      | Access         | Description                                                                                          |
|--------------------|----------------|----------------|------------------------------------------------------------------------------------------------------|
| ldomIOBusLdomIndex | Integer        | Read-only      | Integer that is used as an index into ldomTable that represents the domain that contains the I/O bus |
| ldomIOBusIndex     | Integer        | Not accessible | Integer that is used to index the I/O bus in this table                                              |
| ldomIOBusName      | Display string | Read-only      | Physical I/O device name                                                                             |
| ldomIOBusPath      | Display string | Read-only      | Physical I/O device path                                                                             |
| ldomIOBusOptions   | Display string | Read-only      | Physical I/O device options                                                                          |

## Core Table (ldomCoreTable)

ldomCoreTable describes the core information, such as core-id and cpuset, for all domains.

**TABLE 15–22** Core Table (ldomCoreTable)

| Name              | Data Type      | Access         | Description                                                                                       |
|-------------------|----------------|----------------|---------------------------------------------------------------------------------------------------|
| ldomCoreLdomIndex | Integer        | Read-only      | Integer that is used as an index into ldomTable that represents the domain that contains the core |
| ldomCoreIndex     | Integer        | Not accessible | Integer that is used to index a core in this table                                                |
| ldomCoreID        | Display string | Read-only      | Identifier of a core (core ID)                                                                    |
| ldomCoreCpuSet    | Display string | Read-only      | List of CPUs that is mapped to the core cpuset                                                    |

## Scalar Variables for Logical Domains Version Information

The Logical Domains Manager protocol supports Logical Domains versions, which consists of a major number and a minor number. The Oracle VM Server for SPARC MIB has scalar variables to describe the Logical Domains version information.

TABLE 15–23 Scalar Variables for Logical Domains Version Information

| Name             | Data Type | Access    | Description          |
|------------------|-----------|-----------|----------------------|
| ldomVersionMajor | Integer   | Read-only | Major version number |
| ldomVersionMinor | Integer   | Read-only | Minor version number |

The values for `ldomVersionMajor` and `ldomVersionMinor` are equivalent to the version shown by the `ldm list -p` command. For example:

```
$ ldm ls -p
VERSION 1.5
...

$ snmpget -v1 -c public localhost SUN-LDOM-MIB::ldomVersionMajor.0
SUN-LDOM-MIB::ldomVersionMajor.0 = INTEGER: 1

$ snmpget -v1 -c public localhost SUN-LDOM-MIB::ldomVersionMinor.0
SUN-LDOM-MIB::ldomVersionMinor.0 = INTEGER: 5
```

## Using SNMP Traps

This section describes how to set up your system to send and receive traps. It also describes the traps that you can use to receive change notification for logical domains (domains), as well as other traps that you can use.

## Using Oracle VM Server for SPARC MIB Module Traps

### Sending and Receiving Traps

#### ▼ Send Traps

##### ● Configure the trap.

Edit the `/etc/sma/snmp/snmpd.conf` file to add the directives to define the trap, inform version, and destination.

```
trapcommunity string --> define community string to be used when sending traps
trapsink host[community [port]] --> to send v1 traps
trap2sink host[community [port]] --> to send v2c traps
informsink host[community [port]] --> to send informs
```

For more information, see the [snmpd.conf\(4\)](#) man page.

**Example 15–4** Sending SNMP v1 and v2c Traps

This example sends both v1 and v2c traps to the SNMP trap daemon that runs on the same host. The `/etc/sma/snmp/snmpd.conf` file is updated with the following directives:

```
trapcommunity public
trapsink localhost
trap2sink localhost
```

▼ **Receive Traps**

- **Start the SNMP trap daemon utility.**

For information about the output format options, see the [snmpttrapd\(1M\)](#) man page.

The `snmpttrapd` utility is an SNMP application that receives and logs SNMP TRAP messages. For example, the following `snmpttrapd` command shows that a new domain was created (`ldomTrapDesc = Ldom Created`) with a name of `ldg2` (`ldomName = ldg2`).

```
/usr/sfw/sbin/snmpttrapd -P -F \
"TRAP from %B on %m/%L/%y at %h:%j:%k Enterprise=%N Type=%w SubType=%q \nwith Varbinds: %v \nSecurity info:%P\n\n" \
localhost:162
TRAP from localhost on 5/18/2007 at 16:30:10 Enterprise=. Type=0 SubType=0
with Varbinds: DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (47105)
0:07:51.05 SNMPv2-MIB::snmpTrapOID.0 = OID: SUN-LDOM-MIB::ldomCreate
SUN-LDOM-MIB::ldomIndexNotif = INTEGER: 3 SUN-LDOM-MIB::ldomName = STRING: ldg2
SUN-LDOM-MIB::ldomTrapDesc = STRING: Ldom Created
Security info:TRAP2, SNMP v2c, community public
```

# Oracle VM Server for SPARC MIB Trap Descriptions

This section describes the Oracle VM Server for SPARC MIB traps that you can use.

## Domain Creation (ldomCreate)

This trap notifies you when any domains are created.

TABLE 15–24 Domain Creation Trap (ldomCreate)

| Name           | Data Type      | Description             |
|----------------|----------------|-------------------------|
| ldomIndexNotif | Integer        | Index into ldomTable    |
| ldomName       | Display string | Name of the domain      |
| ldomTrapDesc   | Display string | Description of the trap |

## Domain Destroy (ldomDestroy)

This trap notifies you when any domains are destroyed.

TABLE 15–25 Domain Destroy Trap (ldomDestroy)

| Name           | Data Type      | Description             |
|----------------|----------------|-------------------------|
| ldomIndexNotif | Integer        | Index into ldomTable    |
| ldomName       | Display string | Name of the domain      |
| ldomTrapDesc   | Display string | Description of the trap |

## Domain State Change (ldomStateChange)

This trap notifies you of any domain operating state changes.

TABLE 15–26 Domain State Change Trap (ldomStateChange)

| Name           | Data Type      | Description                  |
|----------------|----------------|------------------------------|
| ldomIndexNotif | Integer        | Index into ldomTable         |
| ldomName       | Display string | Name of the domain           |
| ldomOperState  | Integer        | New state of the domain      |
| ldomStatePrev  | Integer        | Previous state of the domain |
| ldomTrapDesc   | Display string | Description of the trap      |

## Virtual CPU Change (ldomVCpuChange)

This trap notifies you when the number of virtual CPUs in a domain changes.

TABLE 15–27 Domain Virtual CPU Change Trap (ldomVCpuChange)

| Name            | Data Type      | Description                                      |
|-----------------|----------------|--------------------------------------------------|
| ldomIndexNotif  | Integer        | Index into ldomTable                             |
| ldomName        | Display string | Name of the domain that contains the virtual CPU |
| ldomNumVCPU     | Integer        | New number of virtual CPUs for the domain        |
| ldomNumVCPUPrev | Integer        | Previous number of virtual CPUs for the domain   |
| ldomTrapDesc    | Display string | Description of the trap                          |

## Virtual Memory Change (ldomVMemChange)

This trap notifies you when the amount of virtual memory in a domain changes.

TABLE 15–28 Domain Virtual Memory Change Trap (ldomVMemChange)

| Name            | Data Type      | Description                                                                                                                                                                                                                           |
|-----------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomIndexNotif  | Integer        | Index into ldomTable                                                                                                                                                                                                                  |
| ldomName        | Display string | Name of the domain that contains the virtual memory                                                                                                                                                                                   |
| ldomMemSize     | Integer        | Amount of virtual memory for the domain                                                                                                                                                                                               |
| ldomMemSizePrev | Integer        | Previous amount of virtual memory for the domain                                                                                                                                                                                      |
| ldomMemUnit     | Integer        | Memory unit for virtual memory, which is one of the following: <ul style="list-style-type: none"><li>■ 1 is KB</li><li>■ 2 is MB</li><li>■ 3 is GB</li><li>■ 4 is bytes</li></ul> If not specified, the unit value is bytes.          |
| ldomMemUnitPrev | Integer        | Memory unit for previous virtual memory, which is one of the following: <ul style="list-style-type: none"><li>■ 1 is KB</li><li>■ 2 is MB</li><li>■ 3 is GB</li><li>■ 4 is bytes</li></ul> If not specified, the unit value is bytes. |
| ldomTrapDesc    | Display string | Description of the trap                                                                                                                                                                                                               |

## Virtual Disk Service Change (ldomVdsChange)

This trap notifies you when a domain's virtual disk service changes.

TABLE 15–29 Domain Virtual Disk Service Change Trap (ldomVdsChange)

| Name           | Data Type      | Description                                               |
|----------------|----------------|-----------------------------------------------------------|
| ldomIndexNotif | Integer        | Index into ldomTable                                      |
| ldomName       | Display string | Name of the domain that contains the virtual disk service |

**TABLE 15–29** Domain Virtual Disk Service Change Trap (ldomVdsChange) *(Continued)*

| Name               | Data Type      | Description                                                                                                                                                                                       |
|--------------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomVdsServiceName | Display string | Name of the virtual disk service that has changed                                                                                                                                                 |
| ldomChangeFlag     | Integer        | Indicates one of the following changes that occurred to the virtual disk service: <ul style="list-style-type: none"> <li>■ 1 is Added</li> <li>■ 2 is Modified</li> <li>■ 3 is Removed</li> </ul> |
| ldomTrapDesc       | Display string | Description of the trap                                                                                                                                                                           |

## Virtual Disk Change (ldomVdiskChange)

This trap notifies you when a domain's virtual disk changes.

**TABLE 15–30** Virtual Disk Change Trap (ldomVdiskChange)

| Name           | Data Type      | Description                                                                                                                                                                                       |
|----------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomIndexNotif | Integer        | Index into ldomTable                                                                                                                                                                              |
| ldomName       | Display string | Name of the domain that contains the virtual disk device                                                                                                                                          |
| ldomVdiskName  | Display string | Name of the virtual disk device that has changed                                                                                                                                                  |
| ldomChangeFlag | Integer        | Indicates one of the following changes that occurred to the virtual disk service: <ul style="list-style-type: none"> <li>■ 1 is Added</li> <li>■ 2 is Modified</li> <li>■ 3 is Removed</li> </ul> |
| ldomTrapDesc   | Display string | Description of the trap                                                                                                                                                                           |

## Virtual Switch Change (ldomVswChange)

This trap notifies you when a domain's virtual switch changes.

**TABLE 15–31** Virtual Switch Change Trap (ldomVswChange)

| Name           | Data Type      | Description                                                 |
|----------------|----------------|-------------------------------------------------------------|
| ldomIndexNotif | Integer        | Index into ldomTable                                        |
| ldomName       | Display string | Name of the domain that contains the virtual switch service |

TABLE 15–31 Virtual Switch Change Trap (ldomVswChange) (Continued)

| Name               | Data Type      | Description                                                                                                                                                                                     |
|--------------------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomVswServiceName | Display string | Name of the virtual switch service that has changed                                                                                                                                             |
| ldomChangeFlag     | Integer        | Indicates one of the following changes that occurred to the virtual switch service: <ul style="list-style-type: none"><li>■ 1 is Added</li><li>■ 2 is Modified</li><li>■ 3 is Removed</li></ul> |
| ldomTrapDesc       | Display string | Description of the trap                                                                                                                                                                         |

Virtual Network Change (ldomVnetChange)

This trap notifies you when a domain's virtual network changes.

TABLE 15–32 Virtual Network Change Trap (ldomVnetChange)

| Name            | Data Type      | Description                                                                                                                                                                                   |
|-----------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomIndexNotif  | Integer        | Index into ldomTable                                                                                                                                                                          |
| ldomName        | Display string | Name of the domain that contains the virtual network device                                                                                                                                   |
| ldomVnetDevName | Display string | Name of the virtual network device for the domain                                                                                                                                             |
| ldomChangeFlag  | Integer        | Indicates one of the following changes that occurred to the virtual disk service: <ul style="list-style-type: none"><li>■ 1 is Added</li><li>■ 2 is Modified</li><li>■ 3 is Removed</li></ul> |
| ldomTrapDesc    | Display string | Description of the trap                                                                                                                                                                       |

Virtual Console Concentrator Change (ldomVccChange)

This trap notifies you when a domain's virtual console concentrator changes.

TABLE 15–33 Virtual Console Concentrator Change Trap (ldomVccChange)

| Name           | Data Type      | Description                                                       |
|----------------|----------------|-------------------------------------------------------------------|
| ldomIndexNotif | Integer        | Index into ldomTable                                              |
| ldomName       | Display string | Name of the domain that contains the virtual console concentrator |

**TABLE 15–33** Virtual Console Concentrator Change Trap (ldomVccChange) *(Continued)*

| Name           | Data Type      | Description                                                                                                                                                                                               |
|----------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomVccName    | Display string | Name of the virtual console concentrator service that has changed                                                                                                                                         |
| ldomChangeFlag | Integer        | Indicates one of the following changes that occurred to the virtual console concentrator: <ul style="list-style-type: none"> <li>■ 1 is Added</li> <li>■ 2 is Modified</li> <li>■ 3 is Removed</li> </ul> |
| ldomTrapDesc   | Display string | Description of the trap                                                                                                                                                                                   |

## Virtual Console Group Change (ldomVconsChange)

This trap notifies you when a domain's virtual console group changes.

**TABLE 15–34** Virtual Console Group Change Trap (ldomVconsChange)

| Name               | Data Type      | Description                                                                                                                                                                                        |
|--------------------|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomIndexNotif     | Integer        | Index into ldomTable                                                                                                                                                                               |
| ldomName           | Display string | Name of the domain that contains the virtual console group                                                                                                                                         |
| ldomVconsGroupName | Display string | Name of the virtual console group that has changed                                                                                                                                                 |
| ldomChangeFlag     | Integer        | Indicates one of the following changes that occurred to the virtual console group: <ul style="list-style-type: none"> <li>■ 1 is Added</li> <li>■ 2 is Modified</li> <li>■ 3 is Removed</li> </ul> |
| ldomTrapDesc       | Display string | Description of the trap                                                                                                                                                                            |

# Starting and Stopping Domains

This section describes the active management operations that you use to stop and start domains. You can control these active management operations by setting a value for the ldomAdminState property of the Domain Table, ldomTable. See [Table 15–1](#).

# Starting and Stopping a Domain

## ▼ Start a Domain

This procedure describes how to start an existing bound domain. If a domain with the specified domain name does not exist or is not already bound, this operation fails.

### 1 Verify that the *domain-name* domain exists and is bound.

```
ldm list domain-name
```

### 2 Identify *domain-name* in *ldomTable*.

```
snmpwalk -v1 -c public localhost SUN-LDOM-MIB::ldomTable
SUN-LDOM-MIB::ldomName.1 = STRING: primary
SUN-LDOM-MIB::ldomName.2 = STRING: LdomMibTest_1
SUN-LDOM-MIB::ldomAdminState.1 = INTEGER: 0
SUN-LDOM-MIB::ldomAdminState.2 = INTEGER: 0
SUN-LDOM-MIB::ldomOperState.1 = INTEGER: active(1)
SUN-LDOM-MIB::ldomOperState.2 = INTEGER: bound(6)
SUN-LDOM-MIB::ldomNumVCpu.1 = INTEGER: 32
SUN-LDOM-MIB::ldomNumVCpu.2 = INTEGER: 2
SUN-LDOM-MIB::ldomMemSize.1 = INTEGER: 3968
SUN-LDOM-MIB::ldomMemSize.2 = INTEGER: 256
SUN-LDOM-MIB::ldomMemUnit.1 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomMemUnit.2 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomNumCrypto.1 = INTEGER: 8
SUN-LDOM-MIB::ldomNumCrypto.2 = INTEGER: 0
SUN-LDOM-MIB::ldomNumIOBus.1 = INTEGER: 2
SUN-LDOM-MIB::ldomNumIOBus.2 = INTEGER: 0
SUN-LDOM-MIB::ldomUUID.1 = STRING: c2c3d93b-a3f9-60f6-a45e-f35d55c05fb6
SUN-LDOM-MIB::ldomUUID.2 = STRING: af0b05f0-d262-e633-af32-a6c4e81fb81c
SUN-LDOM-MIB::ldomMacAddress.1 = STRING: 00:14:4f:86:63:2a
SUN-LDOM-MIB::ldomMacAddress.2 = STRING: 00:14:4f:fa:78:b9
SUN-LDOM-MIB::ldomHostID.1 = STRING: 0x8486632a
SUN-LDOM-MIB::ldomHostID.2 = STRING: 0x84fa78b9
SUN-LDOM-MIB::ldomFailurePolicy.1 = STRING: ignore
SUN-LDOM-MIB::ldomFailurePolicy.2 = STRING: ignore
SUN-LDOM-MIB::ldomMaster.1 = STRING:
SUN-LDOM-MIB::ldomMaster.2 = STRING:
```

### 3 Start the *domain-name* domain.

Use the `snmpset` command to start the domain by setting a value of 1 to the `ldomAdminState` property. *n* specifies the domain to start.

```
snmpset -v version -c community-string hostname \
SUN-LDOM-MIB::ldomTable.1.ldomAdminState.n = 1
```

### 4 Verify that the *domain-name* domain is active.

#### ■ Use the `ldm list` command.

```
ldm list domain-name
```

- Use the `snmpget` command.

```
snmpget -v version -c community-string hostname SUN-LDOM-MIB::ldomOperState.n
```

### Example 15-5 Starting a Guest Domain

This example verifies that the `LdomMibTest_1` domain exists and is bound before setting the `ldomAdminState` property to 1. Finally, the `ldm list LdomMibTest_1` command verifies that the `LdomMibTest_1` domain is active.

```
ldm list LdomMibTest_1
snmpset -v1 -c private localhost SUN-LDOM-MIB::ldomTable.1.ldomAdminState.2 = 1
ldm list LdomMibTest_1
```

Instead of using the `ldm list` command to retrieve the `LdomMibTest_1` domain's state, you can use the `snmpget` command.

```
snmpget -v1 -c public localhost SUN-LDOM-MIB::ldomOperState.2
```

Note that if the domain is inactive when you use `snmpset` to start the domain, the domain is first bound and then started.

## ▼ Stop a Domain

This procedure describes how to stop a started domain. Any operating system instances that are hosted by the domain are stopped.

### 1 Identify *domain-name* in `ldomTable`.

```
snmpwalk -v1 -c public localhost SUN-LDOM-MIB::ldomTable
SUN-LDOM-MIB::ldomName.1 = STRING: primary
SUN-LDOM-MIB::ldomName.2 = STRING: LdomMibTest_1
SUN-LDOM-MIB::ldomAdminState.1 = INTEGER: 0
SUN-LDOM-MIB::ldomAdminState.2 = INTEGER: 0
SUN-LDOM-MIB::ldomOperState.1 = INTEGER: active(1)
SUN-LDOM-MIB::ldomOperState.2 = INTEGER: bound(6)
SUN-LDOM-MIB::ldomNumVCpu.1 = INTEGER: 32
SUN-LDOM-MIB::ldomNumVCpu.2 = INTEGER: 2
SUN-LDOM-MIB::ldomMemSize.1 = INTEGER: 3968
SUN-LDOM-MIB::ldomMemSize.2 = INTEGER: 256
SUN-LDOM-MIB::ldomMemUnit.1 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomMemUnit.2 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomNumCrypto.1 = INTEGER: 8
SUN-LDOM-MIB::ldomNumCrypto.2 = INTEGER: 0
SUN-LDOM-MIB::ldomNumIOBus.1 = INTEGER: 2
SUN-LDOM-MIB::ldomNumIOBus.2 = INTEGER: 0
SUN-LDOM-MIB::ldomUUID.1 = STRING: c2c3d93b-a3f9-60f6-a45e-f35d55c05fb6
SUN-LDOM-MIB::ldomUUID.2 = STRING: af0b05f0-d262-e633-af32-a6c4e81fb81c
SUN-LDOM-MIB::ldomMacAddress.1 = STRING: 00:14:4f:86:63:2a
SUN-LDOM-MIB::ldomMacAddress.2 = STRING: 00:14:4f:fa:78:b9
SUN-LDOM-MIB::ldomHostID.1 = STRING: 0x8486632a
SUN-LDOM-MIB::ldomHostID.2 = STRING: 0x84fa78b9
SUN-LDOM-MIB::ldomFailurePolicy.1 = STRING: ignore
```

```
SUN-LDOM-MIB::ldomFailurePolicy.2 = STRING: ignore
SUN-LDOM-MIB::ldomMaster.1 = STRING:
SUN-LDOM-MIB::ldomMaster.2 = STRING:
```

## 2 Stop the *domain-name* domain.

Use the `snmpset` command to stop the domain by setting a value of 2 to the `ldomAdminState` property. *n* specifies the domain to stop.

```
snmpset -v version -c community-string hostname \
SUN-LDOM-MIB::ldomTable.1.ldomAdminState.n = 2
```

## 3 Verify that the *domain-name* domain is bound.

- Use the `ldm list` command.

```
ldm list domain-name
```

- Use the `snmpget` command.

```
snmpget -v version -c community-string hostname SUN-LDOM-MIB::ldomOperState.n
```

### Example 15–6 Stopping a Guest Domain

This example sets the `ldomAdminState` property to 2 to stop the guest domain and then uses the `ldm list LdomMibTest_1` command to verify that the `LdomMibTest_1` domain is bound.

```
snmpset -v1 -c private localhost SUN-LDOM-MIB::ldomTable.1.ldomAdminState.2 = 2
ldm list LdomMibTest_1
```

## Logical Domains Manager Discovery

---

Logical Domains Managers can be discovered on a subnet by using multicast messages. The `ldmd` daemon is able to listen on a network for a specific multicast packet. If that multicast message is of a certain type, `ldmd` replies to the caller. This enables `ldmd` to be discovered on systems that are running Oracle VM Server for SPARC.

This chapter provides information about discovering the Logical Domains Manager running on systems on a subnet.

## Discovering Systems Running the Logical Domains Manager

### Multicast Communication

This discovery mechanism uses the same multicast network that is used by the `ldmd` daemon to detect collisions when automatically assigning MAC addresses. To configure the multicast socket, you must supply the following information:

```
#define MAC_MULTI_PORT 64535
#define MAC_MULTI_GROUP "239.129.9.27"
```

By default, *only* multicast packets can be sent on the subnet to which the machine is attached. You can change the behavior by setting the `ldmd/hops` SMF property for the `ldmd` daemon.

### Message Format

The discovery messages must be clearly marked so as not to be confused with other messages. The following multicast message format ensures that discovery messages can be distinguished by the discovery listening process:

```

#include <netdb.h> /* Used for MAXHOSTNAMELEN definition */
#define MAC_MULTI_MAGIC_NO 92792004
#define MAC_MULTI_VERSION 1

enum {
 SEND_MSG = 0,
 RESPONSE_MSG,
 LDMD_DISC_SEND,
 LDMD_DISC_RESP,
};

typedef struct {
 uint32_t version_no;
 uint32_t magic_no;
 uint32_t msg_type;
 uint32_t resv;
 union {
 mac_lookup_t Mac_lookup;
 ldmd_discovery_t Ldmd_discovery;
 } payload;
#define lookup payload.Mac_lookup
#define discovery payload.Ldmd_discovery
} multicast_msg_t;

#define LDMD_VERSION_LEN 32

typedef struct {
 uint64_t mac_addr;
 char source_ip[INET_ADDRSTRLEN];
} mac_lookup_t;

typedef struct {
 char ldmd_version[LDMD_VERSION_LEN];
 char hostname[MAXHOSTNAMELEN];
 struct in_addr ip_address;
 int port_no;
} ldmd_discovery_t;

```

## ▼ Discover Logical Domains Managers Running on Your Subnet

### 1 Open a multicast socket.

Ensure that you use the port and group information specified in [“Multicast Communication” on page 259](#).

### 2 Send a `multicast_msg_t` message over the socket.

The message should include the following:

- Valid value for `version_no`, which is 1 as defined by `MAC_MULTI_VERSION`
- Valid value for `magic_no`, which is 92792004 as defined by `MAC_MULTI_MAGIC_NO`
- `msg_type` of `LDMD_DISC_SEND`

### 3 Listen on the multicast socket for responses from Logical Domains Managers.

The responses must be a `multicast_msg_t` message with the following:

- Valid value for `version_no`
- Valid value for `magic_no`
- `msg_type` set to `LDMD_DISC_RESP`
- Payload consisting of a `ldmd_discovery_t` structure, which contains the following information:
  - `ldmd_version` – Version of the Logical Domains Manager running on the system
  - `hostname` – Host name of the system
  - `ip_address` – IP address of the system
  - `port_no` – Port number being used by the Logical Domains Manager for communications, which should be XMPP port 6482

When listening for a response from Logical Domains Managers, ensure that any auto-allocation MAC collision-detection packets are discarded.



# Using the XML Interface With the Logical Domains Manager

---

This chapter explains the Extensible Markup Language (XML) communication mechanism through which external user programs can interface with Oracle VM Server for SPARC software. These basic topics are covered:

- “XML Transport” on page 263
- “XML Protocol” on page 264
- “Event Messages” on page 268
- “Logical Domains Manager Actions” on page 273
- “Logical Domains Manager Resources and Properties” on page 274
- “XML Schemas” on page 286

## XML Transport

External programs can use the Extensible Messaging and Presence Protocol (XMPP – RFC 3920) to communicate with the Logical Domains Manager. XMPP is supported for both local and remote connections and is on by default. To shut off a remote connection, set the `ldmd/xmpp_enabled` SMF property to `false` and restart the Logical Domains Manager.

```
svccfg -s ldom/ldmd setprop ldmd/xmpp_enabled=false
svcadm refresh ldmd
svcadm restart ldmd
```

---

**Note** – Disabling the XMPP server also prevents domain migration and the dynamic reconfiguration of memory.

---

## XMPP Server

The Logical Domains Manager implements an XMPP server which can communicate with numerous available XMPP client applications and libraries. The Logical Domains Manager uses the following security mechanisms:

- Transport Layer Security (TLS) to secure the communication channel between the client and itself.
- Simple Authentication and Security Layer (SASL) for authentication. PLAIN is the only SASL mechanism supported. You must send in a user name and password to the server, so it can authorize you before allowing monitoring or management operations.

## Local Connections

The Logical Domains Manager detects whether user clients are running on the same domain as itself and, if so, does a minimal XMPP handshake with that client. Specifically, the SASL authentication step after the setup of a secure channel through TLS is skipped. Authentication and authorization are done based on the credentials of the process implementing the client interface.

Clients can choose to implement a full XMPP client or to simply run a streaming XML parser, such as the `libxml2` Simple API for XML (SAX) parser. Either way the client has to handle an XMPP handshake to the point of TLS negotiation. Refer to the XMPP specification for the sequence needed.

## XML Protocol

After communication initialization is complete, Logical Domains-defined XML messages are sent next. There are two general types of XML messages:

- Request and response messages use the `<LDM_interface>` tag. This type of XML message is used for communicating commands and getting results back from the Logical Domains Manager, analogous to executing commands using the command-line interface (CLI). This tag is also used for event registration and unregistration.
- Event messages use the `<LDM_event>` tag. This type of XML message is used to asynchronously report events posted by the Logical Domains Manager.

## Request and Response Messages

The XML interface into Logical Domains has two different formats:

- One format for sending commands into the Logical Domains Manager
- Another format for Logical Domains Manager to respond on the status of the incoming message and the actions requested within that message.

The two formats share many common XML structures, but they are separated in this discussion for a better understanding of the differences between them.

### Request Messages

An incoming XML request to the Logical Domains Manager at its most basic level includes a description of a single command, operating on a single object. More complicated requests can handle multiple commands and multiple objects per command. Following is the structure of a basic XML command.

**EXAMPLE 17-1** Format of a Single Command Operating on a Single Object

```
<LDM_interface version="1.0">
 <cmd>
 <action>Place command here</action>
 <option>Place options for certain commands here</option>
 <data version="3.0">
 <Envelope>
 <References/>
 <!-- Note a <Section> section can be here instead of <Content> -->
 <Content xsi:type="ovf:VirtualSystem_Type" id="Domain name">
 <Section xsi:type="ovf:ResourceAllocationSection_type">
 <Item>
 <rasd:OtherResourceType>LDM Resource Type</rasd:OtherResourceType>
 <gprop:GenericProperty
 key="Property name">Property Value</gprop:GenericProperty>
 </Item>
 </Section>
 <!-- Note: More Sections sections can be placed here -->
 </Content>
 </Envelope>
 </data>
 <!-- Note: More Data sections can be placed here -->
 </cmd>
 <!-- Note: More Commands sections can be placed here -->
</LDM_interface>
```

### The <LDM\_interface> Tag

All commands sent to the Logical Domains Manager must start with the <LDM\_interface> tag. Any document sent into the Logical Domains Manager must have only one <LDM\_interface> tag contained within it. The <LDM\_interface> tag must include a version attribute as shown in [Example 17-1](#).

## The <cmd> Tag

Within the <LDM\_interface> tag, the document must include at least one <cmd> tag. Each <cmd> section must have only one <action> tag. Use the <action> tag to describe the command to run. Each <cmd> tag must include at least one <data> tag to describe the objects on which the command is to operate.

The <cmd> tag can also have an <option> tag, which is used for options and flags that are associated with some commands. The following commands use options:

- The remove-domain command can use the -a option.
- The stop-domain command can use the -f option.
- The cancel-operation command can use the migration or reconf option.
- The add-spconfig command can use the -r *autosave-name* option.
- The remove-spconfig command can use the -r option.
- The list-spconfig command can use the -r [*autosave-name*] option.

## The <data> Tag

Each <data> section contains a description of an object pertinent to the command specified. The format of the data section is based on the XML schema portion of the Open Virtualization Format (OVF) draft specification. That schema defines an <Envelope> section which contains a <References> tag (unused by Logical Domains) and <Content> and <Section> sections.

For Logical Domains, the <Content> section is used to identify and describe a particular domain. The domain name in the id= attribute of the <Content> node identifies the domain. Within the <Content> section are one or more <Section> sections describing resources of the domain as needed by the particular command.

If you only need to identify a domain name, then you do not need to use any <Section> tags. Conversely, if no domain identifier is needed for the command, then you do need to provide a <Section> section, describing the resources needed for the command, outside of a <Content> section, but still within the <Envelope> section.

A <data> section does not need to contain an <Envelope> tag in cases where the object information can be inferred. This situation mainly applies to requests for monitoring all objects applicable to an action, and event registration and unregistration requests.

To allow use of the OVF specification's schema to properly define all types of objects, two additional OVF types have been defined:

- <gprop:GenericProperty> tag
- <Binding> tag

The <gprop:GenericProperty> tag was defined to handle any object's property for which the OVF specification does not have a definition. The property name is defined in the key= attribute

of the node and the value of the property is the contents of the node. The `<binding>` tag is used in the `list-bindings` subcommand output to define resources that are bound to other resources.

## Response Messages

An outgoing XML response closely matches the structure of the incoming request in terms of the commands and objects included, with the addition of a `<Response>` section for each object and command specified, as well as an overall `<Response>` section for the request. The `<Response>` sections provide status and message information as described in [Example 17-2](#). Following is the structure of a response to a basic XML request.

**EXAMPLE 17-2** Format of a Response to a Single Command Operating on a Single Object

```
<LDM_interface version="1.0">
 <cmd>
 <action>Place command here</action>
 <data version="3.0">
 <Envelope>
 <References/>
 <!-- Note a <Section> section can be here instead of <Content> -->
 <Content xsi:type="ovf:VirtualSystem_Type" id="Domain name">
 <Section xsi:type="ovf:ResourceAllocationSection_type">
 <Item>
 <rasd:OtherResourceType>
 LDom Resource Type
 </rasd:OtherResourceType>
 <gprop:GenericProperty>
 key="Property name"
 Property Value
 </gprop:GenericProperty>
 </Item>
 </Section>
 <!-- Note: More <Section> sections can be placed here -->
 </Content>
 </Envelope>
 <response>
 <status>success or failure</status>
 <resp_msg>Reason for failure</resp_msg>
 </response>
 </data>
 <!-- Note: More Data sections can be placed here -->
 <response>
 <status>success or failure</status>
 <resp_msg>Reason for failure</resp_msg>
 </response>
 </cmd>
 <!-- Note: More Command sections can be placed here -->
 <response>
 <status>success or failure</status>
 <resp_msg>Reason for failure</resp_msg>
 </response>
</LDM_interface>
```

## Overall Response

This `<response>` section, which is the direct child of the `<LDM_interface>` section, indicates overall success or failure of the entire request. Unless the incoming XML document is malformed, the `<response>` section includes only a `<status>` tag. If this response status indicates success, all commands on all objects have succeeded. If this response status is a failure and there is no `<resp_msg>` tag, then one of the commands included in the original request failed. The `<resp_msg>` tag is used only to describe some problem with the XML document itself.

## Command Response

The `<response>` section under the `<cmd>` section alerts the user to success or failure of that particular command. The `<status>` tag shows if that command succeeds or fails. As with the overall response, if the command fails, the `<response>` section includes only a `<resp_msg>` tag if the contents of the `<cmd>` section of the request is malformed. Otherwise, the failed status means one of the objects the command ran against caused a failure.

## Object Response

Finally, each `<data>` section in a `<cmd>` section also has a `<response>` section. This shows if the command being run on this particular object passes or fails. If the status of the response is SUCCESS, there is no `<resp_msg>` tag in the `<response>` section. If the status is FAILURE, there are one or more `<resp_msg>` tags in the `<response>` field, depending on the errors encountered when running the command against that object. Object errors can result from problems found when running the command, or a malformed or unknown object.

In addition to the `<response>` section, the `<data>` section can contain other information. This information is in the same format as an incoming `<data>` field, describing the object that caused a failure. See [“The `<data>` Tag” on page 266](#). This additional information is especially useful in the following cases:

- When a command fails against a particular `<data>` section but passes for any additional `<data>` sections
- When an empty `<data>` section is passed into a command and fails for some domains but passes for others

# Event Messages

In lieu of polling, you can subscribe to receive event notifications of certain state changes that occur. There are three types of events to which you can subscribe, individually or collectively. See [“Event Types” on page 270](#) for complete details.

## Registration and Unregistration

Use an `<LDM_interface>` message to register for events. See [“The `<LDM\_interface>` Tag” on page 265](#). The action tag details the type of event for which to register or unregister and the `<data>` section is left empty.

### EXAMPLE 17-3 Example Event Registration Request Message

```
<LDM_interface version="1.0">
 <cmd>
 <action>reg-domain-events</action>
 <data version="3.0"/>
 </cmd>
</LDM_interface>
```

The Logical Domains Manager responds with an `<LDM_interface>` response message stating whether the registration or unregistration was successful.

### EXAMPLE 17-4 Example Event Registration Response Message

```
<LDM_interface version="1.0">
 <cmd>
 <action>reg-domain-events</action>
 <data version="3.0">
 <response>
 <status>success</status>
 </response>
 </data>
 <response>
 <status>success</status>
 </response>
 </cmd>
 <response>
 <status>success</status>
 </response>
</LDM_interface>
```

The action string for each type of event is listed in the events subsection.

## The `<LDM_event>` Messages

Event messages have the same format as an incoming `<LDM_interface>` message with the exception that the start tag for the message is `<LDM_event>`. The action tag of the message is the action that was performed to trigger the event. The data section of the message describes the object associated with the event; the details depend on the type of event that occurred.

### EXAMPLE 17-5 Example `<LDM_event>` Notification

```
<LDM_event version='1.0'>
 <cmd>
 <action>Event command here</action>
```

EXAMPLE 17-5 Example <LDM\_event> Notification (Continued)

```
<data version='3.0'>
 <Envelope
 <References/>
 <Content xsi:type='ovf:VirtualSystem_Type' ovf:id='ldg1'/>
 <Section xsi:type="ovf:ResourceAllocationSection_type">
 <Item>
 <rasd:OtherResourceType>LDom Resource Type</rasd:OtherResourceType>
 <gprop:GenericProperty
 key="Property name">Property Value</gprop:GenericProperty>
 </Item>
 </Section>
 </Envelope>
 </data>
</cmd>
</LDM_event>
```

## Event Types

Following are the event types to which you can subscribe:

- Domain events
- Hardware events
- Progress events
- Resource events

All the events correspond to `ldm` subcommands.

### Domain Events

Domain events describe what actions can be performed directly to a domain. The following table shows the domain events which can be listed in the `<action>` tag in the `<LDM_event>` message.

Domain Events	Domain Events	Domain Events
add-domain	remove-domain	bind-domain
unbind-domain	start-domain	stop-domain
domain-reset	panic-domain	migrate-domain

These events always contain *only* a `<Content>` tag in the OVF data section that describes to which domain the event happened. To register for the domain events, send an `<LDM_interface>` message with the `<action>` tag set to **reg-domain-events**. Unregistering for these events requires an `<LDM_interface>` message with the action tag set to **unreg-domain-events**.

## Hardware Events

Hardware events pertain to changing the physical system hardware. In the case of Oracle VM Server for SPARC software, the only hardware changes that can be made are those to the service processor (SP) when a user adds, removes, or sets an SP configuration. Currently, the only three events of this type are:

- add-spconfig
- set-spconfig
- remove-spconfig

The hardware events always contain *only* a <Section> tag in the OVF data section which describes which SP configuration to which the event is happening. To register for these events, send an <LDM\_interface> message with the <action> tag set to **reg-hardware-events**. Unregistering for these events requires an <LDM\_interface> message with the <action> tag set to **unreg-hardware-events**.

## Progress Events

Progress events are issued for long-running commands, such as a domain migration. These events report the amount of progress that has been made during the life of the command. At this time, only the migration-process event is reported.

Progress events always contain *only* a <Section> tag in the OVF data section that describes the SP configuration affected by the event. To register for these events, send an <LDM\_interface> message with the <action> tag set to reg-hardware-events. Unregistering for these events requires an <LDM\_interface> message with the <action> tag set to unreg-hardware-events.

The <data> section of a progress event consists of a <content> section that describes the affected domain. This <content> section uses an ldom\_info <Section> tag to update progress. The following generic properties are shown in the ldom\_info section:

- --progress – Percentage of the progress made by the command
- --status – Command status, which can be one of ongoing, failed, or done
- --source – Machine that is reporting the progress

## Resource Events

Resource events occur when resources are added, removed, or changed in any domain. The data section for some of these events contains the <Content> tag with a <Section> tag giving a service name in the OVF data section. The following table shows events which can be listed in the <action> tag in the <LDM\_event> message.

Resource Events	Resource Events
add-vdiskserverdevice	remove-vdiskserverdevice

Resource Events	Resource Events
set-vdiskserverdevice	remove-vdiskserver
set-vconscon	remove-vconscon
set-vswitch	remove-vswitch
remove-vdpcs	

The remaining resource events always contain *only* the <Content> tag in the OVF data section that describes to which domain the event happened.

Resource Events	Resource Events	Resource Events
add-vcpu	add-crypto	add-memory
add-io	add-variable	add-vconscon
add-vdisk	add-vdiskserver	add-vnet
add-vswitch	add-vdpcs	add-vdpcc
set-vcpu	set-crypto	set-memory
set-variable	set-vnet	set-vconsole
set-vdisk	remove-vcpu	remove-crypto
remove-memory	remove-io	remove-variable
remove-vdisk	remove-vnet	remove-vdpcc

To register for the resource events, send an <LDM\_interface> message with the <action> tag set to **reg-resource-events**. Unregistering for these events requires an <LDM\_interface> message with the <action> tag set to **unreg-resource-events**.

All Events

You can also register to listen for all three type of events without having to register for each one individually. To register for all three types of events simultaneously, send an <LDM\_interface> message with the <action> tag set to **reg-all-events**. Unregistering for these events require an <LDM\_interface> message with the <action> tag set to **unreg-all-events**.

# Logical Domains Manager Actions

The commands specified in the `<action>` tag, with the exception of `*-*-events` commands, correspond to those of the `ldm` command-line interface. For details about `ldm` subcommands, see the [ldm\(1M\)](#) man page.

---

**Note** – The XML interface does *not* support the verb or command *aliases* supported by the Logical Domains Manager CLI.

---

The supported strings in the `<action>` tag are as follows:

Logical Domains Manager Actions	Logical Domains Manager Actions	Logical Domains Manager Actions
<code>list-bindings</code>	<code>list-services</code>	<code>list-constraints</code>
<code>list-devices</code>	<code>add-domain</code>	<code>remove-domain</code>
<code>list-domain</code>	<code>start-domain</code>	<code>stop-domain</code>
<code>bind-domain</code>	<code>unbind-domain</code>	<code>add-io</code>
<code>remove-io</code>	<code>add-mau</code>	<code>set-mau</code>
<code>remove-mau</code>	<code>add-memory</code>	<code>set-memory</code>
<code>remove-memory</code>	<code>remove-reconf</code>	<code>add-spconfig</code>
<code>set-spconfig</code>	<code>remove-spconfig</code>	<code>list-spconfig</code>
<code>add-variable</code>	<code>set-variable</code>	<code>remove-variable</code>
<code>list-variable</code>	<code>add-vconscon</code>	<code>set-vconscon</code>
<code>remove-vconscon</code>	<code>set-vconsole</code>	<code>add-vcpu</code>
<code>set-vcpu</code>	<code>remove-vcpu</code>	<code>add-vdisk</code>
<code>remove-vdisk</code>	<code>add-vdiskserver</code>	<code>remove-vdiskserver</code>
<code>add-udpcc</code>	<code>remove-udpcc</code>	<code>add-udpccs</code>
<code>remove-udpccs</code>	<code>add-vdiskserverdevice</code>	<code>remove-vdiskserverdevice</code>
<code>add-vnet</code>	<code>set-vnet</code>	<code>remove-vnet</code>
<code>add-vswitch</code>	<code>set-vswitch</code>	<code>remove-vswitch</code>
<code>reg-domain-events</code>	<code>unreg-domain-events</code>	<code>reg-resource-events</code>
<code>unreg-resource-events</code>	<code>reg-hardware-events</code>	<code>unreg-hardware-events</code>
<code>reg-all-events</code>	<code>unreg-all-events</code>	<code>migrate-domain</code>

Logical Domains Manager Actions	Logical Domains Manager Actions	Logical Domains Manager Actions
cancel-operation	set-domain	

# Logical Domains Manager Resources and Properties

Following are the Logical Domains Manager resources and the properties that can be defined for each of those resources. The resources and properties are shown in **bold** type in the XML examples. These examples show resources, not binding output. The constraint output can be used to create input for the Logical Domains Manager actions. The exception to this is domain migration output. See “[Domain Migration](#)” on page 285. Each resource is defined in a <Section> OVF section and is specified by a <rasd:OtherResourceType> tag.

## Domain Information (ldom\_info) Resource

EXAMPLE 17-6 Example ldom\_info XML Output

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="primary">
 <Section xsi:type="ovf:ResourceAllocationSection_type">
 <Item>
 <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
 <uuid>c2c3d93b-a3f9-60f6-a45e-f35d55c05fb6</uuid>
 <rasd:Address>00:03:ba:d8:ba:f6</rasd:Address>
 <gprop:GenericPropertykey="hostid">83d8baf6</gprop:GenericProperty>
 <gprop:GenericProperty key="master">plum</gprop:GenericProperty>
 <gprop:GenericProperty key="failure-policy">reset</gprop:GenericProperty>
 <gprop:GenericProperty key="progress">45%</gprop:GenericProperty>
 <gprop:GenericProperty key="status">ongoing</gprop:GenericProperty>
 <gprop:GenericProperty key="source">dt90-319</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
</Envelope>
```

The ldom\_info resource is always contained within a <Content> section. The following properties within the ldom\_info resource are optional properties:

- <uuid> tag, which specifies the UUID of the domain.
- <rasd:Address> tag, which specifies the MAC address to be assigned to a domain.
- <gprop:GenericPropertykey="failure-policy"> tag, which specifies how slave domains should behave should the master domain fail. The default value is ignore. Following are the valid property values:
  - ignore ignores failures of the master domain (slave domains are unaffected).
  - panic panics any slave domains when the master domain fails.

- `reset` resets any slave domains when the master domain fails.
- `stop` stops any slave domains when the master domain fails.
- `<gprop:GenericPropertykey="hostid">` tag, which specifies the host ID to be assigned to the domain.
- `<gprop:GenericPropertykey="master">` tag, which specifies up to four comma-separated master domain names.
- `<gprop:GenericPropertykey="progress">` tag, which specifies the percentage of progress made by the command.
- `<gprop:GenericPropertykey="source">` tag, which specifies the machine reporting on the progress of the command.
- `<gprop:GenericPropertykey="status">` tag, which specifies the status of the command (done, failed, or ongoing).

## CPU (cpu) Resource

The equivalent of the `add-vcpu`, `set-vcpu`, and `remove-vcpu` XML request actions is to set the value of the `<gpropGenericPropertykey="wcore">` tag as follows:

- If the `-c` option is used, set the `wcore` property to the number of whole cores specified.
- If the `-c` option is *not* used, set the `wcore` property to 0.

Note that the allocation units property, `<rasd:AllocationUnits>`, for the `cpu` resource always specifies the number of virtual CPUs and not the number of cores.

### EXAMPLE 17-7 Example `cpu` XML

The following example shows the XML request equivalent for the `ldm add-vcpu -c 1 ldg1` command:

```
<?xml version="1.0"?>
<LDM_interface version="1.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="./schemas/combined-v3.xsd"
xmlns:ovf="./schemas/envelope"
xmlns:rasd="./schemas/CIM_ResourceAllocationSettingData"
xmlns:vssd="./schemas/CIM_VirtualSystemSettingData"
xmlns:gprop="./schemas/GenericProperty"
xmlns:bind="./schemas/Binding">
 <cmd>
 <action>add-vcpu</action>
 <data version="3.0">
 <Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>cpu</rasd:OtherResourceType>
```

**EXAMPLE 17-7** Example cpu XML (Continued)

```
 <rasd:AllocationUnits>8</rasd:AllocationUnits>
 <gprop:GenericProperty key="wcore">1</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
</Envelope>
</data>
</cmd>
</LDM_interface>
```

A cpu resource is always contained within a <Content> section.

## MAU (mau) Resource

---

**Note** – The mau resource is any supported cryptographic unit on a supported server. Currently, the two cryptographic units supported are the Modular Arithmetic Unit (MAU) and the Control Word Queue (CWQ).

---

**EXAMPLE 17-8** Example mau XML

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>mau</rasd:OtherResourceType>
 <rasd:AllocationUnits>1</rasd:AllocationUnits>
 </Item>
 </Section>
 </Content>
</Envelope>
```

A mau resource is always contained within a <Content> section. The only property is the <rasd:AllocationUnits> tag, which signifies the number of MAUs or other cryptographic units.

## Memory (memory) Resource

**EXAMPLE 17-9** Example memory XML

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
```

**EXAMPLE 17-9** Example memory XML (Continued)

```

 <Item>
 <rasd:OtherResourceType>memory</rasd:OtherResourceType>
 <rasd:AllocationUnits>4G</rasd:AllocationUnits>
 </Item>
 </Section>
</Content>
</Envelope>

```

A memory resource is always contained within a <Content> section. The only property is the <rasd:AllocationUnits> tag, which signifies the amount of memory.

## Virtual Disk Server (vds) Resource

**EXAMPLE 17-10** Example vds XML

```

<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>vds</rasd:OtherResourceType>
 <gprop:GenericProperty
 key="service_name">vdstmp</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
</Envelope>

```

A virtual disk server (vds) resource can be in a <Content> section as part of a domain description, or it can appear on its own in an <Envelope> section. The only property is the <gprop:GenericProperty> tag with a key of service\_name and which contains the name of the vds resource being described.

## Virtual Disk Server Volume (vds\_volume) Resource

**EXAMPLE 17-11** Example vds\_volume XML

```

<Envelope>
 <References/>
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>vds_volume</rasd:OtherResourceType>
 <gprop:GenericProperty key="vol_name">vdsdev0</gprop:GenericProperty>
 <gprop:GenericProperty key="service_name">primary-vds0</gprop:GenericProperty>
 <gprop:GenericProperty key="block_dev">
 opt/SUNWldm/domain_disks/testdisk1</gprop:GenericProperty>
 </Item>
 </Section>

```

**EXAMPLE 17-11** Example vds\_volume XML (Continued)

```
 <gprop:GenericProperty key="vol_opts">ro</gprop:GenericProperty>
 <gprop:GenericProperty key="mpgroup">mpgroup-name</gprop:GenericProperty>
 </Item>
 </Section>
 </Envelope>
```

A vds\_volume resource can be in a <Content> section as part of a domain description, or it can appear on its own in an <Envelope> section. It must have <gprop:GenericProperty> tags with the following keys:

- vol\_name – Name of the volume
- service\_name – Name of the virtual disk server to which this volume is to be bound
- block\_dev – File or device name to be associated with this volume

Optionally, a vds\_volume resource can also have the following properties:

- vol\_opts – One or more of the following, comma-separated, within one string:  
{ro,slice,excl}
- mpgroup – Name of the multipath (failover) group

## Disk (disk) Resource

**EXAMPLE 17-12** Example disk XML

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>disk</rasd:OtherResourceType>
 <gprop:GenericProperty key="vdisk_name">vdisk0</gprop:GenericProperty>
 <gprop:GenericProperty
 key="service_name">primary-vds0</gprop:GenericProperty>
 <gprop:GenericProperty key="vol_name">vdsdev0</gprop:GenericProperty>
 <gprop:GenericProperty key="timeout">60</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
</Envelope>
```

A disk resource is always contained within a <Content> section. It must have <gprop:GenericProperty> tags with the following keys:

- vdisk\_name – Name of the virtual disk
- service\_name – Name of the virtual disk server to which this virtual disk is to be bound
- vol\_name – Virtual disk service device with which this virtual disk is to be associated

Optionally, the disk resource can also have the `timeout` property, which is the timeout value in seconds for establishing a connection between a virtual disk client (vdc) and a virtual disk server (vds). If there are multiple virtual disk (vdisk) paths, then the vdc can try to connect to a different vds, and the timeout ensures that a connection to any vds is established within the specified amount of time.

## Virtual Switch (vsw) Resource

EXAMPLE 17-13 Example vsw XML

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg2">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>vsw</rasd:OtherResourceType>
 <rasd:Address>00:14:4f:fb:ec:00</rasd:Address>
 <gprop:GenericProperty key="service_name">test-vsw1</gprop:GenericProperty>
 <gprop:GenericProperty key="inter_vnet_link">on</gprop:GenericProperty>
 <gprop:GenericProperty key="default-vlan-id">1</gprop:GenericProperty>
 <gprop:GenericProperty key="pvid">1</gprop:GenericProperty>
 <gprop:GenericProperty key="mtu">1500</gprop:GenericProperty>
 <gprop:GenericProperty key="dev_path">switch@0</gprop:GenericProperty>
 <gprop:GenericProperty key="id">0</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
</Envelope>
```

A vsw resource can be either in a `<Content>` section as part of a domain description, or it can appear on its own in an `<Envelope>` section. It *must* have a `<gprop:GenericProperty>` tag with the `service_name` key, which is the name to be assigned to the virtual switch.

Optionally, the vsw resource can also have the following properties:

- `<rasd:Address>` – Assigns a MAC address to the virtual switch
- `default-vlan-id` – Specifies the default virtual local area network (VLAN) to which a virtual network device or virtual switch needs to be a member, in tagged mode. The first VLAN ID (*vid1*) is reserved for the `default-vlan-id`.
- `dev_path` – Path of the network device to be associated with this virtual switch
- `id` – Specifies the ID of a new virtual switch device. By default, ID values are generated automatically, so set this property if you need to match an existing device name in the OS.
- `inter_vnet_link` – Specifies whether to assign LDC channels for inter-vnet communication. The default value is on.

- **linkprop** – Specifies whether the virtual device should get physical link state updates. When the value is **phys - state**, the virtual device gets physical link state updates. When the value is blank, the virtual device does not get physical link state updates. By default, the virtual device does not get physical link state updates.
- **mode** – **sc** for Oracle Solaris Cluster heartbeat support.
- **pvid** – Port virtual local area network (VLAN) identifier (ID) indicates the VLAN of which the virtual network needs to be a member, in untagged mode.
- **mtu** – Specifies the maximum transmission unit (MTU) of a virtual switch, virtual network devices that are bound to the virtual switch, or both. Valid values are in the range of 1500-16000. The **ldm** command issues an error if an invalid value is specified.
- **vid** – Virtual local area network (VLAN) identifier (ID) indicates the VLAN of which a virtual network and virtual switch need to be a member, in tagged mode.

## Network (network) Resource

EXAMPLE 17-14 Example network XML

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>network</rasd:OtherResourceType>
 <gprop:GenericProperty key="linkprop">phys - state</gprop:GenericProperty>
 <gprop:GenericProperty key="vnet_name">ldg1-vnet0</gprop:GenericProperty>
 <gprop:GenericProperty
 key="service_name">primary-vsw0</gprop:GenericProperty>
 <rasd:Address>00:14:4f:fc:00:01</rasd:Address>
 </Item>
 </Section>
 </Content>
</Envelope>
```

A network resource is always contained within a **<Content>** section. It must have **<gprop:GenericProperty>** tags with the following keys:

- **linkprop** – Specifies whether the virtual device should get physical link state updates. When the value is **phys - state**, the virtual device gets physical link state updates. When the value is blank, the virtual device does not get physical link state updates. By default, the virtual device does not get physical link state updates.
- **vnet\_name** – Name of the virtual network (vnet)
- **service\_name** – Name of the virtual switch (vswitch) to which this virtual network is to be bound

Optionally, the network resource can also have the following properties:

- `<rasd:Address>` – Assigns a MAC address to the virtual switch
- `pvid` – Port virtual local area network (VLAN) identifier (ID) indicates the VLAN of which the virtual network needs to be a member, in untagged mode.
- `vid` – Virtual local area network (VLAN) identifier (ID) indicates the VLAN of which a virtual network and virtual switch need to be a member, in tagged mode.
- `mode` – `hybrid` to enable hybrid I/O for that virtual network.

## Virtual Console Concentrator (vcc) Resource

EXAMPLE 17-15 Example vcc XML

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>vcc</rasd:OtherResourceType>
 <gprop:GenericProperty key="service_name">vcc1</gprop:GenericProperty>
 <gprop:GenericProperty key="min_port">6000</gprop:GenericProperty>
 <gprop:GenericProperty key="max_port">6100</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
</Envelope>
```

A vcc resource can be either in a `<Content>` section as part of a domain description, or it can appear on its own in an `<Envelope>` section. It can have `<gprop:GenericProperty>` tags with the following keys:

- `service_name` – Name to be assigned to the virtual console concentrator service
- `min_port` – Minimum port number to be associated with this vcc
- `max_port` – Maximum port number to be associated with this vcc

## Variable (var) Resource

EXAMPLE 17-16 Example var XML

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>var</rasd:OtherResourceType>
 <gprop:GenericProperty key="name">test_var</gprop:GenericProperty>
 <gprop:GenericProperty key="value">test1</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
</Envelope>
```

**EXAMPLE 17-16** Example var XML (Continued)

```
</Item>
</Section>
</Content>
</Envelope>
```

A var resource is always contained within a <Content> section. It can have <gprop:GenericProperty> tags with the following keys:

- name – Name of the variable
- value – Value of the variable

## Physical I/O Device (physio\_device) Resource

**EXAMPLE 17-17** Example physio\_device XML

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
 <gprop:GenericProperty key="name">pci@780</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
</Envelope>
```

A physio\_device resource is always contained within a <Content> section. The only property is the <gprop:GenericProperty> tag with the name key property value, which is the name of the I/O device being described.

## SP Configuration (spconfig) Resource

**EXAMPLE 17-18** Example spconfig XML

```
<Envelope>
 <Section xsi:type="ovf:ResourceAllocationSection_type">
 <Item>
 <rasd:OtherResourceType>spconfig</rasd:OtherResourceType>
 <gprop:GenericProperty
 key="spconfig_name">primary</gprop:GenericProperty>
 <gprop:GenericProperty
 key="spconfig_status">current</gprop:GenericProperty>
 </Item>
 </Section>
</Envelope>
```

A service processor (SP) configuration (spconfig) resource always appears on its own in an <Envelope> section. It can have <gprop:GenericProperty> tags with the following keys

- spconfig\_name – Name of a configuration to be stored on the SP
- spconfig\_status – The current status of a particular SP configuration. This property is used in the output of an ldm list-spconfig command.

## DRM Policy Configuration (policy) Resource

EXAMPLE 17-19 Example policy XML

```
<Envelope>
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>policy</rasd:OtherResourceType>
 <gprop:GenericProperty key="policy_name">test-policy</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_enable">on</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_priority">1</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_vcpu_min">12</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_vcpu_max">13</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_util_lower">8</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_util_upper">9</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_tod_begin">07:08:09</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_tod_end">09:08:07</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_sample_rate">1</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_elastic_margin">8</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_attack">8</gprop:GenericProperty>
 <gprop:GenericProperty key="policy_decay">9</gprop:GenericProperty>
 </Item>
 </Section>
</Envelope>
```

A DRM policy (policy) resource appears in an <Envelope> section and can have <gprop:GenericProperty> tags with the following keys:

- policy\_name – Name of the DRM policy
- policy\_enable – Specifies whether the DRM policy is enabled or disabled
- policy\_priority – Priority of the DRM policy
- policy\_vcpu\_min – Minimum number of virtual CPU resources for a domain
- policy\_vcpu\_max – Maximum number of virtual CPU resources for a domain
- policy\_util\_lower – Lower utilization level at which policy analysis is triggered
- policy\_util\_upper – Upper utilization level at which policy analysis is triggered
- policy\_tod\_begin – Effective start time of the DRM policy
- policy\_tod\_end – Effective stop time of the of the DRM policy
- policy\_sample\_rate – The sample rate, which is the cycle time in seconds

- `policy_elastic_margin` – Amount of buffer between the upper and lower CPU utilization bounds
- `policy_attack` – Maximum amount of a resource to be added during any one resource control cycle
- `policy_decay` – Maximum amount of a resource to be removed during any one resource control cycle

## Virtual Data Plane Channel Service (vdpcs) Resource

EXAMPLE 17-20 Example vdpcs XML

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>vdpcs</rasd:OtherResourceType>
 <gprop:GenericProperty key="service_name">ldg1-vdpcs</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
</Envelope>
```

This resource is only of interest in a Netra DPS environment. A `vdpcs` resource can be either in a `<Content>` section as part of a domain description, or it can appear on its own in an `<Envelope>` section. The only property is the `<gprop:GenericProperty>` tag with the `service_name` key property value, which is the name of the virtual data plane channel service (`vdpcs`) resource being described.

## Virtual Data Plane Channel Client (vdpccl) Resource

EXAMPLE 17-21 Example vdpccl XML

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>vdpccl</rasd:OtherResourceType>
 <gprop:GenericProperty key="vdpccl_name">vdpccl</gprop:GenericProperty>
 <gprop:GenericProperty
 key="service_name">ldg1-vdpcs</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
</Envelope>
```

This resource is only of interest in a Netra DPS environment. A virtual data plane channel client resource is always contained within a <Content> section. It can have <gprop:GenericProperty> tags with the following keys:

- `vdpc_name` – Name of the virtual data plane channel client (vdpc)
- `service_name` – Name of the virtual data plane channel service vdpcs to which this vdpc is to be bound

## Console (console) Resource

EXAMPLE 17-22 Example console XML

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>console</rasd:OtherResourceType>
 <gprop:GenericProperty key="port">6000</gprop:GenericProperty>
 <gprop:GenericProperty key="service_name">vcc2</gprop:GenericProperty>
 <gprop:GenericProperty key="group">group-name</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
</Envelope>
```

A console resource is always contained within a <Content> section. It can have <gprop:GenericProperty> tags with the following keys:

- `port` – Port to which to change this virtual console (console)
- `service_name` – Virtual console concentrator (vcc) service to which to bind this console
- `group` – Name of the group to which to bind this console

## Domain Migration

This example shows what is contained in the <data> section for a migrate-domain subcommand.

EXAMPLE 17-23 Example migrate-domain <data> Section

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="ldg1"/>
 <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="ldg1"/>
 <Section xsi:type="ovf:ResourceAllocationSection_Type">
 <Item>
 <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
```

**EXAMPLE 17-23** Example migrate-domain <data> Section (Continued)

```
<gprop:GenericProperty key="target">target-host</gprop:GenericProperty>
<gprop:GenericProperty key="username">user-name</gprop:GenericProperty>
<gprop:GenericProperty key="password">password</gprop:GenericProperty>
</Item>
</Section>
</Content>
</Envelope>
```

Where:

- First <Content> node (without an <ldom\_info> section) is the source domain to migrate.
- Second <Content> node (with an <ldom\_info> section) is the target domain to which to migrate. The source and target domain names can be the same.
- The <ldom\_info> section for the target domain describes the machine to which to migrate and the details needed to migrate to that machine:
  - target-host is the target machine to which to migrate.
  - user-name is the login user name for the target machine. Must be SASL 64-bit encoded.
  - password is the password to use for logging into the target machine. Must be SASL 64-bit encoded.

---

**Note** – The Logical Domains Manager uses `sasl_decode64()` to decode the target user name and password and uses `sasl_encode64()` to encode these values. SASL 64 encoding is equivalent to base64 encoding.

---

## XML Schemas

The following lists each XML schema file name in the `/opt/SUNWldm/bin/schemas` directory. These schemas are used by the Logical Domains Manager.

- `cim-common.xsd` – `cim-common.xsd` schema
- `cim-rasd.xsd` – `cim-rasd.xsd` schema
- `cim-vssd.xsd` – `cim-vssd.xsd` schema
- `cli-list-constraint-v3.xsd` – `cli-list-constraint-v3.xsd` schema
- `combined-v3.xsd` – LDM\_interface XML schema
- `event-v3.xsd` – LDM\_Event XML schema
- `ldmd-binding.xsd` – `Binding_Type` XML schema
- `ldmd-property.xsd` – `GenericProperty` XML schema
- `ovf-core.xsd` – `ovf-core.xsd` schema
- `ovf-envelope.xsd` – `ovf-envelope.xsd` schema
- `ovf-section.xsd` – `ovf-section.xsd` schema
- `ovf-strings.xsd` – `ovf-strings.xsd` schema

- `ovfenv-core.xsd` – `ovfenv-core.xsd` schema
- `ovfenv-section.xsd` – `ovfenv-section.xsd` schema



# Glossary

---

This list defines terminology, abbreviations, and acronyms in the Oracle VM Server for SPARC documentation.

## A

<b>API</b>	Application programming interface
<b>ASN</b>	Abstract Syntax Notation
<b>auditreduce</b>	Merge and select audit records from audit trail files (see the <a href="#">auditreduce(1M)</a> man page).
<b>auditing</b>	Using Oracle Solaris OS auditing to identify the source of security changes
<b>authorization</b>	Setting up authorization using the Oracle Solaris OS RBAC

## B

<b>bge</b>	Broadcom Gigabit Ethernet driver on Broadcom BCM57xx devices
<b>BSM</b>	Basic Security module
<b>bsmconv</b>	Enable the BSM (see the <a href="#">bsmconv(1M)</a> man page).
<b>bsmunconv</b>	Disable the BSM (see the <a href="#">bsmunconv(1M)</a> man page).

## C

<b>CD</b>	Compact disc
<b>CLI</b>	Command-line interface
<b>CMT</b>	Chip multithreading

<b>compliance</b>	Determining if a system's configuration is in compliance with a predefined security profile
<b>configuration</b>	Name of logical domain configuration that is saved on the service processor
<b>constraints</b>	To the Logical Domains Manager, constraints are one or more resources you want to have assigned to a particular domain. You either receive all the resources you ask to be added to a domain or you get none of them, depending upon the available resources.
<b>control domain</b>	A privileged domain that creates and manages other logical domains and services by using the Logical Domains Manager
<b>CPU</b>	Central processing unit
<b>CWQ</b>	Control Word Queue; cryptographic unit for Oracle's Sun UltraSPARC T2 based platforms

## D

<b>DHCP</b>	Dynamic Host Configuration Protocol
<b>DMA</b>	Direct Memory Access is the ability to directly transfer data between the memory and a device (for example, a network card) without involving the CPU.
<b>DMP</b>	Dynamic Multipathing (Veritas)
<b>domain</b>	See <a href="#">logical domain</a> .
<b>Logical Domains Manager</b>	A CLI to create and manage logical domains and allocate resources to domains
<b>DPS</b>	Data plane software
<b>DR</b>	Dynamic reconfiguration
<b>drd</b>	Oracle Solaris 10 OS dynamic reconfiguration daemon for Logical Domains Manager (see the <a href="#">drd(1M)</a> man page).
<b>DS</b>	Domain Services module (Oracle Solaris 10 OS)
<b>DVD</b>	Digital versatile disc

## E

<b>EFI</b>	Extensible firmware interface
<b>ETM</b>	Encoding Table Management module (Oracle Solaris 10 OS)

---

**F**

<b>FC_AL</b>	Fiber Channel Arbitrated Loop
<b>FMA</b>	Fault Management Architecture
<b>fmd</b>	Oracle Solaris 10 OS fault manager daemon (see the <a href="#">fmd(1M)</a> man page).
<b>format</b>	Disk partitioning and maintenance utility (see the <a href="#">format(1M)</a> man page).
<b>fmthard</b>	Populate label on hard disks (see the <a href="#">fmthard(1M)</a> man page).
<b>FTP</b>	File Transfer Protocol

**G**

<b>Gb</b>	Gigabit
<b>guest domain</b>	Uses services from the I/O and service domains and is managed by the control domain.
<b>GLDv3</b>	Generic LAN Driver version 3.

**H**

<b>hardening</b>	Modifying Oracle Solaris OS configuration to improve security
<b>HDD</b>	Hard disk drive
<b>hypervisor</b>	Firmware layer interposed between the operating system and the hardware layer

**I**

<b>I/O domain</b>	Domain that has direct ownership of and direct access to physical I/O devices and that shares those devices to other logical domains in the form of virtual devices
<b>IB</b>	Infiniband
<b>IDE</b>	Integrated Drive Electronics
<b>IDR</b>	Interim Diagnostics Release
<b>ILOM</b>	Integrated Lights Out Manager
<b>I/O</b>	Input/output devices, such as internal disks and PCIe controllers and their attached adapters and devices

<b>ioctl</b>	Input/output control call
<b>IP</b>	Internet Protocol
<b>IPMP</b>	Internet Protocol Network Multipathing
<b>ISO</b>	International Organization for Standardization

## K

<b>kaio</b>	Kernel asynchronous input/output
<b>KB</b>	Kilobyte
<b>KU</b>	Kernel update

## L

<b>LAN</b>	Local-area network
<b>LDAP</b>	Lightweight Directory Access Protocol
<b>LDC</b>	Logical domain channel
<b>ldm</b>	Logical Domains Manager utility (see the <a href="#">ldm(1M)</a> man page).
<b>ldmd</b>	Logical Domains Manager daemon
<b>lofi</b>	Loopback file
<b>logical domain</b>	A virtual machine comprised of a discrete logical grouping of resources, which has its own operating system and identity within a single computer system. Also called a <i>domain</i> .
<b>LUN</b>	Logical unit number

## M

<b>MAC</b>	Media access control address, which Logical Domains can automatically assign or you can assign manually
<b>MAU</b>	Modular Arithmetic Unit
<b>MB</b>	Megabyte

<b>MD</b>	Machine description in the server database
<b>mem, memory</b>	Memory unit – default size in bytes, or specify gigabytes (G), kilobytes (K), or megabytes (M). Virtualized memory of the server that can be allocated to guest domains.
<b>metadb</b>	Create and delete replicas of the Solaris Volume Manager metadvice state database (see the <a href="#">metadb(1M)</a> man page).
<b>metaset</b>	Configure disk sets (see the <a href="#">metaset(1M)</a> man page).
<b>mhd</b>	Multihost disk control operations (see the <a href="#">mhd(7i)</a> man page).
<b>MIB</b>	Management Information Base
<b>minimizing</b>	Installing the minimum number of core Oracle Solaris OS package necessary
<b>MMF</b>	Multimode fiber
<b>MMU</b>	Memory management unit
<b>mpgroup</b>	Multipathing group name for virtual disk failover
<b>mtu</b>	Maximum transmission unit

## N

<b>NAT</b>	Network Address Translation
<b>ndpsldcc</b>	Netra DPS Logical Domain Channel Client. <i>See also</i> vdpcc.
<b>ndpsldcs</b>	Netra DPS Logical Domain Channel Service. <i>See also</i> vdpccs.
<b>NFS</b>	Network file system
<b>NIS</b>	Network Information Services
<b>NIU</b>	Network Interface Unit (Oracle's Sun SPARC Enterprise T5120 and T5220 servers)
<b>NTS</b>	Network terminal server
<b>NVRAM</b>	Non-volatile random-access memory
<b>nxge</b>	Driver for an NIU 10Gb Ethernet adapter

## O

<b>OID</b>	Object identifier, which is a sequence of numbers that uniquely identifies each object in a MIB
------------	-------------------------------------------------------------------------------------------------

<b>OS</b>	Operating system
<b>OVF</b>	Open Virtualization Format

## **P**

<b>P2V</b>	Logical Domains Physical-to-Virtual Conversion Tool
<b>PA</b>	Physical address
<b>PCI</b>	Peripheral component interconnect bus
<b>PCIe</b>	PCI EXPRESS bus
<b>PCI-X</b>	PCI Extended bus
<b>pcpu</b>	Physical CPU
<b>physio</b>	Physical input/output
<b>PICL</b>	Platform Information and Control Library
<b>picld</b>	PICL daemon (see the <a href="#">picld(1M)</a> man page).
<b>PM</b>	Power management of virtual CPUs and memory
<b>praudit</b>	Print contents of an audit trail file (see the <a href="#">praudit(1M)</a> man page).
<b>PRI</b>	Priority
<b>PROM</b>	Programmable read-only memory

## **R**

<b>RA</b>	Real address
<b>RAID</b>	Redundant Array of Inexpensive Disks
<b>RBAC</b>	Role-Based Access Control
<b>RPC</b>	Remote Procedure Call

## **S**

<b>SASL</b>	Simple Authentication and Security Layer
-------------	------------------------------------------

---

<b>SAX</b>	Simple API for XML parser, which traverses an XML document. The SAX parser is event-based and used mostly for streaming data.
<b>system controller (SC)</b>	Also see service processor
<b>SCSI</b>	Small Computer System Interface
<b>service domain</b>	Logical domain that provides devices, such as virtual switches, virtual console connectors, and virtual disk servers, to other logical domains
<b>SMA</b>	System Management Agent
<b>SMF</b>	Service Management Facility
<b>SMI</b>	Structure of Management Information
<b>SNMP</b>	Simple Network Management Protocol
<b>service processor (SP)</b>	The SP, also known as the system controller (SC), monitors and runs the physical machine.
<b>SSH</b>	Secure Shell
<b>ssh</b>	Secure Shell command (see the <a href="#">ssh(1)</a> man page).
<b>sshd</b>	Secure Shell daemon (see the <a href="#">sshd(1M)</a> man page).
<b>SunVTS</b>	Sun Validation Test Suite
<b>svcadm</b>	Manipulates service instances (see the <a href="#">svcadm(1M)</a> man page).

## T

<b>TCP</b>	Transmission Control Protocol
<b>TLS</b>	Transport Layer Security

## U

<b>UDP</b>	User Datagram Protocol
<b>UFS</b>	UNIX File System
<b>unicast</b>	Network communication that takes place between a single sender and a single receiver
<b>USB</b>	Universal Serial Bus

**uscsi** User SCSI command interface (see the [uscsi\(7I\)](#) man page).

**UTP** Unshielded twisted pair

## V

**var** Variable

**VBSC** Virtual blade system controller

**vcc, vconscn** Virtual console concentrator service with a specific port range to assign to guest domains

**vcons, vconsole** Virtual console for accessing system-level messages. A connection is achieved by connecting to the vconscn service in the control domain at a specific port.

**vcpu** Virtual central processing unit. Each core in a server is represented as a virtual CPU. For example, an 8-core Sun Fire T2000 Server from Oracle has 32 virtual CPUs that can be allocated among the logical domains.

**vdcc** Virtual disk client

**vdisk** A virtual disk is a generic block device associated with different types of physical devices, volumes, or files.

**vdppc** Virtual data plane channel client in a Netra DPS environment

**vdpcs** Virtual data plane channel service in a Netra DPS environment

**vds, vdiskserver** Virtual disk server enables you to import virtual disks into a logical domain

**vdsdev, vdiskserverdevice** Virtual disk server device is exported by the virtual disk server. The device can be an entire disk, a slice on a disk, a file, or a disk volume.

**VLAN** Virtual local area network

**vldc** Virtual logical domain channel service

**vldcc** Virtual logical domain channel client

**vnet** Virtual network device implements a virtual Ethernet device and communicates with other vnet devices in the system by using the virtual network switch (vswitch)

**vNTS** Virtual network terminal service

**vntsd** Oracle Solaris 10 OS virtual network terminal server daemon for Logical Domains consoles (see the [vntsd\(1M\)](#) man page).

**volfs** Volume Management file system (see the [volfs\(7FS\)](#) man page).

**vsw, vswitch** Virtual network switch that connects the virtual network devices to the external network and also switches packets between them

<b>VTOC</b>	Volume table of contents
<b>VxDMP</b>	Veritas Dynamic Multipathing
<b>VxVM</b>	Veritas Volume Manager

## **W**

<b>WAN</b>	Wide-area network
------------	-------------------

## **X**

<b>XFP</b>	eXtreme Fast Path
<b>XML</b>	Extensible Markup Language
<b>XMPP</b>	Extensible Messaging and Presence Protocol

## **Z**

<b>ZFS</b>	Zettabyte File System (Oracle Solaris 10 OS)
<b>zpool</b>	ZFS storage pool (see the <a href="#">zpool(1M)</a> man page).
<b>ZVOL</b>	ZFS Volume Emulation Driver



# Index

---

## A

- assigning
  - endpoint device to an I/O domain, 70–80
  - PCIe bus to an I/O domain, 66–70
- authorization
  - ldm subcommands, 41
  - levels, 41
  - read, 41
  - read and write, 41

## C

- cancel-operation reconf subcommand, 154
- CLI, *See* command-line interface
- command-line interface, 23
- commands
  - ldm(1M), 23
  - ldmconfig(1M), 25, 217, 218
  - ldmp2v(1M), 206
- configuration, selecting to boot, 24
- configuring
  - jumbo frames, 135–138
  - Oracle VM Server for SPARC MIB software, 225–227
- control domain, 22
- CPU clock cycle skip, 167
- CPU core disable, 167
- create I/O domain, entire PCIe bus, 67
- creating, snmpv3 user, 228

## D

- daemons
  - drd, 154
  - ldmd, 23
  - vntsd, 24
- delayed reconfiguration, 154
- direct I/O (DIO), planning, 73
- domain
  - starting, 256–257
  - stopping, 257–258
- domain migration, non-interactive, 150
- domains
  - service, 23
  - types of, 22, 23
- DR, *See* dynamic reconfiguration
- dynamic reconfiguration (DR), 153
- dynamic reconfiguration daemon (drd), 154

## E

- environment variables, setting, 229

## F

- fault and recovery information, providing, 224

## G

- guest domain, 23

**H**

hypervisor, definition, 19

**I**

I/O domain, 65–66, 66–70, 70–80  
    assigning a PCIe bus, 66–70  
    assigning an endpoint device, 70–80  
    creating, 67  
    migration limitations, 65  
    PCI EXPRESS (PCIe) bus, 65–66  
installing, Oracle VM Server for SPARC MIB  
    software, 225–227

**J**

jumbo frames, configuring, 135–138

**L**

LDC, *See* logical domain channel  
ldm subcommands  
    cancel-operation reconf, 154  
    user authorizations, 41  
ldm(1M) command, 23  
ldm(1M) man page, 23  
ldmconfig(1M) command, 25, 217, 218  
ldmd, Logical Domains Manager daemon, 23  
ldmp2v(1M) command, 206  
ldomCoreTable table, 248–249  
ldomCreate trap, 250–251  
ldomCryptoTable table, 247–248  
ldomDestroy trap, 251  
ldomEnvVarsTable table, 232–233  
ldomIOBusTable table, 248  
ldomPolicyTable table, 233–234  
ldomSPConfigTable table, 234  
ldomStateChange trap, 251  
ldomTable table, 231–232  
ldomVccChange trap, 254–255  
ldomVccTable table, 244–245  
ldomVconsChange trap, 255

ldomVconsTable table, 245  
ldomVconsVccRelTable table, 245–246  
ldomVCpuChange trap, 251–252  
ldomVcpuTable table, 236–238  
ldomVdiskChange trap, 253  
ldomVdiskTable table, 241  
ldomVdsChange trap, 252–253  
ldomVdsdevTable table, 240–241  
ldomVdsTable table, 239–240  
ldomVMemChange trap, 252  
ldomVmemPhysBindTable table, 239  
ldomVmemTable table, 238–239  
ldomVnetChange trap, 254  
ldomVnetTable table, 243–244  
ldomVswChange trap, 253–254  
ldomVswTable table, 243  
link-based IPMP, using, 123–126  
loading, Oracle VM Server for SPARC MIB module in  
    SMA, 225–227  
logical domain channel (LDC), 22  
logical domains  
    definition, 19  
    roles, 22  
Logical Domains Manager, 20, 22  
    and Oracle VM Server for SPARC MIB, 224  
    daemon (ldmd), 23  
    discovery mechanism, 259  
    XML schema used with, 263

**M**

Management Information Base (MIB), 221  
memory dynamic reconfiguration (DR), 159  
memory power management (PM), 167  
MIB, 221  
migration, non-interactive, 150  
migration limitations, I/O domain, 65  
multipathing, virtual disk, 93

**N**

non-interactive domain migration, 150

**O**

- Oracle VM Server for SPARC MIB
  - and Logical Domains Manager, 224
  - object tree, 224–225
  - software components, 222
- Oracle VM Server for SPARC MIB module, loading in SMA, 225–227
- Oracle VM Server for SPARC MIB objects, retrieving, 229–231
- Oracle VM Server for SPARC MIB software
  - configuring, 225–227
  - installing, 225–227
  - removing, 225–227
- Oracle VM Server for SPARC MIB tables
  - core table (`ldomCoreTable`), 248–249
  - cryptographic units table (`ldomCryptoTable`), 247–248
  - domain policy table (`ldomPolicyTable`), 233–234
  - domain table (`ldomTable`), 231–232
  - environment variables table (`ldomEnvVarsTable`), 232–233
  - I/O bus table (`ldomIOBusTable`), 248
  - scalar variables for CPU resource pool, 235
  - scalar variables for cryptographic resource pool, 235–236
  - scalar variables for I/O bus resource pool, 236
  - scalar variables for Logical Domains version information, 249
  - scalar variables for memory resource pool, 235
  - service processor configuration table (`ldomSPConfigTable`), 234
  - virtual console concentrator table (`ldomVccTable`), 244–245
  - virtual console group table (`ldomVconsTable`), 245
  - virtual console relationship table (`ldomVconsVccRelTable`), 245–246
  - virtual CPU table (`ldomVcpuTable`), 236–238
  - virtual disk service device table (`ldomVdsdevTable`), 240–241
  - virtual disk service table (`ldomVdsTable`), 239–240
  - virtual disk table (`ldomVdiskTable`), 241
  - virtual memory physical binding table (`ldomVmemPhysBindTable`), 239
  - virtual memory table (`ldomVmemTable`), 238–239

Oracle VM Server for SPARC MIB tables (*Continued*)

- virtual network device device table (`ldomVnetTable`), 243–244
- virtual switch service device table (`ldomVswTable`), 243
- Oracle VM Server for SPARC MIB traps, 250–255
  - domain creation (`ldomCreate`), 250–251
  - domain destroy (`ldomDestroy`), 251
  - domain state change (`ldomStateChange`), 251
  - receiving, 250
  - sending, 249–250
  - virtual console concentrator change (`ldomVccChange`), 254–255
  - virtual console group change (`ldomVconsChange`), 255
  - virtual CPU change (`ldomVCpuChange`), 251–252
  - virtual disk change (`ldomVdiskChange`), 253
  - virtual disk service change (`ldomVdsChange`), 252–253
  - virtual memory change (`ldomVMemChange`), 252
  - virtual network change (`ldomVnetChange`), 254
  - virtual switch change (`ldomVswChange`), 253–254

**P**

- packages, `SUNWldm`, 23
- parsing, XML-based control interface, 224
- PCI EXPRESS (PCIe) bus, 65–66
- physical devices, 22, 23
- physical machine, 22
- planning
  - direct I/O (DIO), 73
- platforms
  - SPARC T3 server, 23
  - UltraSPARC T2 Plus server, 23
- power limit, 167
- power management (PM), 167
- primary domain, 22
  - rebooting, 74–75
- providing
  - fault and recovery information, 224
  - traps, 224

**R**

- read, authorizing, 41
- read and write, authorizing, 41
- rebooting the primary domain, 74–75
- receiving, Oracle VM Server for SPARC MIB traps, 250
- removing, Oracle VM Server for SPARC MIB software, 225–227
- resources
  - See also* virtual devices
  - definition, 21
- retrieving
  - Oracle VM Server for SPARC MIB information, 231–249
  - Oracle VM Server for SPARC MIB objects, 229–231
- roles, logical domains, 22
- root domain, 22

**S**

- sending, Oracle VM Server for SPARC MIB traps, 249–250
- service domain, 22, 23
- service processor (SP), monitoring and running physical machines, 22
- setting
  - environment variables, 229
  - power limit, 167
- snmpv3 user, creating, 228
- SPARC T3 server, 23
- starting, domain, 256–257
- stopping, domain, 257–258
- SUNWldm package, 23
- system controller, *See* service processor (SP)
- system management agent, 223

**T**

- tables, *See* Oracle VM Server for SPARC MIB tables
- traps
  - See* Oracle VM Server for SPARC MIB traps
  - providing, 224

**U**

- UltraSPARC T2 Plus server, 23
- using link-based IPMP, 123–126

**V**

- virtual devices
  - I/O, 23
  - virtual console concentrator (vcc), 24
  - virtual disk client (vdc), 24
  - virtual disk service (vds), 24
  - virtual network (vnet), 23
  - virtual switch (vsw), 24
- virtual disk multipathing, 93
- virtual machine, 22
- virtual network terminal server daemon (vntsd), 24
- vNTS, 244–246

**X**

- XML-based control interface, parsing, 224
- XML schema, Logical Domains Manager used with, 263