

Oracle® VM Server for SPARC 3.2 Administration Guide

Part No: E48732
May 2015

ORACLE®

Copyright © 2007, 2015, Oracle and/or its affiliates. All rights reserved.

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.

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.

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, then the following notice is applicable:

U.S. GOVERNMENT END USERS. Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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.

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.

This software or hardware and documentation may provide access to or information about 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 unless otherwise set forth in an applicable agreement between you and Oracle. 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, except as set forth in an applicable agreement between you and Oracle.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Copyright © 2007, 2015, Oracle et/ou ses affiliés. Tous droits réservés.

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 stipulation expresse 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 END USERS. Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

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.

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. AMD, Opteron, le logo AMD et le logo AMD Opteron sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée d'The Open Group.

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers, sauf mention contraire stipulée dans un contrat entre vous et Oracle. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation, sauf mention contraire stipulée dans un contrat entre vous et Oracle.

Accessibilité de la documentation

Pour plus d'informations sur l'engagement d'Oracle pour l'accessibilité à la documentation, visitez le site Web Oracle Accessibility Program, à l'adresse <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Accès au support électronique

Les clients Oracle qui ont souscrit un contrat de support ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> ou le site <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> si vous êtes malentendant.

Contents

Using This Documentation	11
I Oracle VM Server for SPARC 3.2 Software	13
 1 Overview of the Oracle VM Server for SPARC Software	15
About Oracle VM Server for SPARC and Oracle Solaris OS Versions	15
Hypervisor and Logical Domains	16
Logical Domains Manager	18
Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool	21
Oracle VM Server for SPARC Configuration Assistant	22
Oracle VM Server for SPARC Management Information Base	22
Oracle VM Server for SPARC Troubleshooting	22
 2 Oracle VM Server for SPARC Security	25
Delegating the Management of Logical Domains by Using Rights	25
Controlling Access to a Domain Console by Using Rights	29
Enabling and Using Auditing	37
Using Domain Console Logging	40
 3 Setting Up Services and the Control Domain	43
Output Messages	43
Creating Default Services	44
Initial Configuration of the Control Domain	45
Rebooting to Use Domains	48
Enabling Networking Between the Control/Service Domain and Other Domains (Oracle Solaris 10 Only)	48
Enabling the Virtual Network Terminal Server Daemon	50
 4 Setting Up Guest Domains	51
Creating and Starting a Guest Domain	51
Installing Oracle Solaris OS on a Guest Domain	54
 5 Configuring I/O Domains	61
I/O Domain Overview	61

General Guidelines for Creating an I/O Domain	62
6 Creating a Root Domain by Assigning PCIe Buses	63
Creating a Root Domain by Assigning PCIe Buses	63
7 Creating an I/O Domain by Using Direct I/O	73
Creating an I/O Domain by Assigning PCIe Endpoint Devices	73
Direct I/O Hardware and Software Requirements	75
Current Direct I/O Feature Limitations	76
Planning PCIe Endpoint Device Configuration	77
Rebooting the Root Domain With PCIe Endpoints Configured	78
Making PCIe Hardware Changes	80
Creating an I/O Domain by Assigning a PCIe Endpoint Device	82
8 Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions	89
SR-IOV Overview	89
SR-IOV Hardware and Software Requirements	92
Current SR-IOV Feature Limitations	95
Static SR-IOV	96
Dynamic SR-IOV	97
Enabling I/O Virtualization	98
Planning for the Use of PCIe SR-IOV Virtual Functions	99
Using Ethernet SR-IOV Virtual Functions	100
Using InfiniBand SR-IOV Virtual Functions	120
Using Fibre Channel SR-IOV Virtual Functions	134
I/O Domain Resiliency	149
Rebooting the Root Domain With Non-Resilient I/O Domains Configured	155
9 Using Non-primary Root Domains	157
Non-primary Root Domains Overview	157
Non-primary Root Domain Requirements	158
Non-primary Root Domain Limitations	159
Non-primary Root Domain Examples	160
10 Using Virtual Disks	167
Introduction to Virtual Disks	167
Virtual Disk Identifier and Device Name	169
Managing Virtual Disks	169
Virtual Disk Appearance	172
Virtual Disk Back End Options	173
Virtual Disk Back End	174
Configuring Virtual Disk Multipathing	181

CD, DVD and ISO Images	186
Virtual Disk Timeout	189
Virtual Disk and SCSI	190
Virtual Disk and the <code>format</code> Command	190
Using ZFS With Virtual Disks	191
Using Volume Managers in an Oracle VM Server for SPARC Environment	195
11 Using Virtual Networks	199
Introduction to a Virtual Network	200
Oracle Solaris 10 Networking Overview	200
Oracle Solaris 11 Networking Overview	202
Maximizing Virtual Network Performance	204
Virtual Switch	205
Virtual Network Device	207
Viewing Network Device Configurations and Statistics	210
Controlling the Amount of Physical Network Bandwidth That Is Consumed by a Virtual Network Device	213
Virtual Device Identifier and Network Interface Name	216
Assigning MAC Addresses Automatically or Manually	219
Using Network Adapters With Domains	222
Configuring a Virtual Switch and the Service Domain for NAT and Routing	222
Configuring IPMP in an Oracle VM Server for SPARC Environment	226
Using VLAN Tagging	232
Using Private VLANs	236
Tuning Packet Throughput Performance	241
Using NIU Hybrid I/O	242
Using Link Aggregation With a Virtual Switch	246
Configuring Jumbo Frames	248
Oracle Solaris 11 Networking-Specific Feature Differences	252
12 Migrating Domains	253
Introduction to Domain Migration	253
Overview of a Migration Operation	254
Software Compatibility	255
Security for Migration Operations	255
FIPS 140-2 Mode for Domain Migration	257
Domain Migration Restrictions	259
Migrating a Domain	262
Migrating an Active Domain	263
Migrating Bound or Inactive Domains	270
Monitoring a Migration in Progress	272

Canceling a Migration in Progress	272
Recovering From a Failed Migration	273
Migration Examples	273
13 Managing Resources	277
Resource Reconfiguration	277
Resource Allocation	279
CPU Allocation	280
Configuring the System With Hard Partitions	284
Assigning Physical Resources to Domains	291
Using Memory Dynamic Reconfiguration	295
Using Resource Groups	303
Using Power Management	303
Using Dynamic Resource Management	304
Listing Domain Resources	307
Using Perf-Counter Properties	313
14 Managing Domain Configurations	317
Managing Domain Configurations	317
Available Configuration Recovery Methods	318
Addressing Service Processor Connection Problems	324
15 Handling Hardware Errors	325
Hardware Error-Handling Overview	325
Using FMA to Blacklist or Unconfigure Faulty Resources	325
Recovering Domains After Detecting Faulty or Missing Resources	327
Marking Domains as Degraded	330
Marking I/O Resources as Evacuated	331
16 Performing Other Administration Tasks	333
Entering Names in the CLI	333
Updating Property Values in the /etc/system File	334
Connecting to a Guest Console Over the Network	335
Using Console Groups	335
Stopping a Heavily Loaded Domain Can Time Out	336
Operating the Oracle Solaris OS With Oracle VM Server for SPARC	337
Using Oracle VM Server for SPARC With the Service Processor	338
Configuring Domain Dependencies	339
Determining Where Errors Occur by Mapping CPU and Memory Addresses	343
Using Universally Unique Identifiers	345
Virtual Domain Information Command and API	346
Using Logical Domain Channels	347

Booting a Large Number of Domains	349
Cleanly Shutting Down and Power Cycling an Oracle VM Server for SPARC System	350
Logical Domains Variable Persistence	351
Adjusting the Interrupt Limit	352
Listing Domain I/O Dependencies	354
II Optional Oracle VM Server for SPARC Software	357
17 Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool	359
Oracle VM Server for SPARC P2V Tool Overview	359
Back-End Devices	361
Installing the Oracle VM Server for SPARC P2V Tool	362
Using the <code>ldmp2v</code> Command	365
Known Issues With the Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool	372
18 Oracle VM Server for SPARC Configuration Assistant (Oracle Solaris 10)	375
Using the Configuration Assistant (<code>ldmconfig</code>)	375
19 Using Power Management	379
Using Power Management	379
20 Using the Oracle VM Server for SPARC Management Information Base Software	387
Oracle VM Server for SPARC Management Information Base Overview	387
Installing and Configuring the Oracle VM Server for SPARC MIB Software	392
Managing Security	395
Monitoring Domains	397
Using SNMP Traps	417
Starting and Stopping Domains	425
21 Logical Domains Manager Discovery	429
Discovering Systems Running the Logical Domains Manager	429
22 Using the XML Interface With the Logical Domains Manager	433
XML Transport	433
XML Protocol	434
Event Messages	439
Logical Domains Manager Actions	444
Logical Domains Manager Resources and Properties	446
XML Schemas	464

Contents

Glossary	467
Index	477

Using This Documentation

- **Overview** – 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 3.2 software on supported servers, blades, and server modules. See “[Supported Platforms](#)” in “[Oracle VM Server for SPARC 3.2 Installation Guide](#)”.

Note - The features that are described in this book can be used with all of the supported system software and hardware platforms that are listed in “[Oracle VM Server for SPARC 3.2 Installation Guide](#)”. However, some features are only available on a subset of the supported system software and hardware platforms. For information about these exceptions, see “[What’s New in This Release](#)” in “[Oracle VM Server for SPARC 3.2 Release Notes](#)” and [What’s New in Oracle VM Server for SPARC Software \(<http://www.oracle.com/technetwork/server-storage/vm/documentation/sparc-whatsnew-330281.html>\)](http://www.oracle.com/technetwork/server-storage/vm/documentation/sparc-whatsnew-330281.html).

-
- **Audience** – System administrators who manage virtualization on SPARC servers
 - **Required knowledge** – System administrators on these servers must have a working knowledge of UNIX systems and the Oracle Solaris operating system (Oracle Solaris OS)

Product Documentation Library

Late-breaking information and known issues for this product are included in the documentation library at <http://www.oracle.com/pls/topic/lookup?ctx=E48724>.

Feedback

Provide feedback about this documentation at <http://www.oracle.com/goto/docfeedback>.

PART I

Oracle VM Server for SPARC 3.2 Software

This part introduces the Oracle VM Server for SPARC 3.2 software, which provides highly efficient, enterprise-class virtualization capabilities for SPARC T-Series, SPARC M-Series, and Fujitsu M10 platforms.

Overview of the Oracle VM Server for SPARC Software

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

Oracle VM Server for SPARC provides highly efficient, enterprise-class virtualization capabilities for SPARC T-Series and SPARC M-Series platforms, and Fujitsu M10 platforms. 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 and SPARC M-Series platforms, Fujitsu M10 platforms and the Oracle Solaris OS.

This chapter covers the following topics:

- “About Oracle VM Server for SPARC and Oracle Solaris OS Versions” on page 15
- “Hypervisor and Logical Domains” on page 16
- “Logical Domains Manager” on page 18
- “Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool” on page 21
- “Oracle VM Server for SPARC Configuration Assistant” on page 22
- “Oracle VM Server for SPARC Management Information Base” on page 22
- “Oracle VM Server for SPARC Troubleshooting” on page 22

About Oracle VM Server for SPARC and Oracle Solaris OS Versions

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 “[Fully Qualified Oracle Solaris OS Versions](#)” in “[Oracle VM Server for SPARC 3.2 Installation Guide](#)”.

The version of the Oracle Solaris OS that runs on a guest domain is *independent* of the Oracle Solaris OS version that runs on the primary domain. So, if you run the Oracle Solaris 10 OS in the primary domain, you can still run the Oracle Solaris 11 OS in a guest domain. Likewise, if you run the Oracle Solaris 11 OS in the primary domain, you can still run the Oracle Solaris 10 OS in a guest domain.

Note - Some platforms no longer support the Oracle Solaris 10 OS in the primary domain. Check your platform documentation. You can continue to run the Oracle Solaris 10 OS in the other domains.

The only difference between running the Oracle Solaris 10 OS or the Oracle Solaris 11 OS on the primary domain is the feature differences in each OS.

Hypervisor and Logical Domains

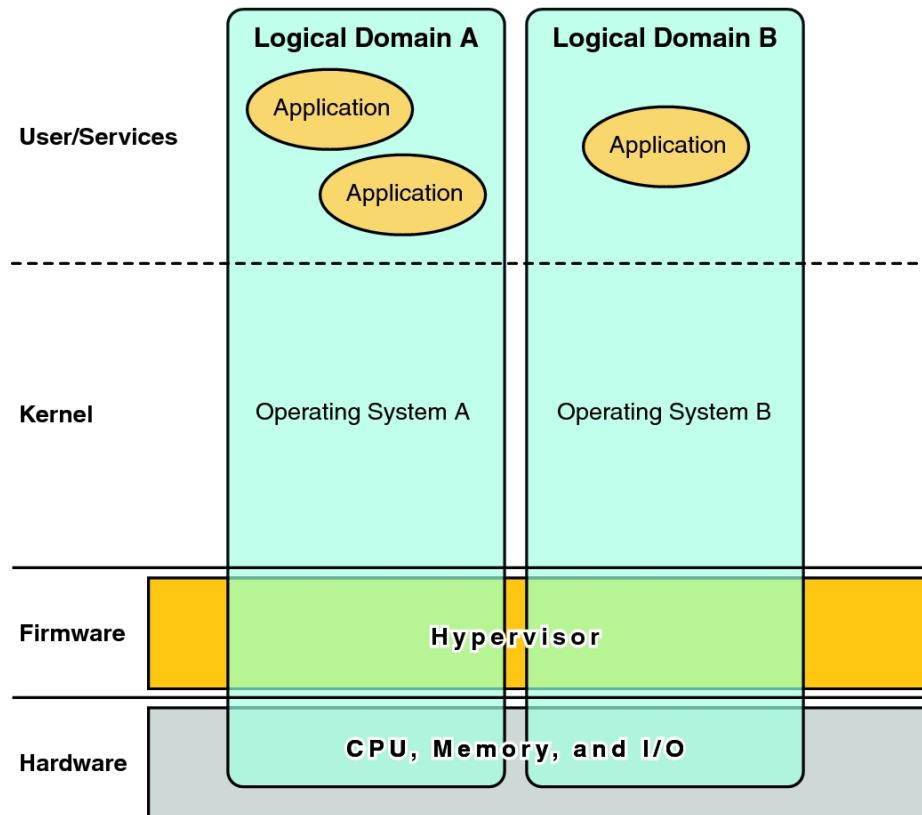
This section provides an overview of the SPARC hypervisor, which supports logical domains.

The SPARC *hypervisor* is a small firmware layer that provides a stable virtualized machine architecture to which an operating system can be written. SPARC 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 perform a power cycle of 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 Oracle VM Server for SPARC functionality:

- User/services (applications)
- Kernel (operating systems)
- Firmware (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 capability 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 you to perform a power cycle 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.

In addition to using the `ldm` command to manage the Oracle VM Server for SPARC software, you can now use Oracle VM Manager.

Oracle VM Manager is a web-based user interface that you can use to manage the Oracle VM environment. Earlier versions of this user interface only managed the Oracle VM Server x86 software, but starting with Oracle VM Manager 3.2 and Oracle VM Server for SPARC 3.0, you can also manage the Oracle VM Server for SPARC software. For more information about Oracle VM Manager, see [Oracle VM Documentation \(`http://www.oracle.com/technetwork/documentation/vm-096300.html`\)](http://www.oracle.com/technetwork/documentation/vm-096300.html).

Logical Domains Manager

The Logical Domains Manager is used to create and manage logical domains, as well as to 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. Logical domains can perform the following roles:

- **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. You can have more than one service domain, and 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 the following:
 - A PCIe root complex.

- A PCIe slot or on-board PCIe device by using the direct I/O (DIO) feature. See [“Creating an I/O Domain by Assigning PCIe Endpoint Devices” on page 73](#).
- A PCIe SR-IOV virtual function. See [Chapter 8, “Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions”](#).

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 an Oracle Sun SPARC Enterprise T5440 server, you can have up to four root domains.

The default root domain is the **primary** domain. You can use **non-primary** domains to act as 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 Oracle VM Server for SPARC. 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 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 an Oracle VM Server for SPARC environment, you can provision up to 128 domains on a system (up to 256 on a Fujitsu M10 server). Some servers, particularly single-processor and some dual-processor systems, have a limited number of I/O buses and physical I/O slots.

As a result, you might be unable to 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. This limitation on the number of physical I/O devices that can be directly accessed is addressed by implementing a virtualized I/O model. See [Chapter 5, “Configuring I/O Domains”](#).

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

Oracle VM Server for SPARC 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), which exports a block device interface
- Virtual disk service (vds), which 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 an Oracle VM Server for SPARC 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 console traffic for all domains, 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 277.

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. For details, see the `ldm(1M)` man page. You can also specify a configuration to boot from the SP, as described in “[Using Oracle VM Server for SPARC With the Service Processor](#)” on page 338.

For information about managing configurations, see “[Managing Domain Configurations](#)” on page 317.

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 the Oracle Solaris 10 OS in a logical domain on a chip multithreading (CMT) system. You can run the `ldmp2v` command from a control domain that runs the Oracle Solaris 10 OS or the Oracle Solaris 11 OS to convert one of the following source systems to a logical domain:

- Any sun4u SPARC based system that runs at least the Solaris 8, Solaris 9, or Oracle Solaris 10 OS
- Any sun4v system that runs the Oracle Solaris 10 OS but does not run the Oracle VM Server for SPARC software

Note - You cannot use the P2V tool to convert an Oracle Solaris 11 physical system to a virtual system.

For information about the tool and about installing it, see [Chapter 17, “Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool”](#). For information about the `ldmp2v(1M)` man page, 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.

Note - The `ldmconfig` command is supported only on Oracle Solaris 10 systems.

The Configuration Assistant is a terminal-based tool.

For more information, see [Chapter 18, “Oracle VM Server for SPARC Configuration Assistant \(Oracle Solaris 10\)”](#) 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 20, “Using the Oracle VM Server for SPARC Management Information Base Software”](#).

Oracle VM Server for SPARC Troubleshooting

You can get information about particular problems with the Oracle VM Server for SPARC software from the following publications:

- “Known Issues” in “Oracle VM Server for SPARC 3.2 Release Notes ”
- Information Center: Overview of Oracle VM Server for SPARC (LDoms) (Doc ID 1589473.2) (https://mosemp.us.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=227880986952919&id=1589473.2&_afrWindowMode=0&_adf.ctrl-state=wu098o5r6_96)

Oracle VM Server for SPARC Security

This chapter describes some security features that you can enable on your Oracle VM Server for SPARC system.

This chapter covers the following topics:

- “[Delegating the Management of Logical Domains by Using Rights](#)” on page 25
- “[Controlling Access to a Domain Console by Using Rights](#)” on page 29
- “[Enabling and Using Auditing](#)” on page 37
- “[Using Domain Console Logging](#)” on page 40

Note - The examples in this book are shown as being performed by superuser. However, you can use profiles instead to have users acquire more fine-grained permissions to perform management tasks.

Delegating the Management of Logical Domains by Using Rights

The Logical Domains Manager package adds two predefined rights profiles to the local rights configuration. These rights profiles delegate administrative privileges to unprivileged users:

- The `LDoms Management` profile permits a user to use all `ldm` subcommands.
- The `LDoms Review` profile permits a user to use all list-related `ldm` subcommands.

These rights profiles can be assigned directly to users or to a role that is then assigned to users. When one of these profiles is assigned directly to a user, you must use the `pfexec` command or a profile shell, such as `pfbash` or `pfksh`, to successfully use the `ldm` command to manage your domains. Determine whether to use roles or rights profiles based on your rights configuration. See “[System Administration Guide: Security Services](#)” or Part III, “[Roles, Rights Profiles, and Privileges](#),” in “[Oracle Solaris 11.1 Administration: Security Services](#)”.

Users, authorizations, rights 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 rights profiles to the local files. To configure profiles and roles in a naming service, see “[System Administration Guide: Naming and Directory Services \(DNS, NIS, and LDAP\)](#)”. For an overview of the authorizations and execution attributes delivered by the Logical Domains Manager package, see “[Logical Domains Manager Profile Contents](#)” on page 29. All of the examples in this chapter assume that the rights configuration uses local files.

Using Rights Profiles and Roles



Caution - Be careful when using the `usermod` and `rolemod` commands to add authorizations, rights profiles, or roles.

- For the Oracle Solaris 10 OS, the `usermod` or `rolemod` command replaces any existing values.
To add values instead of replacing them, specify a comma-separated list of existing values and the new values.
- For the Oracle Solaris 11 OS, add values by using the plus sign (+) for each authorization you add.
For example, the `usermod -A +auth username` command grants the *auth* authorization to the *username* user; similarly for the `rolemod` command.

Managing User Rights Profiles

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

▼ How to Assign a Rights 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](#)” or Part III, “[Roles, Rights Profiles, and Privileges](#),” in “[Oracle Solaris 11.1 Administration: Security Services](#)”.

1. **Become an administrator.**
 - For Oracle Solaris 10, see “[Configuring RBAC \(Task Map\)](#)” in “[System Administration Guide: Security Services](#)”.

- For Oracle Solaris 11.2, see Chapter 1, “About Using Rights to Control Users and Processes,” in “Securing Users and Processes in Oracle Solaris 11.2”.
2. **Assign an administrative profile to a local user account.**

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

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

The following command assigns the LDom Management profile to user sam:

```
# usermod -P "LDom Management" sam
```

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 from assuming that role even though he has the password.

▼ How to Create a Role and Assign the Role to a User

1. **Become an administrator.**

- For Oracle Solaris 10, see “Configuring RBAC (Task Map)” in “System Administration Guide: Security Services”.
- For Oracle Solaris 11.2, see Chapter 1, “About Using Rights to Control Users and Processes,” in “Securing Users and Processes in Oracle Solaris 11.2”.

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 2-1 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:  
Re-enter new Password:  
passwd: password successfully changed for ldm_read  
# useradd -R ldm_read user_1  
# passwd user_1  
New Password:  
Re-enter new Password:  
passwd: password successfully changed for user_1  
# su user_1  
Password:  
$ id  
uid=95555(user_1) gid=10(staff)  
$ roles  
ldm_read  
$ su ldm_read  
Password:  
$ id
```

```
uid=99667(ldm_read) gid=14(sysadmin)
```

Logical Domains Manager Profile Contents

The Logical Domains Manager package adds the following rights profiles to the local rights profile description database:

```
LDoms Power Mgmt Observability:::View LDoms Power Consumption:auths=solaris.ldoms.ldmpower
LDoms Review:::Review LDoms configuration:profiles=LDoms Power Mgmt
Observability;auths=solaris.ldoms.read
LDoms Management:::Manage LDoms domains:profiles=LDoms Power Mgmt
Observability;auths=solaris.ldoms.*
```

The Logical Domains Manager package also adds the following execution attribute that is associated with the LDoms Management profile and the LDoms Power Mgmt Observability profile to the local execution profiles database:

```
LDoms Management:suser:cmd:::/usr/sbin/ldm:privs=file_dac_read,file_dac_search
LDoms Power Mgmt Observability:suser:cmd:::/usr/sbin/ldmpower:privs=file_dac_search
```

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

TABLE 2-1 ldm Subcommands and User Authorizations

ldm Subcommand [†]	User Authorization
add-*	solaris.ldoms.write
bind-domain	solaris.ldoms.write
list	solaris.ldoms.read
list-*	solaris.ldoms.read
panic-domain	solaris.ldoms.write
remove-*	solaris.ldoms.write
set-*	solaris.ldoms.write
start-domain	solaris.ldoms.write
stop-domain	solaris.ldoms.write
unbind-domain	solaris.ldoms.write

[†]Refers to all the resources you can add, list, remove, or set.

Controlling Access to a Domain Console by Using Rights

By default, any user can access all domain consoles. To control access to a domain console, configure the vntsd daemon to perform authorization checking. The vntsd daemon provides a Service Management Facility (SMF) property named vntsd/authorization. This property

can be configured to enable 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 alternative IP address when vntsd/authorization is enabled, vntsd ignores the alternative 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 authorization description database.

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

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

You can control the access to all domain consoles or to a single domain console.

- To control the access to all domain consoles, see “[How to Control Access to All Domain Consoles by Using Roles](#)” on page 30 and “[How to Control Access to All Domain Consoles by Using Rights Profiles](#)” on page 32.
- To control access to a single domain console, see “[How to Control Access to a Single Console by Using Roles](#)” on page 34 and “[How to Control Access to a Single Console by Using Rights Profiles](#)” on page 36.

▼ How to Control Access to All Domain Consoles by Using Roles

1. **Restrict access to a domain console by enabling console authorization checking.**

```
primary# svccfg -s vntsd setprop vntsd/authorization = true  
primary# svcadm refresh vntsd  
primary# svcadm restart vntsd
```

2. **Create a role that has the solaris.vntsd.consoles authorization, which permits access to all domain consoles.**

```
primary# roleadd -A solaris.vntsd.consoles role-name
primary# passwd all_cons
```

3. Assign the new role to a user.

```
primary# usermod -R role-name username
```

Example 2-2 Controlling Access to All Domain Consoles by Using Roles

First, enable console authorization checking to restrict access to a domain console.

```
primary# svccfg -s vntsd setprop vntsd/authorization = true
primary# svcadm refresh vntsd
primary# svcadm restart vntsd
primary# ldm ls
NAME      STATE    FLAGS   CONS   VCPU   MEMORY   UTIL   UPTIME
primary   active   -n-cv-  UART    8      16G      0.2%  47m
ldg1     active   -n--v-  5000    2      1G       0.1%  17h 50m
ldg2     active   -t----  5001    4      2G       25%   11s
```

The following example shows how to create the `all_cons` role with the `solaris.vntsd.consoles` authorization, which permits access to all domain consoles.

```
primary# roleadd -A solaris.vntsd.consoles all_cons
primary# passwd all_cons
New Password:
Re-enter new Password:
passwd: password successfully changed for all_cons
```

This command assigns the `all_cons` role to the `sam` user.

```
primary# usermod -R all_cons sam
```

User `sam` assumes the `all_cons` role and can access any console. For example:

```
$ id
uid=700299(sam) gid=1(other)
$ su all_cons
Password:
$ telnet localhost 5000
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.

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

$ telnet localhost 5001
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.
```

```
Connecting to console "ldg2" in group "ldg2" ....  
Press ~? for control options ..
```

This example shows what happens when an unauthorized user, dana, attempts to access a domain console:

```
$ id  
uid=702048(dana) gid=1(other)  
$ telnet localhost 5000  
Trying 0.0.0.0...  
Connected to 0.  
Escape character is '^]'.  
Connection to 0 closed by foreign host.
```

▼ How to Control Access to All Domain Consoles by Using Rights Profiles

1. Restrict access to a domain console by enabling console authorization checking.

```
primary# svccfg -s vntsd setprop vntsd/authorization = true  
primary# svcadm refresh vntsd  
primary# svcadm restart vntsd
```

2. Create a rights profile with the `solaris.vntsd.consoles` authorization.

- Oracle Solaris 10 OS: Edit the `/etc/security/prof_attr` file.

Include the following entry:

```
LDoms Consoles:::Access LDoms Consoles:auths=solaris.vntsd.consoles
```

- Oracle Solaris 11 OS: Use the `profiles` command to create a new profile.

```
primary# profiles -p "LDoms Consoles" \  
'set desc="Access LDoms Consoles"; set auths=solaris.vntsd.consoles'
```

3. Assign the rights profile to a user.

- Oracle Solaris 10 OS: Assign the rights profile to a user.

```
primary# usermod -P "All,Basic Solaris User,LDoms Consoles" username
```

Be careful to specify any pre-existing profiles when adding the LDoms Consoles profile. The previous command shows that the user already had the All and Basic Solaris User profiles.

- **Oracle Solaris 11 OS: Assign the rights profile to a user.**

```
primary# usermod -P +"LDoms Consoles" username
```

4. Connect to the domain console as the user.

```
$ telnet localhost 5000
```

Example 2-3 Controlling Access to All Domain Consoles by Using Rights Profiles

The following examples show how to use rights profiles to control access to all domain consoles:

- **Oracle Solaris 10:** Create a rights profile with the `solaris.vntsd.consoles` authorization by adding the following entry to the `/etc/security/prof_attr` file:

```
LDoms Consoles:::Access LDoms Consoles:auths=solaris.vntsd.consoles
```

Assign the rights profile to *username*.

```
primary# usermod -P "All,Basic Solaris User,LDoms Consoles" username
```

The following commands show how to verify that the user is `sam` and that the `All`, `Basic Solaris User`, and `LDoms Consoles` rights profiles are in effect. The `telnet` command shows how to access the `ldg1` domain console.

```
$ id
uid=702048(sam) gid=1(other)
$ profiles
All
Basic Solaris User
LDoms Consoles
$ telnet localhost 5000
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.
```

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

- **Oracle Solaris 11:** Use the `profiles` command to create a rights profile with the `solaris.vntsd.consoles` authorization in the rights profile description database.

```
primary# profiles -p "LDoms Consoles" \
'set desc="Access LDoms Consoles"; set auths=solaris.vntsd.consoles'
```

Assign the rights profile to a user.

```
primary# usermod -P +"LDoms Consoles" sam
```

The following commands show how to verify that the user is `sam` and that the All, Basic Solaris User, and LDom Consoles rights profiles are in effect. The `telnet` command shows how to access the `ldg1` domain console.

```
$ id  
uid=702048(sam) gid=1(other)  
$ profiles  
All  
Basic Solaris User  
LDoms Consoles  
$ telnet localhost 5000  
Trying 0.0.0.0...  
Connected to 0.  
Escape character is '^]'.  
  
Connecting to console "ldg1" in group "ldg1" ....  
Press ~? for control options ..
```

▼ How to Control Access to a Single Console by Using Roles

1. **Restrict access to a domain console by enabling console authorization checking.**

```
primary# svccfg -s vntsd setprop vntsd/authorization = true  
primary# svcadm refresh vntsd  
primary# svcadm restart vntsd
```

2. **Add an authorization for a single domain to the authorization description database.**

The authorization name is derived from the name of the domain and has the form `solaris.vntsd.console-domain:`

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

3. **Create a role with the new authorization to permit access only to the console of the domain.**

```
primary# roleadd -A solaris.vntsd.console-domain role-name  
primary# passwd role-name  
New Password:  
Re-enter new Password:  
passwd: password successfully changed for role-name
```

4. **Assign the `role-name` role to a user.**

```
primary# usermod -R role-name username
```

Example 2-4 Accessing a Single Domain Console

This example shows how user **terry** assumes the **ldg1cons** role and accesses the **ldg1** domain console.

First, add an authorization for a single domain, **ldg1**, to the authorization description database.

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

Then, create a role with the new authorization to permit access only to the console of the domain.

```
primary# roleadd -A solaris.vntsd.console-ldg1 ldg1cons
primary# passwd ldg1cons
New Password:
Re-enter new Password:
passwd: password successfully changed for ldg1cons
```

Assign the **ldg1cons** role to user **terry**, assume the **ldg1cons** role, and access the domain console.

```
primary# usermod -R ldg1cons terry
primary# su terry
Password:
$ id
uid=700300(terry) gid=1(other)
$ su ldg1cons
Password:
$ id
uid=700303(ldg1cons) gid=1(other)
$ telnet localhost 5000
Trying 0.0.0.0...
Escape character is '^]'.

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

The following example shows that the user **terry** cannot access the **ldg2** domain console:

```
$ telnet localhost 5001
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.
Connection to 0 closed by foreign host.
```

▼ How to Control Access to a Single Console by Using Rights Profiles

1. Restrict access to a domain console by enabling console authorization checking.

```
primary# svccfg -s vntsd setprop vntsd/authorization = true  
primary# svcadm refresh vntsd  
primary# svcadm restart vntsd
```

2. Add an authorization for a single domain to the authorization description database.

The following example entry adds the authorization for a domain console:

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

3. Create a rights profile with an authorization to access a specific domain console.

- Oracle Solaris 10 OS: Edit the /etc/security/prof_attr file.

```
domain Console:::Access domain  
Console:auths=solaris.vntsd.console-domain
```

This entry must be on a single line.

- Oracle Solaris 11 OS: Use the profiles command to create a new profile.

```
primary# profiles -p "domain Console" \  
'set desc="Access domain Console";  
set auths=solaris.vntsd.console-domain'
```

4. Assign the rights profile to a user.

The following commands assign the profile to a user:

- Oracle Solaris 10 OS: Assign the rights profile.

```
primary# usermod -P "All,Basic Solaris User, domain Console" username
```

Note that the All and Basic Solaris User profiles are required.

- Oracle Solaris 11 OS: Assign the rights profile.

```
primary# usermod -P +"domain Console" username
```

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 that tracks 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 [Part VII, “Auditing in Oracle Solaris,” in “System Administration Guide: Security Services”](#).
- **Oracle Solaris 11 OS:** Use the audit command. See the audit(1M) man page and [Part VII, “Auditing in Oracle Solaris,” in “Oracle Solaris 11.1 Administration: Security Services”](#).

▼ How to Enable Auditing

You must configure and enable the Oracle Solaris auditing feature on your system. Oracle Solaris 11 auditing is enabled by default, but you must still perform some configuration steps.

Note - Pre-existing processes are *not* audited for the virtualization software (vs) class. Ensure that you perform this step *before* regular users log in to the system.

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. **(Oracle Solaris 10) Add the vs class to the /etc/security/audit_control file.**

The following example /etc/security/audit_control fragment shows how you might specify the vs class:

```
dir:/var/audit
flags:lo,vs
minfree:20
```

```
naflags:lo,na
```

3. (Oracle Solaris 10) Enable the auditing feature.

a. Run the bsmconv command.

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

b. Reboot the system.

4. (Oracle Solaris 11) Preselect the vs audit class.

a. Determine which auditing classes are already selected.

Ensure that any audit classes that have already been selected are part of the updated set of classes. The following example shows that the lo class is already selected:

```
# auditconfig -getflags
active user default audit flags = lo(0x1000,0x1000)
configured user default audit flags = lo(0x1000,0x1000)
```

b. Add the vs auditing class.

```
# auditconfig -setflags [class],vs
```

class is zero or more audit classes, separated by commas. You can see the list of audit classes in the /etc/security/audit_class file. Be sure to include the vs class on your Oracle VM Server for SPARC system.

For example, the following command selects both the lo and vs classes:

```
# auditconfig -setflags lo,vs
```

c. (Optional) Log out of the system if you want to audit your processes, either as the administrator or as the configurer.

If you do not want to log out, see “[How to Update the Preselection Mask of Logged In Users](#)” in “[Oracle Solaris 11.1 Administration: Security Services](#)”.

5. Verify that the auditing software is running.

```
# auditconfig -getcond
```

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

▼ How to Disable Auditing

● Disable the auditing feature.

- Oracle Solaris 10 OS:

- 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.

- Oracle Solaris 11 OS:

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

```
# audit -t
```

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

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

▼ How to Review Audit Records

- Use one of the following methods to review vs audit output:
 - Use the **auditreduce** and **praudit** commands to review audit output.

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

- Use the **praudit -x** command to print audit records in XML form.

▼ How to 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.

Using Domain Console Logging

In an Oracle VM Server for SPARC environment, console I/O from the primary domain is directed to the service processor (SP). The console I/O from all other domains is redirected to the service domain that runs the virtual console concentrator, vcc. If the service domain runs the Oracle Solaris 11 OS, the guest domain console output can be logged to a file.

Service domains support console logging for logical domains. While the service domain must run the Oracle Solaris 11 OS, the guest domain being logged can run either the Oracle Solaris 10 OS or the Oracle Solaris 11 OS.

The domain console log is saved to a file on the service domain called `/var/log/vntsd/domain/console-log` that provides the vcc service. You can rotate console log files by using the `logadm` command. See the `logadm(1M)` and `logadm.conf(4)` man pages.

The Oracle VM Server for SPARC software enables you to selectively enable and disable console logging for each logical domain. Console logging is enabled by default.

▼ How to Enable or Disable Console Logging

You must enable or disable console logging for each individual logical domain even if the domains belong to the same console group.

1. **List the current console settings for the domain.**

```
primary# ldm list -o console domain
```

2. **Stop and unbind the domain.**

The domain must be in an inactive and unbound state before you modify the console settings.

```
primary# ldm stop domain
primary# ldm unbind domain
```

3. **Enable or disable console logging.**

- **To enable console logging.**

```
primary# ldm set-vcons log=on domain
```

- **To disable console logging.**

```
primary# ldm set-vcons log=off domain
```

Service Domain Requirements for Domain Console Logging

A domain that is attached to a service domain that runs an OS version older than Oracle Solaris 11.1 *cannot* be logged.

Note - Even if you enable console logging for a domain, the domain's virtual console is not logged if the required support is not available on the service domain.

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 18, “Oracle VM Server for SPARC Configuration Assistant \(Oracle Solaris 10\)”](#).

This chapter covers the following topics:

- [“Output Messages” on page 43](#)
- [“Creating Default Services” on page 44](#)
- [“Initial Configuration of the Control Domain” on page 45](#)
- [“Rebooting to Use Domains” on page 48](#)
- [“Enabling Networking Between the Control/Service Domain and Other Domains \(Oracle Solaris 10 Only\)” on page 48](#)
- [“Enabling the Virtual Network Terminal Server Daemon” on page 50](#)

Output Messages

If a resource cannot be dynamically configured on the control domain, the recommended practice is 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

▼ How to 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 logical domain must communicate outside the box through the virtual switch.

- In Oracle Solaris 10, add a virtual switch service on a network adapter to the control domain.

```
primary# ldm add-vsw net-dev=network-device vsw-service primary
```

For example, the following command adds a virtual switch service (primary-vsw0) on network adapter nxge0 to the control domain (primary):

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

- In Oracle Solaris 11, add a virtual switch service (primary-vsw0) on a network device that you want to use for guest domain networking:

```
primary# ldm add-vsw net-dev=network-device vsw-service primary
```

For example, the following command adds a virtual switch service (`primary-vsw0`) on network device `net0` to the control domain (`primary`):

```
primary# ldm add-vsw net-dev=net0 primary-vsw0 primary
```

You can use the `ldm list-netdev -b` command to determine the backend network devices that are available for the virtual switch. See “[Virtual Switch](#)” on page 205.

You can dynamically update the `net-dev` property value by using the `ldm set-vsw` command.

- The following process applies to the Oracle Solaris 10 OS only and should *not* be performed on an Oracle Solaris 11 system.

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, you are responsible for ensuring 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 \(Oracle Solaris 10 Only\)](#)” on page 48.

```
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.

Configuring the Control Domain

▼ How to Configure the Control Domain

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.

For domain sizing recommendations, see *Oracle VM Server for SPARC Best Practices* (<http://www.oracle.com/technetwork/server-storage/vm/ovmsparc-best-practices-2334546.pdf>).

1. Assign virtual CPUs to the control domain.

Service domains, including the control domain, require CPU and memory resources to perform virtual disk and virtual network I/O operations for guest domains. The amount of CPU and memory resources to allocate depends on the workload of the guest domain.

For example, the following command assigns two CPU cores (16 virtual CPU threads) to the control domain, `primary`. The remainder of the virtual CPU threads are available for guest domains.

```
primary# ldm set-core 2 primary
```

You can dynamically change the actual CPU allocation based on application requirements. Use the `ldm list` command to determine the CPU utilization of the control domain. If the control domain has high CPU utilization, use the `ldm add-core` and `ldm set-core` commands to add CPU resources to a service domain.

2. Determine whether you need cryptographic devices in the control domain.

Note that only UltraSPARC T2, UltraSPARC T2 Plus, and SPARC T3 platforms have cryptographic devices (MAUs). Newer platforms such as SPARC T4 systems and Fujitsu M10 servers already provide cryptographic acceleration, so you do not need to assign cryptographic accelerators to these platforms.

If you are using one of the older processors, assign one cryptographic unit for each CPU whole core in the control domain.

The following example assigns two cryptographic resources to the control domain, `primary`:

```
primary# ldm set-crypto 2 primary
```

3. Initiate a delayed reconfiguration on the control domain.

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

4. Assign memory to the control domain.

For example, the following command assigns 16 Gbytes of memory to the control domain, **primary**. This setup leaves the remainder of the memory available to guest domains.

```
primary# ldm set-memory 16G primary
```

5. Save the domain configuration to the service processor (SP).

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

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

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

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

This **ldm list-config** command shows that the **initial** configuration set will be used after you perform a power cycle.

7. Reboot the control domain to make the reconfiguration changes take effect.

Decreasing the CPU and Memory Resources From the Control Domain's Initial factory-default Configuration

You can use CPU DR to decrease the number of the control domain's cores from an initial factory-default configuration. However, you must use a delayed reconfiguration instead of a memory DR to decrease the control domain's memory.

When in the factory-default configuration, 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.

Note - When the Oracle Solaris OS is installed on a ZFS file system, it automatically sizes and creates swap and dump areas as ZFS volumes in the ZFS root pool based on the amount of physical memory that is present. If you change the domain's memory allocation, it might alter the recommended size of these volumes. The allocations might be larger than needed after reducing control domain memory. For swap space recommendations, see “[Planning for Swap Space](#)” in “[Managing File Systems in Oracle Solaris 11.2](#)”. Before you free disk space, you can can optionally change the swap and dump space. See “[Managing Your ZFS Swap and Dump Devices](#)” in “[Managing ZFS File Systems in Oracle Solaris 11.2](#)”.

▼ How to Decrease the CPU and Memory Resources From the Control Domain's Initial factory-default Configuration

This procedure shows how to decrease the CPU and memory resources from the control domain's initial factory-default configuration. You first use CPU DR to decrease the number of cores and then initiate a delayed reconfiguration before you decrease the amount of memory.

The example values are for CPU and memory sizes for a small control domain that has enough resources to run the ldmd daemon and to perform migrations. However, if you want to use the control domain for additional purposes, you can assign a larger number of cores and more memory to the control domain as needed.

1. **Boot the factory-default configuration.**

2. **Configure the control domain.**

See “[How to Configure the Control Domain](#)” on page 46.

Rebooting to Use 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.

▼ How to Reboot

- **Shut down and reboot the control domain.**

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

Note - Either a reboot or power cycle instantiates the new configuration. Only a power cycle 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 (Oracle Solaris 10 Only)



Caution - This section applies only to an Oracle Solaris 10 system. Do *not* configure the vsw interface on an Oracle Solaris 11 system.

By default, networking between the control domain and other domains in the system is disabled. To enable networking, 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.

Guest domains can automatically communicate with the control domain or service domain as long as the corresponding network back-end device is configured in the same virtual LAN or virtual network.

▼ How to Configure the Virtual Switch as the Primary Interface

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

If necessary, you can 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 3). 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 a Virtual Switch and the Service Domain for NAT and Routing](#)” on page 222.

1. Print the addressing information for all interfaces.

```
primary# ifconfig -a
```

2. Configure the virtual switch network interface.

```
primary# ifconfig vsw0 plumb
```

3. Remove the physical interface for the device that is assigned to the virtual switch (`net-dev`).

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

4. To migrate properties of the physical network device (`nxge0`) to the virtual switch device (`vsw0`), do one of the following:

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

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

- If networking is configured by using DHCP, enable DHCP for the virtual switch.

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

5. 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
```

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.

▼ How to 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 44 for more information.

1. Enable the virtual network terminal server daemon, `vntsd`.

```
primary# svcadm enable vntsd
```

2. 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 18, “Oracle VM Server for SPARC Configuration Assistant \(Oracle Solaris 10\)”](#).

This chapter covers the following topics:

- [“Creating and Starting a Guest Domain” on page 51](#)
- [“Installing Oracle Solaris OS on a Guest Domain” on page 54](#)

Creating and Starting a Guest Domain

The guest domain must run an operating system that is compatible with both the `sun4v` platform and the virtual devices presented by the hypervisor. Currently, this requirement means that you must run at least the Oracle Solaris 10 11/06 OS. Running the Oracle Solaris 10 1/13 OS provides you with all the Oracle VM Server for SPARC 3.2 features. See [“Oracle VM Server for SPARC 3.2 Installation Guide”](#) 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.

Note - A guest domain that has been assigned more than 1024 CPUs cannot run the Oracle Solaris 10 OS. In addition, you cannot use CPU DR to reduce the number of CPUs below 1024 to run the Oracle Solaris 10 OS.

To work around this problem, unbind the guest domain, remove CPUs until you have no more than 1024 CPUs, and then rebind the guest domain. You can then run the Oracle Solaris 10 OS on this guest domain.

▼ How to Create and Start a Guest Domain

1. [Create a logical domain.](#)

The following command would create a guest domain named `ldg1`.

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

2. Add CPUs to the guest domain.

Do one of the following:

■ **Add virtual CPUs.**

The following command would add eight virtual CPUs to guest domain `ldg1`.

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

■ **Add whole cores.**

The following command would add two whole cores to guest domain `ldg1`.

```
primary# ldm add-core 2 ldg1
```

3. Add memory to the guest domain.

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.

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 “[How to Export a ZFS Volume as a Single-Slice Disk](#)” on page 179 and “[Using ZFS With Virtual Disks](#)” on page 191.

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.** This 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 example exports 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 the **auto-boot?** and **boot-device** variables for the guest domain.

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

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

The following example command sets boot-device to vdisk1 for 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.

For Oracle Solaris 11 domains, use the DefaultFixed network configuration profile (NCP). You can enable this profile during or after installation.

During the Oracle Solaris 11 installation, select the Manual networking configuration. After the Oracle Solaris 11 installation, ensure that the DefaultFixed NCP is enabled by using the

`netadm list` command. See “[Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1](#)” and “[Connecting Systems Using Reactive Network Configuration in Oracle Solaris 11.1](#)”.

Memory Size Requirements

The Oracle VM Server for SPARC software does not impose a memory size limitation when you create a domain. The memory size requirement is a characteristic of the guest operating system. Some Oracle VM Server for SPARC functionality might not work if the amount of memory present is smaller than the recommended size. For recommended and minimum memory requirements for the Oracle Solaris 10 OS, see “[System Requirements and Recommendations](#)” in “[Oracle Solaris 10 1/13 Installation Guide: Planning for Installation and Upgrade](#)”. For recommended and minimum memory requirements for the Oracle Solaris 11 OS, see “[Oracle Solaris 11 Release Notes](#)”, “[Oracle Solaris 11.1 Release Notes](#)”, and “[Oracle Solaris 11.2 Release Notes](#)”.

The OpenBoot PROM has a minimum size restriction for a domain. Currently, that restriction is 12 Mbytes. If you have a domain smaller than that size, the Logical Domains Manager will automatically boost the size of the domain to 12 Mbytes. The minimum size restriction for a Fujitsu M10 server is 256 Mbytes. Refer to the release notes for your system firmware for information about memory size requirements.

The memory dynamic reconfiguration (DR) feature enforces 256-Mbyte alignment on the address and size of the memory involved in a given operation. See “[Memory Alignment](#)” on page 297.

▼ How to Install the Oracle Solaris OS on a Guest Domain From a DVD

1. **Insert the Oracle Solaris 10 OS or Oracle Solaris 11 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 example 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 options=ro /dev/dsk/c0t0d0s2 dvd_vol@primary-vds0
primary# ldm add-vdisk vdisk_cd_media dvd_vol@primary-vds0 ldg1
```

5. Verify 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      8       8G       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  -----  60      6G
...
DISK
NAME      VOLUME      TOUT   DEVICE   SERVER
vdisk1    vol1@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 127.0.0.1...
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.

▼ How to Install the Oracle Solaris OS on a Guest Domain From an 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 example 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. Verify 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      8      8G        0.2%  22h 45m
...
VDS
  NAME          VOLUME          OPTIONS          DEVICE
  primary-vds0    vol1           /dev/dsk/c2t1d0s2
  iso_vol          iso_vol@primary-vds0
...
-----
NAME          STATE   FLAGS   CONS   VCPU   MEMORY   UTIL   UPTIME
ldg1         inactive  -----   60     6G
...
DISK
  NAME          VOLUME          TOUT  ID  DEVICE  SERVER  MPGROUP
  vdisk1        vol1@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 '^]'.

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.

▼ **How to Use the Oracle Solaris JumpStart Feature on an Oracle Solaris 10 Guest Domain**

Note - The Oracle Solaris JumpStart feature is available only for the Oracle Solaris 10 OS. See “[Oracle Solaris 10 1/13 Installation Guide: JumpStart Installations](#)”.

To perform an automated installation of the Oracle Solaris 11 OS, you can use the Automated Installer (AI) feature. See “[Transitioning From Oracle Solaris 10 to Oracle Solaris 11.2](#)”.

- **Modify your JumpStart profile to reflect the different disk device name format for the guest domain.**

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.

Note - 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 172.

- **JumpStart profile for installing a UFS root file system.**

See “[Oracle Solaris 10 1/13 Installation Guide: JumpStart Installations](#)”.

Normal UFS Profile

```
filesys c1t1d0s0 free /
filesys c1t1d0s1 2048 swap
filesys c1t1d0s5 120 /spare1
filesys c1t1d0s6 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 1/13 Installation Guide: JumpStart Installations”](#).

Normal ZFS Profile

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

Actual ZFS Profile for Installing a Domain

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


Configuring I/O Domains

This chapter describes I/O domains and how to configure them in an Oracle VM Server for SPARC environment.

This chapter covers the following topics:

- “[I/O Domain Overview](#)” on page 61
- “[General Guidelines for Creating an I/O Domain](#)” on page 62

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, a PCIe endpoint device, or a PCIe SR-IOV virtual function to a domain. Use the `ldm add-io` command to assign a bus, device, or virtual function 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 guest domains.

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

- [Chapter 6, “Creating a Root Domain by Assigning PCIe Buses”](#)
- [Chapter 7, “Creating an I/O Domain by Using Direct I/O”](#)
- [Chapter 8, “Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions”](#)
- [Chapter 9, “Using Non-primary Root Domains”](#)

Note - You cannot migrate a domain that has PCIe buses, PCIe endpoint devices, or SR-IOV virtual functions. For information about other migration limitations, see [Chapter 12, “Migrating Domains”](#).

General Guidelines for Creating an I/O Domain

An I/O domain might have direct access to one or more I/O devices, such as PCIe buses, network interface units (NIUs), PCIe endpoint devices, and PCIe single root I/O virtualization (SR-IOV) virtual functions.

This type of direct access to I/O devices means that more I/O bandwidth is available to provide the following:

- Services to the applications in the I/O domain
- Virtual I/O services to guest domains

The following basic guidelines enable you to effectively use the I/O bandwidth:

- Assign CPU resources at the granularity of CPU cores. Assign one or more CPU cores based on the type of I/O device and the number of I/O devices in the I/O domain.
For example, a 1-Gbps Ethernet device might require fewer CPU cores to use the full bandwidth compared to a 10-Gbps Ethernet device.
- Abide by memory requirements. Memory requirements depend on the type of I/O device that is assigned to the domain. A minimum of 4 Gbytes is recommended per I/O device. The more I/O devices you assign, the more memory you must allocate.
- When you use the PCIe SR-IOV feature, follow the same guidelines for each SR-IOV virtual function that you would use for other I/O devices. So, assign one or more CPU cores and memory (in Gbytes) to fully use the bandwidth that is available from the virtual function.

Note that creating and assigning a large number of virtual functions to a domain that does not have sufficient CPU and memory resources is unlikely to produce an optimal configuration.

SPARC systems, up to and including the SPARC T5 and SPARC M6 platforms, provide a finite number of interrupts, so Oracle Solaris limits the number of interrupts that each device can use. The default limit should match the needs of a typical system configuration but you might need to adjust this value for certain system configurations. For more information, see “[Adjusting the Interrupt Limit](#)” on page 352.

◆ ◆ ◆ C H A P T E R 6

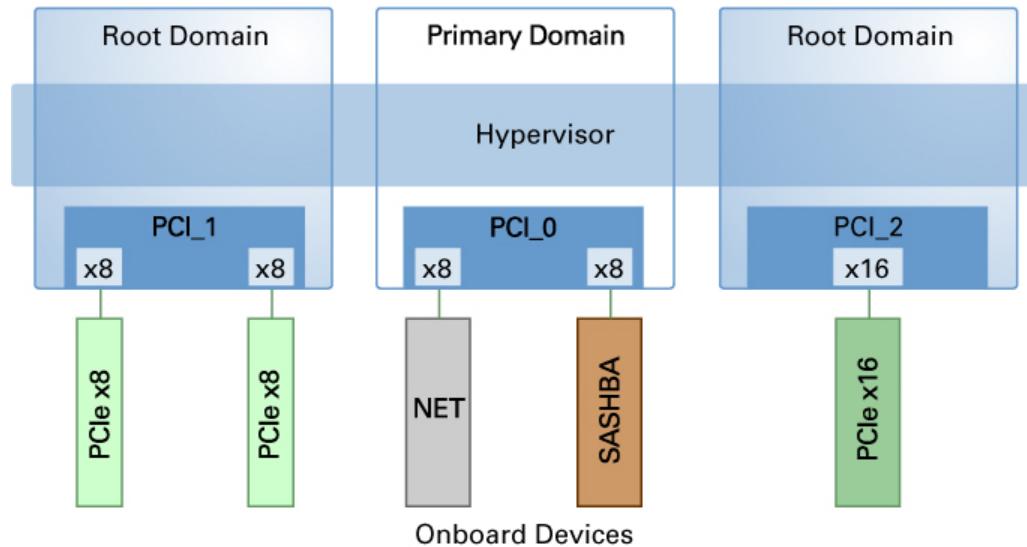
Creating a Root Domain by Assigning PCIe Buses

This chapter describes how to create a root domain by assigning PCIe buses.

Creating a Root Domain by 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 three root complexes, `pci_0`, `pci_1`, and `pci_2`.

FIGURE 6-1 Assigning a PCIe Bus to a Root Domain

The maximum number of root domains that you can create with PCIe buses depends on the number of PCIe buses that are available on the server. Use the `ldm list-io` to determine the number of PCIe buses available on your system.

When you assign a PCIe bus to a root domain, all devices on that bus are owned by that root. You can assign any of the PCIe endpoint devices on that bus to other domains.

When a server is initially configured in an Oracle VM Server for SPARC environment or is using the `factory-default` configuration, the primary domain has access to all the physical device resources. Therefore, the primary domain is the only root domain configured on the system and it owns all the PCIe buses.

Static PCIe Bus Assignment

The static PCIe bus assignment method for a root domain requires you to initiate a delayed reconfiguration on the root domain when assigning or removing a PCIe bus. When you intend to use this method for a domain that does not yet own a PCIe bus, you must stop the domain before you assign the PCIe bus. After you complete the configuration steps on the root domain, you must reboot it. You must use the static method when the Oracle VM Server for SPARC 3.2

firmware is not installed in the system or when the OS version that is installed in the respective domain does not support dynamic PCIe bus assignment.

While the root domain is stopped or in delayed reconfiguration, you can run one or more of the `ldm add-io` and `ldm remove-io` commands before you reboot the root domain. To minimize domain downtime, plan ahead before assigning or removing PCIe buses.

- For root domains, both primary and non-primary, use delayed reconfiguration. After you have added or removed the PCIe buses, reboot the root domain to make the changes take effect.

```
primary# ldm start-reconf root-domain
Add or remove the PCIe bus by using the ldm add-io or ldm remove-io command
primary# ldm stop -r domain-name
```

Note that you can use delayed reconfiguration only if the domain already owns a PCIe bus.

- For non-root domains, stop the domain and then add or remove the PCIe bus.

```
primary# ldm stop domain-name
Add or remove the PCIe bus by using the ldm add-io or ldm remove-io command
primary# ldm start domain-name
```

Dynamic PCIe Bus Assignment

The dynamic PCIe bus assignment feature enables you to dynamically assign or remove a PCIe bus from a root domain.

The dynamic PCIe bus assignment feature is enabled when your system runs the required firmware and software. See “[Dynamic PCIe Bus Assignment Requirements](#)” on page 65. If your system does not run the required firmware and software, the `ldm add-io` and `ldm remove-io` commands fail gracefully.

When enabled, you can run the `ldm add-io` and `ldm remove-io` commands without stopping the root domain or putting the root domain in delayed reconfiguration.

Dynamic PCIe Bus Assignment Requirements

To use the dynamic PCIe bus assignment feature, you must meet the software and firmware requirements for the Oracle VM Server for SPARC 3.2 release.

This feature is supported on SPARC M5, SPARC M6, and Fujitsu M10 platforms.

▼ How to Create a Root Domain by Assigning a PCIe Bus

This example procedure shows how to create a new root 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 SPARC T4-2 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.

Ensure that you do not remove the PCIe buses that host the boot disk and primary network interface from the primary 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.

Ensure that you do not remove a bus that has 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 on-board network port or a PCIe card from a different PCIe slot.

On certain SPARC servers, you can remove a PCIe bus that contains graphics controllers and other devices. However, you cannot add such a PCIe bus to any other domain. Such PCIe buses can only be added to the primary domain.

In this example, the primary domain uses only a ZFS pool (`rpool`) and network interface (`igb0`). 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.

You can add a bus to or remove a bus from a domain by using its device path (`pci@nnn`) or its pseudonym (`pci_n`). The `ldm list-bindings primary` or `ldm list -l -o physio primary` command shows the following:

- `pci@400` corresponds to `pci_0`
- `pci@500` corresponds to `pci_1`
- `pci@600` corresponds to `pci_2`
- `pci@700` corresponds to `pci_3`

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

```
primary# ldm list-io
NAME          TYPE   BUS      DOMAIN  STATUS
---          ---    ---      ---     ---
pci_0         BUS    pci_0    primary  -
```

pci_1	BUS	pci_1	primary
pci_2	BUS	pci_2	primary
pci_3	BUS	pci_3	primary
/SYS/MB/PCIE1	PCIE	pci_0	primary EMP
/SYS/MB/SASHBA0	PCIE	pci_0	primary OCC
/SYS/MB/NET0	PCIE	pci_0	primary OCC
/SYS/MB/PCIE5	PCIE	pci_1	primary EMP
/SYS/MB/PCIE6	PCIE	pci_1	primary EMP
/SYS/MB/PCIE7	PCIE	pci_1	primary EMP
/SYS/MB/PCIE2	PCIE	pci_2	primary EMP
/SYS/MB/PCIE3	PCIE	pci_2	primary EMP
/SYS/MB/PCIE4	PCIE	pci_2	primary EMP
/SYS/MB/PCIE8	PCIE	pci_3	primary EMP
/SYS/MB/SASHBA1	PCIE	pci_3	primary OCC
/SYS/MB/NET2	PCIE	pci_3	primary OCC
/SYS/MB/NET0/IOVNET.PF0	PF	pci_0	primary
/SYS/MB/NET0/IOVNET.PF1	PF	pci_0	primary
/SYS/MB/NET2/IOVNET.PF0	PF	pci_3	primary
/SYS/MB/NET2/IOVNET.PF1	PF	pci_3	primary

2. Determine the device path of the boot disk that must be retained.

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

```
primary# df /
/          (/dev/dsk/c0t5000CCA03C138904d0s0):22755742 blocks  2225374 files
```

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

```
primary# zpool status rpool
  pool: rpool
    state: ONLINE
      scan: none requested
    config:

      NAME                STATE     READ WRITE CKSUM
      rpool              ONLINE      0      0      0
      c0t5000CCA03C138904d0s0  ONLINE      0      0      0
```

3. Obtain information about the system's boot disk.

- For a disk that is managed with Solaris I/O multipathing, determine the PCIe bus under which the boot disk is connected by using the **mpathadm** command.

Starting with the SPARC T3 servers, the internal disks are managed by Solaris I/O multipathing.

- a. Find the initiator port to which the disk is connected.

```
primary# mpathadm show lu /dev/rdsk/c0t5000CCA03C138904d0s0
Logical Unit: /dev/rdsk/c0t5000CCA03C138904d0s2
  mpath-support: libmpscsi_vhci.so
  Vendor: HITACHI
  Product: H106030SDSUN300G
  Revision: A2B0
  Name Type: unknown type
  Name: 5000cca03c138904
  Asymmetric: no
  Current Load Balance: round-robin
  Logical Unit Group ID: NA
  Auto Failback: on
  Auto Probing: NA

  Paths:
    Initiator Port Name: w50800200014100c8
    Target Port Name: w5000cca03c138905
    Override Path: NA
    Path State: OK
    Disabled: no

  Target Ports:
    Name: w5000cca03c138905
    Relative ID: 0
```

b. Determine the PCIe bus on which the initiator port is present.

```
primary# mpathadm show initiator-port w50800200014100c8
Initiator Port: w50800200014100c8
  Transport Type: unknown
  OS Device File: /devices/pci@400/pci@2/pci@0/pci@e/scsi@0/iport@1
```

- **For a disk that is not managed with Solaris I/O multipathing, determine the physical device to which the block device is linked by using the `ls -l` command.**

Use this command for a disk on an UltraSPARC T2 or UltraSPARC T2 Plus system that is not managed with Solaris I/O multipathing.

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 the `pci@400` bus.

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

Identify the primary network interface that is “plumbed” by using the `ifconfig` command. A plumbed interface has streams set up so that the IP protocol can use the device.

■ **Oracle Solaris 10:**

```
primary# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
igb0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 3
    inet 10.129.241.135 netmask ffffff00 broadcast 10.129.241.255
        ether 0:10:e0:e:f1:78
```

■ **Oracle Solaris 11:**

```
primary# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
    inet 127.0.0.1 netmask ff000000
net0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 3
    inet 10.129.241.135 netmask ffffff00 broadcast 10.129.241.255
        ether 0:10:e0:e:f1:78

primary# dladm show-phys net0
LINK          MEDIA           STATE      SPEED DUPLEX   DEVICE
net0          Ethernet         up        1000   full     igb0
```

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

The following command uses the `igb0` network interface:

```
primary# ls -l /dev/igb0
lrwxrwxrwx  1 root      root          46 Oct  1 10:39 /dev/igb0  ->
./devices/pci@500/pci@0/pci@c/network@0:igb0
```

Perform the `ls -l /dev/usbecm` command, as well.

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 is on a bus that you want to assign to another domain, reconfigure the primary domain to use a different network interface.

6. Remove a bus that does not contain the boot disk or the network interface from the primary domain.

In this example, the `pci_2` bus is being removed from the primary domain.

■ **Dynamic method:**

Ensure that the devices in the `pci_2` bus are not in use by the primary domain OS. If they are, this command might fail to remove the bus. Use the static method to forcibly remove the `pci_2` bus.

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

■ **Static method:**

Before you remove the bus, you must initiate a delayed reconfiguration.

```
primary# ldm start-reconf primary
primary# ldm remove-io pci_2 primary
primary# shutdown -y -g0 -i6
```

The bus that the primary domain uses for the boot disk and the network device cannot be assigned to other domains. You can assign any of the other buses to another domain. In this example, the pci@600 is not used by the primary domain, so you can reassign it to another domain.

7. Add a bus to a domain.

In this example, you add the pci_2 bus to the ldg1 domain.

■ **Dynamic method:**

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

■ **Static method:**

Before you add the bus, you must stop the domain.

```
primary# ldm stop-domain ldg1
primary# ldm add-io pci_2 ldg1
primary# ldm start-domain ldg1
```

8. 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.

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

```
primary# ldm list-io
NAME                      TYPE   BUS      DOMAIN  STATUS
----                      ----   ---      -----  -----
pci_0                     BUS    pci_0    primary
pci_1                     BUS    pci_1    primary
pci_2                     BUS    pci_2    ldg1
pci_3                     BUS    pci_3    primary
/SYS/MB/PCIE1              PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA0             PCIE   pci_0    primary  OCC
/SYS/MB/NET0                PCIE   pci_0    primary  OCC
```

```
/SYS/MB/PCIE5          PCIE  pci_1    primary  EMP
/SYS/MB/PCIE6          PCIE  pci_1    primary  EMP
/SYS/MB/PCIE7          PCIE  pci_1    primary  EMP
/SYS/MB/PCIE2          PCIE  pci_2    ldg1     EMP
/SYS/MB/PCIE3          PCIE  pci_2    ldg1     EMP
/SYS/MB/PCIE4          PCIE  pci_2    ldg1     EMP
/SYS/MB/PCIE8          PCIE  pci_3    primary  EMP
/SYS/MB/SASHBA1        PCIE  pci_3    primary  OCC
/SYS/MB/NET2           PCIE  pci_3    primary  OCC
/SYS/MB/NET0/IOVNET.PF0 PF    pci_0    primary
/SYS/MB/NET0/IOVNET.PF1 PF    pci_0    primary
/SYS/MB/NET2/IOVNET.PF0 PF    pci_3    primary
/SYS/MB/NET2/IOVNET.PF1 PF    pci_3    primary
```

This output confirms that PCIe buses `pci_0`, `pci_1`, and `pci_3` and their devices are assigned to the `primary` domain. It also confirms that PCIe bus `pci_2` and its devices are assigned to the `ldg1` domain.

Creating an I/O Domain by Using Direct I/O

This chapter covers the following direct I/O topics:

- “[Creating an I/O Domain by Assigning PCIe Endpoint Devices](#)” on page 73
- “[Direct I/O Hardware and Software Requirements](#)” on page 75
- “[Current Direct I/O Feature Limitations](#)” on page 76
- “[Planning PCIe Endpoint Device Configuration](#)” on page 77
- “[Rebooting the Root Domain With PCIe Endpoints Configured](#)” on page 78
- “[Making PCIe Hardware Changes](#)” on page 80
- “[Creating an I/O Domain by Assigning a PCIe Endpoint Device](#)” on page 82

Creating an I/O Domain by Assigning PCIe Endpoint Devices

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

Note - Because root domains cannot have dependencies on other root domains, a root domain that owns a PCIe bus cannot have its PCIe endpoint devices or SR-IOV virtual functions assigned to another root domain. However, you *can* assign a PCIe endpoint device or virtual function from a PCIe bus to the root domain that owns that bus.

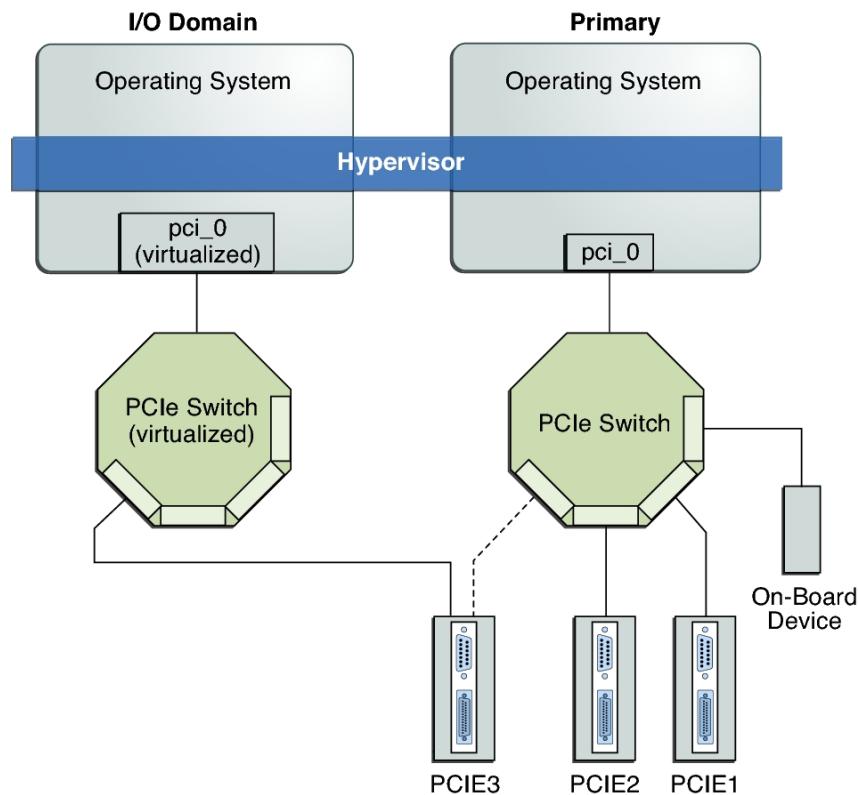
The following diagram shows that the PCIe endpoint device, PCIE3, is assigned to an I/O domain. Both bus `pci_0` and the switch in the I/O domain are virtual. The `PCIE3` endpoint device is no longer accessible in the primary domain.

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 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`.



Caution - You cannot use Oracle Solaris hot-plug operations to hot-remove a PCIe endpoint device after that device is removed from the primary domain by using the `ldm remove-io` command. For information about replacing or removing a PCIe endpoint device, see “[Making PCIe Hardware Changes](#)” on page 80.

FIGURE 7-1 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](#)” on page 75.



Caution - PCIe cards that have a 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 items describe 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](#)” on page 75.
- Only PCIe endpoints that are connected to a PCIe bus assigned to a root domain can be assigned to another domain with the DIO feature.
- I/O domains that use DIO have access to the PCIe endpoint devices only when the root domain is running.
- Rebooting the root domain affects I/O domains that have PCIe endpoint devices. See “[Rebooting the Root Domain With PCIe Endpoints Configured](#)” on page 78. The root domain also performs the following tasks:
 - Initializes and manages the PCIe 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 direct I/O (DIO) feature to assign direct I/O devices to domains, you must run the appropriate software and use supported PCIe cards.

- **Hardware Requirements.** Only certain PCIe cards can be used as a direct I/O endpoint device on an I/O domain. You can still use other cards in your Oracle VM Server for SPARC environment but they cannot be used with the DIO feature. Instead, they can be used for service domains and for I/O domains that have entire root complexes assigned to them.
Refer to your platform's hardware documentation to verify which cards can be used on your platform. For an up-to-date list of supported PCIe cards, see <https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&id=1325454.1>.
- **Software Requirements.** To use the DIO feature, the following domains must run the supported OS:

- **Root domain.** At least the Oracle Solaris 10 9/10 OS plus patch ID 145868-01 or the Oracle Solaris 11 OS.
The recommended practice is for all domains to run at least the Oracle Solaris 10 1/13 OS plus the required patches in <http://www.oracle.com/pls/topic/lookup?ctx=E48724&id=LDSIGreqdrecommendedsolarisos> or the Oracle Solaris 11.1.10.5.0 OS.
- **I/O domain.** Any Oracle Solaris OS that is supported by the platform.

Note - All PCIe cards that are supported on a platform are supported in the root domains. 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.

To add or remove PCIe endpoint devices by using the direct I/O feature, you must first enable I/O virtualization on the PCIe bus itself.

You can use the `ldm set-io` or `ldm add-io` command to set the `iov` property to `on`. You can also use the `ldm add-domain` or `ldm set-domain` command to set the `rc-add-policy` property to `iov`. See the [ldm\(1M\)](#) man page.

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

Current Direct I/O Feature Limitations

For information about how to work around the limitations, see “[Planning PCIe Endpoint Device Configuration](#)” on page 77.

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

Note - The Fujitsu M10 server supports the dynamic reconfiguration of PCIe endpoint devices. You can assign or remove PCIe endpoint devices without rebooting the root domain or stopping the I/O domain.

For up-to-date information about this feature, see *Fujitsu M10/SPARC M10 Systems System Operation and Administration Guide* for your model at <http://www.fujitsu.com/global/services/computing/server/sparc/downloads/manual/>.

SPARC systems, up to and including the SPARC T5 and SPARC M6 platforms, provide a finite number of interrupts, so Oracle Solaris limits the number of interrupts that each device can use.

The default limit should match the needs of a typical system configuration but you might need to adjust this value for certain system configurations. For more information, see “[Adjusting the Interrupt Limit](#)” on page 352.

Planning PCIe Endpoint Device Configuration

Carefully plan ahead when you assign or remove PCIe endpoint devices to avoid root domain downtime. The reboot of the root domain not only affects the services that are available on the root domain itself but also 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.

When in a delayed reconfiguration, you can continue to add or remove more devices and then reboot the root domain only one time to make all the changes take effect.

For an example, see “[How to Create an I/O Domain by Assigning a PCIe Endpoint Device](#)” on page 82.

You must take the following general steps 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 `prttdiag -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 root domain, because reboots affect I/O domains.

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

Note - After you have removed all the devices you want during a delayed reconfiguration, you need to reboot the root domain only 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 root 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 root 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 a PCIe endpoint device to or remove one 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 80.

Rebooting the Root Domain With PCIe Endpoints Configured

The root domain is the owner of the PCIe bus and is responsible for initializing and managing the bus. The root domain must be active and running a version of the Oracle Solaris OS that supports the DIO or SR-IOV feature. Shutting down, halting, or rebooting the root 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 root 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 root domain, you would need to manually stop and start each domain.

Note - An I/O domain cannot start if the associated root domain is not running.

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 root domain.

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

```
primary# ldm set-domain failure-policy=reset primary
primary# ldm set-domain master=primary domain-name
```

EXAMPLE 7-1 Configuring Failure Policy Dependencies for a Configuration With a Non-primary Root Domain and I/O Domains

The following example describes how you can configure failure policy dependencies in a configuration that has a non-primary root domain and I/O domains.

In this example, `ldg1` is a non-primary root domain. `ldg2` is an I/O domain that has either PCIe SR-IOV virtual functions or PCIe endpoint devices assigned from a root complex that is owned by the `ldg1` domain.

```
primary# ldm set-domain failure-policy=stop ldg1
primary# ldm set-domain master=ldg1 ldg2
```

This dependency relationship ensures that the I/O domain is stopped when the `ldg1` root domain reboots.

- If it is the non-primary root domain rebooting, this dependency relationship ensures that the I/O domain is stopped. Start the I/O domain after the non-primary root domain boots.

primary# ldm start ldg2
- If it is the primary domain rebooting, this policy setting stops both the non-primary root domain and the dependent I/O domains. When the primary domain boots, you must start the non-primary root domain first. When the domain boots, start the I/O domain.

```
primary# ldm start ldg1
```

Wait for the `ldg1` domain to become active and then start the I/O domain.

```
primary# ldm start ldg2
```

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 remove-io` command to first remove the card from the root 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 root 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 root 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 remove-io` command. Next, use the `ldm add-io` command to reassign that PCIe card to the root domain. Then, physically replace the PCIe card you assigned to the root domain with a different PCIe card. These steps enable you to avoid a configuration that is unsupported by the DIO feature.

Minimizing Guest Domain Outages When Removing a PCIe Card

While you remove or replace a PCIe card from a system that runs the Oracle VM Server for SPARC software, the domains that depend on this hardware are unavailable. To minimize such guest domain outages, you must prepare your system to use the hotplug capabilities to physically remove the card.

▼ How to Minimize Guest Domain Outages When Removing a PCIe Card

This procedure enables you to avoid an outage to a guest domain that does not have direct I/O or SR-IOV device assigned to it and that has multiple paths configured. Note that this procedure requires two reboots of the primary domain.

Note - This procedure does not apply when the PCIe card is on a root complex owned by a non-primary root domain. Instead, see [How to Replace PCIe Direct I/O Cards Assigned to an Oracle VM Server for SPARC Guest Domain \(Doc ID 1684273.1\) \(\[https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=226878266536565&id=1684273.1&_adf.ctrl-state=bo9fbmr1n_49\]\(https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=226878266536565&id=1684273.1&_adf.ctrl-state=bo9fbmr1n_49\)\)](#).

- 1. Stop the guest domain that has the PCIe slot assigned to it.**

```
primary# ldm stop domain-name
```

- 2. Remove the PCIe slot from the guest domain.**

```
primary# ldm remove-io PCIe-slot domain-name
```

- 3. Stop the guest domains that have PCIe slots and SR-IOV virtual functions assigned to them.**

```
primary# ldm stop domain-name
```

Note - You do not need to stop guest domains that have PCIe buses assigned to them because they might be providing alternate paths to network and disk devices to the guest domains.

- 4. Initiate a delayed reconfiguration on the primary domain so that you can assign this slot to it.**

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

- 5. Add the PCIe slot to the primary domain.**

```
primary# ldm add-io PCIe-slot domain-name
```

- 6. Reboot the primary domain.**

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

- 7. Use the hotplug commands to replace the PCIe card.**

For information about Oracle Solaris OS hotplug capabilities, see [Chapter 2, “Dynamically Configuring Devices,” in “Managing Devices in Oracle Solaris 11.2”](#).

- 8. After the card is replaced, perform the following steps if you must reassign this same PCIe slot to the guest domain:**

- a. Initiate a delayed reconfiguration on the primary domain.**

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

- b. Remove the PCIe slot from the primary domain.**

```
primary# ldm remove-io PCIe-slot domain-name
```

- c. **Reboot the primary domain to cause the removal of the PCIe slot to take effect.**

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

- d. **Reassign the PCIe slot to the guest domain.**

```
primary# ldm add-io PCIe-slot domain-name
```

- e. **Start the guest domains to which you want to assign PCIe slots and SR-IOV virtual functions.**

```
primary# ldm start domain-name
```

Creating an I/O Domain by Assigning a PCIe Endpoint Device

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

Plan all DIO deployments ahead of time to minimize downtime.



Caution - The primary domain loses access to the on-board DVD device if you assign the /SYS/MB/SASHBA1 slot on a SPARC T3-1 or a SPARC T4-1 system to a DIO domain.

The SPARC T3-1 and SPARC T4-1 systems include two DIO slots for on-board storage, which are represented by the /SYS/MB/SASHBA0 and /SYS/MB/SASHBA1 paths. In addition to hosting multiheaded on-board disks, the /SYS/MB/SASHBA1 slot hosts the on-board DVD device. So, if you assign /SYS/MB/SASHBA1 to a DIO domain, the primary domain loses access to the on-board DVD device.

The SPARC T3-2 and SPARC T4-2 systems have a single SASHBA slot that hosts all on-board disks as well as the on-board DVD device. So, if you assign SASHBA to a DIO domain, the on-board disks *and* the on-board DVD device are loaned to the DIO domain and unavailable to the primary domain.

For an example of adding a PCIe endpoint device to create an I/O domain, see “[Planning PCIe Endpoint Device Configuration](#)” on page 77.

Note - In this release, use the `DefaultFixed` NCP to configure datalinks and network interfaces on Oracle Solaris 11 systems.

The Oracle Solaris 11 OS includes the following NCPs:

- `DefaultFixed` – Enables you to use the `dladm` or `ipadm` command to manage networking
- `Automatic` – Enables you to use the `netcfg` or `netadm` command to manage networking

Ensure that the `DefaultFixed` NCP is enabled by using the `netadm list` command. See [Chapter 7, “Using Datalink and Interface Configuration Commands on Profiles,” in “Oracle Solaris Administration: Network Interfaces and Network Virtualization”](#).

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 be determined only in the I/O domains.

```
primary# ldm list-io -l
NAME                      TYPE   BUS      DOMAIN  STATUS
----                      ---    ---      -----  -----
niu_0                     NIU    niu_0   primary
[niu@480]
niu_1                     NIU    niu_1   primary
[niu@580]
pci_0                     BUS    pci_0   primary
[pci@400]
pci_1                     BUS    pci_1   primary
[pci@500]
/SYS/MB/PCIE0              PCIE   pci_0   primary  OCC
[pci@400/pci@2/pci@0/pci@8]
    SUNW,emlx@0/fp/disk
    SUNW,emlx@0/fp/tape
    SUNW,emlx@0/fp@0,0
    SUNW,emlx@0,1/fp/disk
    SUNW,emlx@0,1/fp/tape
    SUNW,emlx@0,1/fp@0,0
/SYS/MB/PCIE2              PCIE   pci_0   primary  OCC
[pci@400/pci@2/pci@0/pci@4]
    pci/scsi/disk
    pci/scsi/tape
    pci/scsi/disk
    pci/scsi/tape
/SYS/MB/PCIE4              PCIE   pci_0   primary  OCC
[pci@400/pci@2/pci@0/pci@0]
    ethernet@0
    ethernet@0,1
    SUNW,qlc@0,2/fp/disk
```

```
SUNW,qlc@0,2/fp@0,0
SUNW,qlc@0,3/fp/disk
SUNW,qlc@0,3/fp@0,0
/SYS/MB/PCIE6
[pci@400/pci@1/pci@0/pci@8]                                PCIE  pci_0    primary  EMP
/SYS/MB/PCIE8
[pci@400/pci@1/pci@0/pci@c]                                PCIE  pci_0    primary  EMP
/SYS/MB/SASHBA
[pci@400/pci@2/pci@0/pci@e]                                PCIE  pci_0    primary  OCC
          scsi@0/iport@1
          scsi@0/iport@2
          scsi@0/iport@4
          scsi@0/iport@8
          scsi@0/iport@80/cdrom@p7,0
          scsi@0/iport@v0
/SYS/MB/NET0
[pci@400/pci@1/pci@0/pci@4]                                PCIE  pci_0    primary  OCC
          network@0
          network@0,1
/SYS/MB/PCIE1
[pci@500/pci@2/pci@0/pci@a]                                PCIE  pci_1    primary  OCC
          SUNW,qlc@0/fp/disk
          SUNW,qlc@0/fp@0,0
          SUNW,qlc@0,1/fp/disk
          SUNW,qlc@0,1/fp@0,0
/SYS/MB/PCIE3
[pci@500/pci@2/pci@0/pci@6]                                PCIE  pci_1    primary  OCC
          network@0
          network@0,1
          network@0,2
          network@0,3
/SYS/MB/PCIE5
[pci@500/pci@2/pci@0/pci@0]                                PCIE  pci_1    primary  OCC
          network@0
          network@0,1
          network@0,1
/SYS/MB/PCIE7
[pci@500/pci@1/pci@0/pci@6]                                PCIE  pci_1    primary  EMP
/SYS/MB/PCIE9
[pci@500/pci@1/pci@0/pci@0]                                PCIE  pci_1    primary  EMP
/SYS/MB/NET2
[pci@500/pci@1/pci@0/pci@5]                                PCIE  pci_1    primary  OCC
          network@0
          network@0,1
          ethernet@0,80
/SYS/MB/NET0/IOVNET.PF0
[pci@400/pci@1/pci@0/pci@4/network@0]                      PF    pci_0    primary
          maxvfs = 7
/SYS/MB/NET0/IOVNET.PF1
[pci@400/pci@1/pci@0/pci@4/network@0,1]                    PF    pci_0    primary
          maxvfs = 7
/SYS/MB/PCIE5/IOVNET.PF0
[pci@500/pci@2/pci@0/pci@0/network@0]                      PF    pci_1    primary
          maxvfs = 63
/SYS/MB/PCIE5/IOVNET.PF1
[pci@500/pci@2/pci@0/pci@0/network@0]                      PF    pci_1    primary
```

```
[pci@500/pci@2/pci@0/pci@0/network@0,1]
    maxvfs = 63
/SYS/MB/NET2/IOVNET.PF0          PF      pci_1      primary
[pci@500/pci@1/pci@0/pci@5/network@0]
    maxvfs = 7
/SYS/MB/NET2/IOVNET.PF1          PF      pci_1      primary
[pci@500/pci@1/pci@0/pci@5/network@0,1]
    maxvfs = 7
```

2. Determine the device path of the boot disk that must be retained.

See Step 2 in “[How to Create a Root Domain by Assigning a PCIe Bus](#)” on page 66.

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

See Step 3 in “[How to Create a Root Domain by Assigning a PCIe Bus](#)” on page 66.

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

See Step 4 in “[How to Create a Root Domain by Assigning a PCIe Bus](#)” on page 66.

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

The following command uses the `igb0` network interface:

```
primary# ls -l /dev/igb0
lrwxrwxrwx 1 root      root        46 Jul 30 17:29 /dev/igb0 ->
./devices/pci@500/pci@0/pci@8/network@0:igb0
```

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 or required by the **primary** domain. Do not remove a bus that has devices (such as network ports) that are used by a domain.

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

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

```
primary# ldm start-reconf primary
primary# ldm set-io iov=on pci_1
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.
primary# ldm remove-io /SYS/MB/PCIE1 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.
-----
primary# ldm remove-io /SYS/MB/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.
-----
primary# ldm remove-io /SYS/MB/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:

```
primary# ldm add-config dio
```

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

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

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

```
primary# ldm list-io
NAME                      TYPE   BUS    DOMAIN  STATUS
----                      ---   ---    -----  -----
niu_0                     NIU    niu_0  primary
niu_1                     NIU    niu_1  primary
pci_0                     BUS    pci_0  primary
pci_1                     BUS    pci_1  primary  IOV
/SYS/MB/PCIE0              PCIE   pci_0  primary  OCC
/SYS/MB/PCIE2              PCIE   pci_0  primary  OCC
/SYS/MB/PCIE4              PCIE   pci_0  primary  OCC
/SYS/MB/PCIE6              PCIE   pci_0  primary  EMP
/SYS/MB/PCIE8              PCIE   pci_0  primary  EMP
/SYS/MB/SASHBA             PCIE   pci_0  primary  OCC
/SYS/MB/NET0               PCIE   pci_0  primary  OCC
/SYS/MB/PCIE1              PCIE   pci_1  primary  OCC
/SYS/MB/PCIE3              PCIE   pci_1  primary  OCC
```

```

/SYS/MB/PCIE5          PCIE  pci_1      OCC
/SYS/MB/PCIE7          PCIE  pci_1      primary EMP
/SYS/MB/PCIE9          PCIE  pci_1      primary EMP
/SYS/MB/NET2           PCIE  pci_1      primary OCC
/SYS/MB/NET0/IOVNET.PF0 PF   pci_0      primary
/SYS/MB/NET0/IOVNET.PF1 PF   pci_0      primary
/SYS/MB/NET2/IOVNET.PF0 PF   pci_1      primary
/SYS/MB/NET2/IOVNET.PF1 PF   pci_1      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 PCIE3 device to the `ldg1` domain.

```
primary# ldm add-io /SYS/MB/PCIE3 ldg1
```

b. Bind and start the `ldg1` domain.

```
primary# ldm bind ldg1
primary# ldm start ldg1
LDom ldg1 started
```

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

Verify that the network device is available and then configure the network device for use in the domain.

- Oracle Solaris 10 OS: Run the following command:

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

- Oracle Solaris 11 OS: Run the following command:

```
primary# dladm show-phys
LINK        MEDIA          STATE     SPEED  DUPLEX  DEVICE
net0        Ethernet       unknown   0       unknown  nxge0
net1        Ethernet       unknown   0       unknown  nxge1
net2        Ethernet       unknown   0       unknown  nxge2
net3        Ethernet       unknown   0       unknown  nxge3
```


Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions

This chapter covers the following PCIe SR-IOV topics:

- “SR-IOV Overview” on page 89
- “SR-IOV Hardware and Software Requirements” on page 92
- “Current SR-IOV Feature Limitations” on page 95
- “Static SR-IOV” on page 96
- “Dynamic SR-IOV” on page 97
- “Enabling I/O Virtualization” on page 98
- “Planning for the Use of PCIe SR-IOV Virtual Functions” on page 99
- “Using Ethernet SR-IOV Virtual Functions” on page 100
- “Using InfiniBand SR-IOV Virtual Functions” on page 120
- “Using Fibre Channel SR-IOV Virtual Functions” on page 134
- “I/O Domain Resiliency” on page 149
- “Rebooting the Root Domain With Non-Resilient I/O Domains Configured” on page 155

SR-IOV Overview

Note - Because root domains cannot have dependencies on other root domains, a root domain that owns a PCIe bus cannot have its PCIe endpoint devices or SR-IOV virtual functions assigned to another root domain. However, you *can* assign a PCIe endpoint device or virtual function from a PCIe bus to the root domain that owns that bus.

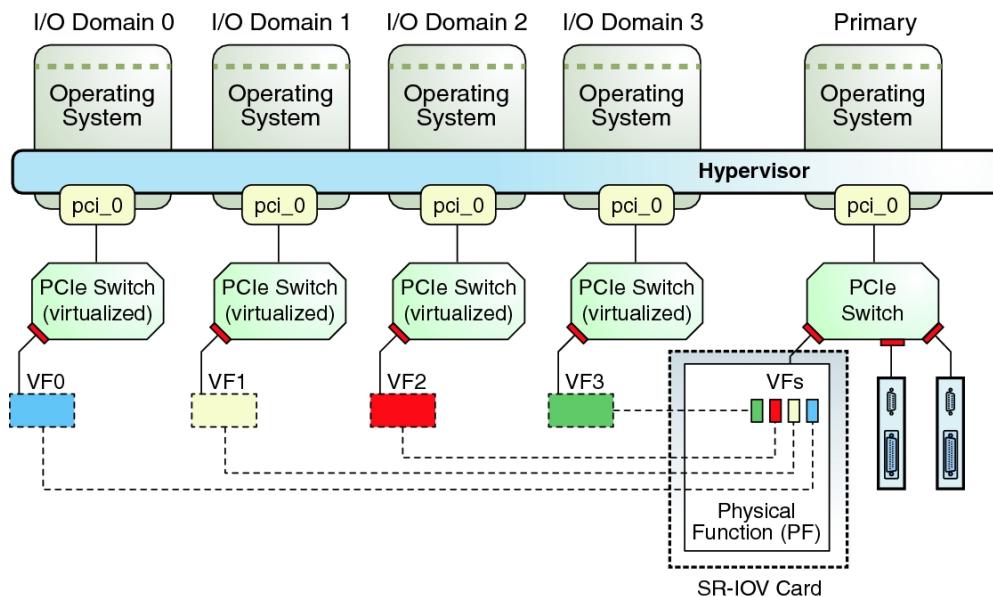
The Peripheral Component Interconnect Express (PCIe) single root I/O virtualization (SR-IOV) implementation is based on version 1.1 of the standard as defined by the PCI-SIG. The SR-IOV standard enables the efficient sharing of PCIe devices among virtual machines and is implemented in the hardware to achieve I/O performance that is comparable to native performance. The SR-IOV specification defines a new standard wherein new devices that are created enable the virtual machine to be directly connected to the I/O device.

A single I/O resource, which is known as a *physical function*, can be shared by many virtual machines. The shared devices provide dedicated resources and also use shared common resources. In this way, each virtual machine has access to unique resources. Therefore, a PCIe device, such as an Ethernet port, that is SR-IOV-enabled with appropriate hardware and OS support can appear as multiple, separate physical devices, each with its own PCIe configuration space.

For more information about SR-IOV, see the [PCI-SIG web site \(<http://www.pcisig.com/>\).](http://www.pcisig.com/)

The following figure shows the relationship between virtual functions and a physical function in an I/O Domain.

FIGURE 8-1 Using Virtual Functions and a Physical Function in an I/O Domain



SR-IOV has the following function types:

- **Physical function** – A PCI function that supports the SR-IOV capabilities as defined by the SR-IOV specification. A physical function contains the SR-IOV capability structure and manages the SR-IOV functionality. Physical functions are fully featured PCIe functions that can be discovered, managed, and manipulated like any other PCIe device. Physical functions can be used to configure and control a PCIe device.
- **Virtual function** – A PCI function that is associated with a physical function. A virtual function is a lightweight PCIe function that shares one or more physical resources with the physical function and with virtual functions that are associated with that physical function. Unlike a physical function, a virtual function can only configure its own behavior.

Each SR-IOV device can have a physical function and each physical function can have up to 256 virtual functions associated with it. This number is dependent on the particular SR-IOV device. The virtual functions are created by the physical function.

After SR-IOV is enabled in the physical function, the PCI configuration space of each virtual function can be accessed by the bus, device, and function number of the physical function. Each virtual function has a PCI memory space, which is used to map its register set. The virtual function device drivers operate on the register set to enable its functionality and the virtual function appears as an actual PCI device. After creation, you can directly assign a virtual function to an I/O domain. This capability enables the virtual function to share the physical device and to perform I/O without CPU and hypervisor software overhead.

You might want to use the SR-IOV feature in your environment to reap the following benefits:

- **Higher performance and reduced latency** – Direct access to hardware from a virtual machines environment
- **Cost reduction** – Capital and operational expenditure savings, which include:
 - Power savings
 - Reduced adapter count
 - Less cabling
 - Fewer switch ports

The Oracle VM Server for SPARC SR-IOV implementation includes both static and dynamic configuration methods. For more information, see “[Static SR-IOV](#)” on page 96 and “[Dynamic SR-IOV](#)” on page 97.

The Oracle VM Server for SPARC SR-IOV feature enables you to perform the following operations:

- Creating a virtual function on a specified physical function
- Destroying a specified virtual function on a physical function
- Assigning a virtual function to a domain
- Removing a virtual function from a domain

To create and destroy virtual functions in the SR-IOV physical function devices, you must first enable I/O virtualization on that PCIe bus. You can use the `ldm set-io` or `ldm add-io` command to set the `iov` property to `on`. You can also use the `ldm add-domain` or `ldm set-domain` command to set the `rc-add-policy` property to `iov`. See the [ldm\(1M\)](#) man page.

Note - On Fujitsu M10 servers, PCIe buses are enabled for I/O virtualization by default.

Assigning a SR-IOV virtual function to a domain creates an implicit dependency on the domain providing the SR-IOV physical function service. You can view these dependencies or view domains that depend on this SR-IOV physical function by using the `ldm list-dependencies` command. See “[Listing Domain I/O Dependencies](#)” on page 354.

SR-IOV Hardware and Software Requirements

The dynamic and static PCIe SR-IOV features are supported on the SPARC T4, SPARC T5, SPARC M5, and SPARC M6 platforms. The dynamic feature is supported on Fujitsu M10 platforms for Ethernet devices only while the other device types require that you use the static method. The SPARC T3 platform supports only the static PCIe SR-IOV feature.

Note - Before you deploy InfiniBand SR-IOV to your Oracle VM Server for SPARC 3.1 environment, review the information in “[InfiniBand SR-IOV Issues](#)” in “[Oracle VM Server for SPARC 3.2 Release Notes](#)”.

- **Hardware Requirements.**

Refer to your platform's hardware documentation to verify which cards can be used on your platform. For an up-to-date list of supported PCIe cards, see <https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&id=1325454.1>.

- **Ethernet SR-IOV.** To use the SR-IOV feature, you can use on-board PCIe SR-IOV devices as well as PCIe SR-IOV plug-in cards. All on-board SR-IOV devices in a given platform are supported unless otherwise explicitly stated in the platform documentation.
- **InfiniBand SR-IOV.** InfiniBand devices are supported on the SPARC T4, SPARC T5, SPARC M5, and SPARC M6 platforms and Fujitsu M10 platforms.
- **Fibre Channel SR-IOV.** Fibre Channel devices are supported on the SPARC T4, SPARC T5, SPARC M5, and SPARC M6 platforms and Fujitsu M10 platforms.

For an up-to-date list of supported devices on Fujitsu M10 platforms, see *Fujitsu M10/SPARC M10 Systems PCI Card Installation Guide* in the product notes for your model at <http://www.fujitsu.com/global/services/computing/server/sparc/downloads/manual/>

- **Firmware Requirements.**

- **Ethernet SR-IOV.** To use the dynamic SR-IOV feature, SPARC T4 systems must run at least version 8.4.0.a of the system firmware. SPARC T5, SPARC M5, and SPARC M6 platforms must run at least version 9.1.0.a of the system firmware. Fujitsu M10 servers must run at least version XCP2210 of the system firmware. The SPARC T3 platform supports only the static SR-IOV feature.

To use the SR-IOV feature, PCIe SR-IOV devices must run at least device firmware version 3.01. Perform the following steps to update the firmware for the Sun Dual 10-Gigabit Ethernet SFP+ PCIe 2.0 network adapters:

1. Determine whether you need to upgrade the FCode version on the device.

Perform these commands from the ok prompt:

```
{0} ok cd path-to-device  
{0} ok .properties
```

The `version` value in the output must be one of the following:

LP	Sun Dual 10GbE SFP+ PCIe 2.0 LP FCode 3.01 4/2/2012
PEM	Sun Dual 10GbE SFP+ PCIe 2.0 EM FCode 3.01 4/2/2012
FEM	Sun Dual 10GbE SFP+ PCIe 2.0 FEM FCode 3.01 4/2/2012

2. Download patch ID 13932765 from [My Oracle Support](https://support.oracle.com/CSP/ui/flash.html#tab=PatchHomePage(page=PatchHomePage&id=h0wvdxy6())) ([https://support.oracle.com/CSP/ui/flash.html#tab=PatchHomePage\(page=PatchHomePage&id=h0wvdxy6\(\)\)](https://support.oracle.com/CSP/ui/flash.html#tab=PatchHomePage(page=PatchHomePage&id=h0wvdxy6()))).
3. Install the patch.

The patch package includes a document that describes how to use the tool to perform the upgrade.

- **InfiniBand SR-IOV.** To use this feature, your system must run at least the following version of the system firmware:
 - **SPARC T4 – 8.4**
 - **SPARC T5, SPARC M5, and SPARC M6 – 9.1.0.x**
 - **Fujitsu M10 Servers – XCP2210**

To support the Dual 40-Gigabit (4x) InfiniBand Host Channel Adapter M2 as an InfiniBand SR-IOV device, the card or express module must run at least version 2.11.2010 of the firmware. You can obtain this version of the firmware by installing the following patches:

- **Low Profile (X4242A)** – Patch ID 16340059
- **Express Module (X4243A)** – Patch ID 16340042

Use the Oracle Solaris 11.1 `fwflash` command to list and update the firmware in the primary domain. To list the current firmware version, use the `fwflash -lc IB` command. To update the firmware, use the `fwflash -f firmware-file -d device` command. See the [fwflash\(1M\)](#) man page.

To use InfiniBand SR-IOV, ensure that InfiniBand switches have at least firmware version 2.1.2. You can obtain this version of the firmware by installing the following patches:

- **Sun Datacenter InfiniBand Switch 36 (X2821A-Z)** – Patch ID 16221424
- **Sun Network QDR InfiniBand GatewaySwitch (X2826A-Z)** – Patch ID 16221538

For information about how to update the firmware, see your InfiniBand switch documentation.

- **Fibre Channel SR-IOV.** To use this feature, your system must run at least the following version of the system firmware:
 - **SPARC T4 – 8.4.2.c**

- **SPARC T5, SPARC M5, and SPARC M6** – 9.1.2.d
- **Fujitsu M10 Servers** – XCP2210

The firmware on the Sun Storage 16 Gb Fibre Channel Universal HBA, Emulex must be at least revision 1.1.60.1 to enable the Fibre Channel SR-IOV feature. The installation instructions are provided with the firmware.



Caution - Only perform the firmware update to the Fibre Channel card if you plan to use the Fibre Channel SR-IOV feature.

- **Software Requirements.** The Oracle Solaris 10 OS is no longer supported for use in an SR-IOV root domain. An SR-IOV root domain must run be at least the Oracle Solaris 11.1.10.5.0 OS unless otherwise noted. So, upgrade your SR-IOV root domains to run at least the Oracle Solaris 11.1.10.5.0 OS. It is recommended that you upgrade the I/O domains to run at least the Oracle Solaris 11.1.10.5.0 OS, as well.
- **Ethernet SR-IOV.** To use the SR-IOV feature, root domains must be running at least the Oracle Solaris 11.1.10.5.0 OS. I/O domains must be running at least the Oracle Solaris 10 1/13 OS.
- **InfiniBand SR-IOV.** The following domains must run the supported Oracle Solaris OS:
 - The primary domain or a non-primary root domain must be running at least the Oracle Solaris 11.1.10.6.0 OS.
 - The I/O domains can run at least the Oracle Solaris 11.1.10.6.0 OS or the Oracle Solaris 10 1/13 OS.
 - Update the `/etc/system` file on any root domain that has an InfiniBand SR-IOV physical function from which you plan to configure virtual functions.

```
set ldc:ldc_maptable_entries = 0x20000
```

For information about correctly creating or updating `/etc/system` property values, see “[Updating Property Values in the /etc/system File](#)” on page 334.

Update the `/etc/system` file on the I/O domain to which you add a virtual function.

```
set rdsv3:rdsv3_fmr_pool_size = 16384
```

- **Fibre Channel SR-IOV.** To use the SR-IOV feature, root domains must be running at least the Oracle Solaris 11.1.17.4.0 OS. I/O domains must be running at least the Oracle Solaris 10 1/13 OS.

See the following for more information about static and dynamic SR-IOV software requirements:

- “[Static SR-IOV Software Requirements](#)” on page 96
- “[Dynamic SR-IOV Software Requirements](#)” on page 97

See the following for more information about the class-specific SR-IOV hardware requirements:

- “[Ethernet SR-IOV Hardware Requirements](#)” on page 100
- “[InfiniBand SR-IOV Hardware Requirements](#)” on page 120
- “[Fibre Channel SR-IOV Hardware Requirements](#)” on page 135

Current SR-IOV Feature Limitations

The SR-IOV feature has the following limitations:

- An I/O domain cannot start if any associated root domain is not running.
- Migration is disabled for any domain that has one or more virtual functions assigned to it.
- You can destroy only the last virtual function that was created for a physical function. So, if you create three virtual functions, the first virtual function that you can destroy must be the third one.
- If an SR-IOV card is assigned to a domain by using the Direct I/O (DIO) feature, the SR-IOV feature is not enabled for that card.
- The PCIe endpoint devices and SR-IOV virtual functions from a particular PCIe bus can be assigned up to a maximum of 15 domains on supported SPARC T-Series and SPARC M-Series systems. On a Fujitsu M10 server you can assign PCIe endpoint devices and SR-IOV virtual functions from a particular PCIe bus to a maximum of 24 domains. The PCIe resources, such as interrupt vectors for each PCIe bus, are divided among the root domain and I/O domains. As a result, the number of devices that you can assign to a particular I/O domain is also limited. Make sure that you do not assign a large number virtual functions to the same I/O domain. For a description of the problems related to SR-IOV, see “[Oracle VM Server for SPARC 3.2 Release Notes](#)”.
- The root domain is the owner of the PCIe bus and is responsible for initializing and managing the bus. The root domain must be active and running a version of the Oracle Solaris OS that supports the SR-IOV feature. Shutting down, halting, or rebooting the root 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 SR-IOV virtual functions is unpredictable when the root 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 root domain, you would need to manually stop and start each domain.

If the I/O domain is resilient, it can continue to operate even if the root domain that is the owner of the PCIe bus becomes unavailable. See “[I/O Domain Resiliency](#)” on page 149.

- SPARC systems, up to and including the SPARC T5 and SPARC M6 platforms, provide a finite number of interrupts, so Oracle Solaris limits the number of interrupts that each device can use. The default limit should match the needs of a typical system configuration but you might need to adjust this value for certain system configurations. For more information, see “[Adjusting the Interrupt Limit](#)” on page 352.

Static SR-IOV

The static SR-IOV method requires that the root domain be in delayed reconfiguration or the I/O domain be stopped while performing SR-IOV operations. After you complete the configuration steps on the root domain, you must reboot it. You must use this method when the Oracle VM Server for SPARC 3.1 firmware is not installed in the system or when the OS version that is installed in the respective domain does not support dynamic SR-IOV.

To create or destroy an SR-IOV virtual function, you first must initiate a delayed reconfiguration on the root domain. Then you can run one or more `ldm create-vf` and `ldm destroy-vf` commands to configure the virtual functions. Finally, reboot the root domain. The following commands show how to create a virtual function on a non-primary root domain:

```
primary# ldm start-reconf root-domain-name
primary# ldm create-vf pf-name
primary# ldm stop-domain -r root-domain-name

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

To statically add a virtual function to or remove one from a guest domain, you must first stop the guest domain. Then perform the `ldm add-io` and `ldm remove-io` commands to configure the virtual functions. After the changes are complete, start the domain. The following commands show how to assign a virtual function in this way:

```
primary# ldm stop guest-domain
primary# ldm add-io vf-name guest-domain
primary# ldm start guest-domain
```

You can also add a virtual function to or remove one from a root domain instead of a guest domain. To add an SR-IOV virtual function to or remove one from a root domain, first initiate a delayed reconfiguration on the root domain. Then, you can run one or more of the `ldm add-io` and `ldm remove-io` commands. Finally, reboot the root domain.

To minimize domain downtime, plan ahead before configuring virtual functions.

Note - InfiniBand SR-IOV devices are supported only with static SR-IOV.

Static SR-IOV Software Requirements

The static SR-IOV features are supported by the Oracle VM Server for SPARC 3.0 software and firmware. See “[PCIe SR-IOV Hardware and Software Requirements](#)” in “[Oracle VM Server for SPARC 3.0 Release Notes](#)”.

You can use the `ldm set-io` or `ldm add-io` command to set the `iov` property to on. You can also use the `ldm add-domain` or `ldm set-domain` command to set the `rc-add-policy` property to `iov`. See the [ldm\(1M\)](#) man page.

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

Dynamic SR-IOV

The dynamic SR-IOV feature removes the following static SR-IOV requirements:

- **Root domain.** Initiate a delayed reconfiguration on the root domain, create or destroy a virtual function, and reboot the root domain
- **I/O domain.** Stop the I/O domain, add or remove a virtual function, and start the I/O domain

With dynamic SR-IOV you can dynamically create or destroy a virtual function without having to initiate a delayed reconfiguration on the root domain. A virtual function can also be dynamically added to or removed from an I/O domain without having to stop the domain. The Logical Domains Manager communicates with the Logical Domains agent and the Oracle Solaris I/O virtualization framework to effect these changes dynamically.

Dynamic SR-IOV Software Requirements

For information about the required PCIe SR-IOV software and firmware versions, see “[SR-IOV Hardware and Software Requirements](#)” on page 92.

Note - If your system does not meet the dynamic SR-IOV software and firmware requirements, you must use the static SR-IOV method to perform SR-IOV-related tasks. See “[Static SR-IOV](#)” on page 96.

Dynamic SR-IOV Configuration Requirements

To dynamically create or destroy a virtual function, ensure that the following conditions are met:

- I/O virtualization has been enabled for a PCIe bus before you begin to configure virtual functions.

- The OS that runs on the root domain and on I/O domains must be at least the Oracle Solaris 11.1.10.5.0 OS or the Oracle Solaris 10 1/13 OS plus the required patches in “[Fully Qualified Oracle Solaris OS Versions](#)” in “[Oracle VM Server for SPARC 3.2 Installation Guide](#)”.
- The physical function device is not configured in the OS or is in a multipathing configuration. For example, you can unplumb an Ethernet SR-IOV device or have it in an IPMP or an aggregation to successfully create or destroy a virtual function.
An operation to create or destroy a virtual function requires that the physical function device driver toggle between the offline and online states. A multipathing configuration permits the device driver to toggle between these states.
- The virtual function is either not in use or in a multipathing configuration before you remove a virtual function from an I/O domain. For example, you can either unplumb an Ethernet SR-IOV virtual function or not use it in an IPMP configuration.

Note - You cannot use aggregation for Ethernet SR-IOV virtual functions because the current multipathing implementation does not support virtual functions.

Enabling I/O Virtualization

Before you can configure SR-IOV virtual functions, you must enable I/O virtualization for the PCIe bus while the root domain is in a delayed reconfiguration. Reboot the domain to make this change take effect.

Note - On Fujitsu M10 servers, PCIe buses are enabled for I/O virtualization by default.

▼ How to Enable I/O Virtualization for a PCIe Bus

This procedure needs to be performed only one time per root complex. The root complex must be running as part of the same SP configuration.

1. **Initiate a delayed reconfiguration on the root domain.**

```
primary# ldm start-reconf root-domain-name
```

2. **Enable I/O virtualization operations for a PCIe bus.**

Perform this step only if I/O virtualization is not enabled already for the bus that has the physical function.

Run one of the following commands:

- **Enable I/O virtualization if the specified PCIe bus already is assigned to a root domain.**

```
primary# ldm set-io iov=on bus
```

- **Enable I/O virtualization while you add a PCIe bus to a root domain.**

```
primary# ldm add-io iov=on bus
```

3. Reboot the root domain.

Run one of the following commands:

- **Reboot the non-primary root domain.**

```
primary# ldm stop-domain -r root-domain
```

- **Reboot the primary root domain.**

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

Planning for the Use of PCIe SR-IOV Virtual Functions

Plan ahead to determine how you want to use virtual functions in your configuration.

Determine which virtual functions from the SR-IOV devices will satisfy your current and future configuration needs.

If you have not yet enabled I/O virtualization, which requires using the static method, combine this step with the steps to create virtual functions. By combining these steps, you need to reboot the root domain only once.

Even when dynamic SR-IOV is available, the recommended practice is to create all the virtual functions at once because you might not be able to create them dynamically after they have been assigned to I/O domains.

In the static SR-IOV case, planning helps you to avoid performing multiple root domain reboots, each of which might negatively affect I/O domains.

For information about I/O domains, see “[General Guidelines for Creating an I/O Domain](#)” on page 62.

Use the following general steps to plan and perform SR-IOV virtual function configuration and assignment:

1. Determine which PCIe SR-IOV physical functions are available on your system and which ones are best suited to your needs.

Use the following commands to identify the required information:

<code>ldm list-io</code>	Identifies the available SR-IOV physical function devices.
<code>prtdiag -v</code>	Identifies which PCIe SR-IOV cards and on-board devices are available.
<code>ldm list-io -l pf-name</code>	Identifies additional information about a specified physical function, such as the maximum number of virtual functions that are supported by the device.
<code>ldm list-io -d pf-name</code>	Identifies the device-specific properties that are supported by the device. See “ Advanced SR-IOV Topics: Ethernet SR-IOV ” on page 113.

2. Enable I/O virtualization operations for a PCIe bus.
See “[How to Enable I/O Virtualization for a PCIe Bus](#)” on page 98.
3. Create the required number of virtual functions on the specified SR-IOV physical function.
Use the following command to create the virtual functions for the physical function:

```
primary# ldm create-vf -n max pf-name
```

For more information, see “[How to Create an Ethernet SR-IOV Virtual Function](#)” on page 102, “[How to Create an InfiniBand Virtual Function](#)” on page 120, and “[How to Create a Fibre Channel SR-IOV Virtual Function](#)” on page 138.

4. Use the `ldm add-config` command to save the configuration to the SP.
For more information, see “[How to Add an Ethernet SR-IOV Virtual Function to an I/O Domain](#)” on page 110, “[How to Add an InfiniBand Virtual Function to an I/O Domain](#)” on page 125, and “[How to Add a Fibre Channel SR-IOV Virtual Function to an I/O Domain](#)” on page 146.

Using Ethernet SR-IOV Virtual Functions

You can use both the static and dynamic SR-IOV methods to manage Ethernet SR-IOV devices.

Ethernet SR-IOV Hardware Requirements

For information about the required PCIe Ethernet SR-IOV hardware, see “[SR-IOV Hardware and Software Requirements](#)” on page 92.

Ethernet SR-IOV Limitations

The Ethernet SR-IOV feature has the following limitations in this release:

- You can enable VLAN configurations of virtual functions by setting either the pvid or the vid property. You cannot set both of these virtual function properties simultaneously.
- You cannot use an SR-IOV virtual function as a back-end device for a virtual switch.

Planning for the Use of Ethernet SR-IOV Virtual Functions

When dynamically creating virtual functions, ensure that the physical functions use multipathing or that they are not plumbed.

If you cannot use multipathing or must plumb the physical function, use the static method to create the virtual functions. See “[Static SR-IOV](#)” on page 96.

Ethernet Device-Specific and Network-Specific Properties

Use the `ldm create-vf` command to set device-specific and network-specific properties of a virtual function. The `unicast-slots` property is device-specific. The `mac-addr`, `alt-mac-addrs`, `mtu`, `pvid`, and `vid` properties are network-specific.

Note that the `mac-addr`, `alt-mac-addrs`, and `mtu` network-specific properties can be changed only when the virtual function is assigned to the primary domain while in a delayed reconfiguration.

Attempts to change these properties fail when the virtual function is assigned as follows:

- When the virtual function is assigned to an active I/O domain: A property change request is rejected because the change must be made when the owning domain is in the inactive or bound state.
- When the virtual function is assigned to a non-primary domain and a delayed reconfiguration is already in effect: A property change request fails with an error message.

The `pvid` and `vid` network-specific properties can be changed without restriction.

Creating Ethernet Virtual Functions

This section describes how to dynamically create and destroy virtual functions. If you cannot use the dynamic methods to perform these actions, initiate a delayed reconfiguration on the root domain before you create or destroy virtual functions.

▼ How to Create an Ethernet SR-IOV Virtual Function

If you cannot use this dynamic method, use the static method instead. See “[Static SR-IOV](#)” on page 96.

1. Identify the physical function device.

```
primary# ldm list-io
```

Note that the name of the physical function includes the location information for the PCIe SR-IOV card or on-board device.

2. If I/O virtualization for the bus that has the physical function is not enabled already, enable it.

Perform this step only if I/O virtualization is not enabled already for the bus that has the physical function.

See “[How to Enable I/O Virtualization for a PCIe Bus](#)” on page 98.

3. Create a single virtual function or multiple virtual functions from an Ethernet physical function either dynamically or statically.

After you create one or more virtual functions, you can assign them to a guest domain.

■ Dynamic method:

- To create multiple virtual functions from a physical function all at the same time, use the following command:

```
primary# ldm create-vf -n number | max pf-name
```

Use the `ldm create-vf -n max` command to create all the virtual functions for that physical function at one time.



Caution - When your system uses an Intel 10-Gbit Ethernet card, maximize performance by creating no more than 31 virtual functions from each physical function.

You can use either the path name or the pseudonym name to specify virtual functions. However, the recommended practice is to use the pseudonym name.

- **To create one virtual function from a physical function, use the following command:**

```
ldm create-vf [mac-addr=num] [alt-mac-addrs=[auto|num1,[auto|num2,...]]]
[pvid=pvid] [vid=vid1,vid2,...] [mtu=size] [name=value...] pf-name
```

Note - If not explicitly assigned, the MAC address is automatically allocated for network devices.

Use this command to create one virtual function for that physical function. You can also manually specify Ethernet class-specific property values.

Note - Sometimes a newly created virtual function is not available for immediate use while the OS probes for IOV devices. Use the `ldm list-io` command to determine whether the parent physical function and its child virtual functions have the `INV` value in the Status column. If they have this value, wait until the `ldm list-io` output no longer shows the `INV` value in the Status column (about 45 seconds) before you use that physical function or any of its child virtual functions. If this status persists, there is a problem with the device.

A device status might be `INV` immediately following a root domain reboot (including that of the primary) or immediately after you use the `ldm create-vf` or `ldm destroy-vf` command.

- **Static method:**

- a. **Initiate a delayed reconfiguration.**

```
primary# ldm start-reconf root-domain-name
```

- b. **Create a single virtual function or multiple virtual functions from an Ethernet physical function.**

Use the same commands as shown previously to dynamically create the virtual functions.

- c. **Reboot the root domain.**

- **To reboot the non-primary root domain:**

```
primary# ldm stop-domain -r root-domain
```

- **To reboot the primary root domain:**

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

Example 8-1 Displaying Information About the Ethernet Physical Function

This example shows information about the /SYS/MB/NET0/IOVNET.PF0 physical function:

- This physical function is from an on-board NET0 network device.
- The IOVNET string indicates that the physical function is a network SR-IOV device.

```
primary# ldm list-io
NAME                      TYPE   BUS    DOMAIN  STATUS
-----
niu_0                     NIU    niu_0  primary
niu_1                     NIU    niu_1  primary
pci_0                     BUS    pci_0  primary
pci_1                     BUS    pci_1  primary
/SYS/MB/PCIE0              PCIE   pci_0  primary  OCC
/SYS/MB/PCIE2              PCIE   pci_0  primary  OCC
/SYS/MB/PCIE4              PCIE   pci_0  primary  OCC
/SYS/MB/PCIE6              PCIE   pci_0  primary  EMP
/SYS/MB/PCIE8              PCIE   pci_0  primary  EMP
/SYS/MB/SASHBA             PCIE   pci_0  primary  OCC
/SYS/MB/NET0                PCIE   pci_0  primary  OCC
/SYS/MB/PCIE1              PCIE   pci_1  primary  OCC
/SYS/MB/PCIE3              PCIE   pci_1  primary  OCC
/SYS/MB/PCIE5              PCIE   pci_1  primary  OCC
/SYS/MB/PCIE7              PCIE   pci_1  primary  EMP
/SYS/MB/PCIE9              PCIE   pci_1  primary  EMP
/SYS/MB/NET2                PCIE   pci_1  primary  OCC
/SYS/MB/NET0/IOVNET.PF0      PF     pci_0  primary
/SYS/MB/NET0/IOVNET.PF1      PF     pci_0  primary
/SYS/MB/PCIE5/IOVNET.PF0      PF     pci_1  primary
/SYS/MB/PCIE5/IOVNET.PF1      PF     pci_1  primary
/SYS/MB/NET2/IOVNET.PF0      PF     pci_1  primary
/SYS/MB/NET2/IOVNET.PF1      PF     pci_1  primary
```

The following command shows more details about the specified physical function. The maxvfs value indicates the maximum number of virtual functions that is supported by the device.

```
primary# ldm list-io -l /SYS/MB/NET0/IOVNET.PF0
NAME                      TYPE   BUS    DOMAIN  STATUS
-----
/SYS/MB/NET0/IOVNET.PF0          PF     pci_0  primary
[pci@400/pci@1/pci@0/pci@4/network@0]
maxvfs = 7
```

Example 8-2 Dynamically Creating an Ethernet Virtual Function Without Setting Optional Properties

This example dynamically creates a virtual function without setting any optional properties. In this case, the MAC address for a network class virtual function is automatically allocated.

Ensure that I/O virtualization is enabled on the pci_0 PCIe bus. See “[How to Enable I/O Virtualization for a PCIe Bus](#)” on page 98.

Now, you can use the `ldm create-vf` command to create the virtual function from the `/SYS/MB/NET0/IOVNET.PF0` physical function.

```
primary# ldm create-vf /SYS/MB/NET0/IOVNET.PF0
Created new vf: /SYS/MB/NET0/IOVNET.PF0.VF0
```

Example 8-3 Dynamically Creating an Ethernet Virtual Function and Setting Properties

This example dynamically creates a virtual function while setting the `mac-addr` property to `00:14:2f:f9:14:c0` and the `vid` property to VLAN IDs 2 and 3.

```
primary# ldm create-vf mac-addr=00:14:2f:f9:14:c0 vid=2,3 /SYS/MB/NET0/IOVNET.PF0
```

Example 8-4 Dynamically Creating an Ethernet Virtual Function With Two Alternate MAC Addresses

This example dynamically creates a virtual function that has two alternate MAC addresses. One MAC address is automatically allocated, and the other is explicitly specified as `00:14:2f:f9:14:c2`.

```
primary# ldm create-vf alt-mac-addrs=auto,00:14:2f:f9:14:c2 /SYS/MB/NET0/IOVNET.PF0
```

Example 8-5 Statically Creating a Virtual Function Without Setting Optional Properties

This example statically creates a virtual function without setting any optional properties. In this case, the MAC address for a network class virtual function is automatically allocated.

First you initiate a delayed reconfiguration on the primary domain and then enable I/O virtualization on the `pci_0` PCIe bus. Because the `pci_0` bus has already been assigned to the primary root domain, use the `ldm set-io` command to enable I/O virtualization.

```
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
will also take effect.
```

```
primary# ldm set-io iov=on pci_0
```

Now, you can use the `ldm create-vf` command to create the virtual function from the `/SYS/MB/NET0/IOVNET.PF0` physical function.

```
primary# ldm create-vf /SYS/MB/NET0/IOVNET.PF0
```

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.

```
Created new vf: /SYS/MB/NET0/IOVNET.PF0.VF0
```

Finally, reboot the primary root domain to make the changes take effect.

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

Example 8-6 Creating Multiple SR-IOV Ethernet Virtual Functions

The following command shows how you can create four virtual functions from the /SYS/MB/NET2/IOVNET.PF1 physical function:

```
primary# ldm create-vf -n 31 /SYS/MB/NET2/IOVNET.PF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF0
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF2
...
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF30
```

Note that the `ldm create-vf -n` command creates multiple virtual functions that are set with default property values, if appropriate. You can later specify non-default property values by using the `ldm set-io` command.

Destroying Ethernet Virtual Functions

A virtual function can be destroyed if it is not currently assigned to a domain. A virtual function can be destroyed only in the reverse sequential order of creation, so only the last virtual function that was created can be destroyed. The resulting configuration is validated by the physical function driver.

▼ How to Destroy an Ethernet SR-IOV Virtual Function

If you cannot use this dynamic method, use the static method instead. See “[Static SR-IOV](#)” on page 96.

1. Identify the physical function device.

```
primary# ldm list-io
```

2. Destroy single a virtual function or multiple virtual functions either dynamically or statically.

■ **Dynamic method:**

■ **To destroy some or all of the virtual functions from a physical function at one time, use the following command:**

```
primary# ldm destroy-vf -n number | max pf-name
```

Use the `ldm destroy-vf -n max` command to destroy all the virtual functions for that physical function at one time.

If you specify *number* as an argument to the **-n** option, the last *number* of virtual functions are destroyed. Use this method as it performs this operation with only one physical function device driver state transition.

You can use either the path name or the pseudonym name to specify virtual functions. However, the recommended practice is to use the pseudonym name.

- **To destroy a specified virtual function:**

```
primary# ldm destroy-vf vf-name
```

Due to delays in the affected hardware device and in the OS, the affected physical function and any remaining child virtual functions might not be available for immediate use. Use the **ldm list-io** command to determine whether the parent physical function and its child virtual functions have the **INV** value in the Status column. If they have this value, wait until the **ldm list-io** output no longer shows the **INV** value in the Status column (about 45 seconds). At that time, you can safely use that physical function or any of its child virtual functions. If this status persists, there is a problem with the device.

A device status might be **INV** immediately following a root domain reboot (including that of the primary) or immediately after you use the **ldm create-vf** or **ldm destroy-vf** command.

- **Static method:**

- a. **Initiate a delayed reconfiguration.**

```
primary# ldm start-reconf root-domain-name
```

- b. **Destroy either a single virtual function or multiple virtual functions.**

- **To destroy all of the virtual functions from the specified physical function at the same time, use the following command:**

```
primary# ldm destroy-vf -n number | max pf-name
```

You can use either the path name or the pseudonym name to specify virtual functions. However, the recommended practice is to use the pseudonym name.

- **To destroy a specified virtual function:**

```
primary# ldm destroy-vf vf-name
```

- c. **Reboot the root domain.**

- **To reboot the non-primary root domain:**

```
primary# ldm stop-domain -r root-domain
```

■ **To reboot the primary root domain:**

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

Example 8-7 Destroying an Ethernet Virtual Function

This example shows how to dynamically destroy the /SYS/MB/NET0/IOVNET.PF0.VF0 virtual function.

```
primary# ldm destroy-vf /SYS/MB/NET0/IOVNET.PF0.VF0
```

The following example shows how to statically destroy the /SYS/MB/NET0/IOVNET.PF0.VF0 virtual function.

```
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
will also take effect.
```

```
primary# ldm destroy-vf /SYS/MB/NET0/IOVNET.PF0.VF0
primary# shutdown -i6 -g0 -y
```

Example 8-8 Destroying Multiple Ethernet SR-IOV Virtual Functions

This example shows the results of destroying all the virtual functions from the /SYS/MB/NET2/IOVNET.PF1 physical function. The ldm list-io output shows that the physical function has seven virtual functions. The ldm destroy-vf command destroys all virtual functions, and the final ldm list-io output shows that none of the virtual functions remain.

```
primary# ldm list-io
...
/SYS/MB/NET2/IOVNET.PF1          PF      pci_1
/SYS/MB/NET2/IOVNET.PF1.VF0      VF      pci_1
/SYS/MB/NET2/IOVNET.PF1.VF1      VF      pci_1
/SYS/MB/NET2/IOVNET.PF1.VF2      VF      pci_1
/SYS/MB/NET2/IOVNET.PF1.VF3      VF      pci_1
/SYS/MB/NET2/IOVNET.PF1.VF4      VF      pci_1
/SYS/MB/NET2/IOVNET.PF1.VF5      VF      pci_1
/SYS/MB/NET2/IOVNET.PF1.VF6      VF      pci_1
primary# ldm destroy-vf -n max /SYS/MB/NET2/IOVNET.PF1
primary# ldm list-io
...
/SYS/MB/NET2/IOVNET.PF1          PF      pci_1      ldg1
```

Modifying Ethernet SR-IOV Virtual Functions

The `ldm set-io vf-name` command modifies the current configuration of a virtual function by changing the property values or by setting new properties. This command can modify both the network-specific properties and the device-specific properties. For information about device-specific properties, see “[Advanced SR-IOV Topics: Ethernet SR-IOV](#)” on page 113.

If you cannot use this dynamic method, use the static method instead. See “[Static SR-IOV](#)” on page 96.

You can use the `ldm set-io` command to modify the following properties:

- `mac-addr`, `alt-mac-addrs`, and `mtu`

To change these virtual function properties, stop the domain that owns the virtual function, use the `ldm set-io` command to change the property values, and start the domain.

- `pvid` and `vid`

You can dynamically change these properties while the virtual functions are assigned to a domain. Note that doing so might result in a change to the network traffic of an active virtual function; setting the `pvid` property enables a transparent VLAN. Setting the `vid` property to specify VLAN IDs permits VLAN traffic to those specified VLANs.

- **Device-specific properties**

Use the `ldm list-io -d pf-name` command to view the list of valid device-specific properties. You can modify these properties for both the physical function and the virtual function. You must use the static method to modify device-specific properties. See “[Static SR-IOV](#)” on page 96. For more information about device-specific properties, see “[Advanced SR-IOV Topics: Ethernet SR-IOV](#)” on page 113.

▼ How to Modify Ethernet SR-IOV Virtual Function Properties

1. Identify the physical function device.

```
primary# ldm list-io
```

Note that the name of the physical function includes the location information for the PCIe SR-IOV card or on-board device.

2. Modify a virtual function property.

```
ldm set-io name=value [name=value...] vf-name
```

Example 8-9 Modifying Ethernet Virtual Function Properties

These examples describe how to use the `ldm set-io` command to set properties on an Ethernet virtual function.

- The following example modifies properties of the specified virtual function, /SYS/MB/NET0/IOVNET.PF0.VF0, to be part of VLAN IDs 2, 3, and 4.

```
primary# ldm set-io vid=2,3,4 /SYS/MB/NET0/IOVNET.PF0.VF0
```

Note that this command dynamically changes the VLAN association for a virtual function. To use these VLANs, the VLAN interfaces in the I/O domains must be configured by using the appropriate Oracle Solaris OS networking commands.

- The following example sets the pvid property value to 2 for the /SYS/MB/NET0/IOVNET.PF0.VF0 virtual function, which transparently makes the virtual function part of VLAN 2. Namely, the virtual function will not view any tagged VLAN traffic.

```
primary# ldm set-io pvid=2 /SYS/MB/NET0/IOVNET.PF0.VF0
```

- The following example assigns three automatically allocated alternate MAC addresses to a virtual function. The alternate addresses enable the creation of Oracle Solaris 11 virtual network interface cards (VNICs) on top of a virtual function. Note that to use VNICs, you must run the Oracle Solaris 11 OS in the domain.

Note - Before you run this command, stop the domain that owns the virtual function.

```
primary# ldm set-io alt-mac-addrs=auto,auto,auto /SYS/MB/NET0/IOVNET.PF0.VF0
```

- The following example sets the device-specific unicast-slots property to 12 for the specified virtual function. To find the device-specific properties that are valid for a physical function, use the ldm list-io -d pf-name command.

```
primary# ldm set-io unicast-slots=12 /SYS/MB/NET0/IOVNET.PF0.VF0
```

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.

Adding and Removing Ethernet SR-IOV Virtual Functions on I/O Domains

▼ How to Add an Ethernet SR-IOV Virtual Function to an I/O Domain

If you cannot dynamically remove the virtual function, use the static method. See “[Static SR-IOV](#)” on page 96.

1. Identify the virtual function that you want to add to an I/O domain.

```
primary# ldm list-io
```

2. Add a virtual function dynamically or statically.

- To dynamically add a virtual function:

```
primary# ldm add-io vf-name domain-name
```

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the pseudonym name. *domain-name* specifies the name of the domain to which you add the virtual function.

The device path name for the virtual function in the domain is the path shown in the *list-io -l* output.

- To statically add a virtual function:

- a. Initiate a delayed reconfiguration and then add the virtual function.

```
primary# ldm start-reconf root-domain-name
primary# ldm add-io vf-name domain-name
```

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the pseudonym name. *domain-name* specifies the name of the domain to which you add the virtual function. The specified guest must be in the inactive or bound state.

The device path name for the virtual function in the domain is the path shown in the *list-io -l* output.

- b. Reboot the root domain.

- To reboot the non-primary root domain:

```
primary# ldm stop-domain -r root-domain
```

- To reboot the primary root domain:

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

Example 8-10 Adding an Ethernet Virtual Function

This example shows how to dynamically add the /SYS/MB/NET0/IOVNET.PF0.VF0 virtual function to the ldg1 domain.

```
primary# ldm add-io /SYS/MB/NET0/IOVNET.PF0.VF0 ldg1
```

If you cannot add the virtual function dynamically, use the static method:

```
primary# ldm stop-domain ldg1
primary# ldm add-io /SYS/MB/NET0/IOVNET.PF0.VF0 ldg1
primary# ldm start-domain ldg1
```

▼ How to Remove an Ethernet Virtual SR-IOV Function From an I/O Domain

If you cannot dynamically remove the virtual function, use the static method. See “[Static SR-IOV](#)” on page 96.



Caution - Before removing the virtual function from the domain, ensure that it is not critical for booting that domain.

1. Identify the virtual function that you want to remove from an I/O domain.

```
primary# ldm list-io
```

2. Remove a virtual function either dynamically or statically.

- To dynamically remove a virtual function:

```
primary# ldm remove-io vf-name domain-name
```

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the device pseudonym. *domain-name* specifies the name of the domain from which you remove the virtual function.

- To statically remove a virtual function:

- a. Stop the I/O domain.

```
primary# ldm stop-domain domain-name
```

- b. Remove the virtual function.

```
primary# ldm remove-io vf-name domain-name
```

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the device pseudonym. *domain-name* specifies the name of the domain from which you remove the virtual function. The specified guest must be in the inactive or bound state.

- c. Start the I/O domain.

```
primary# ldm start-domain domain-name
```

Example 8-11 Dynamically Removing an Ethernet Virtual Function

This example shows how to dynamically remove the /SYS/MB/NET0/IOVNET.PF0.VF0 virtual function from the ldg1 domain.

```
primary# ldm remove-io /SYS/MB/NET0/IOVNET.PF0.VF0 ldg1
```

If the command succeeds, the virtual function is removed from the ldg1 domain. When ldg1 is restarted, the specified virtual function no longer appears in that domain.

If you cannot remove the virtual function dynamically, use the static method:

```
primary# ldm stop-domain ldg1
primary# ldm remove-io /SYS/MB/NET0/IOVNET.PF0.VF0 ldg1
primary# ldm start-domain ldg1
```

Advanced SR-IOV Topics: Ethernet SR-IOV

This section describes some advanced topics related to using SR-IOV virtual functions.

Advanced Network Configuration for Virtual Functions

When you use SR-IOV virtual functions, note the following issues:

- SR-IOV virtual functions can only use the MAC addresses that are assigned by the Logical Domains Manager. If you use other Oracle Solaris OS networking commands to change the MAC address on the I/O domain, the commands might fail or might not function properly.
- At this time, link aggregation of SR-IOV network virtual functions in the I/O domain is not supported. If you attempt to create a link aggregation, it might not function as expected.
- You can create virtual I/O services and assign them to I/O domains. These virtual I/O services can be created on the same physical function from which virtual functions are also created. For example, you can use an on-board 1-Gbps network device (net0 or igb0) as a network back-end device for a virtual switch and also statically create virtual functions from the same physical function device.

Booting an I/O Domain by Using an SR-IOV Virtual Function

An SR-IOV virtual function provides similar capabilities to any other type of PCIe device, such as the ability to use a virtual function as a logical domain boot device. For example, a network virtual function can be used to boot over the network to install the Oracle Solaris OS in an I/O domain.

Note - When booting the Oracle Solaris OS from a virtual function device, verify that the Oracle Solaris OS that is being loaded has virtual function device support. If so, you can continue with the rest of the installation as planned.

SR-IOV Device-Specific Properties

SR-IOV physical function device drivers can export device-specific properties. These properties can be used to tune the resource allocation of both the physical function and its virtual functions. For information about the properties, see the man page for the physical function driver, such as the [igb\(7D\)](#) and [ixgbe\(7D\)](#) man pages.

The `ldm list-io -d` command shows device-specific properties that are exported by the specified physical function device driver. The information for each property includes its name, brief description, default value, maximum values, and one or more of the following flags:

P	Applies to a physical function
V	Applies to a virtual function
R	Read-only or informative parameter only

```
primary# ldm list-io -d pf-name
```

Use the `ldm create-vf` or `ldm set-io` command to set the read-write properties for a physical function or a virtual function. Note that to set a device-specific property, you must use the static method. See [“Static SR-IOV” on page 96](#).

The following example shows the device-specific properties that are exported by the on-board Intel 1-Gbps SR-IOV device:

```
primary# ldm list-io -d /SYS/MB/NET0/IOVNET.PF0
Device-specific Parameters
-----
max-config-vfs
    Flags = PR
    Default = 7
    Descr = Max number of configurable VFs
max-vf-mtu
    Flags = VR
    Default = 9216
    Descr = Max MTU supported for a VF
max-vlans
    Flags = VR
    Default = 32
    Descr = Max number of VLAN filters supported
```

```

pvid-exclusive
  Flags = VR
  Default = 1
  Descr = Exclusive configuration of pvid required
unicast-slots
  Flags = PV
  Default = 0 Min = 0 Max = 24
  Descr = Number of unicast mac-address slots

```

The following example sets the `unicast-slots` property to 8:

```
primary# ldm create-vf unicast-slots=8 /SYS/MB/NET0/IOVNET.PF0
```

Creating VNICs on SR-IOV Virtual Functions

The creation of Oracle Solaris 11 VNICs is supported on SR-IOV virtual functions. However, the number of VNICs that is supported is limited to the number of alternate MAC addresses (`alt-mac-addrs` property) assigned to the virtual function. Make sure that you assign a sufficient number of alternate MAC addresses when you use VNICs on the virtual function. Use the `ldm create-vf` or `ldm set-io` command to set the `alt-mac-addrs` property with the alternate MAC addresses.

The following example shows the creation of four VNICs on an SR-IOV virtual function. The first command assigns alternate MAC addresses to the virtual function device. This command uses the automatic allocation method to allocate four alternate MAC addresses to the `/SYS/MB/NET0/IOVNET.PF0.VF0` virtual function device:

```
primary# ldm set-io alt-mac-addrs=auto,auto,auto,auto /SYS/MB/NET0/IOVNET.PF0.VF0
```

The next command starts the `ldg1` I/O domain. Because the `auto-boot?` property is set to `true` in this example, the Oracle Solaris 11 OS is also booted in the I/O domain.

```
primary# ldm start ldg1
```

The following command uses the Oracle Solaris 11 `dladm` command in the guest domain to show virtual function that has alternate MAC addresses. This output shows that the `net30` virtual function has four alternate MAC addresses.

```

guest# dladm show-phys -m
LINK      SLOT      ADDRESS          INUSE CLIENT
net0       primary   0:14:4f:fa:b4:d1 yes    net0
net25      primary   0:14:4f:fa:c9:eb no     --
net30       primary   0:14:4f:fb:de:4c no     --
              1        0:14:4f:f9:e8:73 no     --
              2        0:14:4f:f8:21:58 no     --
              3        0:14:4f:fa:9d:92 no     --
              4        0:14:4f:f9:8f:1d no     --

```

The following commands create four VNICs. Note that attempts to create more VNICs than are specified by using alternate MAC addresses will fail.

```
guest# dladm create-vnic -l net30 vnic0
guest# dladm create-vnic -l net30 vnic1
guest# dladm create-vnic -l net30 vnic2
guest# dladm create-vnic -l net30 vnic3
guest# dladm show-link
      LINK      CLASS     MTU     STATE    OVER
  net0       phys    1500     up      --
  net25      phys    1500     up      --
  net30      phys    1500     up      --
  vnic0      vnic    1500     up    net30
  vnic1      vnic    1500     up    net30
  vnic2      vnic    1500     up    net30
  vnic3      vnic    1500     up    net30
```

Using an SR-IOV Virtual Function to Create an I/O Domain

The following procedure explains how to create an I/O domain that includes PCIe SR-IOV virtual functions.

▼ How to Create an I/O Domain by Assigning an SR-IOV Virtual Function to It

Plan ahead to minimize the number of reboots of the root domain, which minimizes downtime.

Before You Begin Before you begin, ensure that you have enabled I/O virtualization for the PCIe bus that is the parent of the physical function from which you create virtual functions. See “[How to Enable I/O Virtualization for a PCIe Bus](#)” on page 98.

1. **Identify an SR-IOV physical function to share with an I/O domain that uses the SR-IOV feature.**

```
primary# ldm list-io
```

2. **Create one or more virtual functions for the physical function.**

```
primary# ldm create-vf pf-name
```

You can run this command for each virtual function that you want to create. You can also use the -n option to create more than one virtual function from the same physical function in a single command. See [Example 8-6](#) and the [ldm\(1M\)](#) man page.

Note - This command fails if other virtual functions have already been created from the associated physical function and if any of those virtual functions are bound to another domain.

3. View the list of available virtual functions on the root domain.

```
primary# ldm list-io
```

4. Assign the virtual function that you created in Step 2 to its target I/O domain.

```
primary# ldm add-io vf-name domain-name
```

Note - If the OS in the target I/O domain does not support dynamic SR-IOV, you must use the static method. See “[Static SR-IOV](#)” on page [96](#).

5. Verify that the virtual function is available on the I/O domain.

The following Oracle Solaris 11 command shows the availability of the virtual function:

```
guest# dladm show-phys
```

Example 8-12 Dynamically Creating an I/O Domain by Assigning an SR-IOV Virtual Function to It

The following dynamic example shows how to create a virtual function, /SYS/MB/NET0/IOVNET.PF0.VF0, for a physical function, /SYS/MB/NET0/IOVNET.PF0, and assign the virtual function to the ldg1 I/O domain.

This example assumes that the following circumstances are true:

- The OS on the primary domain supports dynamic SR-IOV operations
- The pci_0 bus is assigned to the primary domain and has been initialized for I/O virtualization operations
- The /SYS/MB/NET0/IOVNET.PF0 physical function belongs to the pci_0 bus
- The /SYS/MB/NET0/IOVNET.PF0 physical function does not have any existing virtual functions assigned to domains
- The ldg1 domain is active and booted and its OS supports dynamic SR-IOV operations

Create the virtual function from the /SYS/MB/NET0/IOVNET.PF0 physical function.

```
primary# ldm create-vf /SYS/MB/NET0/IOVNET.PF0
Created new vf: /SYS/MB/NET0/IOVNET.PF0.VF0
```

Add the /SYS/MB/NET0/IOVNET.PF0.VF0 virtual function to the ldg1 domain.

```
primary# ldm add-io /SYS/MB/NET0/IOVNET.PF0.VF0 ldg1
```

The following command shows that the virtual function has been added to the ldg1 domain.

```
primary# ldm list-io
NAME          TYPE   BUS     DOMAIN   STATUS
-----
niu_0         NIU    niu_0   primary
niu_1         NIU    niu_1   primary
pci_0         BUS    pci_0   primary   IOV
pci_1         BUS    pci_1   primary
/SYS/MB/PCIE0 PCIE   pci_0   primary   OCC
/SYS/MB/PCIE2 PCIE   pci_0   primary   OCC
/SYS/MB/PCIE4 PCIE   pci_0   primary   OCC
/SYS/MB/PCIE6 PCIE   pci_0   primary   EMP
/SYS/MB/PCIE8 PCIE   pci_0   primary   EMP
/SYS/MB/SASHBA PCIE   pci_0   primary   OCC
/SYS/MB/NET0  PCIE   pci_0   primary   OCC
/SYS/MB/PCIE1 PCIE   pci_1   primary   OCC
/SYS/MB/PCIE3 PCIE   pci_1   primary   OCC
/SYS/MB/PCIE5 PCIE   pci_1   primary   OCC
/SYS/MB/PCIE7 PCIE   pci_1   primary   EMP
/SYS/MB/PCIE9 PCIE   pci_1   primary   EMP
/SYS/MB/NET2  PCIE   pci_1   primary   OCC
/SYS/MB/NET0/IOVNET.PF0 PF    pci_0   primary
/SYS/MB/NET0/IOVNET.PF1 PF    pci_0   primary
/SYS/MB/PCIE5/IOVNET.PF0 PF    pci_1   primary
/SYS/MB/PCIE5/IOVNET.PF1 PF    pci_1   primary
/SYS/MB/NET2/IOVNET.PF0 PF    pci_1   primary
/SYS/MB/NET2/IOVNET.PF1 PF    pci_1   primary
/SYS/MB/NET0/IOVNET.PF0.VF0 VF    pci_0   ldg1
```

Example 8-13 Statically Creating an I/O Domain by Assigning an SR-IOV Virtual Function to It

The following static example shows how to create a virtual function, /SYS/MB/NET0/IOVNET.PF0.VF0, for a physical function, /SYS/MB/NET0/IOVNET.PF0, and assign the virtual function to the ldg1 I/O domain.

This example assumes that the following circumstances are true:

- The OS on the primary domain does not support dynamic SR-IOV operations
- The pci_0 bus is assigned to the primary domain and has not been initialized for I/O virtualization operations
- The /SYS/MB/NET0/IOVNET.PF0 physical function belongs to the pci_0 bus
- The /SYS/MB/NET0/IOVNET.PF0 physical function does not have any existing virtual functions assigned to domains
- The ldg1 domain is active and booted and its OS does not support dynamic SR-IOV operations
- The ldg1 domain has the auto-boot? property set to true so that the domain boots automatically when the domain is started

First, initiate a delayed reconfiguration on the primary domain, enable I/O virtualization, and create the virtual function from the /SYS/MB/NET0/IOVNET.PF0 physical function.

```

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
will also take effect.

primary# ldm set-io iov=on pci_0
primary# ldm create-vf /SYS/MB/NET0/IOVNET.PF0

-----
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.
-----
Created new vf: /SYS/MB/NET0/IOVNET.PF0.VF0

```

Next, shut down the primary domain.

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

Stop the ldg1 domain, add the virtual function, and start the domain.

```

primary# ldm stop ldg1
primary# ldm add-io /SYS/MB/NET0/IOVNET.PF0.VF0 ldg1
primary# ldm start ldg1

```

The following command shows that the virtual function has been added to the ldg1 domain.

NAME	TYPE	BUS	DOMAIN	STATUS
niu_0	NIU	niu_0	primary	
niu_1	NIU	niu_1	primary	
pci_0	BUS	pci_0	primary	IOV
pci_1	BUS	pci_1	primary	
/SYS/MB/PCIE0	PCIE	pci_0	primary	OCC
/SYS/MB/PCIE2	PCIE	pci_0	primary	OCC
/SYS/MB/PCIE4	PCIE	pci_0	primary	OCC
/SYS/MB/PCIE6	PCIE	pci_0	primary	EMP
/SYS/MB/PCIE8	PCIE	pci_0	primary	EMP
/SYS/MB/SASHBA	PCIE	pci_0	primary	OCC
/SYS/MB/NET0	PCIE	pci_0	primary	OCC
/SYS/MB/PCIE1	PCIE	pci_1	primary	OCC
/SYS/MB/PCIE3	PCIE	pci_1	primary	OCC
/SYS/MB/PCIE5	PCIE	pci_1	primary	OCC
/SYS/MB/PCIE7	PCIE	pci_1	primary	EMP
/SYS/MB/PCIE9	PCIE	pci_1	primary	EMP
/SYS/MB/NET2	PCIE	pci_1	primary	OCC
/SYS/MB/NET0/IOVNET.PF0	PF	pci_0	primary	
/SYS/MB/NET0/IOVNET.PF1	PF	pci_0	primary	
/SYS/MB/PCIE5/IOVNET.PF0	PF	pci_1	primary	
/SYS/MB/PCIE5/IOVNET.PF1	PF	pci_1	primary	
/SYS/MB/NET2/IOVNET.PF0	PF	pci_1	primary	
/SYS/MB/NET2/IOVNET.PF1	PF	pci_1	primary	
/SYS/MB/NET0/IOVNET.PF0.VF0	VF	pci_0	ldg1	

Using InfiniBand SR-IOV Virtual Functions

Only the static SR-IOV feature is supported for InfiniBand SR-IOV devices.

To minimize downtime, run all of the SR-IOV commands as a group while the root domain is in delayed reconfiguration or a guest domain is stopped. The SR-IOV commands that are limited in this way are the `ldm create-vf`, `ldm destroy-vf`, `ldm add-io`, and `ldm remove-io` commands.

Typically, virtual functions are assigned to more than one guest domain. A reboot of the root domain affects all of the guest domains that have been assigned the root domain's virtual functions.

Because an unused InfiniBand virtual function has very little overhead, you can avoid downtime by creating the necessary virtual functions ahead of time, even if they are not used immediately.

InfiniBand SR-IOV Hardware Requirements

For information about the required PCIe InfiniBand SR-IOV hardware, see “[SR-IOV Hardware and Software Requirements](#)” on page 92.

For InfiniBand SR-IOV support, the root domain must be running at least the Oracle Solaris 11.1.10.5.0 OS. The I/O domains can run the Oracle Solaris 10 1/13 OS plus patch 148888-04 or at least the Oracle Solaris 11.1.10.5.0 OS.

Creating and Destroying InfiniBand Virtual Functions

▼ How to Create an InfiniBand Virtual Function

This procedure describes how to create an InfiniBand SR-IOV virtual function.

1. **Initiate a delayed reconfiguration on the root domain.**

```
primary# ldm start-reconf root-domain-name
```

2. **Enable I/O virtualization by setting `iov=on`.**

Perform this step only if I/O virtualization is not enabled already for the bus that has the physical function.

```
primary# ldm set-io iov=on bus
```

3. **Create one or more virtual functions that are associated with the physical functions from that root domain.**

```
primary# ldm create-vf pf-name
```

You can run this command for each virtual function that you want to create. You can also use the -n option to create more than one virtual function from the same physical function in a single command. See [Example 8-6](#) and the [ldm\(1M\)](#) man page.

4. **Reboot the root domain.**

Run one of the following commands:

- **Reboot the non-primary root domain.**

```
primary# ldm stop-domain -r root-domain
```

- **Reboot the primary root domain.**

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

Example 8-14 Creating an InfiniBand Virtual Function

The following example shows information about the /SYS/MB/RISER1/PCIE4/IOVIB.PF0 physical function:

- This physical function is in PCIE slot 4.
- The IOVIB string indicates that the physical function is an InfiniBand SR-IOV device.

NAME	TYPE	BUS	DOMAIN	STATUS
pci_0	BUS	pci_0	primary	
niu_0	NIU	niu_0	primary	
/SYS/MB/RISER0/PCIE0	PCIE	pci_0	primary	EMP
/SYS/MB/RISER1/PCIE1	PCIE	pci_0	primary	EMP
/SYS/MB/RISER2/PCIE2	PCIE	pci_0	primary	EMP
/SYS/MB/RISER0/PCIE3	PCIE	pci_0	primary	OCC
/SYS/MB/RISER1/PCIE4	PCIE	pci_0	primary	OCC
/SYS/MB/RISER2/PCIE5	PCIE	pci_0	primary	EMP
/SYS/MB/SASHBA0	PCIE	pci_0	primary	OCC
/SYS/MB/SASHBA1	PCIE	pci_0	primary	OCC
/SYS/MB/NET0	PCIE	pci_0	primary	OCC
/SYS/MB/NET2	PCIE	pci_0	primary	OCC
/SYS/MB/RISER0/PCIE3/IOVIB.PF0	PF	pci_0	primary	
/SYS/MB/RISER1/PCIE4/IOVIB.PF0	PF	pci_0	primary	
/SYS/MB/NET0/IOVNET.PF0	PF	pci_0	primary	
/SYS/MB/NET0/IOVNET.PF1	PF	pci_0	primary	
/SYS/MB/NET2/IOVNET.PF0	PF	pci_0	primary	

```
/SYS/MB/NET2/IOVNET.PF1          PF      pci_0      primary
```

The following command shows more details about the specified physical function. The `maxvfs` value indicates the maximum number of virtual functions that are supported by the device.

```
primary# ldm list-io -l /SYS/MB/RISER1/PCIE4/IOVIB.PF0
NAME                      TYPE   BUS    DOMAIN  STATUS
-----
/SYS/MB/RISER1/PCIE4/IOVIB.PF0      PF      pci_0      primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0]
maxvfs = 64
```

The following example shows how to create a static virtual function. First, initiate a delayed reconfiguration on the primary domain and enable I/O virtualization on the `pci_0` PCIe bus. Because the `pci_0` bus has been assigned already to the primary root domain, use the `ldm set-io` command to enable I/O virtualization.

```
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
will also take effect.

primary# ldm set-io iov=on pci_0
-----
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.
-----
```

Now, use the `ldm create-vf` command to create a virtual function from the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0` physical function.

```
primary# ldm create-vf /SYS/MB/RISER1/PCIE4/IOVIB.PF0
-----
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.
-----
Created new vf: /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0
```

Note that you can create more than one virtual function during the same delayed reconfiguration. The following command creates a second virtual function:

```
primary# ldm create-vf /SYS/MB/RISER1/PCIE4/IOVIB.PF0
-----
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.
-----
Created new vf: /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1
```

Finally, reboot the primary root domain to make the changes take effect.

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

```

Shutdown started.

Changing to init state 6 - please wait
...

```

▼ How to Destroy an InfiniBand Virtual Function

This procedure describes how to destroy an InfiniBand SR-IOV virtual function.

A virtual function can be destroyed if it is not currently assigned to a domain. A virtual function can be destroyed only in the reverse sequential order of creation, so only the last virtual function that was created can be destroyed. The resulting configuration is validated by the physical function driver.

1. **Initiate a delayed reconfiguration on the root domain.**

```
primary# ldm start-reconf root-domain-name
```

2. **Destroy one or more virtual functions that are associated with the physical functions from that root domain.**

```
primary# ldm destroy-vf vf-name
```

You can run this command for each virtual function that you want to destroy. You can also use the -n option to destroy more than one virtual function from the same physical function in a single command. See [Example 8-8](#) and the [ldm\(1M\)](#) man page.

3. **Reboot the root domain.**

Run one of the following commands:

- **Reboot the non-primary root domain.**

```
primary# ldm stop-domain -r root-domain
```

- **Reboot the primary root domain.**

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

Example 8-15 Destroying an InfiniBand Virtual Function

The following example shows how to destroy a static InfiniBand virtual function, /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1.

The `ldm list-io` command shows information about the buses, physical functions, and virtual functions.

```
primary# ldm list-io
```

NAME	TYPE	BUS	DOMAIN	STATUS
---	---	---	---	---
pci_0	BUS	pci_0	primary	IOV
...				
/SYS/MB/RISER1/PCIE4/IOVIB.PF0	PF	pci_0	primary	
...				
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0	VF	pci_0		
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1	VF	pci_0		

You can obtain more details about the physical function and related virtual functions by using the `ldm list-io -l` command.

```
primary# ldm list-io -l /SYS/MB/RISER1/PCIE4/IOVIB.PF0
NAME          TYPE   BUS   DOMAIN STATUS
-----
/SYS/MB/RISER1/PCIE4/IOVIB.PF0      PF    pci_0   primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0]
maxvfs = 64
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0    VF    pci_0
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,1]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1    VF    pci_0
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,2]
```

A virtual function can be destroyed only if it is unassigned to a domain. The DOMAIN column of the `ldm list-io -l` output shows the name of any domain to which a virtual function is assigned. Also, virtual functions must be destroyed in the reverse order of their creation. Therefore, in this example, you must destroy the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1` virtual function before you can destroy the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0` virtual function.

After you identify the proper virtual function, you can destroy it. First, initiate a delayed reconfiguration.

```
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
will also take effect.

primary# ldm destroy-vf /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1
-----
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.
-----
```

You can issue more than one `ldm destroy-vf` command while in delayed reconfiguration. Thus, you could also destroy the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0`.

Finally, reboot the primary root domain to make the changes take effect.

```
primary# shutdown -i6 -g0 -y
Shutdown started.
```

```
Changing to init state 6 - please wait
...

```

Adding and Removing InfiniBand Virtual Functions on I/O Domains

▼ How to Add an InfiniBand Virtual Function to an I/O Domain

This procedure describes how to add an InfiniBand SR-IOV virtual function to an I/O domain.

1. Stop the I/O domain.

```
primary# ldm stop-domain domain-name
```

2. Add one or more virtual functions to the I/O domain.

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the pseudonym name. *domain-name* specifies the name of the domain to which you add the virtual function. The specified I/O domain must be in the inactive or bound state.

```
primary# ldm add-io vf-name domain-name
```

3. Start the I/O domain.

```
primary# ldm start-domain domain-name
```

Example 8-16 Adding an InfiniBand Virtual Function

The following example shows how to add the /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2 virtual function to the iodom1 I/O domain.

First, identify the virtual function that you want to assign.

```
primary# ldm list-io
NAME                                TYPE   BUS      DOMAIN STATUS
-----
pci_0                               BUS    pci_0   primary  IOV
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0       PF     pci_0   primary
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0   VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1   VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2   VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3   VF     pci_0
```

To add a virtual function to an I/O domain, it must be unassigned. The DOMAIN column indicates the name of the domain to which the virtual function is assigned. In this case, the /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2 is not assigned to a domain.

To add a virtual function to a domain, the domain must be in the inactive or bound state.

```
primary# ldm list-domain
NAME      STATE    FLAGS   CONS   VCPU   MEMORY   UTIL   NORM   UPTIME
primary   active   -n-cv-  UART   32     64G      0.2%  0.2%   56m
iodom1   active   -n----  5000   8      8G       33%   33%   25m
```

The `ldm list-domain` output shows that the `iodom1` I/O domain is active, so it must be stopped.

```
primary# ldm stop iodom1
LDom iodom1 stopped
primary# ldm list-domain
NAME      STATE    FLAGS   CONS   VCPU   MEMORY   UTIL   NORM   UPTIME
primary   active   -n-cv-  UART   32     64G      0.0%  0.0%   57m
iodom1   bound    -----  5000   8      8G       0.0%  0.0%   57m
```

Now you can add the virtual function to the I/O domain.

```
primary# ldm add-io /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2 iodom1
primary# ldm list-io
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2           VF      pci_0      iodom1
```

Note that you can add more than one virtual function while an I/O domain is stopped. For example, you might add other unassigned virtual functions such as /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3 to `iodom1`. After you add the virtual functions, you can restart the I/O domain.

```
primary# ldm start iodom1
LDom iodom1 started
primary# ldm list-domain
NAME      STATE    FLAGS   CONS   VCPU   MEMORY   UTIL   NORM   UPTIME
primary   active   -n-cv-  UART   32     64G      1.0%  1.0%   1h 18m
iodom1   active   -n----  5000   8      8G       36%   36%   1m
```

▼ How to Remove an InfiniBand Virtual Function From an I/O Domain

This procedure describes how to remove an InfiniBand SR-IOV virtual function from an I/O domain.

1. Stop the I/O domain.

```
primary# ldm stop-domain domain-name
```

2. Remove one or more virtual functions from the I/O domain.

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the device pseudonym. *domain-name* specifies the name of the domain from which you remove the virtual function. The specified I/O domain must be in the inactive or bound state.

Note - Before removing the virtual function from the I/O domain, ensure that it is not critical for booting that domain.

```
primary# ldm remove-io vf-name domain-name
```

3. Start the I/O domain.

```
primary# ldm start-domain domain-name
```

Example 8-17 Removing an InfiniBand Virtual Function

The following example shows how to remove the /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2 virtual function from the iodom1 I/O domain.

First, identify the virtual function that you want to remove.

```
primary# ldm list-io
NAME                      TYPE   BUS      DOMAIN STATUS
-----
pci_0                     BUS    pci_0   primary  IOV
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0          VF     pci_0   primary
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0      VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1      VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2      VF     pci_0   iodom1
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3      VF     pci_0   iodom1
```

The DOMAIN column shows the name of the domain to which the virtual function is assigned. The /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2 virtual function is assigned to iodom1.

To remove a virtual function from an I/O domain, the domain must be inactive or bound state. Use the `ldm list-domain` command to determine the state of the domain.

```
primary# ldm list-domain
NAME      STATE   FLAGS  CONS   VCPU  MEMORY   UTIL  NORM  UPTIME
primary   active  -n-cv-  UART   32    64G     0.3%  0.3%  29m
iodom1   active  -n----  5000   8     8G      17%   17%   11m
```

In this case, the iodom1 domain is active and so must be stopped.

```
primary# ldm stop iodom1
LDOM iodom1 stopped
primary# ldm list-domain
NAME      STATE   FLAGS  CONS   VCPU  MEMORY   UTIL  NORM  UPTIME
```

```
primary      active      -n-cv-  UART    32    64G    0.0%  0.0%  31m
iodom1      bound       -----  5000     8     8G
```

Now you can remove the /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2 virtual function from iodom1.

```
primary# ldm remove-io /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2 iodom1
primary# ldm list-io
NAME                      TYPE   BUS    DOMAIN STATUS
----                      ---   ---  -----
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2      VF     pci_0
...
```

Note that the DOMAIN column for the virtual function is now empty.

You can remove more than one virtual function while an I/O domain is stopped. In this example, you could also remove the /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3 virtual function. After you remove the virtual functions, you can restart the I/O domain.

```
primary# ldm start iodom1
LDom iodom1 started
primary# ldm list-domain
NAME      STATE      FLAGS  CONS   VCPU  MEMORY  UTIL  NORM  UPTIME
primary   active     -n-cv-  UART    32    64G    0.3%  0.3%  39m
iodom1   active     -n----  5000     8     8G    9.4%  9.4%  5s
```

Adding and Removing InfiniBand Virtual Functions to Root Domains

▼ How to Add an InfiniBand Virtual Function to a Root Domain

This procedure describes how to add an InfiniBand SR-IOV virtual function to a root domain.

1. **Initiate a delayed reconfiguration.**

```
primary# ldm start-reconf root-domain
```

2. **Add one or more virtual functions to the root domain.**

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the pseudonym name. *root-domain-name* specifies the name of the root domain to which you add the virtual function.

```
primary# ldm add-io vf-name root-domain-name
```

3. **Reboot the root domain.**

Run one of the following commands:

- **Reboot the non-primary root domain.**

```
primary# ldm stop-domain -r root-domain-name
```

- **Reboot the primary root domain.**

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

▼ How to Remove an InfiniBand Virtual Function From a Root Domain

This procedure describes how to remove an InfiniBand SR-IOV virtual function from a root domain.

1. **Initiate a delayed reconfiguration.**

```
primary# ldm start-reconf root-domain
```

2. **Remove one or more virtual functions from the root domain.**

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the pseudonym name. *root-domain-name* specifies the name of the root domain to which you add the virtual function.

```
primary# ldm remove-io vf-name root-domain-name
```

3. **Reboot the root domain.**

Run one of the following commands:

- **Reboot the non-primary root domain.**

```
primary# ldm stop-domain -r root-domain-name
```

- **Reboot the primary root domain.**

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

Advanced SR-IOV Topics: InfiniBand SR-IOV

This section describes how to identify InfiniBand physical and virtual functions as well as to correlate the Logical Domains Manager and the Oracle Solaris view of InfiniBand physical and virtual functions.

Listing InfiniBand SR-IOV Virtual Functions

The following example shows different ways to display information about the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0` physical function. A physical function name that includes the `IOVIB` string indicates that it is an InfiniBand SR-IOV device.

```
primary# ldm list-io
NAME                      TYPE   BUS      DOMAIN  STATUS
----                      ----   ---      -----  -----
pci_0                     BUS    pci_0   primary  IOV
niu_0                     NIU    niu_0   primary
/SYS/MB/RISER0/PCIE0      PCIE   pci_0   primary  EMP
/SYS/MB/RISER1/PCIE1      PCIE   pci_0   primary  EMP
/SYS/MB/RISER2/PCIE2      PCIE   pci_0   primary  EMP
/SYS/MB/RISER0/PCIE3      PCIE   pci_0   primary  OCC
/SYS/MB/RISER1/PCIE4      PCIE   pci_0   primary  OCC
/SYS/MB/RISER2/PCIE5      PCIE   pci_0   primary  EMP
/SYS/MB/SASHBA0           PCIE   pci_0   primary  OCC
/SYS/MB/SASHBA1           PCIE   pci_0   primary  OCC
/SYS/MB/NET0               PCIE   pci_0   primary  OCC
/SYS/MB/NET2               PCIE   pci_0   primary  OCC
/SYS/MB/RISER0/PCIE3/IOVIB.PF0 PF    pci_0   primary
/SYS/MB/RISER1/PCIE4/IOVIB.PF0 PF    pci_0   primary
/SYS/MB/NET0/IONET.PF0     PF    pci_0   primary
/SYS/MB/NET0/IONET.PF1     PF    pci_0   primary
/SYS/MB/NET2/IONET.PF0     PF    pci_0   primary
/SYS/MB/NET2/IONET.PF1     PF    pci_0   primary
/SYS/MB/RISER0/PCIE3/IOVIB.PF0.VF0 VF    pci_0   primary
/SYS/MB/RISER0/PCIE3/IOVIB.PF0.VF1 VF    pci_0   primary
/SYS/MB/RISER0/PCIE3/IOVIB.PF0.VF2 VF    pci_0   iodom1
/SYS/MB/RISER0/PCIE3/IOVIB.PF0.VF3 VF    pci_0   iodom1
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0 VF    pci_0   primary
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1 VF    pci_0   primary
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2 VF    pci_0   iodom1
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3 VF    pci_0   iodom1
```

The `ldm list-io -l` command provides more detailed information about the specified physical function device, `/SYS/MB/RISER1/PCIE4/IOVIB.PF0`. The `maxvfs` value shows that the maximum number of virtual functions supported by the physical device is 64. For each virtual function that is associated with the physical function, the output shows the following:

- Function name
- Function type
- Bus name
- Domain name
- Optional status of the function
- Device path

This `ldm list-io -l` output shows that `VF0` and `VF1` are assigned to the `primary` domain and that `VF2` and `VF3` are assigned to the `iodom1` I/O domain.

```
primary# ldm list-io -l /SYS/MB/RISER1/PCIE4/IOVIB.PF0
NAME                      TYPE   BUS    DOMAIN  STATUS
-----
/SYS/MB/RISER1/PCIE4/IOVIB.PF0      PF    pci_0  primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0]
maxvfs = 64
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0      VF    pci_0  primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,1]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1      VF    pci_0  primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,2]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2      VF    pci_0  iodom1
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,3]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3      VF    pci_0  iodom1
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,4]
```

Identifying InfiniBand SR-IOV Functions

This section describes how to identify the InfiniBand SR-IOV devices on an Oracle Solaris 11 and an Oracle Solaris 10 system.

Use the `ldm list-io -l` command to show the Oracle Solaris device path name that is associated with each physical function and virtual function.

```
primary# ldm list-io -l /SYS/MB/RISER1/PCIE4/IOVIB.PF0
NAME                      TYPE   BUS    DOMAIN  STATUS
-----
/SYS/MB/RISER1/PCIE4/IOVIB.PF0      PF    pci_0  primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0]
maxvfs = 64
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0      VF    pci_0  primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,1]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1      VF    pci_0  primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,2]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2      VF    pci_0  iodom1
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,3]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3      VF    pci_0  iodom1
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,4]
```

Oracle Solaris 11:

Use the Oracle Solaris 11 `dladm show-phys -L` command to match each IP over InfiniBand (IPoIB) instance to its physical card. For example, the following command shows which IPoIB instances use the card in slot PCIE4, which is the same card shown in the previous `ldm list-io -l` example.

```
primary# dladm show-phys -L | grep PCIE4
net5          ibp0      PCIE4/PORT1
net6          ibp1      PCIE4/PORT2
net19         ibp8      PCIE4/PORT1
net9          ibp9      PCIE4/PORT2
```

net18	ibp4	PCIE4/PORT1
net11	ibp5	PCIE4/PORT2

Each InfiniBand host channel adapter (HCA) device has a globally unique ID (GUID). There are also GUIDs for each port (typically there are two ports to an HCA). An InfiniBand HCA GUID uniquely identifies the adapter. The port GUID uniquely identifies each HCA port and plays a role similar to a network device's MAC address. These 16-hexadecimal digit GUIDs are used by InfiniBand management tools and diagnostic tools.

Use the Oracle Solaris 11 `dladm show-ib` command to obtain GUID information about the InfiniBand SR-IOV devices. Physical functions and virtual functions for the same device have related HCA GUID values. The 11th hexadecimal digit of the HCA GUID shows the relationship between a physical function and its virtual functions. Note that leading zeros are suppressed in the HCAGUID and PORTGUID columns.

For example, physical function **PF0** has two virtual functions, **VF0** and **VF1**, which are assigned to the primary domain. The 11th hexadecimal digit of each virtual function is incremented by one from the related physical function. So, if the GUID for the **PF0** is 8, the GUIDs for **VF0** and **VF1** will be 9 and A, respectively.

The following `dladm show-ib` command output shows that the **net5** and **net6** links belong to the physical function **PF0**. The **net19** and **net9** links belong to **VF0** of the same device while the **net18** and **net11** links belong to **VF1**.

```
primary# dladm show-ib
LINK      HCAGUID      PORTGUID      PORT STATE  PKEYS
net6      21280001A17F56 21280001A17F58 2   up    FFFF
net5      21280001A17F56 21280001A17F57 1   up    FFFF
net19     21290001A17F56 14050000000001 1   up    FFFF
net9      21290001A17F56 14050000000008 2   up    FFFF
net18     212A0001A17F56 14050000000002 1   up    FFFF
net11     212A0001A17F56 14050000000009 2   up    FFFF
```

The device in the following Oracle Solaris 11 `dladm show-phys` output shows the relationship between the links and the underlying InfiniBand port devices (**ibpX**).

```
primary# dladm show-phys
LINK      MEDIA      STATE      SPEED  DUPLEX  DEVICE
...
net6      Infiniband  up       32000  unknown  ibp1
net5      Infiniband  up       32000  unknown  ibp0
net19     Infiniband  up       32000  unknown  ibp8
net9      Infiniband  up       32000  unknown  ibp9
net18     Infiniband  up       32000  unknown  ibp4
net11     Infiniband  up       32000  unknown  ibp5
```

Use the `ls -l` command to show the actual InfiniBand port (IB port) device paths. An IB port device is a child of a device path that is shown in the `ldm list-io -l` output. A physical function has a one-part unit address such as `pciex15b3,673c@0` while virtual functions have a

two-part unit address, `pcie15b3,1002@0,2`. The second part of the unit address is one higher than the virtual function number. (In this case, the second component is 2, so this device is virtual function 1.) The following output shows that `/dev/ibp0` is a physical function and `/dev/ibp5` is a virtual function.

```
primary# ls -l /dev/ibp0
lrwxrwxrwx 1 root      root          83 Apr 18 12:02 /dev/ibp0 ->
./devices/pci@400/pci@1/pci@0/pcie15b3,673c@0/hermon@0/ibport@1,0,ipib:ibp0
primary# ls -l /dev/ibp5
lrwxrwxrwx 1 root      root          85 Apr 22 23:29 /dev/ibp5 ->
./devices/pci@400/pci@1/pci@0/pcie15b3,1002@0,2/hermon@3/ibport@2,0,ipib:ibp5
```

You can use the OpenFabrics `ibv_devices` command to view the OpenFabrics device name and the node (HCA) GUID. When virtual functions are present, the Type column indicates whether the function is physical or virtual.

```
primary# ibv_devices
device           node GUID        type
-----
mlx4_4          0002c90300a38910  PF
mlx4_5          0021280001a17f56  PF
mlx4_0          0002cb0300a38910  VF
mlx4_1          0002ca0300a38910  VF
mlx4_2          00212a0001a17f56  VF
mlx4_3          0021290001a17f56  VF
```

Oracle Solaris 10:

In an Oracle Solaris 10 guest I/O domain, use the `dladm show-dev` command to show each IPoIB instance, which has a name in the form `ibdxx`.

```
# dladm show-dev
vnet0      link: up      speed: 0      Mbps      duplex: unknown
ibd0      link: up      speed: 32000  Mbps      duplex: unknown
ibd1      link: up      speed: 32000  Mbps      duplex: unknown
ibd2      link: up      speed: 32000  Mbps      duplex: unknown
ibd3      link: up      speed: 32000  Mbps      duplex: unknown
```

You can use the `ls -l` command on the HCA path names in the `/devices/` directory to extract an HCA and its HCA GUID.

```
# ls -l /devices/ib\:[0-9]*
crw-r--r-- 1 root      sys      67,  0 Jun 12 16:27 /devices/ib:212B0001A17F56
crw-r--r-- 1 root      sys      67,  0 Jun 12 16:27 /devices/ib:212C0001A17F56
```

The GUIDs in the `ibv_devices` output (note the 11th hexadecimal digit, "B" and "C" in this case) indicate that these are virtual functions assigned to the Oracle Solaris 10 domain. You can obtain more information about the IPoIB instances by using the `ls -l` command on the `/dev` IPoIB path names.

```
# ls -l /dev/ibd*
```

```
lrwxrwxrwx 1 root      other        29 May 23 16:26 /dev/ibd ->
./devices/pseudo/clone@0:ibd
lrwxrwxrwx 1 root      root         89 May 31 10:52 /dev/ibd0 ->
./devices/pci@400/pci@1/pci@0/pci@0/pciex15b3,1002@0,3/hermon@0/ibport@1,ffff,ipib:ibd0
lrwxrwxrwx 1 root      root         89 May 31 10:52 /dev/ibd1 ->
./devices/pci@400/pci@1/pci@0/pci@0/pciex15b3,1002@0,3/hermon@0/ibport@2,ffff,ipib:ibd1
lrwxrwxrwx 1 root      root         89 Jun 12 18:36 /dev/ibd2 ->
./devices/pci@400/pci@1/pci@0/pci@0/pciex15b3,1002@0,4/hermon@1/ibport@1,ffff,ipib:ibd2
lrwxrwxrwx 1 root      root         89 Jun 12 18:36 /dev/ibd3 ->
./devices/pci@400/pci@1/pci@0/pci@0/pciex15b3,1002@0,4/hermon@1/ibport@2,ffff,ipib:ibd3
```

Each path begins with the device path that is shows in the `ldm list-io -l` output. The virtual functions such as `pciex15b3,1002@0,4` have a two-part unit address where the second part of the unit address is one higher than the virtual function number (in this case, VF3).

The `ibport` device has a three-part unit address followed by a colon and then the IPoIB device instance name. The first part of the unit address is the port number. The second is the partition key (P-key) hexadecimal value. Note that InfiniBand P-key values are similar to VLANs for Ethernet. The third part is a the string `ipib`.

The `ls -l /dev/ibd3` command output shows that the `ibd3` IPoIB instance uses port 2 and P-key value `ffff`.

Using Fibre Channel SR-IOV Virtual Functions

Starting with the Oracle VM Server for SPARC 3.1.1 release provides support for SR-IOV Fibre Channel. An SR-IOV Fibre Channel host bus adapter (HBA) might have one or more ports each of which appears as SR-IOV physical function. You can identify Fibre Channel physical functions by the `IOVFC` string in its device name.

Each Fibre Channel physical function has unique port and node world-wide name (WWN) values that are provided by the card manufacturer. When you create virtual functions from a Fibre Channel physical function, the virtual functions behave like a Fibre Channel HBA device. Each virtual function must have a unique identity that is specified by the port WWN and node WWN of the SAN fabric. You can use the Logical Domains Manager to automatically or manually assign the port and node WWNs. By assigning your own values, you can fully control the identity of any virtual function.

The Fibre Channel HBA virtual functions use the N_Port ID Virtualization (NPIV) method to log in to the SAN fabric. Because of this NPIV requirement, you must connect the Fibre Channel HBA port to an NPIV-capable Fibre Channel switch. The virtual functions are managed entirely by the hardware or the firmware of the SR-IOV card. Other than these exceptions, Fibre Channel virtual functions work and behave the same way as a non-SR-IOV Fibre Channel HBA device. The SR-IOV virtual functions have the same capabilities as the non-SR-IOV devices, so all types of SAN storage devices are supported in either configuration.

The virtual functions' unique port and node WWN values enable a SAN administrator to assign storage to the virtual functions in the same way as he would for any non-SR-IOV Fibre Channel HBA port. This management includes zoning, LUN masking, and quality of service (QoS). You can configure the storage so that it is accessible exclusively to a specific logical domain without being visible to the physical function in the root domain.

You can use both the static and dynamic SR-IOV methods to manage Fibre Channel SR-IOV devices.

Fibre Channel SR-IOV Hardware Requirements

For information about the required PCIe Fibre Channel SR-IOV hardware, see “[SR-IOV Hardware and Software Requirements](#)” on page 92.

- **Control domain.**
 - **Oracle Solaris 11 OS.**
 - **QLogic cards.** At least the Oracle Solaris 11.2.0.42.0 OS
 - **Emulex cards.** At least the Oracle Solaris 11.1.17.4.0 OS
 - **Oracle Solaris 10 OS.** Only on SPARC T-Series platforms and Fujitsu M10 servers.
 - **QLogic cards.** At least the Oracle Solaris 10 1/13 OS, Oracle Solaris 10 patch ID 150400-11, plus device driver patch ID 149175-06
 - **Emulex cards.** At least the Oracle Solaris 10 1/13 OS, Oracle Solaris 10 patch ID 150400-11, plus device driver patch ID 149173-04 for Fujitsu M10 servers. Not supported for the SR-IOV feature on SPARC T-Series and SPARC M-Series systems.
- **I/O domain.**
 - **Oracle Solaris 11 OS.**
 - **QLogic cards.** At least the Oracle Solaris 11.2.0.42.0 OS
 - **Emulex cards.** At least the Oracle Solaris 11.1.17.4.0 OS
 - **Oracle Solaris 10 OS.**
 - **QLogic cards.** At least the Oracle Solaris 10 1/13 OS, Oracle Solaris 10 patch ID 150400-11, plus device driver patch ID 149175-06
 - **Emulex cards.** At least the Oracle Solaris 10 1/13 OS, Oracle Solaris 10 patch ID 150400-11, plus device driver patch ID 149173-04

Fibre Channel SR-IOV Requirements and Limitations

The Fibre Channel SR-IOV feature has the following recommendations and limitations:

- The SR-IOV card must run the latest version of firmware that supports the SR-IOV feature.
- The Fibre Channel PCIe card must be connected to a Fibre Channel switch that supports NPIV and that is compatible with the PCIe card.
- The Logical Domains Manager properly autogenerates unique `port-wwn` and `node-wwn` property values by connecting the control domains of all systems to the same SAN fabric and by being part of the same multicast domain.

If you cannot configure this environment, you must manually provide the `node-wwn` and `port-wwn` values when you create the virtual function. This behavior ensures that there are no naming conflicts. See “[World-Wide Name Allocation for Fibre Channel Virtual Functions](#)” on page 137.

Fibre Channel Device Class-Specific Properties

You can use the `ldm create-vf` or the `ldm set-io` commands to set the following Fibre Channel virtual function properties:

<code>bw-percent</code>	Specifies the percentage of the bandwidth to be allocated to the Fibre Channel virtual function. Valid values are from 0 to 100. The total bandwidth value assigned to a Fibre Channel physical function's virtual functions cannot exceed 100. The default value is 0 so that the virtual function gets a fair share of the bandwidth that is not already reserved by other virtual functions that share the same physical function.
<code>node-wwn</code>	Specifies the node world-wide name (WWN) for the Fibre Channel virtual function. Valid values are non-zero. By default, this value is allocated automatically. If you manually specify this value, you must also specify a value for the <code>port-wwn</code> property. For more information, see “ World-Wide Name Allocation for Fibre Channel Virtual Functions ” on page 137.
<code>port-wwn</code>	Specifies the port WWN for the Fibre Channel virtual function. Valid values are non-zero. By default, this value is allocated automatically. If you manually specify this value, you must also specify a value for the <code>node-wwn</code> property. For more information, see “ World-Wide Name Allocation for Fibre Channel Virtual Functions ” on page 137.

You cannot modify the `node-wwn` or the `port-wwn` property values while the Fibre Channel virtual function is in use. However, you can modify the `bw-percent` property value dynamically even when the Fibre Channel virtual function is in use.

World-Wide Name Allocation for Fibre Channel Virtual Functions

The Logical Domains Manager supports both the automatic allocation and the manual assignment of world-wide names for the Fibre Channel virtual functions.

Automatic World-Wide Name Allocation

Logical Domains Manager allocates a unique MAC address from its automatic MAC address allocation pool and creates IEEE format node-wwn and port-wwn property values.

```
port-wwn = 10:00:XX:XX:XX:XX  
node-wwn = 20:00:XX:XX:XX:XX
```

XX:XX:XX:XX:XX:XX is the automatically allocated MAC address.

This automatic allocation method produces unique WWNs when the control domains of all systems that are connected to the same Fibre Channel fabric are also connected by Ethernet and are part of the same multicast domain. If you cannot meet this requirement, you must manually assign unique WWNs, which is required on the SAN.

Manual World-Wide Name Allocation

You can construct unique WWNs by using any method. This section describes how to create WWNs from the Logical Domains Manager manual MAC address allocation pool. You must guarantee the uniqueness of the WWNs you allocate.

Logical Domains Manager has a pool of 256,000 MAC addresses that are available for manual allocation in the 00:14:4F:FC:00:00 - 00:14:4F:FF:FF:FF range.

The following example shows the port-wwn and node-wwn property values based on the 00:14:4F:FC:00:01 MAC address:

```
port-wwn = 10:00:00:14:4F:FC:00:01  
node-wwn = 20:00:00:14:4F:FC:00:01
```

00:14:4F:FC:00:01 is the manually allocated MAC address. For information about automatic MAC address allocation, see “[Assigning MAC Addresses Automatically or Manually](#)” on page 219.

Note - It is best to manually assign the WWNs to ensure predictable configuration of the SAN storage.

You must use the manual WVN allocation method when all systems are not connected to the same multicast domain by Ethernet. You can also use this method to guarantee that the same WWNs are used when Fibre Channel virtual functions are destroyed and re-created.

Creating Fibre Channel SR-IOV Virtual Functions

This section describes how to dynamically create and destroy virtual functions. If you cannot use the dynamic methods to perform these actions, initiate a delayed reconfiguration on the root domain before you create or destroy virtual functions.

▼ How to Create a Fibre Channel SR-IOV Virtual Function

If you cannot use this dynamic method, use the static method instead. See “[Static SR-IOV](#)” on page 96.

1. Identify the physical function device.

```
primary# ldm list-io
```

Note that the name of the physical function includes the location information for the PCIe SR-IOV card or on-board device.

2. If I/O virtualization for the bus that has the physical function is not enabled already, enable it.

Perform this step only if I/O virtualization is not enabled already for the bus that has the physical function.

See “[How to Enable I/O Virtualization for a PCIe Bus](#)” on page 98.

3. Create a single virtual function or multiple virtual functions from a physical function either dynamically or statically.

After you create one or more virtual functions, you can assign them to a guest domain.

■ Dynamic method:

■ To create multiple virtual functions from a physical function all at the same time, use the following command:

```
primary# ldm create-vf -n number | max pf-name
```

Use the `ldm create-vf -n max` command to create all the virtual functions for that physical function at one time. This command automatically allocates the port and node WWNs for each virtual function and sets the `bw-percent` property to the default value, which is 0. This value specifies that fair share bandwidth is allocated to all virtual functions.

Tip - Create all virtual functions for the physical function at once. If you want to manually assign WWNs, first create all of the virtual functions and then use the `ldm set-io` command to manually assign your WWN values for each virtual function. This technique minimizes the number of state transitions when creating virtual functions from a physical function.

You can use either the path name or the pseudonym name to specify virtual functions. However, the recommended practice is to use the pseudonym name.

- **To create one virtual function from a physical function, use the following command:**

```
ldm create-vf [bw-percent=value] [port-wwn=value node-wwn=value] pf-name
```

You can also manually specify Fibre Channel class-specific property values.

Note - Sometimes a newly created virtual function is not available for immediate use while the OS probes for IOV devices. Use the `ldm list-io` command to determine whether the parent physical function and its child virtual functions have the INV value in the Status column. If they have this value, wait until the `ldm list-io` output no longer shows the INV value in the Status column (about 45 seconds) before you use that physical function or any of its child virtual functions. If this status persists, there is a problem with the device.

A device status might be INV immediately following a root domain reboot (including that of the primary) or immediately after you use the `ldm create-vf` or `ldm destroy-vf` command.

- **Static method:**

- a. **Initiate a delayed reconfiguration.**

```
primary# ldm start-reconf root-domain-name
```

- b. **Create a single virtual function or multiple virtual functions from a physical function.**

Use the same commands as shown previously to dynamically create the virtual functions.

- c. **Reboot the root domain.**

- **To reboot the non-primary root domain:**

```
primary# ldm stop-domain -r root-domain
```

- **To reboot the primary root domain:**

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

Example 8-18 Displaying Information About the Fibre Channel Physical Function

This example shows information about the /SYS/MB/PCIE7/IOVFC.PF0 physical function:

- This physical function is from a board in a PCIe slot, PCIE7.
- The IOVFC string indicates that the physical function is a Fibre Channel SR-IOV device.

```
primary# ldm list-io
NAME                                     TYPE   BUS      DOMAIN  STATUS
-----
pci_0                                    BUS    pci_0    primary  IOV
pci_1                                    BUS    pci_1    rootdom1 IOV
niu_0                                    NIU    niu_0    primary
niu_1                                    NIU    niu_1    primary
/SYS/MB/PCIE0                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE2                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE4                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE6                           PCIE   pci_0    primary  EMP
/SYS/MB/PCIE8                           PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA                          PCIE   pci_0    primary  OCC
/SYS/MB/NET0                            PCIE   pci_0    primary  OCC
/SYS/MB/PCIE1                           PCIE   pci_1    rootdom1 OCC
/SYS/MB/PCIE3                           PCIE   pci_1    rootdom1 OCC
/SYS/MB/PCIE5                           PCIE   pci_1    rootdom1 OCC
/SYS/MB/PCIE7                           PCIE   pci_1    rootdom1 OCC
/SYS/MB/PCIE9                           PCIE   pci_1    rootdom1 OCC
/SYS/MB/NET2                            PCIE   pci_1    rootdom1 OCC
/SYS/MB/NET0/IONET.PF0                  PF     pci_0    primary
/SYS/MB/NET0/IONET.PF1                  PF     pci_0    primary
/SYS/MB/PCIE5/IONET.PF0                PF     pci_1    rootdom1
/SYS/MB/PCIE5/IONET.PF1                PF     pci_1    rootdom1
/SYS/MB/PCIE7/IOVFC.PF0                PF     pci_1    rootdom1
/SYS/MB/PCIE7/IOVFC.PF1                PF     pci_1    rootdom1
/SYS/MB/NET2/IONET.PF0                  PF     pci_1    rootdom1
/SYS/MB/NET2/IONET.PF1                  PF     pci_1    rootdom1
```

The following command shows more details about the specified physical function. The maxvfs value indicates the maximum number of virtual functions that is supported by the device.

```
primary# ldm list-io -l /SYS/MB/PCIE7/IOVFC.PF0
NAME                                     TYPE   BUS      DOMAIN  STATUS
-----
/SYS/MB/PCIE7/IONET.PF0                  PF     pci_0    rootdom1
[pci@400/pci@1/pci@0/pci@6/SUNW,fcdev@0]
maxvfs = 8
```

Example 8-19 Dynamically Creating a Fibre Channel Virtual Function Without Setting Optional Properties

This example dynamically creates a virtual function without setting any optional properties. In this case, the `ldm create-vf` command automatically allocates the default bandwidth percentage, port world-wide name (WWN), and node WWN values.

Ensure that I/O virtualization is enabled on the `pci_1` PCIe bus. See “[How to Enable I/O Virtualization for a PCIe Bus](#)” on page 98.

You can use the `ldm create-vf` command to create all the virtual functions from the `/SYS/MB/PCIE7/IOVFC.PF0` physical function.

```
primary# ldm create-vf -n max /SYS/MB/PCIE7/IOVFC.PF0
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF0
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF1
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF2
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF3
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF4
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF5
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF6
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF7
```

Example 8-20 Dynamically Creating a Fibre Channel Virtual Function and Setting Properties

This example dynamically creates a virtual function while setting the `bw-percent` property value to 25 and specifies the port and node WWNs.

```
primary# ldm create-vf port-wwn=10:00:00:14:4F:FC:00:01 \
node-wwn=20:00:00:14:4F:FC:00:01 bw-percent=25 /SYS/MB/PCIE7/IOVFC.PF0
```

Example 8-21 Statically Creating a Fibre Channel Virtual Function Without Setting Optional Properties

This example statically creates a virtual function without setting any optional properties. In this case, the `ldm create-vf` command automatically allocates the default bandwidth percentage, port world-wide name (WWN), and node WWN values.

First you initiate a delayed reconfiguration on the `rootdom1` domain. Then, enable I/O virtualization on the `pci_1` PCIe bus. Because the `pci_1` bus has already been assigned to the `rootdom1` root domain, use the `ldm set-io` command to enable I/O virtualization.

```
primary# ldm start-reconf rootdom1
Initiating a delayed reconfiguration operation on the rootdom1 domain.
All configuration changes for other domains are disabled until the rootdom1
domain reboots, at which time the new configuration for the rootdom1 domain
will also take effect.
```

```
primary# ldm set-io iov=on pci_1
```

Now, you can use the `ldm create-vf` command to create all the virtual functions from the `/SYS/MB/PCIE7/IOVFC.PF0` physical function.

```
primary# ldm create-vf -n max /SYS/MB/PCIE7/IOVFC.PF0

-----
Notice: The rootdom1 domain is in the process of a delayed reconfiguration.
Any changes made to the rootdom1 domain will only take effect after it reboots.
-----

Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF0
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF1
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF2
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF3
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF4
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF5
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF6
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF7
```

Finally, reboot the rootdom1 root domain to make the changes take effect in one of the following ways:

- rootdom1 is a non-primary root domain

```
primary# ldm stop-domain -r rootdom1
```
- rootdom1 is the primary domain

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

Destroying Fibre Channel SR-IOV Virtual Functions

A virtual function can be destroyed if it is not currently assigned to a domain. A virtual function can be destroyed only in the reverse sequential order of creation, so only the last virtual function that was created can be destroyed. The resulting configuration is validated by the physical function driver.

▼ How to Destroy a Fibre Channel SR-IOV Virtual Function

If you cannot use this dynamic method, use the static method instead. See “[Static SR-IOV](#)” on page 96.

1. Identify the physical function device.

```
primary# ldm list-io
```

2. Destroy a single virtual function or multiple virtual functions either dynamically or statically.

- Dynamic method:

- **To destroy all of the virtual functions from a physical function at one time, use the following command:**

```
primary# ldm destroy-vf -n number | max pf-name
```

You can use either the path name or the pseudonym name to specify virtual functions. However, the recommended practice is to use the pseudonym name.

Use the `ldm destroy-vf -n max` command to destroy all the virtual functions for that physical function at one time.

If you specify *number* as an argument to the *-n* option, the last *number* of virtual functions are destroyed. Use this method as it performs this operation with only one physical function device driver state transition.

- **To destroy a specified virtual function:**

```
primary# ldm destroy-vf vf-name
```

Due to delays in the affected hardware device and in the OS, the affected physical function and any remaining child virtual functions might not be available for immediate use. Use the `ldm list-io` command to determine whether the parent physical function and its child virtual functions have the INV value in the Status column. If they have this value, wait until the `ldm list-io` output no longer shows the INV value in the Status column (about 45 seconds). At that time, you can safely use that physical function or any of its child virtual functions. If this status persists, there is a problem with the device.

A device status might be INV immediately following a root domain reboot (including that of the primary) or immediately after you use the `ldm create-vf` or `ldm destroy-vf` command.

- **Static method:**

- a. **Initiate a delayed reconfiguration.**

```
primary# ldm start-reconf root-domain-name
```

- b. **Destroy either a single virtual function or multiple virtual functions.**

- **To destroy all of the virtual functions from the specified physical function at the same time, use the following command:**

```
primary# ldm destroy-vf -n number | max pf-name
```

You can use either the path name or the pseudonym name to specify virtual functions. However, the recommended practice is to use the pseudonym name.

- **To destroy a specified virtual function:**

```
primary# ldm destroy-vf vf-name
```

- c. **Reboot the root domain.**

- **To reboot the non-primary root domain:**

```
primary# ldm stop-domain -r root-domain
```

- **To reboot the primary root domain:**

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

Example 8-22 Dynamically Destroying Multiple Fibre Channel SR-IOV Virtual Functions

This example shows the results of destroying all the virtual functions from the /SYS/MB/PCIE5/IOVFC.PF1 physical function. The ldm list-io output shows that the physical function has eight virtual functions. The ldm destroy-vf -n max command destroys all the virtual functions, and the final ldm list-io output shows that none of the virtual functions remain.

```
primary# ldm list-io
...
/SYS/MB/PCIE5/IOVFC.PF1          PF      pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF0      VF      pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF1      VF      pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF2      VF      pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF3      VF      pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF4      VF      pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF5      VF      pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF6      VF      pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF7      VF      pci_1
primary# ldm destroy-vf -n max /SYS/MB/PCIE5/IOVFC.PF1
primary# ldm list-io
...
/SYS/MB/PCIE5/IOVFC.PF1          PF      pci_1
```

Example 8-23 Destroying a Fibre Channel Virtual Function

This example shows how to statically destroy the virtual functions from the /SYS/MB/PCIE7/IOVFC.PF0 physical function.

```
primary# ldm start-reconf rootdom1
Initiating a delayed reconfiguration operation on the rootdom1 domain.
All configuration changes for other domains are disabled until the rootdom1
domain reboots, at which time the new configuration for the rootdom1 domain
will also take effect.
```

```
primary# ldm destroy-vf -n max /SYS/MB/PCIE7/IOVFC.PF0
primary# ldm stop-domain -r rootdom1
```

Modifying Fibre Channel SR-IOV Virtual Functions

The `ldm set-io` command modifies the current configuration of a virtual function by changing the property values or by setting new properties.

If you cannot use this dynamic method, use the static method instead. See “[Static SR-IOV](#)” on page 96.

You can use the `ldm set-io` command to modify the `bw-percent`, `port-wwn`, and `node-wwn` properties.

You can dynamically change only the `bw-percent` property while the virtual functions are assigned to a domain.

▼ How to Modify Fibre Channel SR-IOV Virtual Function Properties

1. Identify the physical function device.

```
primary# ldm list-io
```

Note that the name of the physical function includes the location information for the PCIe SR-IOV card or on-board device.

2. Modify a virtual function property.

```
ldm set-io [bw-percent=value] [port-wwn=value node-wwn=value] pf-name
```

Unlike the `bw-percent` property value, which you can dynamically change at any time, you can dynamically modify the `port-wwn` and `node-wwn` property values only when the virtual function is not assigned to a domain.

Example 8-24 Modifying Fibre Channel SR-IOV Virtual Function Properties

This example modifies the properties of the specified virtual function, `/SYS/MB/PCIE7/IOVFC.PF0.VF0`, to specify the bandwidth percentage and the port and node WWN values.

```
primary# ldm set-io port-wwn=10:00:00:14:4f:fc:f4:7c \
node-wwn=20:00:00:14:4f:fc:f4:7c bw-percent=25 /SYS/MB/PCIE7/IOVFC.PF0.VF0
```

Adding and Removing Fibre Channel SR-IOV Virtual Functions on I/O Domains

▼ How to Add a Fibre Channel SR-IOV Virtual Function to an I/O Domain

If you cannot dynamically remove the virtual function, use the static method. See “[Static SR-IOV](#)” on page 96.

1. Identify the virtual function that you want to add to an I/O domain.

```
primary# ldm list-io
```

2. Add a virtual function either dynamically or statically.

- To dynamically add a virtual function:

```
primary# ldm add-io vf-name domain-name
```

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the pseudonym name. *domain-name* specifies the name of the domain to which you add the virtual function.

The device path name for the virtual function in the domain is the path shown in the `list-io -l` output.

- To statically add a virtual function:

- a. Stop the domain and then add the virtual function.

```
primary# ldm stop-domain domain-name  
primary# ldm add-io vf-name domain-name
```

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the pseudonym name. *domain-name* specifies the name of the domain to which you add the virtual function. The specified guest must be in the inactive or bound state.

The device path name for the virtual function in the domain is the path shown in the `list-io -l` output.

- b. Restart the domain.

```
primary# ldm start-domain domain-name
```

Example 8-25 Adding a Fibre Channel Virtual Function

This example shows how to dynamically add the /SYS/MB/PCIE7/IOVFC.PF0.VF0 virtual function to the ldg2 domain.

```
primary# ldm add-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
```

If you cannot add the virtual function dynamically, use the static method:

```
primary# ldm stop-domain ldg2
primary# ldm add-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
primary# ldm start-domain ldg2
```

▼ How to Remove a Fibre Channel SR-IOV Virtual Function From an I/O Domain

If you cannot use this dynamic method, use the static method instead. See “[Static SR-IOV](#)” on page 96.



Caution - Before removing the virtual function from the domain, ensure that it is not critical for booting that domain.

1. **Identify the virtual function that you want to remove from an I/O domain.**

```
primary# ldm list-io
```

2. **Remove a virtual function either dynamically or statically.**

- **To dynamically remove a virtual function:**

```
primary# ldm remove-io vf-name domain-name
```

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the device pseudonym. *domain-name* specifies the name of the domain from which you remove the virtual function.

- **To statically remove a virtual function:**

- a. **Stop the I/O domain.**

```
primary# ldm stop-domain domain-name
```

- b. **Remove the virtual function.**

```
primary# ldm remove-io vf-name domain-name
```

vf-name is the pseudonym name or the path name of the virtual function. The recommended practice is to use the device pseudonym. *domain-name* specifies the

name of the domain from which you remove the virtual function. The specified guest must be in the inactive or bound state.

c. **Start the I/O domain.**

```
primary# ldm start-domain domain-name
```

Example 8-26 Dynamically Removing a Fibre Channel Virtual Function

This example shows how to dynamically remove the /SYS/MB/PCIE7/IOVFC.PF0.VF0 virtual function from the ldg2 domain.

```
primary# ldm remove-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
```

If the command succeeds, the virtual function is removed from the ldg2 domain. When ldg2 is restarted, the specified virtual function no longer appears in that domain.

If you cannot remove the virtual function dynamically, use the static method:

```
primary# ldm stop-domain ldg2
primary# ldm remove-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
primary# ldm start-domain ldg2
```

Advanced SR-IOV Topics: Fibre Channel SR-IOV

This section describes some advanced topics related to using Fibre Channel SR-IOV virtual functions.

Accessing a Fibre Channel Virtual Function in a Guest Domain

The ldg2 console log shows the operations of the assigned Fibre Channel virtual function device. Use the fcadm command to view and access the Fibre Channel virtual function device.

```
ldg2# fcadm hba-port
HBA Port WWN: 100000144ffb8a99
  Port Mode: Initiator
  Port ID: 13d02
  OS Device Name: /dev/cfg/c3
  Manufacturer: Emulex
  Model: 7101684
  Firmware Version: 7101684 1.1.60.1
  FCode/BIOS Version: Boot:1.1.60.1 Fcode:4.03a4
  Serial Number: 4925382+133400002R
  Driver Name: emlx
  Driver Version: 2.90.15.0 (2014.01.22.14.50)
  Type: N-port
```

```

State: online
Supported Speeds: 4Gb 8Gb 16Gb
Current Speed: 16Gb
Node WNN: 200000144ffb8a99
NPIV Not Supported

```

Use the **format** command to show the visible LUNs.

```

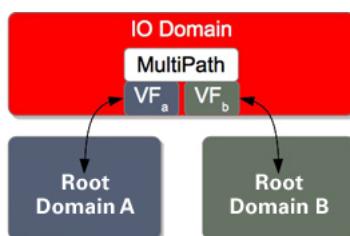
lbg2# format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
 0. c2d0 <Unknown-Unknown-0001-25.00GB>
   /virtual-devices@100/channel-devices@200/disk@0
 1. c3t21000024FF4C4BF8d0 <SUN-COMSTAR-1.0-10.00GB>
   /pci@340/pci@1/pci@0/pci@6/SUNW,emlxs@0,2/fp@0,0/ssd@w21000024ff4c4bf8,0
Specify disk (enter its number): ^D
lbg2#

```

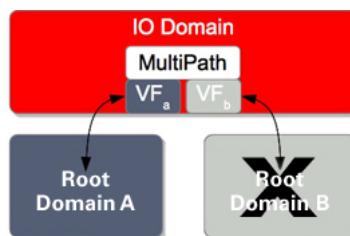
I/O Domain Resiliency

I/O domain resiliency improves the availability and performance of an I/O domain by enabling it to continue to run even when one of its associated root domains is interrupted. When a root domain is interrupted, the I/O domains that use its services continue to run by enabling its affected devices to fail over to the alternate I/O path. When the root domain returns to service, the affected devices in the resilient I/O domain are also returned to service and the failover capabilities are restored.

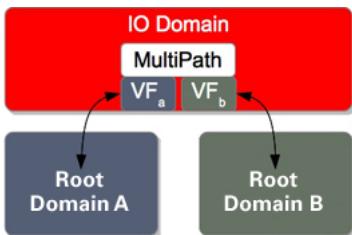
The following diagrams show and describe what happens when one of the configured root domains fails and what happens when the root domain returns to service.



Each root domain provides a virtual function to the I/O domain. The I/O domain uses virtual device multipathing such as IPMP for network devices and MPxIO for Fibre Channel devices.



When root domain B is interrupted by a panic or reboot, virtual function B is suspended in the I/O domain and then multipathing engages to route all I/O through root domain A.



When root domain B is restored to service, virtual function B resumes operation in the I/O domain. The multipath group is restored to full redundancy.

In this configuration, the virtual function could be a vnet device, which means that the I/O domain can be configured with any combination of virtual functions or vnet devices. Note that the Fibre Channel devices can only be virtual functions.

You can create a configuration where you have both resilient and non-resilient I/O domains. For an example, see “[Example – Using Resilient and Non-Resilient Configurations](#)” on page 155.

Resilient I/O Domain Requirements

A resilient I/O domain must meet the following requirements:

- Runs at least the Oracle Solaris 11.2.8.0.0 (SRU 8) OS and its primary domain runs at least the Oracle VM Server for SPARC 3.2 software.
- Uses multipathing to create failover configurations for virtual functions and virtual devices. For network multipathing, the IPMP group can have virtual functions and vnet devices.
- Has the `master` property value set to the name of a root domain whose `failure-policy` property is set to `ignore`. Any other failure policy setting, such as `stop`, `reset`, or `panic`, supersedes I/O resiliency and the I/O domain is interrupted.
- Uses only SR-IOV virtual functions that support I/O domain resiliency. See <https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&id=1325454.1>.

Note - If the I/O domain only uses I/O-resilient devices but does not configure multipathing, the applications are impacted until the root domain returns to service and the affected devices are restored.

I/O Domain Resiliency Limitations

- When you hotplug an SR-IOV card to the root domain and then assign virtual functions to it to an I/O domain, the I/O domain might fail to provide resiliency when the root domain

fails. Therefore, you should only add the SR-IOV card while the root domain is down. Then assign the virtual functions after the root domain boots.

- If you have a resilient I/O domain and then assign a device in one of the following ways, then the I/O domain is no longer resilient:
 - Add a virtual function from a card that does not support I/O resiliency
 - Directly assign a device by using the direct I/O feature

In that case, set the failure-policy from ignore to reset or stop.

Configuring Resilient I/O Domains

▼ How to Configure a Resilient I/O Domain

Before You Begin Use only the PCIe cards that support the I/O domain resiliency feature. See <https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&id=1325454.1>.

Ensure that the I/O domain, root domain, service domain, and primary domain run at least the Oracle Solaris 11.2.8.0.0 (SRU 8) OS and the Logical Domains Manager 3.2 software.

1. **On the root domain, set the failure-policy property to ignore.**

```
primary# ldm set-domain failure-policy=ignore root-domain-name
```

Note - If you add any devices to the I/O domain that are not supported for resiliency, that domain is no longer resilient. So, reset the failure-policy property value to stop, reset, or panic.

For information about domain dependencies, see “[Configuring Domain Dependencies](#)” on page 339.

2. **On the I/O domain, set the master property to the name of the root domain.**

```
primary# ldm set-domain master=root-domain-name I/O-domain-name
```

3. **Configure multipathing across the paths.**

- **Ethernet. Use IPMP to configure multipathing across the paths.**

For information about using IPMP to configure multipathing, see [Administering TCP/IP Networks, IPMI, and IP Tunnels in Oracle Solaris 11.2](#).

- **Fibre Channel. Use MPxIO to configure multipathing across the paths.**

For information about using MPxIO to configure multipathing, see [Managing SAN Devices and Multipathing Oracle Solaris 11.2](#).

Example 8-27 Using IPMP to Configure Multipathing With Ethernet SR-IOV Functions

This example shows how to use IPMP to configure network virtual-function devices for a resilient I/O domain. For more information, see [Administering TCP/IP Networks, IPMI, and IP Tunnels in Oracle Solaris 11.2](#).

1. Identify two Ethernet SR-IOV physical functions that are assigned to different root domains.

In this example, the `root-1` and `root-2` root domains have Ethernet SR-IOV physical functions.

```
primary# ldm list-io | grep root-1 | grep PF
/SYS/PCI-EM8/IOVNET.PF0          PF      pci_1    root-1
primary# ldm list-io | grep root-2 | grep PF
/SYS/RIO/NET2/IOVNET.PF0          PF      pci_2    root-2
```

2. Create two Ethernet virtual functions on each of the specified physical functions.

```
primary# ldm create-vf /SYS/MB/NET0/IOVNET.PF0
Created new vf: /SYS/PCI-EM8/IOVNET.PF0.VF0
primary# ldm create-vf /SYS/RIO/NET2/IOVNET.PF0
Created new vf: /SYS/RIO/NET2/IOVNET.PF0.VF0
```

3. Assign the Ethernet virtual functions to the `io-1` I/O domain.

```
primary# ldm add-io /SYS/PCI-EM8/IOVNET.PF0.VF0 io-1
primary# ldm add-io /SYS/RIO/NET2/IOVNET.PF0.VF0 io-1
```

4. Configure the Ethernet virtual functions into an IPMP group on the I/O domain.

- a. Identify the newly added network devices, `net1` and `net2`, on the I/O domain.

```
io-1# dladm show-phys
LINK      MEDIA     STATE   SPPEED  DUPLEX   DEVICE
net0      Ethernet   up      0        unknown  vnet0
net1      Ethernet   up      1000    full     igbvf0
net2      Ethernet   up      1000    full     igbvf1
```

- b. Create IP interfaces for the newly added network devices.

```
io-1# ipadm create-ip net1
io-1# ipadm create-ip net2
```

- c. Create the `ipmp0` IPMP group for the two network interfaces.

```
io-1# ipadm create-ipmp -i net1 -i net2 ipmp0
```

- d. Assign an IP address to the IPMP group.

This example configures the DHCP option.

```
io-1# ipadm create-addr -T dhcp ipmp0/v4
e. Check the status of the IPMP group interface.
```

```
io-1# ipmpstat -g
```

Example 8-28 Using MPxIO to Configure Multipathing With Fibre Channel SR-IOV Functions

This example shows how to use MPxIO to configure Fibre Channel virtual-function devices for a resilient I/O domain. For more information, see *Managing SAN Devices and Multipathing Oracle Solaris 11.2*.

1. Identify two Fibre Channel SR-IOV physical functions that are assigned to different root domains.

In this example, the root-1 and root-2 root domains have Fibre Channel SR-IOV physical functions.

```
primary# ldm list-io | grep root-1 | grep PF
/SYS/PCI-EM4/IOVFC.PF0          PF      pci_1    root-1
primary# ldm list-io | grep root-2 | grep PF
/SYS/PCI-EM15/IOVFC.PF0         PF      pci_2    root-2
```

2. Create two virtual functions on each of the specified physical functions.

For more information, see “[How to Create a Fibre Channel SR-IOV Virtual Function](#)” on page 138.

```
primary# ldm create-vf port-wwn=10:00:00:14:4f:fc:60:00 \
node-wwn=20:00:00:14:4f:fc:60:00 /SYS/PCI-EM4/IOVFC.PF0
Created new vf: /SYS/PCI-EM4/IOVFC.PF0.VF0
primary# ldm create-vf port-wwn=10:00:00:14:4f:fc:70:00 \
node-wwn=20:00:00:14:4f:fc:70:00 /SYS/PCI-EM15/IOVFC.PF0
Created new vf: /SYS/PCI-EM15/IOVFC.PF0.VF0
```

3. Add the newly created virtual functions to the io-1 I/O domain.

```
primary# ldm add-io /SYS/PCI-EM4/IOVFC.PF0.VF0 io-1
primary# ldm add-io /SYS/PCI-EM15/IOVFC.PF0.VF0 io-1
```

4. Determine whether MPxIO is enabled on the I/O domain by using the `prtconf -v` command.

If the output for the `fp` device includes the following device property setting, MPxIO is enabled:

```
mpxio-disable="no"
```

If the `mpxio-disable` property is set to yes, update the property value to no in the `/etc/driver/driv/fp.conf` file and then reboot the I/O domain.

If the `mpxio-disable` device property does not appear in the `prtconf -v` output, add the `mpxio-disable="no"` entry to the `/etc/driver/drv/fp.conf` file and then reboot the I/O domain.

5. Check the status of MPxIO group.

```
io-1# mpathadm show LU

Logical Unit: /dev/rdsk/c0t600A0B80002A384600003D6B544EECD0d0s2
  mpath-support: libmpscsi_vhci.so
  Vendor: SUN
  Product: CSM200_R
  Revision: 0660
  Name Type: unknown type
  Name: 600a0b80002a384600003d6b544eecd0
  Asymmetric: yes
  Current Load Balance: round-robin
  Logical Unit Group ID: NA
  Auto Failback: on
  Auto Probing: NA

  Paths:
    Initiator Port Name: 100000144ffc6000
    Target Port Name: 201700a0b82a3846
    Override Path: NA
    Path State: OK
    Disabled: no

    Initiator Port Name: 100000144ffc7000
    Target Port Name: 201700a0b82a3846
    Override Path: NA
    Path State: OK
    Disabled: no

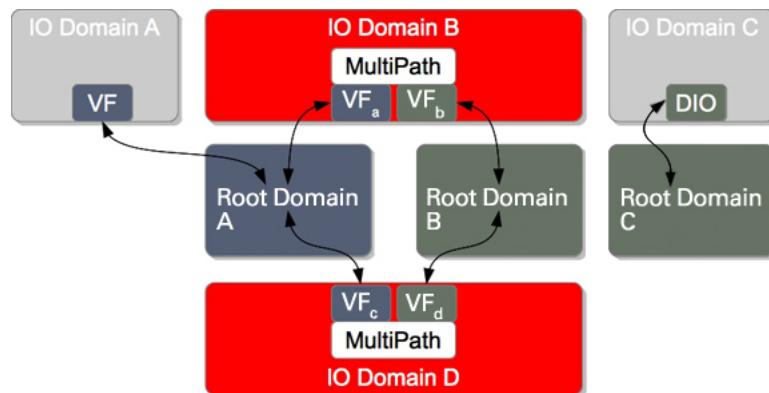
  Target Port Groups:
    ID: 1
    Explicit Failover: yes
    Access State: active
    Target Ports:
      Name: 201700a0b82a3846
      Relative ID: 0
```

Example – Using Resilient and Non-Resilient Configurations

You can use configurations with both resilient and non-resilient domains.

The following figure shows that I/O domain A and I/O domain C are not resilient because neither use multipathing. I/O domain A has a virtual function and I/O domain C has a direct I/O device.

FIGURE 8-2 Configuration With Resilient and Non-Resilient I/O Domains



I/O domain B and I/O domain D are resilient. I/O domains A, B, and D depend on root domain A. I/O domains B and D depend on root domain B. I/O domain C depends on root domain C.

If root domain A is interrupted, I/O domain A is interrupted as well. I/O domains B and D, however, fail over to alternate paths and continue to run applications. If root domain C is interrupted, I/O domain C fails in the way specified by the `failure-policy` property value of root domain C.

Rebooting the Root Domain With Non-Resilient I/O Domains Configured

Note - If your I/O domain is resilient, it can continue to operate even when the root domain that services it is interrupted. For information about configuring resilient I/O domains, see “[I/O Domain Resiliency](#)” on page 149.

As with PCIe slots in the I/O domain, the concerns that are described in “[Rebooting the Root Domain With PCIe Endpoints Configured](#)” on page 78 also pertain to the virtual functions that are assigned to an I/O domain.

Note - An I/O domain cannot start if the associated root domain is not running.

Using Non-primary Root Domains

This chapter covers the following non-primary root domain topics:

- “[Non-primary Root Domains Overview](#)” on page 157
- “[Non-primary Root Domain Requirements](#)” on page 158
- “[Non-primary Root Domain Limitations](#)” on page 159
- “[Non-primary Root Domain Examples](#)” on page 160

Non-primary Root Domains Overview

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 **primary** domain is the default root domain.

You can perform direct I/O and SR-IOV operations on PCIe buses that are assigned to any root domain. You can now perform the following operations for all root domains, including non-primary root domains:

- Show the status of PCIe slots
- Show the SR-IOV physical functions that are present
- Assign a PCIe slot to an I/O domain or a root domain
- Remove a PCIe slot from an I/O domain or a root domain
- Create a virtual function from its physical function
- Destroy a virtual function
- Assign a virtual function to another domain
- Remove a virtual function from another domain

The Logical Domains Manager obtains the PCIe endpoint devices and SR-IOV physical function devices from the Logical Domains agents that run in the non-primary root domains. This information is cached while the root domain is down after it is first discovered but only until the root domain is booted.

You can perform direct I/O and SR-IOV operations only when the root domain is active. Logical Domains Manager operates on the actual devices that are present at that time. The cached data might be refreshed when the following operations occur:

- The Logical Domains agent is restarted in the specified root domain
- A hardware change, such as a hot-plug operation, is performed in the specified root domain

Use the `ldm list-io` command to view the PCIe endpoint device status. The output also shows the sub-devices and physical function devices from the root complexes that are owned by each non-primary root domain.

You can use apply the following commands to any root domain:

- `ldm add-io`
- `ldm remove-io`
- `ldm set-io`
- `ldm create-vf`
- `ldm destroy-vf`
- `ldm start-reconf`
- `ldm cancel-reconf`

Delayed reconfiguration support has been extended to include non-primary root domains. However, it can be used *only* to run the `ldm add-io`, `ldm remove-io`, `ldm set-io`, `ldm create-vf` and `ldm destroy-vf` commands. The delayed reconfiguration can be used for any operation that cannot be completed by using dynamic operations such as the following:

- Performing direct I/O operations
- Creating and destroying virtual functions from a physical function that does not meet the dynamic SR-IOV configuration requirements.



Caution - Plan ahead to minimize the number of reboots of the root domain, which minimizes downtime.

Non-primary Root Domain Requirements

Non-primary root domains can be used by the direct I/O and SR-IOV features. This feature is supported on the SPARC T4, SPARC T5, SPARC M5, and SPARC M6 platforms and Fujitsu M10 platforms.

- **Hardware Requirements.**

In addition to the PCIe cards for the direct I/O and SR-IOV described in <https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&id=1325454.1>, other PCIe cards can be used, but

not for DIO and SR-IOV. To determine which cards you can use on your platform, see your platform's hardware documentation.

- **Firmware Requirements.**

SPARC T4 platforms must run at least version 8.4.0.a of the system firmware.

SPARC T5, SPARC M5, and SPARC M6 platforms must run at least version 9.1.0.x of the system firmware.

Fujitsu M10 servers must run at least version XCP2210 of the system firmware.

- **Software Requirements.**

All domains must be running at least the Oracle Solaris 11.1.10.5.0 OS or the Oracle Solaris 10 1/13 OS plus the required patches in <http://www.oracle.com/pls/topic/lookup?ctx=E48724&id=LDSIGreqdrecommendedsolarisos> or the Oracle Solaris 11.1.10.5.0 OS.

Non-primary Root Domain Limitations

Use of the non-primary root domain has the following limitations:

- An I/O domain cannot start if the associated root domain is not running.
- Support for delayed reconfiguration has been extended to the non-primary root domains. Only the following commands can be run until that root domain has been rebooted or the delayed reconfiguration has been canceled:
 - `ldm add-io`
 - `ldm remove-io`
 - `ldm set-io`
 - `ldm create-vf`
 - `ldm destroy-vf`
- The root domain must be active and booted to perform the following operations:
 - Creating and destroying SR-IOV virtual functions
 - Adding and removing PCIe slots
 - Adding and removing SR-IOV virtual functions
- You must initiate a delayed reconfiguration on the root domain when you perform the `ldm add-io` and `ldm remove-io` direct I/O operations for PCIe slots.
- When your configuration does not meet the dynamic I/O virtualization requirements, you must use delayed reconfiguration for the following SR-IOV virtual function operations:
 - `ldm create-vf`
 - `ldm destroy-vf`
 - `ldm add-io`
 - `ldm remove-io`

- `ldm set-io`
- The reboot of a root domain affects any I/O domain that has a device from the PCIe buses that the root domain owns. See “[Rebooting the Root Domain With PCIe Endpoints Configured](#)” on page 78.
- You cannot assign an SR-IOV virtual function or a PCIe slot from one root domain to another root domain. This limitation prevents circular dependencies.

Non-primary Root Domain Examples

The following examples describe how to enable I/O virtualization for a PCIe bus, manage direct I/O devices on non-primary root domains, and manage SR-IOV virtual functions on non-primary root domains.

Enabling I/O Virtualization for a PCIe Bus

The following example shows how to enable I/O virtualization by using the `ldm add-io` and `ldm set-io` commands.

The following SPARC T4-2 I/O configuration shows that bus `pci_1` already has been removed from the primary domain.

```
primary# ldm list-io
NAME                      TYPE   BUS      DOMAIN  STATUS
----                      ---    ---      -----  -----
pci_0                     BUS    pci_0   primary  IOV
pci_1                    BUS    pci_1
niu_0                     NIU    niu_0   primary
niu_1                     NIU    niu_1   primary
/SYS/MB/PCIE0              PCIE   pci_0   primary  OCC
/SYS/MB/PCIE2              PCIE   pci_0   primary  OCC
/SYS/MB/PCIE4              PCIE   pci_0   primary  OCC
/SYS/MB/PCIE6              PCIE   pci_0   primary  EMP
/SYS/MB/PCIE8              PCIE   pci_0   primary  EMP
/SYS/MB/SASHBA             PCIE   pci_0   primary  OCC
/SYS/MB/NET0               PCIE   pci_0   primary  OCC
/SYS/MB/PCIE1              PCIE   pci_1   UNK
/SYS/MB/PCIE3              PCIE   pci_1   UNK
/SYS/MB/PCIE5              PCIE   pci_1   UNK
/SYS/MB/PCIE7              PCIE   pci_1   UNK
/SYS/MB/PCIE9              PCIE   pci_1   UNK
/SYS/MB/NET2               PCIE   pci_1   UNK
/SYS/MB/NET0/IOVNET.PF0    PF     pci_0   primary
/SYS/MB/NET0/IOVNET.PF1    PF     pci_0   primary
```

The following listing shows that the guest domains are in the bound state:

```
primary# ldm list
NAME      STATE    FLAGS   CONS   VCPU  MEMORY  UTIL   NORM  UPTIME
primary   active   -n-cv-  UART   8     8G      0.6%  0.6%  8m
rootdom1 bound    -----  5000   8     4G
ldg2      bound    -----  5001   8     4G
ldg3      bound    -----  5002   8     4G
```

The following `ldm add-io` command adds the `pci_1` bus to the `rootdom1` domain with I/O virtualization enabled for that bus. The `ldm start` command starts the `rootdom1` domain.

```
primary# ldm add-io iov=on pci_1 rootdom1
primary# ldm start rootdom1
LDom rootdom1 started
```

If a specified PCIe bus is assigned already to a root domain, use the `ldm set-io` command to enable I/O virtualization.

```
primary# ldm start-reconf rootdom1
primary# ldm set-io iov=on pci_1
primary# ldm stop-domain -r rootdom1
```

The root domain must be running its OS before you can configure the I/O devices. Connect to the console of the `rootdom1` guest domain and then boot the OS of the `rootdom1` root domain if your guest domains are not already set to autoboot.

```
primary# telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^].
Connecting to console "rootdom1" in group "rootdom1" ....
Press ~? for control options ..
ok> boot
...
primary#
```

The following command shows that the `pci_1` PCIe bus and its children are now owned by the `rootdom1` root domain.

```
primary# ldm list-io
NAME          TYPE   BUS      DOMAIN  STATUS
---          ---   ---      -----  -----
pci_0         BUS    pci_0    primary  IOV
pci_1         BUS    pci_1    rootdom1 IOV
niu_0         NIU    niu_0    primary
niu_1         NIU    niu_1    primary
/SYS/MB/PCIE0 PCIE   pci_0    primary  OCC
/SYS/MB/PCIE2 PCIE   pci_0    primary  OCC
/SYS/MB/PCIE4 PCIE   pci_0    primary  OCC
/SYS/MB/PCIE6 PCIE   pci_0    primary  EMP
/SYS/MB/PCIE8 PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA PCIE   pci_0    primary  OCC
/SYS/MB/NET0  PCIE   pci_0    primary  OCC
```

/SYS/MB/PCIE1	PCIE	pci_1	rootdom1	OCC
/SYS/MB/PCIE3	PCIE	pci_1	rootdom1	OCC
/SYS/MB/PCIE5	PCIE	pci_1	rootdom1	OCC
/SYS/MB/PCIE7	PCIE	pci_1	rootdom1	OCC
/SYS/MB/PCIE9	PCIE	pci_1	rootdom1	EMP
/SYS/MB/NET2	PCIE	pci_1	rootdom1	OCC
/SYS/MB/NET0/IOVNET.PF0	PF	pci_0	primary	
/SYS/MB/NET0/IOVNET.PF1	PF	pci_0	primary	
/SYS/MB/PCIE5/IOVNET.PF0	PF	pci_1	rootdom1	
/SYS/MB/PCIE5/IOVNET.PF1	PF	pci_1	rootdom1	
/SYS/MB/NET2/IOVNET.PF0	PF	pci_1	rootdom1	
/SYS/MB/NET2/IOVNET.PF1	PF	pci_1	rootdom1	

Managing Direct I/O Devices on Non-primary Root Domains

The following example shows how to manage direct I/O devices on non-primary root domains.

The following command produces an error because it attempts to remove a slot from the root domain while it is still active:

```
primary# ldm remove-io /SYS/MB/PCIE7 ldg1
Dynamic I/O operations on PCIe slots are not supported.
Use start-reconf command to trigger delayed reconfiguration and make I/O
changes statically.
```

The following command shows the correct method of removing a slot by first initiating a delayed reconfiguration on the root domain.

```
primary# ldm start-reconf ldg1
Initiating a delayed reconfiguration operation on the ldg1 domain.
All configuration changes for other domains are disabled until the ldg1
domain reboots, at which time the new configuration for the ldg1 domain
will also take effect.
primary# ldm remove-io /SYS/MB/PCIE7 ldg1
-----
Notice: The ldg1 domain is in the process of a delayed reconfiguration.
Any changes made to the ldg1 domain will only take effect after it reboots.
-----
primary# ldm stop-domain -r ldg1
```

The following `ldm list-io` command verifies that the `/SYS/MB/PCIE7` slot is no longer on the root domain.

```
primary# ldm list-io
NAME                                     TYPE   BUS     DOMAIN  STATUS
-----                                     ---   ---     -----  -----
pci_0                                      BUS   pci_0   primary  IOV
pci_1                                      BUS   pci_1   ldg1    IOV
```

niu_0	NIU	niu_0	primary
niu_1	NIU	niu_1	primary
/SYS/MB/PCIE0	PCIE	pci_0	primary OCC
/SYS/MB/PCIE2	PCIE	pci_0	primary OCC
/SYS/MB/PCIE4	PCIE	pci_0	primary OCC
/SYS/MB/PCIE6	PCIE	pci_0	primary EMP
/SYS/MB/PCIE8	PCIE	pci_0	primary EMP
/SYS/MB/SASHBA	PCIE	pci_0	primary OCC
/SYS/MB/NET0	PCIE	pci_0	primary OCC
/SYS/MB/PCIE1	PCIE	pci_1	ldg1 OCC
/SYS/MB/PCIE3	PCIE	pci_1	ldg1 OCC
/SYS/MB/PCIE5	PCIE	pci_1	ldg1 OCC
/SYS/MB/PCIE7	PCIE	pci_1	OCC
/SYS/MB/PCIE9	PCIE	pci_1	ldg1 EMP
/SYS/MB/NET2	PCIE	pci_1	ldg1 OCC
/SYS/MB/NET0/IOVNET.PF0	PF	pci_0	primary
/SYS/MB/NET0/IOVNET.PF1	PF	pci_0	primary
/SYS/MB/PCIE5/IOVNET.PF0	PF	pci_1	ldg1
/SYS/MB/PCIE5/IOVNET.PF1	PF	pci_1	ldg1
/SYS/MB/NET2/IOVNET.PF0	PF	pci_1	ldg1
/SYS/MB/NET2/IOVNET.PF1	PF	pci_1	ldg1

The following commands assign the /SYS/MB/PCIE7 slot to the ldg2 domain. The ldm start command starts the ldg2 domain.

```
primary# ldm add-io /SYS/MB/PCIE7 ldg2
primary# ldm start ldg2
LDom ldg2 started
```

Managing SR-IOV Virtual Functions on Non-primary Root Domains

These commands create two virtual functions from each of the two physical functions that belong to the non-primary root domain.

```
primary# ldm create-vf /SYS/MB/PCIE5/IOVNET.PF0
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF0
primary# ldm create-vf /SYS/MB/PCIE5/IOVNET.PF0
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF1
primary# ldm create-vf /SYS/MB/NET2/IOVNET.PF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF0
primary# ldm create-vf /SYS/MB/NET2/IOVNET.PF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF1
```

You can also use the -n option to create the two virtual functions by using the following two commands:

```
primary# ldm create-vf -n 2 /SYS/MB/PCIE5/IOVNET.PF0
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF0
```

```
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF1
primary# ldm create-vf -n 2 /SYS/MB/NET2/IOVNET.PF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF0
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF1
```

If you were unable to dynamically create the virtual functions on a given physical function, initiate a delayed reconfiguration to create them statically.

```
primary# ldm start-reconf ldg1
primary# ldm create-vf /SYS/MB/PCIE5/IOVNET.PF0
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF0
primary# ldm create-vf /SYS/MB/PCIE5/IOVNET.PF0
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF1
primary# ldm create-vf /SYS/MB/NET2/IOVNET.PF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF0
primary# ldm create-vf /SYS/MB/NET2/IOVNET.PF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF1
```

Then reboot the root domain, ldg1, to effect the changes.

```
primary# ldm stop-domain -r ldg1
```

The following command shows the new virtual functions.

```
primary# ldm list-io
NAME                                TYPE   BUS      DOMAIN  STATUS
-----  -----  -----  -----
pci_0                               BUS    pci_0   primary  IOV
pci_1                               BUS    pci_1   ldg1    IOV
niu_0                               NIU    niu_0   primary
niu_1                               NIU    niu_1   primary
/SYS/MB/PCIE0                      PCIE   pci_0   primary  OCC
/SYS/MB/PCIE2                      PCIE   pci_0   primary  OCC
/SYS/MB/PCIE4                      PCIE   pci_0   primary  OCC
/SYS/MB/PCIE6                      PCIE   pci_0   primary  EMP
/SYS/MB/PCIE8                      PCIE   pci_0   primary  EMP
/SYS/MB/SASHBA                     PCIE   pci_0   primary  OCC
/SYS/MB/NET0                        PCIE   pci_0   primary  OCC
/SYS/MB/PCIE1                      PCIE   pci_1   ldg1    OCC
/SYS/MB/PCIE3                      PCIE   pci_1   ldg1    OCC
/SYS/MB/PCIE5                      PCIE   pci_1   ldg1    OCC
/SYS/MB/PCIE7                      PCIE   pci_1   ldg2    OCC
/SYS/MB/PCIE9                      PCIE   pci_1   ldg1    EMP
/SYS/MB/NET2                        PCIE   pci_1   ldg1    OCC
/SYS/MB/NET0/IOVNET.PF0            PF     pci_0   primary
/SYS/MB/NET0/IOVNET.PF1            PF     pci_0   primary
/SYS/MB/PCIE5/IOVNET.PF0          PF     pci_1   ldg1
/SYS/MB/PCIE5/IOVNET.PF1          PF     pci_1   ldg1
/SYS/MB/NET2/IOVNET.PF0          PF     pci_1   ldg1
/SYS/MB/NET2/IOVNET.PF1          PF     pci_1   ldg1
/SYS/MB/PCIE5/IOVNET.PF0.VF0      VF     pci_1
/SYS/MB/PCIE5/IOVNET.PF0.VF1      VF     pci_1
/SYS/MB/NET2/IOVNET.PF1.VF0      VF     pci_1
/SYS/MB/NET2/IOVNET.PF1.VF1      VF     pci_1
```

The following command dynamically adds the /SYS/MB/PCIE5/IOVNET.PF0.VF1 virtual function to the ldg1 non-primary root domain:

```
primary# ldm add-io /SYS/MB/PCIE5/IOVNET.PF0.VF1 ldg1
```

The following command dynamically adds the /SYS/MB/NET2/IOVNET.PF1.VF0 virtual function to the ldg2 domain:

```
primary# ldm add-io /SYS/MB/NET2/IOVNET.PF1.VF0 ldg2
```

The following command adds the /SYS/MB/NET2/IOVNET.PF1.VF1 virtual function to the bound ldg3 domain:

```
primary# ldm add-io /SYS/MB/NET2/IOVNET.PF1.VF1 ldg3
primary# ldm start ldg3
LDom ldg3 started
```

Connect to the console of the ldg3 domain and then boot its OS.

The following output shows that all the assignments appear as expected. One virtual function is unassigned so it can be assigned dynamically to the ldg1, ldg2, or ldg3 domain.

NAME	TYPE	BUS	DOMAIN	STATUS
pci_0	BUS	pci_0	primary	IOV
pci_1	BUS	pci_1	ldg1	IOV
niu_0	NIU	niu_0	primary	
niu_1	NIU	niu_1	primary	
/SYS/MB/PCIE0	PCIE	pci_0	primary	OCC
/SYS/MB/PCIE2	PCIE	pci_0	primary	OCC
/SYS/MB/PCIE4	PCIE	pci_0	primary	OCC
/SYS/MB/PCIE6	PCIE	pci_0	primary	EMP
/SYS/MB/PCIE8	PCIE	pci_0	primary	EMP
/SYS/MB/SASHBA	PCIE	pci_0	primary	OCC
/SYS/MB/NET0	PCIE	pci_0	primary	OCC
/SYS/MB/PCIE1	PCIE	pci_1	ldg1	OCC
/SYS/MB/PCIE3	PCIE	pci_1	ldg1	OCC
/SYS/MB/PCIE5	PCIE	pci_1	ldg1	OCC
/SYS/MB/PCIE7	PCIE	pci_1	ldg2	OCC
/SYS/MB/PCIE9	PCIE	pci_1	ldg1	EMP
/SYS/MB/NET2	PCIE	pci_1	ldg1	OCC
/SYS/MB/NET0/IOVNET.PF0	PF	pci_0	primary	
/SYS/MB/NET0/IOVNET.PF1	PF	pci_0	primary	
/SYS/MB/PCIE5/IOVNET.PF0	PF	pci_1	ldg1	
/SYS/MB/PCIE5/IOVNET.PF1	PF	pci_1	ldg1	
/SYS/MB/NET2/IOVNET.PF0	PF	pci_1	ldg1	
/SYS/MB/NET2/IOVNET.PF1	PF	pci_1	ldg1	
/SYS/MB/PCIE5/IOVNET.PF0.VF0	VF	pci_1		
/SYS/MB/PCIE5/IOVNET.PF0.VF1	VF	pci_1	ldg1	
/SYS/MB/NET2/IOVNET.PF1.VF0	VF	pci_1	ldg2	
/SYS/MB/NET2/IOVNET.PF1.VF1	VF	pci_1	ldg3	

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 167
- “Virtual Disk Identifier and Device Name” on page 169
- “Managing Virtual Disks” on page 169
- “Virtual Disk Appearance” on page 172
- “Virtual Disk Back End Options” on page 173
- “Virtual Disk Back End” on page 174
- “Configuring Virtual Disk Multipathing” on page 181
- “CD, DVD and ISO Images” on page 186
- “Virtual Disk Timeout” on page 189
- “Virtual Disk and SCSI” on page 190
- “Virtual Disk and the `format` Command” on page 190
- “Using ZFS With Virtual Disks” on page 191
- “Using Volume Managers in an Oracle VM Server for SPARC Environment” on page 195

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 is sent. 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.

Note - You can refer to a disk either by using `/dev/dsk` or `/dev/rdsk` as part of the disk path name. Either reference produces the same result.



Caution - Do not use the `d0` device to represent the entire disk. This device represents the entire disk only when the disk has an EFI label and not a VTOC label. Using the `d0` device results in the virtual disk being a single-slice disk, which might cause you to corrupt the disk label if you write the beginning of the disk.

Instead, use the `s2` slice to virtualize the entire disk. The `s2` slice is independent of the label.

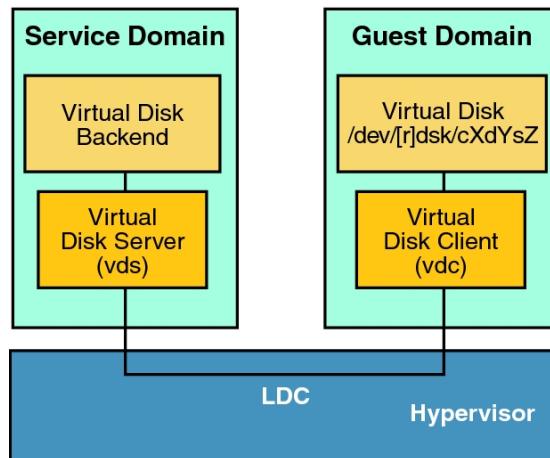
The virtual disk back end can be physical or logical. Physical devices can include the following:

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

Logical devices can be any of the following:

- A file on a local file system, such as ZFS or UFS, or on a remote file system that is made available by means of NFS
- A logical volume from a volume manager, such as ZFS, VxVM, or Solaris Volume Manager
- Any disk pseudo device accessible from the service domain

FIGURE 10-1 Virtual Disks With Oracle VM Server for SPARC



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 domain-name
```

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.

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 173 for a description of virtual disk options. See “[Virtual Disk Timeout](#)” on page 189 for a description of the virtual disk timeout.

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.

Assigning a virtual disk device to a domain creates an implicit dependency on the domain providing the virtual disk service. You can view these dependencies or view domains that depend on the virtual disk service by using the `ldm list-dependencies` command. See “[Listing Domain I/O Dependencies](#)” on page 354.

▼ How to 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 domain-name
```

You can specify a custom 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 169.

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

▼ How to Export a Virtual Disk Back End Multiple Times



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.**

```
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.**

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
```

▼ How to Change Virtual Disk Options

For more information about virtual disk options, see “[Virtual Disk Back End Options](#)” on page 173.

- After a back end is exported from the service domain, you can change the virtual disk options.

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

▼ How to Change the Timeout Option

For more information about virtual disk options, see “[Virtual Disk Back End Options](#)” on page 173.

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

```
primary# ldm set-vdisk timeout=seconds disk-name domain-name
```

▼ How to Remove a Virtual Disk

1. **Remove a virtual disk from a guest domain.**

```
primary# ldm rm-vdisk disk-name domain-name
```

2. **Stop exporting the corresponding back end from the service domain.**

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

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.



Caution - Single-slice disks do not have device IDs. If a device ID is required, use a full physical disk backend.

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 eight slices (*s0* to *s7*). This type of disk is visible with the `format(1M)` command. The disk's partition table can be changed using either the `fmthard` or `format` 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 be exported only 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 eight slices (*s0* to *s7*). However, only the first slice (*s0*) is usable. This type of 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`). This type of disk was not visible with the `format` 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 be accessed only 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 restriction 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 might 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` or `luxadm` 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 With Solaris Volume Manager](#)” on page 196 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 197 for more information.
- If you are exporting the same virtual disk back end multiple times from the same virtual disk service, see “[How to Export a Virtual Disk Back End Multiple Times](#)” on page 170 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.

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 174.

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.

▼ How to Export a Physical Disk as a Virtual Disk



Caution - When configuring virtual disks, ensure that each virtual disk references a distinct physical (back-end) resource, such as a physical disk, a disk slice, a file, or a volume.

Some disks, such as FibreChannel and SAS, have a “dual-ported” nature, which means that the same disk can be referenced by two different paths. Ensure that the paths you assign to different domains do not refer to the same physical 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 -1 /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 regardless of whether 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.

▼ How to 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 -1 /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 eight devices, because the disk is a single-slice disk, only the first slice (`s0`) is usable.

▼ How to Export Slice 2

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

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

File and Volume Exporting

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 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` 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 - You must run the `format` command in the guest domain to create partitions.

▼ How to 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-Mbyte blank file to get a 100-Mbyte 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 eight slices.

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

```
ldg1# ls -1 /dev/dsk/c0d5s*
/dev/dsk/c0d5s0
/dev/dsk/c0d5s1
/dev/dsk/c0d5s2
/dev/dsk/c0d5s3
/dev/dsk/c0d5s4
/dev/dsk/c0d5s5
/dev/dsk/c0d5s6
/dev/dsk/c0d5s7
```

▼ How to Export a ZFS Volume as a Full Disk

1. Create a ZFS volume to use as a full disk.

The following example shows how to create a ZFS volume, zdisk0, to use as a full 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-Mbyte volume to result in a 100-Mbyte virtual disk.

2. From the control domain, export the corresponding device to that ZFS volume.

```
primary# ldm add-vdsdev /dev/zvol/dsk/ldoms/domain/test/zdisk0 \
zdisk0@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 volume to a guest domain.

The following example shows how to assign the volume, zdisk0, to the 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, verify that the disk is accessible and is a full disk.

A full disk is a regular disk with eight slices.

The following example shows how to list the disk, `c0d9`, and verify that it is accessible and is a full disk:

```
ldg1# ls -1 /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
```

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.

▼ How to 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-Mbyte volume to get a 100-Mbyte 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 -1 /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

If you have a configuration exporting volumes as virtual disks, 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 3.2 software, and set the `slice` option for all volumes you want to export as single-slice disks. See the [ldm\(1M\)](#) man page.
- Add the following line to the `/etc/system` file on the service domain.

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

For information about correctly creating or updating `/etc/system` property values, see [“Updating Property Values in the /etc/system File” on page 334](#).

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 [†]	Single-slice disk [‡]

Back End	No Slice Option	Slice Option Set
Disk slice (not slice 2)	Single-slice disk*	Single-slice disk
File	Full disk	Single-slice disk
Volume, including ZFS, Solaris Volume Manager, or VxVM	Full disk	Single-slice disk

[†]Export the entire disk.

[‡]Export only slice 2

* A slice is always exported as a 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

Using the loopback file (`lofi`) driver to export a file as a virtual disk adds an extra driver layer and affects 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 Exporting](#)” on page 177.

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` 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.

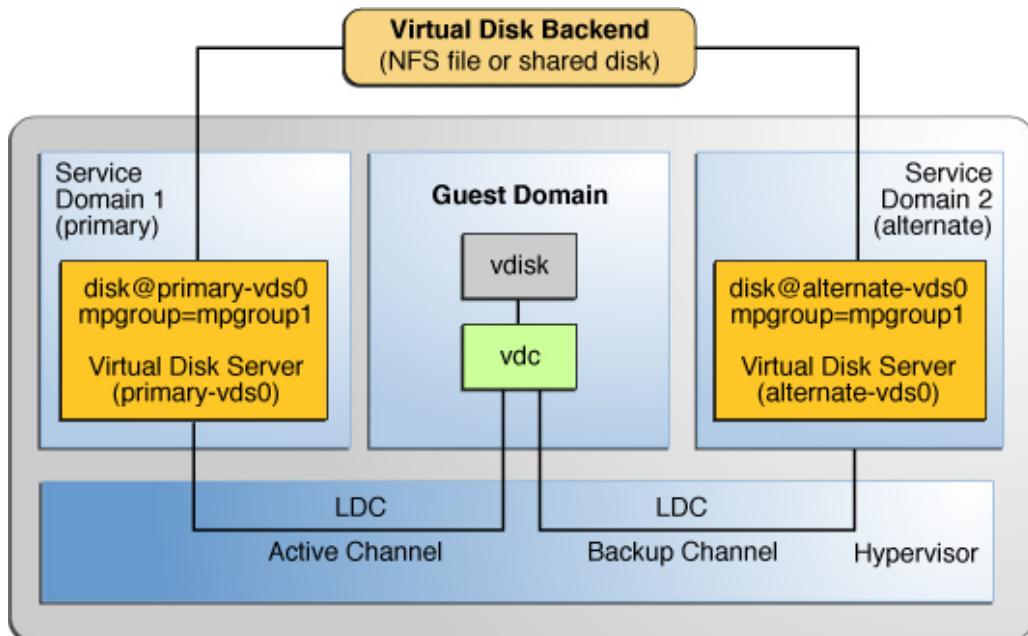
Note - You cannot use mpgroups and SCSI reservation together.

Note that 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 that is used as an example in the procedure “[How to Configure Virtual Disk Multipathing](#)” on page 183. In this example, a multipathing group named `mpgroup1` is used to create a virtual disk, whose back end is accessible from two service domains: primary and alternate.

FIGURE 10-2 Configuring Virtual Disk Multipathing



Virtual Disk Multipathing and Virtual Disk Timeout

With virtual disk multipathing, the path that is used to access the back end automatically changes if the back end becomes inaccessible by means of the currently active path. This path change occurs independently of the value of the virtual disk `timeout` property.

The virtual disk `timeout` property specifies the amount of time after which an I/O fails when no service domain is available to process the I/O. This timeout applies to all virtual disks, even those that use virtual disk multipathing.

As a consequence, setting a virtual disk timeout when virtual disk multipathing is configured can prevent multipathing from working correctly, especially with a small timeout value. So, avoid setting a virtual disk timeout for virtual disks that are part of a multipathing group.

For more information, see “[Virtual Disk Timeout](#)” on page 189.

▼ How to Configure Virtual Disk Multipathing

See [Figure 10-2](#).

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

```
primary# ldm add-vdsdev mpgroup=mpgroup1 backend-path1 volume@primary-vds0
```

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.**

```
primary# ldm add-vdsdev mpgroup=mpgroup1 backend-path2 volume@alternate-vds0
```

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.**

```
primary# ldm add-vdisk disk-name volume@primary-vds0 domain-name
```

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.

Example 10-1 Using an Mpgroup to Add a LUN to the Virtual Disk Service of Both Primary and Alternate Domains

The following shows how to create a LUN and add it to the virtual disk service for both primary and alternate domains by using the same mpgroup:

To determine which domain to use first when accessing the LUN, specify the associated path when adding the disk to the domain.

- Create the virtual disk devices:

```
primary# ldm add-vdsdev mpgroup=ha lun1@primary-vds0  
primary# ldm add-vdsdev mpgroup=ha lun1@alternate-vds0
```

- To use the LUN from primary-vds0 first, perform the following command:

```
primary# ldm add-vdisk disk1 lun1@primary-vds0 gd0
```

- To use the LUN from alternate-vds0 first, perform the following command:

```
primary# ldm add-vdisk disk1 lun1@alternate-vds0 gd0
```

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 one of the service domains it has been associated with. If this service domain becomes unavailable, the virtual disk attempts to access its back end through another 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.

Dynamic Path Selection

Starting with the Oracle VM Server for SPARC 3.2 software, you can dynamically select the path to be used for a virtual disk on guest domains that run at least the Oracle Solaris 11.2.1.0.0 (SRU 1) OS.

Dynamic path selection occurs when the first path in an mpgroup disk is changed by using the `ldm set-vdisk` command to set the `volume` property to a value in the form `volume-name@service-name`. An active domain that supports dynamic path selection can switch to only the selected path. If the updated drivers are not running, this path is selected when the Oracle Solaris OS reloads the disk instance or at the next domain reboot.

The dynamic path selection feature enables you to dynamically perform the following steps while the disk is in use:

- Specify the disk path to be tried first by the guest domain when attaching the disk
- Change the currently active path to the one that is indicated for already attached multipathing disks

Using the `ldm add-vdisk` command with an mpgroup disk now specifies the path indicated by `volume-name@service-name` as the selected path with which to access the disk.

The selected disk path is listed first in the set of paths provided to the guest domain irregardless of its rank when the associated mpgroup was created.

You can use the `ldm set-vdisk` command on bound, inactive, and active domains. When used on active domains, this command permits you to choose only the selected path of the mpgroup disk.

The `ldm list-bindings` command shows the following information:

- The STATE column for each mpgroup path indicates one of the following values:
 - active – Current active path of the mpgroup
 - standby – Path is not currently used
 - unknown – Domain does not support dynamic path selection, the device is not attached, or an error prevents the path state from being retrieved
- The disk paths are listed in the order that is used for choosing the active path
- The volume that is associated with the disk is the selected path for the mpgroup and is listed first.

The following example shows that the selected path is `vol-ldg2@opath-ldg2` and that the currently used active path is going through the `ldg1` domain. You might see this situation if the selected path could not be used and the second possible path was used instead. Even if the selected path comes online, the non-selected path continues to be used. To make the first path active again, re-issue the `ldm set-vdisk` command to set the `volume` property to the name of the path you want.

DISK

NAME	VOLUME	TOUT	ID	DEVICE	SERVICE	MPGROUP
disk	disk-ldg4@primary-vds0	0	disk@0	primary		
tdiskgroup	vol-ldg2@opath-ldg2	1	disk@1	ldg2		testdiskgroup
PORT	MPGROUP	VOLUME	MPGROUP	SERVER	STATE	
2	vol-ldg2@opath-ldg2	ldg2			standby	
0	vol-ldg1@opath-vds	ldg1			active	
1	vol-prim@primary-vds0	primary			standby	

If you use the `ldm set-vdisk` command on an mpgroup disk of a bound domain that does not run at least the Oracle Solaris 11.2.1.0.0 (SRU 1) OS, the operation changes the order of the path priorities and the new path can be used first during next disk attach or reboot or if the OBP needs to access it.

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 can export only 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` service must not use the CD or DVD. See “[How to Export a CD or DVD From the Service Domain to the Guest Domain](#)” on page 186 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.

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

1. **From the service domain, check whether the volume management daemon, `vold`, is running and online.**

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

2. If the volume management daemon is running and online, as in the example in Step 1, do the following:

- a. In the /etc/vold.conf file, 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/rdsk/c1t0d0s2 | 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/c1t0d0s2 cdrom@primary-vds0
```

5. Assign the exported CD or DVD to the guest domain.

The following command shows how to assign the exported CD or DVD to domain ldg1:

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

Exporting a CD or DVD Multiple Times

A CD or DVD can be exported multiple times and assigned to different guest domains. See “[How to Export a Virtual Disk Back End Multiple Times](#)” on page 170 for more information.

▼ How to Export an ISO Image From the Control Domain to Install a Guest Domain

Before You Begin 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:

```
primary# ldm list
NAME      STATE    FLAGS   CONS   VCPU   MEMORY   UTIL   UPTIME
primary   active   -n-cv  SP     8      8G       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`.

```
primary# 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`.

```
primary# 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`.

```
primary# ldm add-vdisk s10-dvd dvd-iso@primary-vds0 ldom1
```

4. Restart the guest domain.

In this example, the logical domain is `ldom1`.

```
primary# ldm start-domain ldom1
LDom ldom1 started
# ldm list
NAME      STATE    FLAGS   CONS   VCPU   MEMORY   UTIL   UPTIME
primary   active   -n-cv  SP     8      8G       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.

```
primary# 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, in some cases, file systems or applications might not want the I/O operation to block but rather to fail and report an error if the service domain is down for too long. You can now 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.

Set this timeout by using one of the following methods:

- Using the `ldm add-vdisk` command.

```
ldm add-vdisk timeout=seconds disk-name volume-name@service-name domain-name
```

- Using the `ldm set-vdisk` command.

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

- Adding the following line to the `/etc/system` file on the guest domain.

```
set vdc:vdc_timeout=seconds
```

For information about correctly creating or updating `/etc/system` property values, see [“Updating Property Values in the `/etc/system` File” on page 334](#).

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.

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).

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` and multihost disk control operations `mhd`. Other virtual disks, such as virtual disks having a file or a volume as a back end, do not support these interfaces.

Note - You cannot use mpgroups and SCSI reservation together.

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. Doing so can lead to an unexpected disk state.

Virtual Disk and the `format` Command

The `format` 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 you to do the following:

- Store disk images in ZFS volumes or ZFS files
- Use snapshots to back up disk images
- Use clones to duplicate disk images and provision additional domains

Refer to “[Oracle Solaris ZFS Administration Guide](#)” for more information about using 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 by using the `mkfile` command. This command can take some time to complete,

especially if the file to be created 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 show how to store disk images using a ZFS volume or a ZFS file. The syntax to export a ZFS volume or file is the same but the path to the back end is different.

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.

EXAMPLE 10-2 Storing a Disk Image Using a ZFS Volume

First, create a 20-Gbyte image on a ZFS volume.

```
primary# zfs create -V 20gb ldmpool/ldg1/disk0
```

Then, export the ZFS volume as a virtual disk.

```
primary# ldm add-vdsdev /dev/zvol/dsk/ldmpool/ldg1/disk0 ldg1_disk0@primary-vds0
```

Assign the ZFS volume to the ldg1 guest domain.

```
primary# ldm add-vdisk disk0 ldg1_disk0@primary-vds0 ldg1
```

EXAMPLE 10-3 Storing a Disk Image Using a ZFS File

First, create a 20-Gbyte disk image on a ZFS volume and create the ZFS file.

```
primary# zfs create ldmpool/ldg1/disk0
primary# mkfile 20g /ldmpool/ldg1/disk0/file
```

Then, export the ZFS file as a virtual disk.

```
primary# ldm add-vdsdev /ldmpool/ldg1/disk0/file ldg1_file0@primary-vds0
```

Assign the ZFS file to the ldg1 guest domain.

```
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. You can ensure that a disk is not in use in a guest domain in one of the following ways:

- Stop and unbind the guest domain. This solution is the safest, and 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.
- Unmount any slices of the disk you want to snapshot that are used in the guest domain, and ensure that no slice is in use in 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.

EXAMPLE 10-4 Creating a Snapshot of a Disk Image

This example creates a snapshot of the disk image that was created for the `ldg1` domain.

```
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. The cloned image then 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 Oracle Solaris 11 `sysconfig unconfigure` command or the Oracle Solaris 10 `sys-unconfig` 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 rather 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.

▼ How to Create a Snapshot of a Disk Image of an Unconfigured System

1. **Bind and start the original domain.**
2. **Unconfigure the system.**
 - **Oracle Solaris 11 OS: Run the `sysconfig unconfigure` command.**
 - **Oracle Solaris 10 OS: Run the `sys-unconfig` command.**
- When this operation completes, the domain halts.
3. **Stop and unbind the domain, do *not* reboot it.**
4. **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.

5. **Clone this image to create a new domain which, when first booted, asks for the configuration of the system.**

Using Volume Managers in an Oracle VM Server for SPARC Environment

This section describes using volume managers in an Oracle VM Server for SPARC environment.

Using Virtual Disks With Volume Managers

Any 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. Data stored on the virtual disk from the guest domain can therefore 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/rdsk/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 With 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.

```
primary# 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/c10t600C0FF0000000000015153295A4B100d0s1 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 VxVM is installed on your system and 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 whether Veritas DMP is enabled by checking the multipathing information in the `vxdisk list` output. 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 `vxdmpadm` command. For example:

```
vxdmpadm -f disable path=/dev/dsk/c4t12d0s2
```

## Using Volume Managers With Virtual Disks

This section describes using volume managers with virtual disks.

### Using ZFS With 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 With 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 metadevice state database, `metadb`, 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`. 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 With 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 200
- “Oracle Solaris 10 Networking Overview” on page 200
- “Oracle Solaris 11 Networking Overview” on page 202
- “Maximizing Virtual Network Performance” on page 204
- “Virtual Switch” on page 205
- “Virtual Network Device” on page 207
- “Viewing Network Device Configurations and Statistics” on page 210
- “Controlling the Amount of Physical Network Bandwidth That Is Consumed by a Virtual Network Device” on page 213
- “Virtual Device Identifier and Network Interface Name” on page 216
- “Assigning MAC Addresses Automatically or Manually” on page 219
- “Using Network Adapters With Domains” on page 222
- “Configuring a Virtual Switch and the Service Domain for NAT and Routing” on page 222
- “Configuring IPMP in an Oracle VM Server for SPARC Environment” on page 226
- “Using VLAN Tagging” on page 232
- “Using Private VLANs” on page 236
- “Tuning Packet Throughput Performance” on page 241
- “Using NIU Hybrid I/O” on page 242
- “Using Link Aggregation With a Virtual Switch” on page 246
- “Configuring Jumbo Frames” on page 248
- “Oracle Solaris 11 Networking-Specific Feature Differences” on page 252

Oracle Solaris OS networking changed a great deal between the Oracle Solaris 10 OS and the Oracle Solaris 11 OS. For information about issues to consider, see “[Oracle Solaris 10 Networking Overview](#)” on page 200, “[Oracle Solaris 11 Networking Overview](#)” on page 202, and “[Oracle Solaris 11 Networking-Specific Feature Differences](#)” on page 252.

## Introduction to a Virtual Network

A virtual network enables domains to communicate with each other without using any external physical networks. A virtual network also can enable 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.

Oracle Solaris networking differs greatly between the Oracle Solaris 10 OS and the Oracle Solaris 11 OS. The following two sections provide overview information about networking for each OS.

---

**Note** - Oracle Solaris 10 networking behaves the same as it would on a domain or a system. The same is true for Oracle Solaris 11 networking. For more information about Oracle Solaris OS networking, see [Oracle Solaris 10 Documentation](#) and [Oracle Solaris 11.1 Documentation](#).

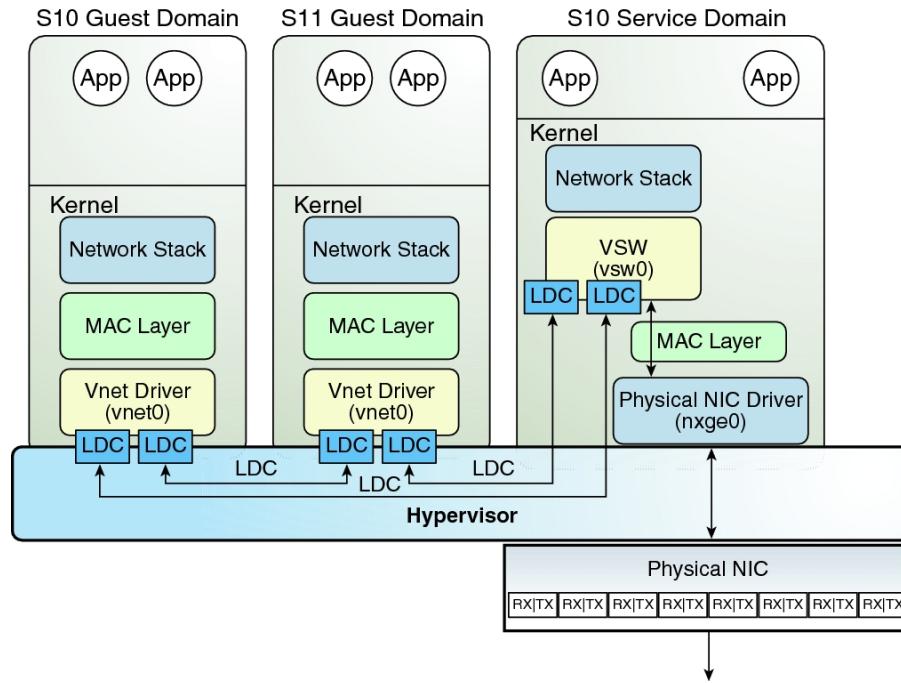
---

The feature differences between Oracle Solaris 10 and Oracle Solaris 11 networking are described in [“Oracle Solaris 11 Networking Overview” on page 202](#).

---

## Oracle Solaris 10 Networking Overview

The following diagram shows that a guest domain that runs the Oracle Solaris 11 OS is fully compatible with an Oracle Solaris 10 service domain. The only differences are features added or enhanced in the Oracle Solaris 11 OS.

**FIGURE 11-1** Oracle VM Server for SPARC Network Overview for the Oracle Solaris 10 OS

The previous diagram shows interface names such as `nxge0`, `vsw0`, and `vnet0` which apply to the Oracle Solaris 10 OS only. Also note the following:

- The virtual switch in the service domain is connected to the guest domains, which enables guest domains to communicate with each other.
- The virtual switch is also connected to the physical network interface `nxge0`, which enables guest domains to communicate with the physical network.
- The virtual switch network interface `vsw0` is created in the service domain, which enables the two guest domains to communicate with the service domain.
- The virtual switch network interface `vsw0` in the service domain can be configured by using the Oracle Solaris 10 `ifconfig` command.
- The virtual network device `vnet0` in an Oracle Solaris 10 guest domain can be configured as a network interface by using the `ifconfig` command.
- The virtual network device `vnet0` in an Oracle Solaris 11 guest domain might appear with a generic link name, such as `net0`. It can be configured as a network interface by using the `ipadm` command.

The virtual switch behaves like a regular physical network switch and switches network packets between the different systems, such as guest domains, the service domain, and the physical network, to which it is connected. The `vsw` driver provides the network device functionality that enables the virtual switch to be configured as a network interface.

## Oracle Solaris 11 Networking Overview

The Oracle Solaris 11 OS introduced many new networking features, which are described in the Oracle Solaris 11 networking documentation at [Oracle Solaris 11.1 Documentation](#).

The following Oracle Solaris 11 networking features are important to understand when you use the Oracle VM Server for SPARC software:

- All network configuration is performed by the `ipadm` and `dladm` commands.
- The “vanity name by default” feature generates generic link names, such as `net0`, for all physical network adapters. This feature also generates generic names for virtual switches (`vswn`) and virtual network devices (`vnetn`), which appear like physical network adapters to the OS. To identify the generic link name that is associated with a physical network device, use the `dladm show-phys` command.

By default in Oracle Solaris 11, physical network device names use generic “vanity” names. Generic names, such as `net0`, are used instead of device driver names, such as `nxge0`, which were used in Oracle Solaris 10.

The following command creates a virtual switch for the primary domain by specifying the generic name, `net0`, instead of a driver name, such as `nxge0`:

```
primary# ldm add-vsw net-dev=net0 primary-vsw0 primary
```

- The Oracle Solaris 11 OS uses virtual network interface cards (VNICs) to create internal virtual networks.  
A [VNIC](#) is a virtual instantiation of a physical network device that can be created from the physical network device and assigned to a zone.
- Use the Oracle Solaris 11 `DefaultFixed` network configuration profile (NCP) when configuring the Oracle VM Server for SPARC software.

For Oracle Solaris 11 domains, use the `DefaultFixed` NCP. You can enable this profile during or after installation. During an Oracle Solaris 11 installation, select the `Manual` networking configuration.

- Do not replace the primary network interface with the virtual switch (`vsw`) interface. The control domain can use the existing primary network interface to communicate with the guest domains that have virtual network devices connected to the same virtual switch.
- Do not use the physical network adapter's MAC address for the virtual switch because using the physical adapter's MAC address for the virtual switch conflicts with the primary network interface.

**Note** - In this release, use the `DefaultFixed` NCP to configure datalinks and network interfaces on Oracle Solaris 11 systems.

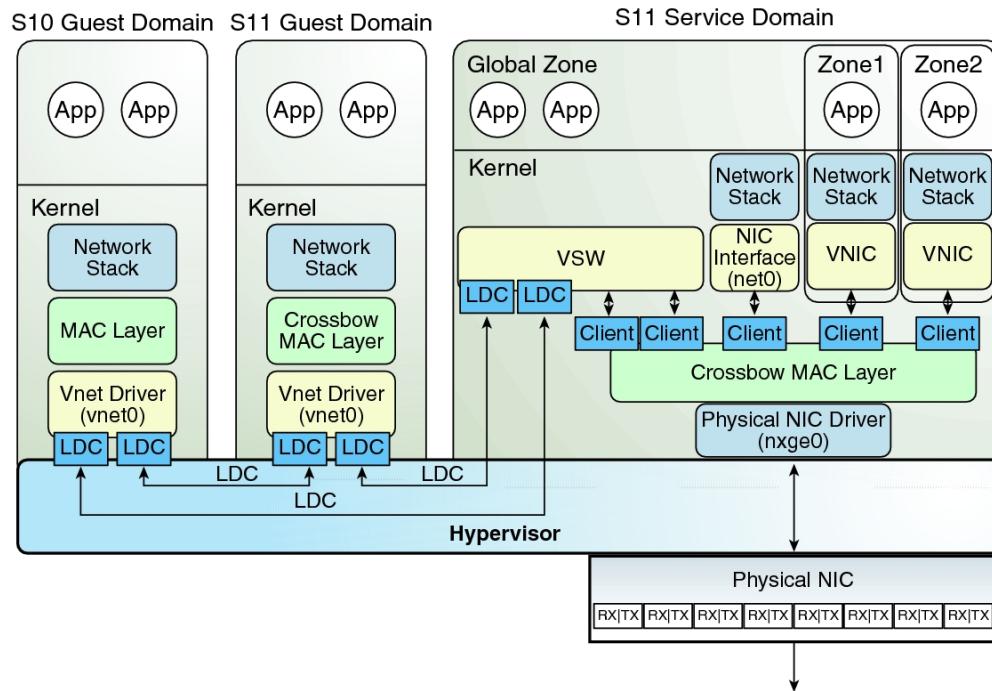
The Oracle Solaris 11 OS includes the following NCPs:

- `DefaultFixed` – Enables you to use the `dladm` or `ipadm` command to manage networking
- `Automatic` – Enables you to use the `netcfg` or `netadm` command to manage networking

Ensure that the `DefaultFixed` NCP is enabled by using the `netadm list` command. See [Chapter 7, “Using Datalink and Interface Configuration Commands on Profiles,” in “Oracle Solaris Administration: Network Interfaces and Network Virtualization”](#).

The following diagram shows that a guest domain that runs the Oracle Solaris 10 OS is fully compatible with an Oracle Solaris 11 service domain. The only differences are features added or enhanced in the Oracle Solaris 11 OS.

**FIGURE 11-2** Oracle VM Server for SPARC Network Overview for the Oracle Solaris 11 OS



The diagram shows that network device names, such as `nxge0` and `vnet0`, can be represented by generic link names, such as `netn` in Oracle Solaris 11 domains. Also note the following:

- The virtual switch in the service domain is connected to the guest domains, which enables guest domains to communicate with each other.
- The virtual switch is also connected to the physical network device `nxge0`, which enables guest domains to communicate with the physical network.

The virtual switch also enables guest domains to communicate with the service domain network interface `net0` and with VNICs on the same physical network device as `nxge0`. This includes communication between the guest domains and the Oracle Solaris 11 service domain. Do not configure the virtual switch itself (the `vsw0` device) as a network device, as this functionality is no longer supported.

- The virtual network device `vnet0` in an Oracle Solaris 10 guest domain can be configured as a network interface by using the `ifconfig` command.
- The virtual network device `vnet0` in an Oracle Solaris 11 guest domain might appear with a generic link name, such as `net0`. It can be configured as a network interface by using the `ipadm` command.

A virtual switch behaves like a regular physical network switch and switches network packets between the different systems to which it is connected. A system can be a guest domain, a service domain, or a physical network.

## Maximizing Virtual Network Performance

You can achieve high transfer rates for guest and external networks and for guest-to-guest communications when you configure your platform and the domains as described in this section. The virtual network stack introduces support for large segment offload (LSO), which produces high TCP performance without requiring the use of jumbo frames.

## Hardware and Software Requirements

Meet the following requirements to maximize the network performance for your domains:

- **Hardware requirements.** These performance improvements are available only for the SPARC T4, SPARC T5, SPARC M5, and SPARC M6 systems.
- **System firmware requirements.** These SPARC systems must run the latest system firmware. See “[Fully Qualified System Firmware Versions](#)” in “[Oracle VM Server for SPARC 3.2 Installation Guide](#)”.
- **Oracle Solaris OS requirements.** Ensure that the service domain and guest domain run the following Oracle Solaris OS versions:
  - **Service domain.** At least the Oracle Solaris 11.1.9.0.0 OS or the Oracle Solaris 10 OS with the 150031-03 patch.

- **Guest domain.** At least the Oracle Solaris 11.1.9.0.0 OS or the Oracle Solaris 10 OS with the 150031-03 patch.
- **CPU and memory requirements.** Ensure that you assign sufficient CPU and memory resources to the service domain and the guest domains.
  - **Service domain.** Because the service domain acts as a data proxy for the guest domains, assign at least 2 CPU cores and at least 4 Gbytes of memory to the service domain.
  - **Guest domain.** Configure each guest domain to be able to drive at least 10-Gbps performance. Assign at least 2 CPU cores and at least 4 Gbytes of memory to each guest domain.

## Configuring Your Domains to Maximize the Performance of Your Virtual Network

In previous versions of Oracle VM Server for SPARC and the Oracle Solaris OS, you could improve your network performance by configuring jumbo frames. This configuration is no longer required and unless required for another reason, using the standard MTU value of 1500 for your service and guest domains is best.

To achieve the improved networking performance, set the `extended-mapin-space` property to `on` for the service domain and the guest domains, which is the default setting for the Oracle VM Server for SPARC 3.1 software and supported system firmware.

```
primary# ldm set-domain extended-mapin-space=on domain-name
```

To check the `extended-mapin-space` property value, run the following command:

```
primary# ldm ls -l domain-name |grep extended-mapin
extended-mapin-space=on
```

---

**Note** - A change to the `extended-mapin-space` property value triggers a delayed reconfiguration on the primary domain. This situation requires a primary domain reboot. You also must first stop the guest domains before you change this property value.

---

## 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 also associated with a physical network interface, network communication is permitted between guest domains and the physical network over the physical network interface. When running in an Oracle Solaris 10

service domain, a virtual switch also has a network interface, `vswn`, which permits the service domain to communicate with the other domains that are connected to that virtual switch. The virtual switch can be used like any regular network interface and configured with the Oracle Solaris 10 `ifconfig` command.

Assigning a virtual network device to a domain creates an implicit dependency on the domain providing the virtual switch. You can view these dependencies or view domains that depend on this virtual switch by using the `ldm list-dependencies` command. See “[Listing Domain I/O Dependencies](#)” on page 354.

In an Oracle Solaris 11 service domain, the virtual switch cannot be used as a regular network interface. If the virtual switch is connected to a physical network interface, communication with the service domain is possible by using this physical interface. If configured without a physical interface, you can enable communication with the service domain by using an etherstub as the network device (`net-dev`) that is connected with a VNIC.

To determine which network device to use as the back-end device for the virtual switch, search for the physical network device in the `dladm show-phys` output or use the `ldm list-netdev` command to list the network devices for logical domains.

---

**Note** - When a virtual switch is added to an Oracle Solaris 10 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 \(Oracle Solaris 10 Only\)](#)” on page 48 for instructions.

This situation occurs *only* for the Oracle Solaris 10 OS and *not* for the Oracle Solaris 11 OS.

---

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. See the [ldm\(1M\)](#) man page.

When you create a virtual switch on a VLAN tagged instance of a NIC or an aggregation, you must specify the NIC (`nxge0`), the aggregation (`aggr3`), or the vanity name (`net0`) as the value of the `net-dev` property when you use the `ldm add-vsw` or `ldm set-vsw` command.

---

**Note** - Starting with the Oracle Solaris 11.2.1.0.0 (SRU 1) OS, you can dynamically update the `net-dev` property value by using the `ldm set-vsw` command. In previous Oracle Solaris OS releases, using the `ldm set-vsw` command to update the `net-dev` property value in the primary domain causes the primary domain to enter a delayed reconfiguration.

---

You cannot add a virtual switch on top of an InfiniBand IP-over-InfiniBand (IPoIB) network device. Although the `ldm add-vsw` and `ldm add-vnet` commands appear to succeed, no data

will flow because these devices transport IP packets by means of the InfiniBand transport layer. The virtual switch only supports Ethernet as a transport layer.

The following examples explain how to create a virtual switch on a physical network adapter:

- **Oracle Solaris 10 OS:** The following command creates a virtual switch on a physical network adapter called nxge0:

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

For more information about configuring a virtual switch as a network interface, see [“Enabling Networking Between the Control/Service Domain and Other Domains \(Oracle Solaris 10 Only\)” on page 48](#).

- **Oracle Solaris 11 OS:** The following command creates a virtual switch on a physical network adapter called net0:

```
primary# ldm add-vsw net-dev=net0 primary-vsw0 primary
```

The following example uses the `ldm list-netdev -b` command to show only the valid virtual switch back-end devices for the svcdom service domain.

```
primary# ldm list-netdev -b svcdom
DOMAIN
svcdom

NAME CLASS MEDIA STATE SPEED OVER LOC
----- -----
net0 PHYS ETHER up 10000 ixgbe0 /SYS/MB/RISER1/PCIE
net1 PHYS ETHER unknown 0 ixgbe1 /SYS/MB/RISER1/PCIE4
net2 ESTUB ETHER unknown 0 --
net3 ESTUB ETHER unknown 0 --
ldoms-estub.vsw0 ESTUB ETHER unknown 0 --
```

## Virtual Network Device

A virtual network 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 Oracle Solaris 10 `ifconfig` command or the Oracle Solaris 11 `ipadm` command.

---

**Note** - For Oracle Solaris 11, the devices are assigned generic names, so `vnetn` would use a generic name, such as `net0`.

---

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. See the [ldm\(1M\)](#) man page.

See the information about Oracle VM Server for SPARC networking for Oracle Solaris 10 and Oracle Solaris 11 in [Figure 11-1](#) and [Figure 11-2](#), respectively.

## Inter-Vnet LDC Channels

By default, 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.

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.

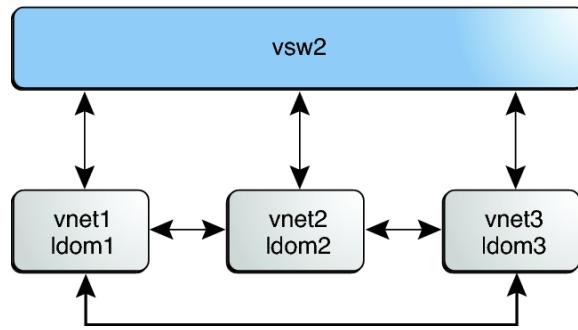
---

**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.

---

You can use the `ldm add-vsw` and the `ldm set-vsw` commands to specify a value of `on` or `off` for the `inter-vnet-link` property.

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.

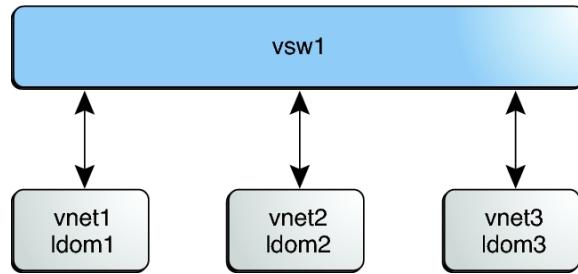
**FIGURE 11-3** Virtual Switch Configuration That Uses Inter-Vnet Channels

The following figure shows the same virtual switch configuration with the `inter-vnet-link` property set to `off`. The inter-vnet LDC channels are not allocated. 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 11-4** Virtual Switch Configuration That Does Not Use Inter-Vnet Channels

## Viewing Network Device Configurations and Statistics

The `ldm list-netdev` and `ldm list-netstat` commands enable you to view information about the network devices in the system and the network statistics, respectively. As a result, you have a centralized view of the network devices and statistics in a given physical domain.

To use these commands, you must run at least the Oracle Solaris 11.2 SRU 1 OS in the guest domain.

### EXAMPLE 11-1 Listing Network Device Configuration Information

The following example shows a short listing of the network devices for the `ldg1` domain by using the `ldm list-netdev` command.

```
primary# ldm list-netdev ldg1

DOMAIN
ldg1

NAME CLASS MEDIA STATE SPEED OVER LOC
---- ---- ---- ---- ---- ----
net0 VNET ETHER up 0 -- primary-vsw0/vne_t0_ldg1
net3 PHYS ETHER up 10000 -- /SYS/MB/RISER1/PCIE4
net4 VSW ETHER up 10000 -- ldg1-vsw1
net1 PHYS ETHER up 10000 -- /SYS/MB/RISER1/PCIE4
net5 VNET ETHER up 0 -- ldg1-vsw1/vnet1_ldg1
net6 VNET ETHER up 0 -- ldg1-vsw1/vnet2_ldg1
agg2 AGGR ETHER unknown 0 net1,net3 --
ldoms-vsw0.vport3 VNIC ETHER unknown 0 ldg1-vsw1/vnet2_ldg1
ldoms-vsw0.vport2 VNIC ETHER unknown 0 ldg1-vsw1/vnet1_ldg1
ldoms-vsw0.vport1 VNIC ETHER unknown 0 ldg1-vsw1/vnet2_ldg3
ldoms-vsw0.vport0 VNIC ETHER unknown 0 ldg1-vsw1/vnet2_ldg2
```

### EXAMPLE 11-2 Listing Detailed Network Device Configuration Information

The following example shows a detailed listing of the network devices for the `ldg1` domain by using the `ldm list-netdev -l` command.

```
primary# ldm list-netdev -l ldg1

DOMAIN
ldg1

NAME CLASS MEDIA STATE SPEED OVER LOC
---- ---- ---- ---- ---- ----
net0 VNET ETHER up 0 -- primary-vsw0/vnet0_ldg1
[/virtual-devices@100/channel-devices@200/network@0]
MTU : 1500 [1500-1500]
```

```

IPADDR : 10.129.241.200/255.255.255.0
MAC_ADDRS : 00:14:4f:fb:9c:d2

net3 PHYS ETHER up 10000 -- /SYS/MB/RISER1/PCIE4
[/pci@400/pci@1/pci@0/pci@0/network@0]
MTU : 1500 [576-15500]
MAC_ADDRS : a0:36:9f:0a:c5:d2

net4 VSW ETHER up 10000 -- ldg1-vsw1
[/virtual-devices@100/channel-devices@200/virtual-network-switch@0]
MTU : 1500 [1500-1500]
IPADDR : 192.168.1.2/255.255.255.0
MAC_ADDRS : 00:14:4f:fb:61:6e

net1 PHYS ETHER up 10000 -- /SYS/MB/RISER1/PCIE4
[/pci@400/pci@1/pci@0/pci@0/network@0,1]
MTU : 1500 [576-15500]
MAC_ADDRS : a0:36:9f:0a:c5:d2

net5 VNET ETHER up 0 -- ldg1-vsw1/vnet1_ldg1
[/virtual-devices@100/channel-devices@200/network@1]
MTU : 1500 [1500-1500]
IPADDR : 0.0.0.0 /255.0.0.0
: fe80::214:4fff:fef8:5062/ffc0::
MAC_ADDRS : 00:14:4f:f8:50:62

net6 VNET ETHER up 0 -- ldg1-vsw1/vnet2_ldg1
[/virtual-devices@100/channel-devices@200/network@2]
MTU : 1500 [1500-1500]
IPADDR : 0.0.0.0 /255.0.0.0
: fe80::214:4fff:fef8:af92/ffc0::
MAC_ADDRS : 00:14:4f:f8:af:92

aggr2 AGGR ETHER unknown 0 net1,net3 --
MODE : TRUNK
POLICY : L2,L3
LACP_MODE : ACTIVE
MEMBER : net1 [PORTSTATE = attached]
MEMBER : net3 [PORTSTATE = attached]
MAC_ADDRS : a0:36:9f:0a:c5:d2

ldoms-vsw0.vport3 VNIC ETHER unknown 0 -- ldg1-vsw1/vnet2_ldg1
MTU : 1500 [576-1500]
MAC_ADDRS : 00:14:4f:f8:af:92

ldoms-vsw0.vport2 VNIC ETHER unknown 0 -- ldg1-vsw1/vnet1_ldg1
MTU : 1500 [576-1500]
MAC_ADDRS : 00:14:4f:f8:50:62

ldoms-vsw0.vport1 VNIC ETHER unknown 0 -- ldg1-vsw1/vnet2_ldg3
MTU : 1500 [576-1500]
MAC_ADDRS : 00:14:4f:f9:d3:88

ldoms-vsw0.vport0 VNIC ETHER unknown 0 -- ldg1-vsw1/vnet2_ldg2

```

```
MTU : 1500 [576-1500]
MAC_ADDR : 00:14:4f:fa:47:f4
 : 00:14:4f:f9:65:b5
 : 00:14:4f:f9:60:3f
```

**EXAMPLE 11-3** Listing Network Device Statistics

The **ldm list-netstat** command shows network statistics for one or more domains in the system.

The following example shows the default network statistics for all domains in the system.

```
primary# ldm list-netstat

DOMAIN
primary

NAME IPACKETS RBYTES OPACKETS OBYTES
---- -----
net3 0 0 0 0
net0 2.72M 778.27M 76.32K 6.01M
net4 2.72M 778.27M 76.32K 6.01M
net6 2 140 1.30K 18.17K
net7 0 0 0 0
net2 0 0 0 0
net1 0 0 0 0
aggr1 0 0 0 0
ldoms-vsw0.vport0 935.40K 74.59M 13.15K 984.43K
ldoms-vsw0.vport1 933.26K 74.37M 11.42K 745.15K
ldoms-vsw0.vport2 933.24K 74.37M 11.46K 747.66K
ldoms-vsw1.vport1 202.26K 17.99M 179.75K 15.69M
ldoms-vsw1.vport0 202.37K 18.00M 189.00K 16.24M

DOMAIN
ldg1

NAME IPACKETS RBYTES OPACKETS OBYTES
---- -----
net0 5.19K 421.57K 68 4.70K
net3 0 0 2.07K 256.93K
net4 0 0 4.37K 560.17K
net1 0 0 2.29K 303.24K
net5 149 31.19K 78 17.00K
net6 147 30.51K 78 17.29K
aggr2 0 0 0 0
ldoms-vsw0.vport3 162 31.69K 52 14.11K
ldoms-vsw0.vport2 163 31.74K 51 13.76K
ldoms-vsw0.vport1 176 42.99K 25 1.50K
ldoms-vsw0.vport0 158 40.19K 45 4.42K

DOMAIN
ldg2
```

| NAME   | IPACKETS | RBYTES  | OPACKETS | OBYTES  |
|--------|----------|---------|----------|---------|
| net0   | 5.17K    | 418.90K | 71       | 4.88K   |
| net1   | 2.70K    | 201.67K | 2.63K    | 187.01K |
| net2   | 132      | 36.40K  | 1.51K    | 95.07K  |
| <hr/>  |          |         |          |         |
| DOMAIN |          |         |          |         |
| ldg3   |          |         |          |         |
| NAME   | IPACKETS | RBYTES  | OPACKETS | OBYTES  |
| net0   | 5.16K    | 417.43K | 72       | 4.90K   |
| net1   | 2.80K    | 206.12K | 2.67K    | 190.36K |
| net2   | 118      | 35.00K  | 1.46K    | 87.78K  |

## Controlling the Amount of Physical Network Bandwidth That Is Consumed by a Virtual Network Device

The bandwidth resource control feature enables you to limit the physical network bandwidth consumed by a virtual network device. This feature is supported on a service domain that runs at least the Oracle Solaris 11 OS and is configured with a virtual switch. Oracle Solaris 10 service domains silently ignore network bandwidth settings. This feature ensures that one guest domain does not take over the available physical network bandwidth and leave none for the others.

Use the `ldm add-vnet` and `ldm set-vnet` commands to specify the bandwidth limit by providing a value for the `maxbw` property. Use the `ldm list-bindings` or the `ldm list-domain -o` `network` command to view the `maxbw` property value for an existing virtual network device. The minimum bandwidth limit is 10 Mbps.

## Network Bandwidth Limitations

---

**Note** - This feature is not supported by a Hybrid I/O-enabled virtual network device. The `maxbw` property is not enforced for Hybrid mode virtual networks because the Hybrid I/O assigns a specific unit of hardware resource that cannot be changed to limit bandwidth. To limit a virtual network device's bandwidth, you must disable the Hybrid mode.

---

The bandwidth resource control applies only to the traffic that goes through the virtual switch. Thus, inter-vnet traffic is not subjected to this limit. If you do not have a physical backend device configured, you can ignore bandwidth resource control.

The minimum supported bandwidth limit depends on the Oracle Solaris network stack in the service domain. The bandwidth limit can be configured with any desired high value. There

is no upper limit. The bandwidth limit ensures only that the bandwidth does not exceed the configured value. Thus, you can configure a bandwidth limit with a value greater than the link speed of the physical network device that is assigned to the virtual switch.

## Setting the Network Bandwidth Limit

Use the `ldm add-vnet` command to create a virtual network device and specify the bandwidth limit by providing a value for the `maxbw` property.

```
primary# ldm add-vnet maxbw=limit if-name vswitch-name domain-name
```

Use the `ldm set-vnet` command to specify the bandwidth limit for an existing virtual network device.

```
primary# ldm set-vnet maxbw=limit if-name domain-name
```

You can also clear the bandwidth limit by specifying a blank value for the `maxbw` property:

```
primary# ldm set-vnet maxbw= if-name domain-name
```

The following examples show how to use the `ldm` command to specify the bandwidth limit. The bandwidth is specified as an integer with a unit. The unit is `M` for megabits-per-second or `G` for gigabits-per-second. The unit is megabits-per-second if you do not specify a unit.

### EXAMPLE 11-4 Setting the Bandwidth Limit When Creating a Virtual Network Device

The following command creates a virtual network device (`vnet0`) that has a bandwidth limit of 100 Mbps.

```
primary# ldm add-vnet maxbw=100M vnet0 primary-vsw0 ldg1
```

The following command would issue an error message when attempting to set a bandwidth limit below the minimum value, which is 10 Mbps.

```
primary# ldm add-vnet maxbw=1M vnet0 primary-vsw0 ldg1
```

### EXAMPLE 11-5 Setting the Bandwidth Limit on an Existing Virtual Network Device

The following commands sets the bandwidth limit to 200 Mbps on the existing `vnet0` device.

Depending on the real-time network traffic pattern, the amount of bandwidth might not reach the specified limit of 200 Mbps. For example, the bandwidth might be 95 Mbps, which does not exceed the 200 Mbps limit.

```
primary# ldm set-vnet maxbw=200M vnet0 ldg1
```

The following command sets the bandwidth limit to 2 Gbps on the existing `vnet0` device.

Because there is no upper limit on bandwidth in the MAC layer, you can still set the limit to be 2 Gbps even if the underlying physical network speed is less than 2 Gbps. In such a case, there is no bandwidth limit effect.

```
primary# ldm set-vnet maxbw=2G vnet0 ldg1
```

**EXAMPLE 11-6** Clearing the Bandwidth Limit on an Existing Virtual Network Device

The following command clears the bandwidth limit on the specified virtual network device (`vnet0`). By clearing this value, the virtual network device uses the maximum bandwidth available, which is provided by the underlying physical device.

```
primary# ldm set-vnet maxbw= vnet0 ldg1
```

**EXAMPLE 11-7** Viewing the Bandwidth Limit of an Existing Virtual Network Device

The `ldm list-bindings` command shows the value of the `maxbw` property for the specified virtual network device, if defined.

The following command shows that the `vnet0` virtual network device has a bandwidth limit of 15 Mbps. If no bandwidth limit is set, the `MAXBW` field is blank.

```
primary# ldm list-bindings
...
VSW
 NAME MAC NET-DEV ID DEVICE LINKPROP
 primary-vsw0 00:14:4f:f9:95:97 net0 0 switch@0 1

 DEFAULT-VLAN-ID PVID VID MTU MODE INTER-VNET-LINK
 1 1 1500 on

 PEER MAC PVID VID MTU MAXBW LINKPROP INTERVNETLINK
 vnet0@ldg1 00:14:4f:fb:b8:c8 1 1500 15

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

 NETWORK
 NAME SERVICE ID DEVICE
 vnet0 primary-vsw0@primary 0 network@0

 MAC MODE PVID VID MTU MAXBW LINKPROP
 00:14:4f:fb:b8:c8 1 1500 15

 PEER MAC MODE PVID VID
 primary-vsw0@primary 00:14:4f:f9:95:97 1
```

```
MTU MAXBW LINKPROP
1500
```

You can also use the `dladm show-linkprop` command to view the `maxbw` property value as follows:

```
primary# dladm show-linkprop -p maxbw
LINK PROPERTY PERM VALUE EFFECTIVE DEFAULT POSSIBLE
...
ldoms-vsw0.vport0 maxbw rw 15 15 -- --
```

## 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.

```
primary# ldm add-vsw [id=switch-id] vswitch-name domain-name
primary# ldm add-vnet [id=network-id] if-name vswitch-name domain-name
```

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 |
| ...   |                      |           |                   |        |      |     |      |

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*.

---

**Note** - On Oracle Solaris 11 systems, generic link names in the form of *netn* are assigned to both *vswn* and *vnetn*. Use the `ldadm show-phys` command to identify which *netn* names map to the *vswn* and *vnetn* devices.

---



**Caution** - The Oracle Solaris OS preserves the mapping between the name of a network interface and a virtual switch or a virtual network device 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 make the existing system configuration unusable. This situation 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.

## Finding the Oracle Solaris OS Network Interface Name (Oracle Solaris 11 OS)

On Oracle Solaris 11 systems, you can use the `ldm list-netdev` command to find the Oracle Solaris OS network interface names. For more information, see the [ldm\(1M\)](#) man page.

The following example shows the `ldm list-netdev` and `ldm list -o network` commands. The `ldm list -o network` command shows the virtual network devices in the NAME field. The `ldm list-netdev` output shows the corresponding OS interface name in the NAME column.

```
primary# ldm list -o network ldg1
...
NETWORK
 NAME SERVICE ID DEVICE MAC MODE
 PVID VID MTU MAXBW LINKPROP
```

```
vnet0-ldg1 primary-vsw0@primary 0 network@0 00:14:4f:fa:eb:4e 1
 1500
vnet1-ldg1 svcdom-vsw0@svcdom 1 network@1 00:14:4f:f8:53:45 4
 1500
 PVLAN :400,community

primary# ldm list-netdev ldg1
DOMAIN
ldg1

NAME CLASS MEDIA STATE SPEED OVER LOC

net0 VNET ETHER up 0 vnet0 primary-vsw0/vnet0-ldg1
net1 VNET ETHER up 0 vnet1 svcdom-vsw0/vnet1-ldg1
net2 VNET ETHER unknown 0 vnet2 svcdom-vsw1/vnet2-ldg1
```

To verify that the `ldm list-netdev` output is correct, run the `dladm show-phys` and `dladm show-linkprop -p mac-address` commands from the `ldg1`:

```
ldg1# dladm show-phys
LINK MEDIA STATE SPEED DUPLEX DEVICE
net0 Ethernet up 0 unknown vnet0
net1 Ethernet up 0 unknown vnet1
net2 Ethernet unknown 0 unknown vnet2

ldg1# dladm show-linkprop -p mac-address
LINK PROPERTY PERM VALUE EFFECTIVE DEFAULT POSSIBLE
net0 mac-address rw 0:14:4f:fa:eb:4e 0:14:4f:fa:eb:4e 0:14:4f:fa:eb:4e --
net1 mac-address rw 0:14:4f:f8:53:45 0:14:4f:f8:53:45 0:14:4f:f8:53:45 --
```

## ▼ How to Find Oracle Solaris OS Network Interface Name (Oracle Solaris 10 OS)

This procedure describes how to find the Oracle Solaris OS network interface name in `ldg1` that corresponds to `net-c`. This example also shows differences if you are looking for the network interface name of a virtual switch instead of a virtual network device. In this example procedure, guest domain `ldg1` contains two virtual network devices, `net-a` and `net-c`.

1. Use the `ldm` command to find the virtual network device number for `net-c`.

```
primary# 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. **Find the corresponding network interface on `ldg1` by logging into `ldg1` and finding the entry for this device number under `/devices`.**

```
primary# uname -n
ldg1
primary# 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.**

- **Oracle Solaris 10 OS:**

```
primary# 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
```

- **Oracle Solaris 11 OS:**

First, you must determine the name of the interface to specify for `vnet1` by using the `dladm show-phys` command.

```
primary# dladm show-phys |grep vnet1
net2 Ethernet up 0 unknown vnet1
```

Then use the following command to determine the MAC address of `net2`.

```
primary# dladm show-linkprop -p mac-address net2
LINK PROPERTY PERM VALUE EFFECTIVE DEFAULT POSSIBLE
net2 mac-address rw 00:14:4f:f8:dd:68 00:14:4f: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, and a virtual switch, 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 uses 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 behavior 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 Domains

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`

---

**Note** - In Oracle Solaris 11, the allocation of MAC addresses for VNICs uses addresses outside these ranges.

---

## Automatic Assignment Algorithm

When you do not specify a MAC address when creating a 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. 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 221. If the address is already assigned, the Logical Domains Manager iterates, choosing another address and again checking for collisions. This process 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, the 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. If a different device on another Oracle VM Server for SPARC-enabled system has already been assigned that MAC address, the Logical Domains Manager on that system sends a response containing the MAC address in question. If the requesting Logical Domains Manager receives a response, it notes 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 the following:

- 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.

---

## Using Network Adapters With Domains

In an Oracle Solaris 10 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 a Virtual Switch and the Service Domain for NAT and Routing](#)” on page 222 for information about how to use non-GLDv3 compliant network adapters.

---

**Note** - GLDv3 compliance is not an issue for Oracle Solaris 11 environments.

---

For more information about using link aggregation, see “[Using Link Aggregation With a Virtual Switch](#)” on page 246.

### ▼ How to Determine Whether a Network Adapter Is GLDv3-Compliant (Oracle Solaris 10)

This procedure applies only to Oracle Solaris 10 domains.

- **Determine whether the network adapter is GLDv3-compliant.**

The following example uses bge0 as the network device name.

```
primary# dladm show-link bge0
bge0 type: non-vlan mtu: 1500 device: bge0
```

The value of the type: field is one of the following:

- GLDv3-compliant drivers have a type of non-vlan or vlan.
- Non-GLDv3-compliant drivers have a type of legacy.

## Configuring a Virtual Switch and the Service Domain for NAT and Routing

In the Oracle Solaris 10 OS, the virtual switch (vsw) is a layer-2 switch, which 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 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 essential to provide external connectivity to the domains when the physical network adapter is not GLDv3-compliant.

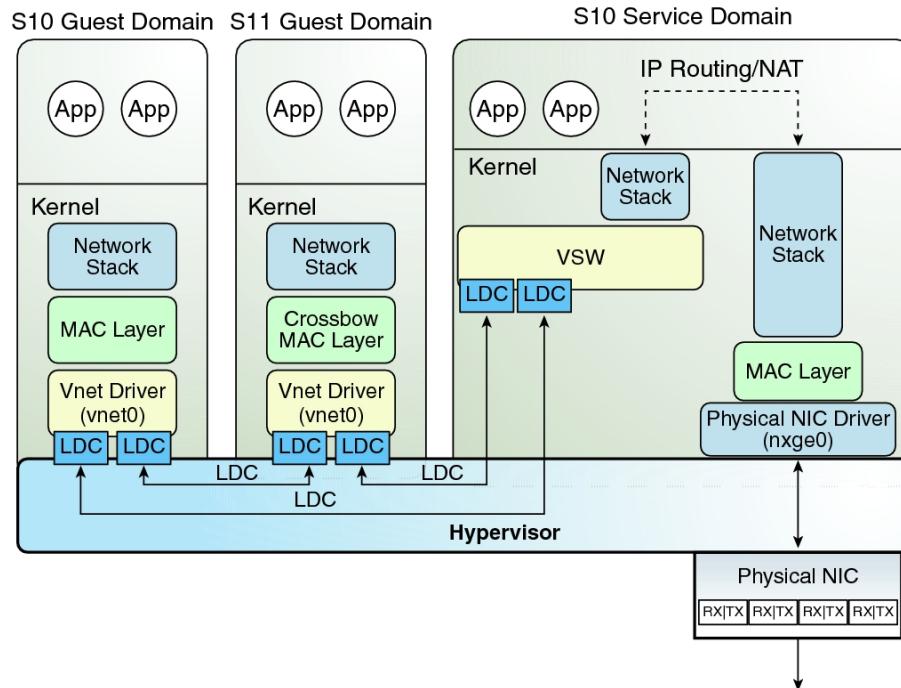
The advantages of this configuration are:

- 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.

## Configuring NAT on an Oracle Solaris 10 System

The following diagram shows how a virtual switch can be used to configure Network Address Translation (NAT) in a service domain to provide external connectivity for guest domains.

**FIGURE 11-5** Virtual Network Routing



## ▼ How to Set Up a Virtual Switch to Provide External Connectivity to Domains (Oracle Solaris 10)

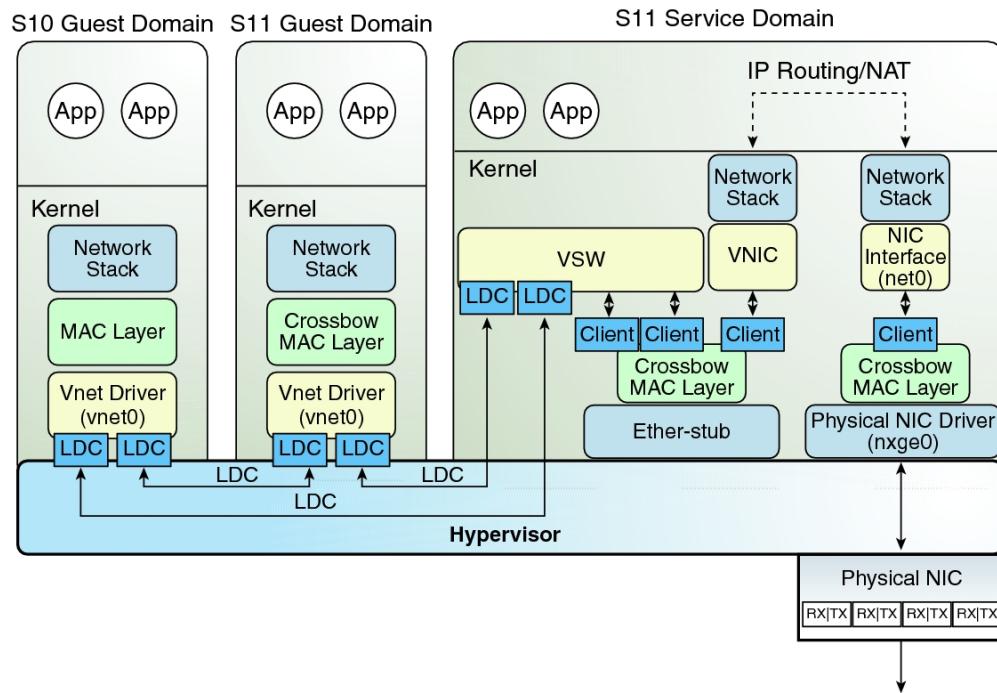
1. **Create a virtual switch that does not have an associated physical device.**  
If assigning an address, ensure that the virtual switch has a unique MAC address.  

```
primary# ldm add-vsw [mac-addr=xx:xx:xx:xx:xx:xx] 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 “[How to Configure the Virtual Switch as the Primary Interface](#)” on page 49 for more information about creating the virtual switch.
3. **Configure the virtual switch device for DHCP, if needed.**  
See “[How to Configure the Virtual Switch as the Primary Interface](#)” on page 49 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 more information about IP routing, see “[Packet Forwarding and Routing on IPv4 Networks](#)” in “[Oracle Solaris Administration: IP Services](#)”.

## Configuring NAT on an Oracle Solaris 11 System

The Oracle Solaris 11 network virtualization features include `etherstub`, which is a pseudo network device. This device provides functionality similar to physical network devices but only for private communications with its clients. This pseudo device can be used as a network back-end device for a virtual switch that provides the private communications between virtual networks. By using the `etherstub` device as a back-end device, guest domains can also communicate with VNICs on the same `etherstub` device. Using the `etherstub` device in this way enables guest domains to communicate with zones in the service domain. Use the `dladm create-etherstub` command to create an `etherstub` device.

The following diagram shows how virtual switches, `etherstub` devices, and VNICs can be used to set up Network Address Translation (NAT) in a service domain.

**FIGURE 11-6** Virtual Network Routing

You might consider using persistent routes. For more information, see “[Troubleshooting Issues When Adding a Persistent Route](#)” in “[Troubleshooting Network Administration Issues in Oracle Solaris 11.2](#)” and “[Creating Persistent \(Static\) Routes](#)” in “[Configuring and Administering Network Components in Oracle Solaris 11.2](#)”.

## ▼ How to Set Up a Virtual Switch to Provide External Connectivity to Domains (Oracle Solaris 11)

1. Create an Oracle Solaris 11 etherstub device.

```
primary# dladm create-etherstub stub0
```

2. Create a virtual switch that uses stub0 as the physical back-end device.

```
primary# ldm add-vsw net-dev=stub0 primary-stub-vsw0 primary
```

3. Create a VNIC on the stub0 device.

```
primary# dladm create-vnic -l stub0 vnic0
```

**4. Configure vnic0 as the network interface.**

```
primary# ipadm create-ip vnic0
primary# ipadm create-addr -T static -a 192.168.100.1/24 vnic0/v4static
```

**5. Enable IPv4 forwarding and create NAT rules.**

See “[Setting IP Interface Properties](#)” in “[Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1](#)” and “[Packet Forwarding and Routing on IPv4 Networks](#)” in “[Oracle Solaris Administration: IP Services](#)”.

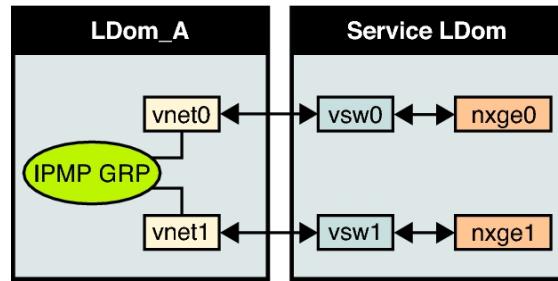
## Configuring IPMP in an Oracle VM Server for SPARC Environment

The Oracle VM Server for SPARC software supports link-based IP network multipathing (IPMP) with virtual network devices. When configuring an IPMP group with virtual network devices, configure the group to use link-based detection. If you are 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 (vnet0 and vnet1) connected to separate virtual switch instances (vsw0 and vsw1) in the service domain, which in turn use two different physical interfaces. The physical interfaces are nxge0 and nxge1 in Oracle Solaris 10 and net0 and net1 in Oracle Solaris 11. The diagram shows the Oracle Solaris 10 physical interface names.

If a physical link failure occurs 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.

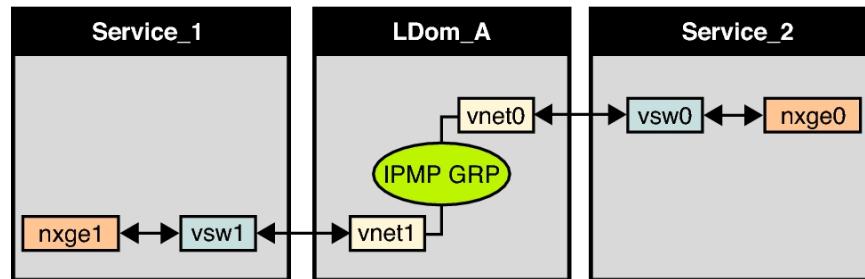
**FIGURE 11-7** Two Virtual Networks Connected to Separate Virtual Switch Instances


---

**Note** - This diagram shows the configuration on an Oracle Solaris 10 system. For an Oracle Solaris 11 system, only the interface names change to use the generic names, such as `net0` and `net1` for `nxge0` and `nxge1`, respectively.

---

You can achieve further reliability 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 11-8** Virtual Network Devices Each Connected to Different Service Domains


---

**Note** - This diagram shows the configuration on an Oracle Solaris 10 system. For an Oracle Solaris 11 system, only the interface names change to use the generic names, such as `net0` and `net1` for `nxge0` and `nxge1`, respectively.

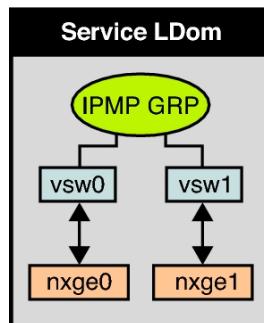
---

For more information, see the Oracle Solaris 10 “[Oracle Solaris Administration: IP Services](#)” or “Establishing an Oracle Solaris Network” in the [Oracle Solaris 11.1 Information Library](#).

## Configuring and Using IPMP in the Service Domain

You can configure IPMP 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 a failover to the other virtual switch device in the IPMP group. The two physical interfaces are `nxge0` and `nxge1` in Oracle Solaris 10 and `net0` and `net1` in Oracle Solaris 11. The diagram shows the Oracle Solaris 10 physical interface names.

**FIGURE 11-9** Two Virtual Switch Interfaces Configured as Part of an IPMP Group



---

**Note** - The diagram shows the configuration on an Oracle Solaris 10 system. For an Oracle Solaris 11 system, only the interface names change to use the generic names, such as `net0` and `net1` for `nxge0` and `nxge1`, respectively.

---

## Using Link-Based IPMP in Oracle VM Server for SPARC 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 configuration is enabled by default and does not require you to perform additional configuration steps.

Sometimes detecting physical network link state changes might be necessary. 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, you might need to obtain and report the physical link status to the virtual network device and its stack.

You can use the `linkprop=phys-state` option 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. In addition, the link status is also logged in the `/var/adm/messages` file.

For Oracle Solaris 10, see the [dladm\(1M\)](#) and [ifconfig\(1M\)](#) man pages. For Oracle Solaris 11, see the [dladm\(1M\)](#), [ipadm\(1M\)](#), and [ipmpstat\(1M\)](#) man pages.

---

**Note** - You can run both link-state-unaware and link-state-aware vnet and vsw drivers concurrently on an Oracle VM Server for SPARC 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 1/13 OS, and run at least version 1.3 of the Logical Domains Manager.

---

## ▼ How to 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 an administrator.

- For Oracle Solaris 10, see “[Configuring RBAC \(Task Map\)](#)” in “[System Administration Guide: Security Services](#)”.
- For Oracle Solaris 11.2, see [Chapter 1, “About Using Rights to Control Users and Processes,”](#) in “[Securing Users and Processes in Oracle Solaris 11.2](#)”.

## 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`.

---

```
primary# ldm add-vnet linkprop=phys-state if-name vswitch-name domain-name
```

The following example enables physical link status updates for `ldom1_vnet0` connected to `primary-vsw0` on the logical domain `ldom1`:

```
primary# ldm add-vnet linkprop=phys-state ldom1_vnet0 primary-vsw0 ldom1
```

- Modify an existing virtual network device by specifying `linkprop=phys-state` when running the `ldm set-vnet` command.

```
primary# ldm set-vnet linkprop=phys-state if-name domain-name
```

The following example enables physical link status updates for `vnet0` on the logical domain `ldom1`:

```
primary# ldm set-vnet linkprop=phys-state ldom1_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 `ldom1_vnet0` on the logical domain `ldom1`:

```
primary# ldm set-vnet linkprop= ldom1_vnet0 ldom1
```

### Example 11-8 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.

```
primary# ldm add-vnet ldom1_vnet0 primary-vsw0 ldom1
primary# ldm add-vnet ldom1_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.

- **Oracle Solaris 10 OS:** Use the `ifconfig` command.

```
primary# ifconfig vnet0 plumb
primary# ifconfig vnet1 plumb
primary# ifconfig vnet0 group ipmp0
primary# ifconfig vnet1 group ipmp0
```

The second and third commands configure the `ipmp0` interface with the IP address, as appropriate.

- **Oracle Solaris 11 OS:** Use the `ipadm` command.

Note that `net0` and `net1` are the Oracle Solaris 11 vanity names for `vnet0` and `vnet1`, respectively.

```
primary# ipadm create-ip net0
primary# ipadm create-ip net1
primary# ipadm create-ipmp ipmp0
primary# ipadm add-ipmp -i net0 -i net1 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.

```
primary# ldm add-vnet linkprop=phys-state ldom1_vnet0 primary-vsw0 ldom1
primary# ldm add-vnet linkprop=phys-state ldom1_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:

- **Oracle Solaris 10 OS:** Use the `ifconfig` command.

```
primary# ifconfig vnet0 plumb
primary# ifconfig vnet1 plumb
primary# ifconfig vnet0 192.168.1.1/24 up
primary# ifconfig vnet1 192.168.1.2/24 up
primary# ifconfig vnet0 group ipmp0
primary# ifconfig vnet1 group ipmp0
```

- **Oracle Solaris 11 OS:** Use the `ipadm` command.

Note that `net0` and `net1` are the vanity names for `vnet0` and `vnet1`, respectively.

```
primary# ipadm create-ip net0
primary# ipadm create-ip net1
primary# ipadm create-ipmp ipmp0
primary# ipadm add-ipmp -i net0 -i net1 ipmp0
primary# ipadm create-addr -T static -a 192.168.1.1/24 ipmp0/v4addr1
primary# ipadm create-addr -T static -a 192.168.1.2/24 ipmp0/v4addr2
```

## Using VLAN Tagging

The Oracle VM Server for SPARC software supports 802.1Q VLAN-Tagging in the network infrastructure.

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 virtual network device in a guest domain. Use the Oracle Solaris 10 `ifconfig` command or the Oracle Solaris 11 `dladm` and `ipadm` commands to create a VLAN interface over a virtual network device. The creation method is the same as the method used to configure a VLAN interface over any other physical network device. The additional requirement in the Oracle VM Server for SPARC environment is that you must use the `ldm` command to assign the virtual network to the corresponding VLANs. See the [ldm\(1M\)](#) man page.

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 virtual network 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 virtual network

using the `pvid=` and `vid=` arguments to the `ldm add-vnet` command. This information configures the virtual switch to support multiple VLANs in the Oracle VM Server for SPARC 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

The Port VLAN ID (PVID) specifies the VLAN of which the virtual network device must 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 virtual network 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 virtual network 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 virtual network instance `0` using one of the following commands and if the `pvid=` argument for the `vnet` has been specified as `10`, the `vnet0` interface would be implicitly assigned to belong to VLAN 10. Note that the following commands show the `vnet0` interface names, which pertain to Oracle Solaris 10. For Oracle Solaris 11, use the generic name instead, such as `net0`.

- **Oracle Solaris 10 OS:** Use the `ifconfig` command.

```
ifconfig vnet0 plumb
```

- **Oracle Solaris 11 OS:** Use the `ipadm` command.

```
ipadm create-ip net0
```

## VLAN ID

The VID ID (VID) specifies the VLAN of which a virtual network device or virtual switch must 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.

## ▼ How to 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.

```
primary# ldm add-vsw net-dev=nxge0 pvid=21 vid=20 primary-vsw0 primary
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 20 | Subnet 192.168.1.0 (netmask: 255.255.255.0) |
|---------|---------------------------------------------|

|         |                                             |
|---------|---------------------------------------------|
| VLAN 21 | Subnet 192.168.2.0 (netmask: 255.255.255.0) |
|---------|---------------------------------------------|

|         |                                             |
|---------|---------------------------------------------|
| VLAN 22 | Subnet 192.168.3.0 (netmask: 255.255.255.0) |
|---------|---------------------------------------------|

#### a. Create the VLAN interface in the service (primary) domain.

- Oracle Solaris 10 OS: Use the `ifconfig` command.

```
primary# ifconfig vsw0 plumb
primary# ifconfig vsw0 192.168.2.100 netmask 0xffffffff broadcast + up
primary# ifconfig vsw20000 plumb
primary# ifconfig vsw20000 192.168.1.100 netmask 0xffffffff broadcast + up
```

- Oracle Solaris 11 OS: Use the `dladm` and `ipadm` commands.

```
primary# dladm create-vlan -l net0 -v20 vlan20
primary# ipadm create-ip vlan20
primary# ipadm create-addr -T static -a 192.168.1.100/24 vlan20/ipv4
```

#### b. Create the VLAN interface in the guest (`ldom1`) domain.

- Oracle Solaris 10 OS: Use the `ifconfig` command.

```
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 10 OS, refer to “[Administering Virtual Local Area Networks](#)” in “[Oracle Solaris Administration: IP Services](#)”.

■ **Oracle Solaris 11 OS: Use the `dladm` and `ipadm` commands.**

```
ldom1# dladm create-vlan -l net0 -v21
ldom1# ipadm create-ip net0
ldom1# ipadm create-addr -T static -a 192.168.1.101/24 net0/ipv4
ldom1# ipadm create-ip net21000
ldom1# ipadm create-addr -T static -a 192.168.2.101/24 net21000/ipv4
ldom1# ipadm create-ip net22000
ldom1# ipadm create-addr -T static -a 192.168.3.101/24 net22000/ipv4
```

For more information about how to configure VLAN interfaces in the Oracle Solaris 11 OS, refer to [Chapter 3, “Configuring Virtual Networks by Using Virtual Local Area Networks,”](#) in “[Managing Network Datalinks in Oracle Solaris 11.2](#)”.

## ▼ How to Install a Guest Domain When the Install Server Is in a VLAN

Be careful when using the Oracle Solaris JumpStart feature to install a guest domain over the network when the installation server is in a VLAN. This feature is supported only on Oracle Solaris 10 systems.

For more information about using the Oracle Solaris JumpStart feature to install a guest domain, see “[How to Use the Oracle Solaris JumpStart Feature on an Oracle Solaris 10 Guest Domain](#)” on page 58.

**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 ldom
```

Do not configure any tagged VLANs (`vid`) for that virtual network device. You must do this because the OpenBoot PROM (OBP) is not aware of VLANs and cannot handle VLAN-tagged network packets.

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
```

You can now add the virtual network device to additional VLANs in tagged mode.

## Using Private VLANs

The private VLAN (PVLAN) mechanism enables you to divide a regular VLAN into sub-VLANs to isolate network traffic. The PVLAN mechanism is defined in [RFC 5517](http://tools.ietf.org/html/rfc5517) (<http://tools.ietf.org/html/rfc5517>). Usually, a regular VLAN is a single broadcast domain, but when configured with PVLAN properties, the single broadcast domain is partitioned into smaller broadcast subdomains while keeping the existing Layer 3 configuration. When you configure a PVLAN, the regular VLAN is called the *primary VLAN* and the sub-VLANs are called *secondary VLANs*.

When two virtual networks use the same VLAN ID on a physical link, all broadcast traffic is passed between the two virtual networks. However, when you create virtual networks that use PVLAN properties, the packet-forwarding behavior might not apply to all situations.

The following table shows the broadcast packet-forwarding rules for isolated and community PVLANs.

**TABLE 11-1** Broadcast Packet-Forwarding Rules

|             | Isolated | Community A | Community B |
|-------------|----------|-------------|-------------|
| Isolated    | No       | No          | No          |
| Community A | No       | Yes         | No          |
| Community B | No       | No          | Yes         |

For example, when both the vnet0 and vnet1 virtual networks are isolated on the net0 network, net0 does not pass broadcast traffic between the two virtual networks. However, when the net0 network receives traffic from an isolated VLAN, the traffic is not passed to the isolated ports that are related to the VLAN. This situation occurs because the isolated virtual network accepts only traffic from the primary VLAN.

The inter-vnet-links feature supports the communication restrictions of isolated and community PVLANs. Inter-vnet-links are disabled for isolated PVLANs and are enabled only for virtual networks that are in the same community for community PVLANs. Direct traffic from other virtual networks outside of the community is not permitted.

---

**Note** - If a target service domain does not support the PVLAN feature, the migration of a guest domain that is configured for PVLAN might fail.

---

## PVLAN Requirements

You can configure PVLANs by using the `ldm add-vnet` and `ldm set-vnet` commands. Use these commands to set the `pvlan` property. Note that you must also specify the `pvid` property to successfully configure the PVLAN.

This feature requires at least the Oracle Solaris 11.2.4.0.0 (SRU 4) OS.

To configure a PVLAN, you must specify the following information:

- **Primary VLAN ID.** The primary VLAN ID is the port VLAN ID (PVID) that is used to configure a PVLAN for a single virtual network device. This configuration ensures that a guest domain does receive VLAN packets. Note that you cannot configure VIDs with a PVLAN. This value is represented by the `pvid` property.
- **Secondary VLAN ID.** A secondary VLAN ID is used by a particular VLAN to provide PVLAN functionality. You specify this information as the `secondary-vid` part of the `pvlan` value. `secondary-vid` is an integer value in the range of 1-4094. A primary VLAN can have many secondary VLANs with the following restrictions:
  - Neither the primary VLAN ID nor the secondary VLAN ID can be the same as the default VLAN ID.
  - The primary VLAN ID and the secondary VLAN ID cannot have the same values for both isolated and community PVLAN types.
  - Each primary VLAN can configure only one isolated PVLAN. So, you cannot create two isolated PVLANs that use the same primary VLAN ID.
  - A primary VLAN can have multiple community VLANs with the following restrictions:
    - A primary VLAN ID cannot be used as secondary VLAN ID to create another community PVLAN.

For example, if you have a community PVLAN with a primary VLAN ID of 3 and a secondary VLAN ID of 100, you cannot create another community PVLAN that uses 3 as the secondary VLAN ID.

- A secondary VLAN ID cannot be used as primary VLAN ID to create a community PVLAN.

For example, if you have a community PVLAN with a primary VLAN ID of 3 and a secondary VLAN ID of 100, you cannot create another community PVLAN that uses 100 as the primary VLAN ID.

- The secondary VLAN ID cannot already be used as a VLAN ID for regular virtual networks or VNICs.



**Caution** - The Logical Domains Manager can validate only the configuration of the virtual networks on a particular virtual switch. If a PVLAN configuration is set up for Oracle Solaris VNICs on the same back-end device, ensure that the same requirements are met across all VNICs and virtual networks.

- **PVLAN type.** You specify this information as the *pvlan-type* part of the *pvlan* value. *pvlan-type* is one of the following values:
  - **isolated.** The ports that are associated with an isolated PVLAN are isolated from all of the peer virtual networks and Oracle Solaris virtual NICs on the back-end network device. The packets reach only the external network based on the values you specified for the PVLAN.
  - **community.** The ports that are associated with a community PVLAN can communicate with other ports that are in the same community PVLAN but are isolated from all other ports. The packets reach the external network based on the values you specified for the PVLAN.

## Configuring PVLANS

This section includes tasks that describes how to create PVLANS and list information about PVLANS.

### Creating a PVLAN

You can configure a PVLAN by setting the *pvlan* property value by using the `ldm add-vnet` or `ldm set-vnet` command. See the [ldm\(1M\)](#) man page.

You can use the following commands to create or remove a PVLAN:

- Use `ldm add-vnet` to create a PVLAN:

```
ldm add-vnet pvid=port-VLAN-ID pvlan=secondary-vid,pvlan-type if-name vswitch-name domain-name
```

The following command shows how to create a virtual network with a PVLAN that has a primary *vlan-id* of 4, a secondary *vlan-id* of 200, and a *pvlan-type* of *isolated*.

```
primary# ldm add-vnet pvid=4 pvlan=200,isolated vnet1 primary-vsw0 ldg1
```

- Use `ldm set-vnet` to create a PVLAN:

```
ldm set-vnet pvid=port-VLAN-ID pvlan=secondary-vid,pvlan-type if-name domain-name
```

The following command shows how to create a virtual network with a PVLAN that has a primary *vlan-id* of 3, a secondary *vlan-id* of 300, and a *pvlan-type* of community.

```
primary# ldm add-vnet pvid=3 pvlan=300,community vnet1 primary-vsw0 ldg1
```

- Use `ldm set-vnet` to remove a PVLAN:

```
ldm set-vnet pvlan= if-name vswitch-name domain-name
```

The following command removes the PVLAN configuration for the `vnet0` virtual network. The result of this command is that the specified virtual network is a regular VLAN that uses the *vlan-id* that you specified when you configured the PVLAN.

```
primary# ldm set-vnet pvlan= vnet0 primary-vsw0 ldg1
```

## Viewing PVLAN Information

You can view information about a PVLAN by using several of the Logical Domains Manager listing subcommands. See the [ldm\(1M\)](#) man page.

You can use the following commands to view PVLAN information.

- Use `ldm list-domain -o network` to list PVLAN information:

```
ldm list-domain [-e] [-l] -o network [-p] [domain-name...]
```

The following examples show information about PVLAN configuration on the `ldg1` domain by using the `ldm list-domain -o network` command.

- The following `ldm list-domain` command shows information about the PVLAN configuration on the `ldg1` domain.

```
primary# ldm list-domain -o network ldg1
NAME
ldg1

MAC
00:14:4f:fa:bf:0f

NETWORK
NAME SERVICE ID DEVICE MAC
vnet0 primary-vsw0@primary 0 network@0 00:14:4f:f8:03:ed
MODE PVID VID MTU MAXBW LINKPROP
 1 3 1500 1700
PVLAN : 200,community
```

- The following `ldm list-domain` command shows PVLAN configuration information in a parseable form for the `ldg1` domain.

```
primary# ldm list-domain -o network -p ldg1
VERSION 1.13
DOMAIN|name=ldg1|
MAC|mac-addr=00:14:4f:fa:bf:0f
VNET|name=vnet0|dev=network@0|service=primary-vsw0@primary
|mac-addr=00:14:4f:f8:03:ed|mode=|pvid=1|vid=3|mtu=1500|linkprop=|id=0
|alt-mac-addrs=|maxbw=1700|protect=|priority=|cos=|pvlan=200,community
```

- Use `ldm list-bindings` to list PVLAN information:

```
ldm list-bindings [-e] [-p] [domain-name...]
```

The following examples show information about PVLAN configuration on the `ldg1` domain by using the `ldm list-bindings` command.

- The following `ldm list-bindings` command shows information about the PVLAN configuration on the `ldg1` domain.

```
primary# ldm list-bindings
...
NETWORK
NAME SERVICE ID DEVICE MAC
vnet0 primary-vsw0@primary 0 network@0 00:14:4f:f8:03:ed
MODE PVID VID MTU MAXBW LINKPROP
 1 3 1500 1700
 PVLAN :200,community
PEER MAC MODE PVID VID MTU MAXBW LINKPROP
primary-vsw0@primary 00:14:4f:f8:fe:5e 1
```

- The following `ldm list-bindings` command shows PVLAN configuration information in a parseable form for the `ldg1` domain.

```
primary# ldm list-bindings -p
...
VNET|name=vnet0|dev=network@0|service=primary-vsw0@primary
|mac-addr=00:14:4f:f8:03:ed|mode=|pvid=1|vid=3|mtu=1500|linkprop=
|id=0|alt-mac-addrs=|maxbw=1700|protect=|priority=|cos=|pvlan=200,community
|peer=primary-vsw0@primary|mac-addr=00:14:4f:f8:fe:5e|mode=|pvid=1|vid=
|mtu=1500|maxbw=
```

- Use `ldm list-constraints` to list PVLAN information:

```
ldm list-constraints [-x] [domain-name...]
```

The following shows the output generated by running the `ldm list-constraints` command:

```
primary# ldm list-constraints -x ldg1
...
```

```

<Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>network</rasd:OtherResourceType>
 <rasd:Address>auto-allocated</rasd:Address>
 <gprop:GenericProperty key="vnet_name">vnet0</gprop:GenericProperty>
 <gprop:GenericProperty key="service_name">primary-vsw0</gprop:GenericProperty>
 <gprop:GenericProperty key="pvid">1</gprop:GenericProperty>
 <gprop:GenericProperty key="vid">3</gprop:GenericProperty>
 <gprop:GenericProperty key="pvlan">200,community</gprop:GenericProperty>
 <gprop:GenericProperty key="maxbw">1700000000</gprop:GenericProperty>
 <gprop:GenericProperty key="device">network@0</gprop:GenericProperty>
 <gprop:GenericProperty key="id">0</gprop:GenericProperty>
 </Item>

```

## Tuning Packet Throughput Performance

You can use the `ldm add-vnet` and `ldm set-vnet` commands to set the following data link property values to tune packet throughput performance:

|                         |                                                          |
|-------------------------|----------------------------------------------------------|
| <code>priority</code>   | Specifies the CPU packet-processing priority             |
| <code>cos</code>        | Specifies the IEEE 802.1p link service class of the link |
| <code>protection</code> | Specifies packet traffic security type                   |

For information about valid and default property values, see the [ldm\(1M\)](#) man page.

### EXAMPLE 11-9 Setting and Viewing Data Link Packet Properties

The following example shows how to use the `ldm set-vnet` command to set the `priority`, `protection`, and `cos` property values in a single command. You can also use the `ldm add-vnet` command to add a new virtual network that uses the specified data link property values.

```
primary# ldm set-vnet allowed-ips=192.168.100.1,192.168.100.2 \
allowed-dhcpcids=oracle1@system1.company.com, \
00:14:4f:f9:d3:88,system2,00:14:4f:fb:61:6e cos=7 priority=high \
protection=restricted,mac-nospoof,ip-nospoof,dhcp-nospoof vnet3_ldg3 ldg3
```

The `ldm list -o network` command shows the data link property values on the `ldg3` domain that you set with the previous `ldm set-vnet` command. The protection values are `mac-nospoof`, `restricted`, `ip-nospoof` for the `192.168.100.1,192.168.100.2` MAC address, and `dhcp-nospoof` for `system1@company.com,00:14:4f:f9:d3:88,system2,00:14:4f:fb:61:6e`. The `priority` is set to `high` and the class of service (`cos`) is set to `7`.

```
primary# ldm list -o network ldg3
```

```
NAME
ldg3

MAC
00:14:4f:fa:3b:49

VSW
NAME MAC NET-DEV ID DEVICE LINKPROP DEFAULT-VLAN-ID
PVID VID MTU MODE INTER-VNET-LINK
ldg3_vsw1 00:14:4f:f9:65:b5 net3 0 switch@0 1
 1500 on

NETWORK
NAME SERVICE ID DEVICE MAC MODE PVID VID
MTU MAXBW LINKPROP
vnet3 primary-vsw1@primary 0 network@0 00:14:4f:fa:47:f4 1
1500
vnet1_ldg3 primary-vsw1@primary 1 network@1 00:14:4f:fb:90:b7 1
1500
vnet2_ldg3 ldg1_vsw1@ldg1 2 network@2 00:14:4f:fb:61:6e 1
1500
vnet3_ldg3 ldg1_vsw1@ldg1 3 network@3 00:14:4f:fb:25:70 1
 00:14:4f:fb:7a:8d
 00:14:4f:fb:6e:32
 00:14:4f:fb:1a:2f
1500 1000
PROTECTION: mac-nospoof
 restricted
 ip-nospoof/192.168.100,1,192.168.100.2
 dhcp-nospoof/system1@oracle.us.oracle.com,
 00:14:4f:f9:d3:88,system2,00:14:4f:fb:61:6e
PRIORITY : high
COS : 7
```

## 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 enable 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 due to resource availability or virtual machine migration.

The hybrid I/O architecture is well-suited for the Network Interface Unit (NIU) on Oracle Sun UltraSPARC T2, SPARC T3, and SPARC T4 platforms. An NIU is a network I/O interface that is integrated on the 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.

---

**Note** - The NIU Hybrid I/O feature is deprecated in favor of SR-IOV. Oracle VM Server for SPARC 3.2 is the last software release to include this feature.

---

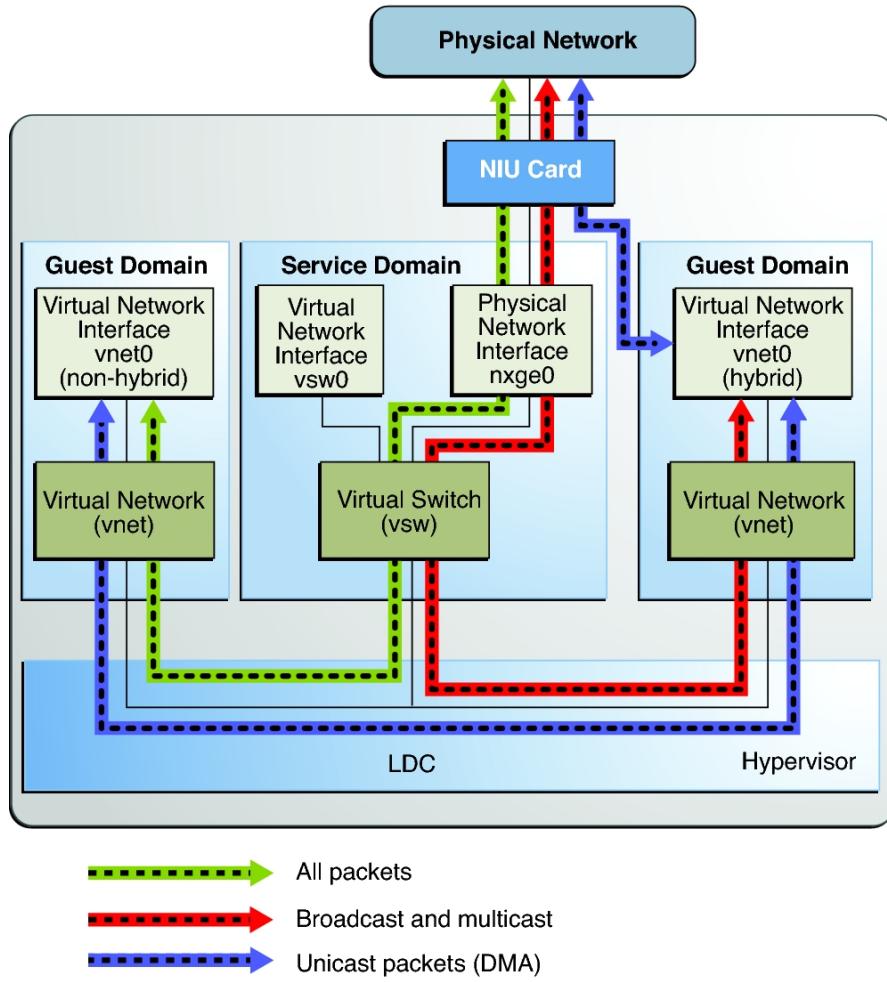
NIU hybrid I/O is available for Oracle Sun UltraSPARC T2, SPARC T3, and SPARC T4 platforms. This feature is enabled by an optional hybrid mode that provides for a virtual network (vnet) device where the DMA hardware resources are loaned to a vnet device in a guest domain for improved performance. In 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.

---

**FIGURE 11-10** Hybrid Virtual Networking



**Note** - This diagram shows the configuration on an Oracle Solaris 10 system. For an Oracle Solaris 11 system, only the interface names change to use the generic names, such as net0 for nxge0.

The hybrid mode applies only for the vnet devices that are associated with a virtual switch (vsw) configured to use an NIU network device. Because the shareable DMA hardware resources are limited, up to only three vnet devices per vsw can have DMA hardware resources.

assigned at a given time. If more than three vnet devices have the hybrid mode enabled, the assignment is done on a first-come, first-served basis. Because there are two NIU network devices in a system, there can be a total of six vnet devices on two different virtual switches with DMA hardware resources assigned.

Note the following points when using this feature:

- Hybrid mode option for a vnet device is treated as a suggestion only, so 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, you can 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. Because 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.

---

- The hybrid mode option cannot be changed dynamically while the guest domain is active.
- The DMA hardware resources are assigned only when an active vnet device 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.

## ▼ How to 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.

```
primary# 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 or SPARC T4-1 server.

```
primary# grep nxge /etc/path_to_inst
"/niu@480/network@0" 0 "nxge"
"/niu@480/network@1" 1 "nxge"
```

2. **Oracle Solaris 11 OS only:** Identify the link name that corresponds to the NIU network device, such as `nxge0`.

```
primary# dladm show-phys -L |grep nxge0
net2 nxge0 /SYS/MB
```

3. **Configure a virtual switch.**

- **Oracle Solaris 10 OS:**

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

- **Oracle Solaris 11 OS:**

The following example uses `net2` instead of `nxge0`.

```
primary# ldm add-vsw net-dev=net2 primary-vsw0 primary
```

## ▼ How to Enable or Disable Hybrid Mode

Hybrid mode is disabled by default for a vnet device and must be explicitly enabled.

- **Use the `ldm` command to enable and disable hybrid mode.**
  - **To enable a hybrid mode for a vnet device while it is being created:**  
`primary# ldm add-vnet mode=hybrid vnet01 primary-vsw0 ldom01`
  - **To disable hybrid mode for a vnet device:**  
`primary# ldm set-vnet mode= vnet01 ldom01`

## Using Link Aggregation With a Virtual Switch

A 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 “[Oracle Solaris Administration: 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.

---

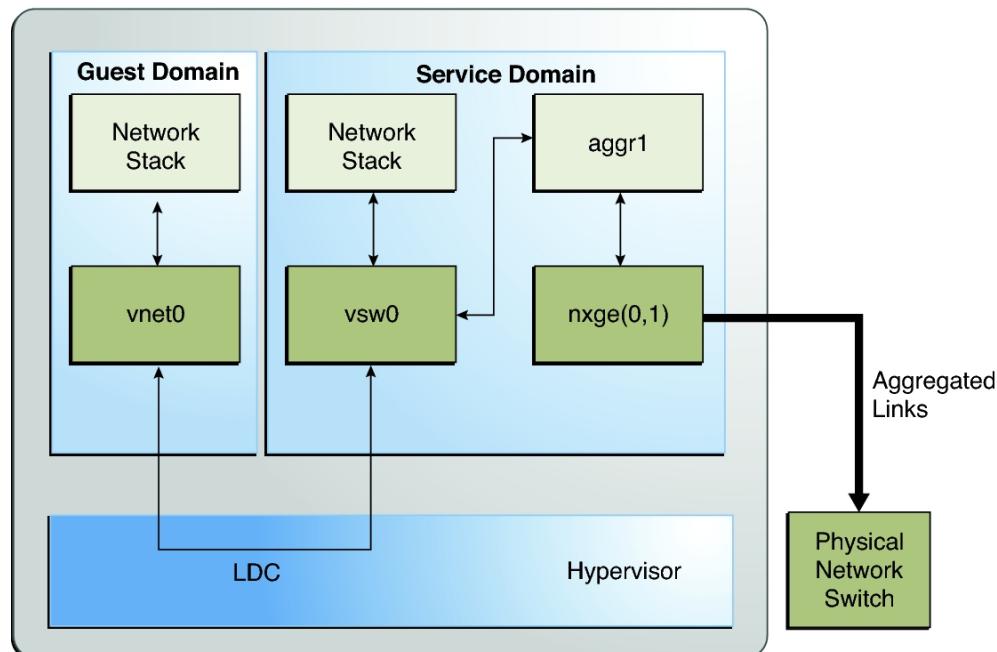
**Note** - You cannot group the virtual network devices (vnet and vsw) into a link aggregation.

---

You can create and use the virtual switch that is configured to use a link aggregation in the service domain. See “[How to Configure the Virtual Switch as the Primary Interface](#)” on page 49.

The following figure illustrates a virtual switch configured to use an aggregation, aggr1, over physical interfaces nxge0 and nxge1.

**FIGURE 11-11** Configuring a Virtual Switch to Use a Link Aggregation




---

**Note** - This diagram shows the configuration on an Oracle Solaris 10 system. For an Oracle Solaris 11 system, only the interface names change to use the generic names, such as net0 and net1 for nxge0 and nxge1, respectively.

---

## Configuring Jumbo Frames

The Oracle VM Server for SPARC virtual switch (*vsw*) and virtual network (*vnet*) devices can now support Ethernet frames with payload sizes larger than 1500 bytes. These drivers are therefore now able to increase network throughput.

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.

If the required MTU value for the virtual network device should be less than that supported by the virtual switch, you can specify an MTU value directly on a virtual network device.

---

**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 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 Oracle Solaris 10 *nxge* driver, see the [nxge\(7D\)](#) man page.

---

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 re-enable the virtual network device to obtain the MTU value from the virtual switch device, run the following command:

```
primary# ldm set-vnet mtu= vnet-name domain-name
```

Note that enabling jumbo frames for a virtual network device automatically enables jumbo frames for any hybrid I/O 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.

## ▼ How to Configure Virtual Network and Virtual Switch Devices to Use Jumbo Frames

1. Log in to the control domain.
2. Become an administrator.
  - For Oracle Solaris 10, see “Configuring RBAC (Task Map)” in “System Administration Guide: Security Services”.
  - For Oracle Solaris 11.2, see Chapter 1, “About Using Rights to Control Users and Processes,” in “Securing Users and Processes in Oracle Solaris 11.2”.

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.

```
primary# ldm add-vsw net-dev=device 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.

```
primary# ldm set-vsw net-dev=device 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.

**Example**      Configuring Jumbo Frames on Virtual Switch and Virtual Network Devices  
**11-10**

- 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.

```
primary# 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.

```
primary# ldm add-vnet vnet01 primary-vsw0 ldom1
```

Depending on the version of the Oracle Solaris OS that is running, do the following:

- **Oracle Solaris 10 OS:** 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`.

```
primary# ifconfig vsw0 plumb
primary# ifconfig vsw0 192.168.1.100/24 up
primary# ifconfig vsw0
vsw0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 9000 index 5
 inet 192.168.1.100 netmask ffffff00 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`.

```
primary# ifconfig vnet0 plumb
primary# ifconfig vnet0 192.168.1.101/24 up
primary# ifconfig vnet0
vnet0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 9000 index 4
 inet 192.168.1.101 netmask ffffff00 broadcast 192.168.1.255
 ether 0:14:4f:f9:c4:13
```

- **Oracle Solaris 11 OS:** Use the `ipadm` command to view the `mtu` property value of the primary interface.

```
ipadm show-ifprop -p mtu net0
IFNAME PROPERTY PROTO PERM CURRENT PERSISTENT DEFAULT POSSIBLE
net0 mtu ipv4 rw 9000 -- 9000 68-9000
```

The `ipadm` command creates the virtual network interface in the guest domain, `ldom1`. The `ipadm show-ifprop` command output shows that the value of the `mtu` property is `9000`.

```
primary# ipadm create-ip net0
primary# ipadm create-addr -T static -a 192.168.1.101/24 net0/ipv4
primary# ipadm show-ifprop -p mtu net0
IFNAME PROPERTY PROTO PERM CURRENT PERSISTENT DEFAULT POSSIBLE
```

```
net0 mtu ipv4 rw 9000 -- 9000 68-9000
```

- The following example shows how 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 requires a different MTU.

- **Oracle Solaris 10 OS:** Use the **ifconfig** command.

```
primary# ifconfig vnet0 mtu 4000
primary# ifconfig vnet0
vnet0: flags=1201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS,FIXEDMTU>
 mtu 4000 index 4
 inet 192.168.1.101 netmask ffffff00 broadcast 192.168.1.255
 ether 0:14:4f:f9:c4:13
```

- **Oracle Solaris 11 OS:** Use the **ipadm** command.

```
primary# ipadm set-ifprop -p mtu=4000 net0
primary# ipadm show-ifprop -p mtu net0
IFNAME PROPERTY PROTO PERM CURRENT PERSISTENT DEFAULT POSSIBLE
net0 mtu ipv4 rw 4000 -- 9000 68-9000
```

## Compatibility With Older (Jumbo-Unaware) Versions of the vnet and vsw Drivers (Oracle Solaris 10)

---

**Note** - This section applies only to the Oracle Solaris 10 OS.

---

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 Oracle VM Server for SPARC 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 limitation applies to both client guest domains and the service domain.

Therefore, while jumbo frames are enabled, ensure that all virtual devices in the Oracle VM Server for SPARC network are upgraded to use the new drivers that support jumbo frames. You must be running at least Logical Domains 1.2 to configure jumbo frames.

## Oracle Solaris 11 Networking-Specific Feature Differences

Some of the Oracle VM Server for SPARC networking features work differently when a domain runs the Oracle Solaris 10 OS as compared to the Oracle Solaris 11 OS. The feature differences for the Oracle VM Server for SPARC virtual network device and virtual switch when the Oracle Solaris 11 OS is run in a domain are as follows:

- **Configuring the `vswn` device as the primary network interface to enable a service domain to communicate with guest domains**

This configuration is required only for domains that run the Oracle Solaris 10 OS. For Oracle Solaris 11, a virtual switch uses the Oracle Solaris 11 network stack, automatically enabling its virtual network devices to communicate with the network interface that corresponds to its back-end device, such as `net0`. You can no longer use the `vswn` device itself in Oracle Solaris 11 service domains.

- **Using an Oracle Solaris 11 etherstub device as a back-end device to create a private virtual switch**

If not connected to a back-end device, a virtual switch provides communication only between guest domains and not between guest domains and the service domain. Using an etherstub as a back-end device enables a guest domain to communicate with a zone (including the global zone) that is configured in an Oracle Solaris 11 service domain. This configuration is accomplished by using a VNIC connected to that etherstub.

- **Using generic names for the virtual switch and virtual network devices**

The Oracle Solaris 11 OS assigns generic names for `vswn` and `vnetn` devices. Ensure that you do not create a virtual switch with the back-end device that is another `vsw` or `vnet` device. Use the `dladm show-phys` command to see the actual physical devices that are associated with generic network device names.

- **Using VNICs on the virtual switch and virtual network devices**

You *cannot* use VNICs on `vswn` devices. An attempt to create a VNIC on `vswn` fails. See “[Oracle Solaris 11: Zones Configured With an Automatic Network Interface Might Fail to Start](#)” in “[Oracle VM Server for SPARC 3.2 Release Notes](#)”.

## 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 253
- “[Overview of a Migration Operation](#)” on page 254
- “[Software Compatibility](#)” on page 255
- “[Security for Migration Operations](#)” on page 255
- “[FIPS 140-2 Mode for Domain Migration](#)” on page 257
- “[Domain Migration Restrictions](#)” on page 259
- “[Migrating a Domain](#)” on page 262
- “[Migrating an Active Domain](#)” on page 263
- “[Migrating Bound or Inactive Domains](#)” on page 270
- “[Performing a Dry Run](#)” on page 263
- “[Monitoring a Migration in Progress](#)” on page 272
- “[Canceling a Migration in Progress](#)” on page 272
- “[Recovering From a Failed Migration](#)” on page 273
- “[Performing Non-Interactive Migrations](#)” on page 263
- “[Migration Examples](#)” on page 273

---

**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 3.2 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 *live migration* feature 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 might 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

To achieve the best migration performance, ensure that both the source machine and the target machine are running the latest version of the Logical Domains Manager.

## 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 recopied to the target machine. In this way, there is 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:

- The Logical Domains Manager version running on both machines must be either the current version or the most recent previously released version.
- Both the source and target machines must have a compatible version of firmware installed to support live migration. Both machines must be running at least the minimum version of the firmware supported with this release of the Oracle VM Server for SPARC software.

For more information, see [“Version Restrictions for Migration” on page 259](#).

## 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 some cases. In particular, a user other than superuser must use the `LDoms Management` rights profile. However, if you perform a migration with SSL certificates, users are not required to be authenticated on both the target and source machines and you cannot specify another user.

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 12-3](#). 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 263](#).

- **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.

On platforms that have cryptographic units, 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.

The speed of a migration operation is automatically improved on platforms that have cryptographic instructions in the CPU. This improvement occurs because the SSL operations can be carried out by the cryptographic instructions rather than in software.

- **FIPS 140-2.** You can configure your system to perform domain migrations to use the Oracle Solaris FIPS 140-2 certified OpenSSL libraries. See “[FIPS 140-2 Mode for Domain Migration](#)” on page 257.

## Configuring SSL Certificates for Migration

To perform certificate-based authentication, use the -c option with the `ldm migrate-domain` command. This option is mutually exclusive with the password file and alternate user options. If the -c option is not specified, the migration operation performs password authentication.

### ▼ How to Configure SSL Certificates for Migration

To configure SSL certificates, you must perform the steps in this task on both the source machine and the target machine.

1. **Create the `/var/opt/SUNWldm/trust` directory if it does not already exist.**
2. **Securely copy the remote `ldmd` certificate to the local `ldmd` trusted certificate directory.**

The remote `ldmd` certificate is the `/var/opt/SUNWldm/server.crt` on the remote host. The local `ldmd` trusted certificate directory is `/var/opt/SUNWldm/trust`. Call the remote certificate file `remote-hostname.pem`.

3. **Create a symbolic link from the certificate in the `ldmd` trusted certificate directory to `/etc/certs/CA`.**

Set the `REMOTE` variable to `remote-host`.

```
localhost# ln -s /var/opt/SUNWldm/trust/${REMOTE}.pem /etc/certs/CA/
```

4. **Restart the `svc:/system/ca-certificates` service.**

```
localhost# svcadm restart svc:/system/ca-certificates
```

5. **Verify that the configuration is operational.**

```
localhost# openssl verify /var/opt/SUNWldm/trust/${REMOTE}.pem
/var/opt/SUNWldm/trust/remote-hostname.pem: OK
```

6. **Restart the `ldmd` daemon.**

```
localhost# svcadm restart ldmd
```

## Removing SSL Certificates

If you remove a .pem file from the /var/opt/SUNWldm/trust and /etc/certs/CA directories, you must restart the svc:/system/ca-certificates service and then the ldmd service. Note that any migrations that use that .pem file are still permitted until the services restart.

```
localhost# svcadm restart svc:/system/ca-certificates
localhost# svcadm restart ldmd
```

## FIPS 140-2 Mode for Domain Migration

You can configure your Logical Domains Manager to perform domain migrations that use the Oracle Solaris FIPS 140-2 certified OpenSSL libraries. When the Logical Domains Manager is in FIPS 140-2 mode, you can use it only to migrate domains to another system that has the Logical Domains Manager running in FIPS 140-2 mode. Attempts to migrate to a non-FIPS system are rejected. If the Logical Domains Manager is not in FIPS 140-2 mode, it cannot migrate to a Logical Domains Manager that is in FIPS 140-2 mode.

To successfully start the Logical Domains Manager in FIPS 140-2 mode, you must enable the FIPS mediator. For step-by-step instructions, see “[How to Run Logical Domains Manager in FIPS 140-2 Mode](#)” on page 257.

For more information and showing how to use the FIPS 140-capable OpenSSL implementation, see “[How to Switch to the FIPS 140-Capable OpenSSL Implementation](#)” in “[Managing Encryption and Certificates in Oracle Solaris 11.2](#)” and “[Example of Enabling Two Applications in FIPS 140 Mode on an Oracle Solaris System](#)” in “[Oracle SuperCluster M7-8 Owner’s Guide: Overview](#)”.

## ▼ How to Run Logical Domains Manager in FIPS 140-2 Mode

**Before You Begin** Before you can run the Logical Domains Manager in FIPS 140-2 mode, ensure that you are running at least version 3.2 of the Logical Domains Manager and that the primary domain runs at least the Oracle Solaris 11.2 OS.

### 1. Install and enable the FIPS 140-2 OpenSSL mediator.

#### a. Install the FIPS 140-2 OpenSSL mediator if necessary.

This package should be installed by default when you install the Oracle Solaris 11.2 OS.

```
pkg install openssl-fips-140
```

**b. List the current OpenSSL mediator.**

```
pkg mediator openssl
MEDIATOR VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
openssl vendor local default
```

**c. List the available OpenSSL mediators.**

```
pkg mediator -a openssl
MEDIATOR VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
openssl vendor vendor default
openssl system system fips-140
```



**Caution** - The OpenSSL implementation to which you switch must exist in the system. If you switch to an implementation that is not in the system, the system might become unusable.

---

**d. Enable the FIPS 140-2 mediator.**

```
pkg set-mediator -I fips-140 openssl
```

**e. Reboot.**

```
reboot
```

**f. Confirm that the FIPS 140-2 mediator is set.**

```
pkg mediator openssl
MEDIATOR VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
openssl system local fips-140
```

**2. Configure the ldmd Daemon to use FIPS 140-2 mode.**

**a. Put the ldmd daemon in FIPS 140-2 mode.**

```
svccfg -s ldoms/ldmd setprop ldmd/fips1402_enabled = true
```

**b. Restart the ldmd daemon.**

```
svcadm refresh ldmd
svcadm restart ldmd
```

## ▼ How to Revert the Logical Domains Manager to the Default Mode From FIPS 140-2 Mode

1. Stop using the FIPS 140-2 OpenSSL mediator by reverting to the default OpenSSL mediator.

Only perform this step if the FIPS 140-2 mediator is not required for other applications.

```
pkg set-mediator -I default openssl
```

2. Reboot.

```
reboot
```

3. Configure the ldmd daemon to use the default mode.

```
svccfg -s ldoms/ldmd setprop ldmd/fips1402_enabled = false
```

4. Restart the ldmd daemon.

```
svcadm refresh ldmd
svcadm restart ldmd
```

## Domain Migration Restrictions

The following sections describe restrictions for domain migration. The Logical Domains Manager software and the system firmware versions must be compatible to permit migrations. Also, you must meet certain CPU requirements to ensure a successful domain migration.

Live migration is not qualified and supported on all combinations of the source and target platforms and system firmware versions. For those combinations that cannot perform a live migration, you can perform a cold migration instead.

## Version Restrictions for Migration

This section describes version restrictions for performing live migrations.

- **Logical Domains Manager version.** You can perform a live migration in either direction when one system runs the latest version of the Logical Domains Manager and the other system runs at least the immediately preceding version of the Logical Domains Manager.
- **Oracle Solaris OS version.** You can perform a live migration of a guest domain that runs at least the Oracle Solaris 10 9/10 OS. You *cannot* perform a live migration of a guest domain

that runs the Oracle Solaris 10 10/09 OS or earlier Oracle Solaris OS versions. You can still boot these older Oracle Solaris OS versions and perform cold migrations of such domains.

- **System firmware version.** In general, you can perform a live migration between two systems when both the source and target machines support the appropriate minimum system firmware versions.

The following list shows the platforms that support live migration and the associated minimum system firmware version are:

- **UltraSPARC T2 and UltraSPARC T2 Plus platforms** – Version 7.4.5
- **SPARC T3 and SPARC T4 platforms** – Version 8.2.2.c
- **SPARC T5, SPARC M5, and SPARC M6 platforms** – All system firmware versions
- **Fujitsu M10 platforms** – All XCP versions

However, some specific platform and firmware combinations do not support live migration. Attempting to perform the live migration of a domain from a system that runs at least system firmware version 8.4 or XCP2210 to a system that runs an older system firmware version fails. The failure occurs because of a hypervisor API mismatch between the newer and older system firmware versions. In this instance, the following message is issued:

```
primary# ldm migrate ldg1 root@target-name
Target Password:
Domain ldg1 is using features of the system firmware that are not supported in
the version of the firmware running on the target machine.
Domain Migration of LDom ldg1 failed
```

Note that you can perform the live migration of a domain from a system that runs system firmware version 8.3 to a system that runs at least system firmware version 8.4 unless the target machine is a SPARC M5-32 system. For more information, see “[Domain Migrations From SPARC T4 Systems That Run System Firmware 8.3 to SPARC T5, SPARC M5, or SPARC M6 Systems Are Erroneously Permitted](#)” in “[Oracle VM Server for SPARC 3.2 Release Notes](#)”.

System firmware versions 8.4, 9.1, and XCP2230 introduced support for EFI GPT disk labels. By default, virtual disks that are installed when running at least the Oracle Solaris 11.1 OS on those systems have an EFI GPT disk label. You cannot read this disk label on older versions of firmware (such as 9.0.x, 8.3, 7.x, or XCP2221). This situation precludes you from performing a live migration or a cold migration to a system that runs a system firmware version without EFI GPT support. Note that a cold migration also fails in this situation, which is different than the previous limitations.

To determine whether your virtual disk has an EFI GPT disk label, run the `devinfo -i` command on the raw device. The following examples show whether the virtual disk has an SMI VTOC or an EFI GPT disk label:

- **SMI VTOC disk label.** When your virtual disk has an SMI VTOC, you can perform a migration to firmware regardless of whether it supports EFI.

This example indicates that the device has a VTOC label because the `devinfo -i` command reports device-specific information:

```
devinfo -i /dev/rdsk/c2d0s2
/dev/rdsk/c2d0s2 0 0 73728 512 2
```

- **EFI GPT disk label.** When your virtual disk has an EFI GPT disk label, you can perform a migration only to firmware that has EFI support.

This example indicates that the device has an EFI GPT disk label because the `devinfo -i` command reports an error:

```
devinfo -i /dev/rdsk/c1d0s0
devinfo: /dev/rdsk/c1d0s0: This operation is not supported on EFI
labeled devices
```

## CPU Restrictions for Migration

If the domain to be migrated is running an Oracle Solaris OS version older than the Oracle Solaris 10 1/13 OS, you might see the following message during the migration:

```
Domain domain-name is not running an operating system that is
compatible with the latest migration functionality.
```

The following CPU requirements and restrictions apply when you run an OS prior to the Oracle Solaris 10 1/13 OS:

- Full cores must be allocated to the migrated domain. If the number of threads in the domain to be migrated is less than a full core, the extra threads are unavailable to any domain until after the migrated domain is rebooted.
- After a migration, CPU dynamic reconfiguration (DR) is disabled for the migrated domain until it has been rebooted. At that time, you can use CPU DR on the migrated domain.
- The target machine must have enough free full cores to provide the number of threads that are required for the migrated domain. After the migration, if a full core is only partially used by the migrated domain, any extra threads are unavailable to any domain until after the migrated domain is rebooted.

These restrictions also apply when you attempt to migrate a domain that is running in OpenBoot or in the kernel debugger. See [Migrating a Domain From the OpenBoot PROM or a Domain That Is Running in the Kernel Debugger](#).

## Version Restrictions for Cross-CPU Migration

You cannot perform live migrations between an UltraSPARC T2, UltraSPARC T2 Plus, or SPARC T3 system and a SPARC T5, SPARC M5, or SPARC M6 system.

- The SPARC T4 system must run system firmware version 8.7
- The SPARC T5, SPARC M5, or SPARC M6 system must run system firmware version 9.4

- Both the source and target machines must run the Oracle VM Server for SPARC 3.2 software

## Migration Restrictions for Setting perf-counters

Take care when performing migrations of domains that have the `perf-counters` property value set.

Before you perform the migration of a domain that has the `perf-counters` property value set to `global`, ensure that no other domain on the target machine has the `perf-counters` property set to `global`.

During a migration operation, the `perf-counters` property is treated differently based on whether the performance access capability is available on the source machine, the target machine, or both.

The `perf-counters` property value is treated as follows:

- **Source machine only.** The `perf-counters` property value is not propagated to the target machine.
- **Target machine only.** The `perf-counters` property value on the machine to be migrated is updated to be equivalent to `perf-counters=`.
- **Source and target machines.** The `perf-counters` property value is propagated from the domain to be migrated to the migrated domain on the target machine.

For more information about the `perf-counters` property, see “[Using Perf-Counter Properties](#)” on page 313 and the [ldm\(1M\)](#) man page.

## 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.

---

**Note** - If you migrate a domain, any named resources that you assigned by using the `cid` and `mblock` properties are dropped. Instead, the domain uses anonymous resources on the target system.

---

For information about migrating an active domain while it continues to run, see “[Migrating an Active Domain](#)” on page 263. For information about migrating a bound or inactive domain, see “[Migrating Bound or Inactive Domains](#)” on page 270.

For information about the migration options and operands, see the [ldm\(1M\)](#) man page.

---

**Note** - After a domain migration completes, save a new configuration to the SP of both the source and target systems. As a result, the state of the migrated domain is correct if either the source or target system undergoes a power cycle.

---

## 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, a dry run could succeed and an actual migration fail, and vice versa.

---

## Performing Non-Interactive Migrations

Use the SSL certificate method to perform non-interactive migration operations. You can still use the legacy `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](#)” on page 259.

---

**Tip** - You can reduce the overall migration time by adding more virtual CPUs to the primary domain on both the source and target machines. Having at least two whole cores in each primary domain is recommended but not required.

---

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 Oracle Solaris 10 NTP client, see “[Managing Network Time Protocol \(Tasks\)](#)” in “[System Administration Guide: Network Services](#)”. To configure a domain as an Oracle Solaris 11 NTP client, see “[Managing Network Time Protocol \(Tasks\)](#)” in “[Introduction to Oracle Solaris 11 Network Services](#)”.

---

**Note** - During the suspend phase at the end of a migration, a guest domain might experience a slight delay. This delay should not result in any noticeable interrupt to network communications, especially if the protocol includes a retry mechanism such as TCP or if a retry mechanism exists at the application level such as NFS over UDP. However, if the guest domain runs a network-sensitive application such as Routing Information Protocol (RIP), the domain might experience a short delay or interrupt when attempting an operation. This delay would occur during the short period when the guest network interface is being torn down and re-created during the suspension phase.

---

## Domain Migration Requirements for CPUs

Following are the requirements and restrictions on CPUs when you perform a migration:

- The target machine must have sufficient free virtual CPUs to accommodate the number of virtual CPUs in use by the domain to be migrated.
- Setting the `cpu-arch` property on the guest domain enables you to migrate the domain between systems that have different processor types. Note that the guest domain must be in a bound or inactive state to change the `cpu-arch` value.

The supported `cpu-arch` property values are as follows:

- `native` uses CPU-specific hardware features to enable a guest domain to migrate *only* between platforms that have the same CPU type. `native` is the default value.
- `migration-class1` is a cross-CPU migration family for SPARC platforms starting with the SPARC T4. These platforms support hardware cryptography during and after these migrations so that there is a lower bound to the supported CPUs.

This value is not compatible with UltraSPARC T2, UltraSPARC T2 Plus, or SPARC T3 platforms, or Fujitsu M10 platforms.

- `sparc64-class1` is a cross-CPU migration family for SPARC64 platforms. Because the `sparc64-class1` value is based on SPARC64 instructions, it has a greater number of

instructions than the generic value. Therefore, it does not have a performance impact compared to the generic value.

This value is compatible only with Fujitsu M10 servers.

- generic uses the lowest common CPU hardware features that are used by all platforms to enable a guest domain to perform a CPU-type-independent migration.

The following `isainfo -v` commands show the instructions that are available on a system when `cpu-arch=generic` and when `cpu-arch=migration-class1`.

- `cpu-arch=generic`

```
isainfo -v
64-bit sparcv9 applications
 asi_blk_init vis2 vis popc
32-bit sparc applications
 asi_blk_init vis2 vis popc v8plus div32 mul32
```

- `cpu-arch=migration-class1`

```
isainfo -v
64-bit sparcv9 applications
 crc32c cbcond pause mont mpmul sha512 sha256 sha1 md5
 camellia des aes ima hpc vis3 fmaf asi_blk_init vis2
 vis popc
32-bit sparc applications
 crc32c cbcond pause mont mpmul sha512 sha256 sha1 md5
 camellia des aes ima hpc vis3 fmaf asi_blk_init vis2
 vis popc v8plus div32 mul32
```

Using the generic value might result in reduced performance of the guest domain compared to using the native value. The possible performance degradation occurs because the guest domain uses only generic CPU features that are available on all supported CPU types instead of using the native hardware features of a particular CPU. By not using these features, the generic value enables the flexibility of migrating the domain between systems that use CPUs that support different features.

If migrating a domain between at least SPARC T4 systems, you can set `cpu-arch=migration-class1` to improve the guest domain performance. While the performance is improved from using the generic value, the native value still provides the best performance for the guest domain.

Use the `psrinfo -pv` command when the `cpu-arch` property is set to `native` to determine the processor type, as follows:

```
psrinfo -pv
The physical processor has 2 virtual processors (0 1)
 SPARC-T5 (chipid 0, clock 3600 MHz)
```

Note that when the `cpu-arch` property is set to a value other than native, the `psrinfo -pv` output does not show the platform type. Instead, the command shows that the `sun4v-cpu` CPU module is loaded.

```
psrinfo -pv
The physical processor has 2 cores and 13 virtual processors (0-12)
 The core has 8 virtual processors (0-7)
 The core has 5 virtual processors (8-12)
 sun4v-cpu (chipid 0, clock 3600 MHz)
```

## Migration Requirements for Memory

The target machine must have sufficient free memory to accommodate the migration of a domain. In addition, the following properties must be maintained across the migration:

- You must be able 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 domain to be migrated, `ldg1`, also has 8 Gbytes of free memory that is laid out in two memory blocks. The target has the memory laid out in three memory blocks some of which are too small.

```
source# ldm ls -o memory ldg1
NAME
ldg1

MEMORY
RA PA SIZE
0x80000000 0x400000000 2G
0x400000000 0x880000000 6G

target# ldm ls-devices mem
MEMORY
PA SIZE
0x180880000000 5632M
0x301f70000000 2G
0x381b20000000 512M
```

Given this memory layout situation, the migration fails:

```
source# ldm migrate -n ldg1 target
Target Password:
Free memory layout and congruency requirements prevent binding the
memory block with PA 0x880000000, RA 0x400000000, and size 6G
Domain Migration of LDom ldg1 would fail if attempted
```

## 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.

For more information, see “[Migration Requirements for PCIe Endpoint Devices](#)” on page 268 and “[Migration Requirements for PCIe SR-IOV Virtual Functions](#)” on page 268.

## 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. This shared storage can be a SAN disk, or storage that is available by means of the NFS or iSCSI protocols. 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 any available vcc device in the control domain. If no available ports are available in the control domain, the console is created by using an available port on an available vcc device in a service domain. The migration fails if there is a conflict with the default group name.

## 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 feature, see “[Creating an I/O Domain by Assigning PCIe Endpoint Devices](#)” on page 73.

## Migration Requirements for PCIe SR-IOV Virtual Functions

You cannot perform a domain migration on an I/O domain that is configured with PCIe SR-IOV virtual functions.

For information about the SR-IOV feature, see [Chapter 8, “Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions”](#).

## 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.

Note that the NIU Hybrid I/O feature is deprecated in favor of SR-IOV. Oracle VM Server for SPARC 3.2 is the last software release to include this feature.

## Migration Requirements for Cryptographic Units

On platforms that have 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).

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. You are not permitted to initiate a delayed reconfiguration operation while a migration is in progress.

## Migrating While an Active Domain Has the Power Management Elastic Policy in Effect

You can perform a live migration when the power management (PM) elastic policy is in effect on either the source machine or the target machine.

## 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 From the OpenBoot PROM or a Domain That Is Running in the Kernel Debugger

Performing a domain migration requires coordination between the Logical Domains Manager and the Oracle Solaris 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 (kmdb), this coordination is not possible. As a result, the migration attempt fails.

When a domain to be migrated is running in OpenBoot, you will see the following message:

```
primary# ldm migrate ldg1 system2
Migration is not supported while the domain ldg1 is in the 'OpenBoot Running' state
Domain Migration of LDom ldg1 failed
```

When a domain to be migrated is running in the kernel debugger (kmdb), you will see the following message:

```
primary# ldm migrate ldg1 system2
Migration is not supported while the domain ldg1 is in the 'Solaris debugging' state
Domain Migration of LDom ldg1 failed
```

## 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. Therefore, you can migrate between different platform types, such as SPARC T3 to SPARC T5 platforms or Fujitsu M10 platforms, because no runtime state is being copied across.

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.



**Caution** - When you migrate a bound domain, the virtual disk back-end options and `mpgroup` values are not checked because no runtime state information is exchanged with the target machine. This check *does* occur when you migrate an active domain.

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.

**Note** - After a domain migration completes, save a new configuration to the SP of both the source and target systems. As a result, the state of the migrated domain is correct if either the source or target system undergoes a power cycle.

## Migration Requirements for Virtual I/O Devices

For an inactive domain, no checks are performed of the virtual I/O (VIO) constraints. Therefore, 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. This requirement applies to bound domains but not to inactive domains.

For information about the direct I/O (DIO) feature, see “[Creating an I/O Domain by Assigning PCIe Endpoint Devices](#)” on page 73.

## Migration Requirements for PCIe SR-IOV Virtual Functions

You cannot perform a domain migration on an I/O domain that is configured with PCIe SR-IOV virtual functions. This requirement applies to bound domains but not to inactive domains.

For information about the SR-IOV feature, see [Chapter 8, “Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions”](#).

## 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:

- s – The domain that is the source of the migration.
- + – The migrated domain that is the target of the migration.
- e – An error has occurred that requires user intervention.

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% t5-sys-2
```

## Cancelling 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.

## Recovering From a Failed Migration

The migration operation terminates if the network connection is lost 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 12-1 Using SSL Certificates to Perform a Guest Domain Migration

This example shows how to migrate the `ldg1` domain to a machine called `t5-sys-2`. Before the migration operation starts, you must have configured the SSL certificates on both the source and target machines. See “[How to Configure SSL Certificates for Migration](#)” on page 256.

```
ldm migrate-domain -c ldg1 t5-sys-2
```

### EXAMPLE 12-2 Migrating a Guest Domain

This example shows how to migrate the `ldg1` domain to a machine called `t5-sys-2`.

```
ldm migrate-domain ldg1 t5-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 t5-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, `t5-sys-2`.

**EXAMPLE 12-3** 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 (`t5-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@t5-sys-2:ldg-tgt
Target Password:
```

**EXAMPLE 12-4** 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:
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 12-5** 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 `t5-sys-1`.

```
ldm list -o status ldg-tgt
NAME
ldg-tgt

STATUS
OPERATION PROGRESS SOURCE
migration 55% t5-sys-1
```

**EXAMPLE 12-6** 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 t5-sys-2.

```
ldm list -o status -p ldg-src
VERSION 1.6
DOMAIN|name=ldg-src|
STATUS
|op=migration|progress=42|error=no|target=t5-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 277
- “Resource Allocation” on page 279
- “CPU Allocation” on page 280
- “Configuring the System With Hard Partitions” on page 284
- “Assigning Physical Resources to Domains” on page 291
- “Using Memory Dynamic Reconfiguration” on page 295
- “Using Resource Groups” on page 303
- “Using Power Management” on page 303
- “Using Dynamic Resource Management” on page 304
- “Listing Domain Resources” on page 307
- “Using Perf-Counter Properties” on page 313

### 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 and the Oracle Solaris 11 OS
- **Virtual I/O devices** – Supported in at least the Oracle Solaris 10 10/08 OS and the Oracle Solaris 11 OS
- **Cryptographic units** – Supported in at least the Oracle Solaris 10 1/13 OS and the Oracle Solaris 11 OS
- **Memory** – See “[Using Memory Dynamic Reconfiguration](#)” on page 295
- **CPU whole cores** – See “[Fully Qualified Oracle Solaris OS Versions](#)” in “[Oracle VM Server for SPARC 3.2 Installation Guide](#)”
- **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 if no OS is running

In general, 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.

Delayed reconfiguration operations are restricted to the control domain. You can run a limited number of commands while a delayed reconfiguration on the root domain is in progress to support operations that cannot be completed dynamically. These subcommands are `add-io`, `set-io`, `remove-io`, `create-vf`, and `destroy-vf`. You can also run the `ldm start-reconf` command on the root domain. For all other domains, you must stop the domain to modify the configuration unless the resource can be dynamically reconfigured.

While a delayed reconfiguration is in progress, other reconfiguration requests for that domain are deferred until it is rebooted or stopped and started.

The `ldm cancel-reconf` command cancels delayed reconfiguration operations on the 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-reconf` command if any other `ldm remove-*` commands have already performed a delayed reconfiguration operation on virtual I/O devices. The `ldm cancel-reconf` command fails in this circumstance.

---

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 a Domain Might Fail](#)” in “[Oracle VM Server for SPARC 3.2 Release Notes](#)”. To remove large amounts of memory from the control domain, see “[Decrease the Control Domain's Memory](#)” on page 297.

---

**Note -** When the primary domain is in a delayed reconfiguration state, resources that are managed by Oracle VM Server for SPARC are power-managed *only* after the primary domain reboots. Resources that are managed directly by the OS, such as CPUs that are managed by the Solaris Power Aware Dispatcher, are not affected by this state.

---

## Only One CPU Configuration Operation Is Permitted to Be Performed During a Delayed Reconfiguration

Do not attempt to perform more than one CPU configuration operation on the primary domain while it is in a delayed reconfiguration. If you attempt more CPU configuration requests, they will be rejected.

**Workaround:** Perform one of the following actions:

- Cancel the delayed reconfiguration, start another one, and request the configuration changes that were lost from the previous delayed reconfiguration.
- Reboot the control domain with the incorrect CPU count and then make the allocation corrections after the domain reboots.

## Resource Allocation

The resource allocation mechanism uses resource allocation constraints 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.



**Caution** - Do not create a circular dependency between two domains in which each domain provides services to the other. Such a configuration creates a single point of failure condition where an outage in one domain causes the other domain to become unavailable. Circular dependency configurations also prevent you from unbinding the domains after they have been bound initially.

The Logical Domains Manager does not prevent the creation of circular domain dependencies.

If the domains cannot be unbound due to a circular dependency, remove the devices that cause the dependency and then attempt to unbind the domains.

---

## CPU Allocation

When you run threads from the same core in separate domains, you might experience unpredictable and poor performance. The Oracle VM Server for SPARC software uses the CPU affinity feature to optimize CPU allocation during the logical domain binding process, which occurs before you can start the domain. This feature attempts to keep threads from the same core allocated to the same logical domain because this type of allocation improves cache sharing between the threads within the same core.

CPU affinity attempts to avoid the sharing of cores among domains unless there is no other recourse. When a domain has been allocated a partial core and requests more strands, the strands from the partial core are bound first, and then another free core is located to fulfill the request, if necessary.

The CPU allocation mechanism uses the following constraints for CPU resources:

- **Whole-core constraint.** This constraint specifies that CPU cores are allocated to a domain rather than virtual CPUs. As long as the domain does not have the max-cores constraint enabled, the whole-core constraint can be added or removed by using the `ldm set-core` or `ldm set-vcpu` command, respectively. The domain can be inactive, bound, or active. However, enough cores must be available to satisfy the request to apply the constraint. As a worst-case example, if a domain that shares cores with another domain requests the whole-core constraint, cores from the free list would need to be available to satisfy the request. As a best-case example, all the virtual CPUs in the core are already on core boundaries, so the constraint is applied without changes to CPU resources.
- **Maximum number of cores (max-cores) constraint.** This constraint specifies the maximum number of cores that can be assigned to a bound or active domain.

## ▼ How to Apply the Whole-Core Constraint

Ensure that the domain has the whole-core constraint enabled prior to setting the max-cores constraint.

1. **Apply the whole-core constraint on the domain.**

```
primary# ldm set-core 1 domain-name
```

2. **Verify that the domain has the whole-core constraint enabled.**

```
primary# ldm ls -o resmgmt domain-name
```

Notice that max-cores is set to unlimited. The domain cannot be used in conjunction with hard partitioning until the max-cores constraint is enabled.

### Example 13-1 Applying the Whole-Core Constraint

This example shows how to apply the whole-core constraint on the ldg1 domain. The first command applies the constraint, while the second command verifies that it is enabled.

```
primary# ldm set-core 1 ldg1
primary# ldm ls -o resmgmt ldg1
NAME
ldg1

CONSTRAINT
cpu=whole-core
max-cores=unlimited
```

## ▼ How to Apply the Max-Cores Constraint

Ensure that the domain has the whole-core constraint enabled prior to setting the max-cores constraint.

You can only enable, modify, or disable the max-cores constraint on an inactive domain, not on a domain that is bound or active. Before you update the max-cores constraint on the control domain, you must first initiate a delayed reconfiguration.

1. **Enable the max-cores constraint on the domain.**

```
primary# ldm set-domain max-cores=max-number-of-CPU-cores domain-name
```

---

**Note** - 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 “orphaned.”

---

2. **Verify that the whole-core constraint is enabled.**

```
primary# ldm ls -o resmgmt domain-name
```

3. **Bind and restart the domain.**

```
primary# ldm bind domain-name
primary# ldm start domain-name
```

Now, you can use the domain with hard partitioning.

**Example 13-2 Applying the Max-Cores Constraint**

This example shows how to constrain max-cores to three cores by setting the `max-cores` property, and verifying that the constraint is enabled:

```
primary# ldm set-domain max-cores=3 ldg1
primary# ldm ls -o resmgmt ldg1
NAME
ldg1

CONSTRAINT
cpu=whole-core
max-cores=3
```

Now, you can use the domain with hard partitioning.

The following example removes the max-cores constraint from the unbound and inactive `ldg1` domain, but leaves the whole-core constraint as-is.

```
primary# ldm stop ldg1
primary# ldm unbind ldg1
primary# ldm set-domain max-cores=unlimited ldg1
```

Alternately, to remove both the max-cores constraint and the whole-core constraint from the `ldg1` domain, assign virtual CPUs instead of cores, as follows:

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

In either case, bind and restart the domain.

```
primary# ldm bind ldg1
```

```
primary# ldm start ldg1
```

## 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 283
- “[Dynamic Resource Management](#)” on page 283

### 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-core`, `ldm set-core`, or `ldm remove-core` 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 expected interactions between the whole-core constraint and DRM are as follows:

- 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.

## Configuring the System With Hard Partitions

This section describes hard partitioning with the Oracle VM Server for SPARC software, and how to use hard partitioning to conform to the Oracle CPU licensing requirements.

For information about Oracle's hard partitioning requirements for software licenses, see [Partitioning: Server/Hardware Partitioning \(`http://www.oracle.com/us/corporate/pricing/partitioning-070609.pdf`\)](http://www.oracle.com/us/corporate/pricing/partitioning-070609.pdf).

- **CPU cores and CPU threads.** The Oracle VM Server for SPARC software runs on SPARC T-Series and SPARC M-Series platforms, and Fujitsu M10 platforms. The processors that are used in these systems have multiple CPU cores, each of which contains multiple CPU threads.
- **Hard partitioning and CPU whole cores.** Beginning with the Oracle VM Server for SPARC 2.0 release, hard partitioning is enforced by using CPU whole-core configurations. A CPU whole-core configuration has domains that are allocated CPU whole cores instead of individual CPU threads. By default, a domain is configured to use CPU threads.

When binding a domain in a whole-core configuration, the system provisions the specified number of CPU cores and all its CPU threads to the domain. Using a CPU whole-core configuration limits the number of CPU cores that can be dynamically assigned to a bound or active domain.
- **Oracle hard partition licensing.** To conform to the Oracle hard partition licensing requirement, you must use at least the Oracle VM Server for SPARC 2.0 release. You must also use CPU whole cores as follows:
  - A domain that runs applications that use Oracle hard partition licensing must be configured with CPU whole cores.
  - A domain that does not run applications that use Oracle hard partition licensing is not required to be configured with CPU whole cores. For example, if you do not run any Oracle applications in the control domain, that domain is not required to be configured with CPU whole cores.

## Checking the Configuration of a Domain

You use the `ldm list -o` command to determine whether a domain is configured with CPU whole cores and how to list the CPU cores that are assigned to a domain.

- To determine whether the domain is configured with CPU whole cores:

```
primary# ldm list -o resmgmt domain-name
```

Verify that the whole-core constraint appears in the output and that the `max-cores` property specifies the maximum number of CPU cores that are configured for the domain. See the [`ldm\(1M\)`](#) man page.

The following command shows that the `ldg1` domain is configured with CPU whole cores and a maximum of five cores:

```
primary# ldm list -o resmgmt ldg1
NAME
ldg1
```

```
CONSTRAINT
whole-core
max-cores=5
```

- When a domain is bound, CPU cores are assigned to the domain. To list the CPU cores that are assigned to a domain:

```
primary# ldm list -o core domain-name
```

The following command shows the cores that are assigned to the `ldg1` domain:

```
primary# ldm list -o core ldg1
NAME
ldg1
```

```
CORE
CID PCPUSET
1 (8, 9, 10, 11, 12, 13, 14, 15)
2 (16, 17, 18, 19, 20, 21, 22, 23)
```

## Configuring a Domain With CPU Whole Cores

The tasks in this section explain how to create a new domain with CPU whole cores, how to configure an existing domain with CPU whole cores, and how to configure the `primary` domain with CPU whole cores.

---

**Note** - The `ldm` subcommands that are used to assign whole cores changed in the Oracle VM Server for SPARC 2.2 release.

The tasks and examples in this section use the new commands that were introduced with the Oracle VM Server for SPARC 2.2 software.

If you are using version 2.0 or 2.1 of the Logical Domains Manager to assign whole cores to domains, use the `ldm add-vcpu -c`, `ldm set-vcpu -c`, and `ldm remove-vcpu -c` commands instead of the `ldm add-core`, `ldm set-core`, and `ldm remove-core` commands, respectively.

---

Oracle VM Server for SPARC 3.2 is the last software release to use the `-c` option in this way.

Use the following command to configure a domain to use CPU whole cores:

```
ldm set-core number-of-CPU-cores domain
```

This command also specifies the maximum number of CPU cores for the domain, which is the CPU cap. See the [ldm\(1M\)](#) man page.

The CPU cap and the allocation of CPU cores is handled by separate commands. By using these commands, you can independently allocate CPU cores, set a cap, or both. The allocation unit can be set to cores even when no CPU cap is in place. However, running the system in this mode is *not* acceptable for configuring hard partitioning on your Oracle VM Server for SPARC system.

- Allocate the specified number of CPU cores to a domain by using the `add-core`, `set-core`, or `rm-core` subcommand.
- Set the CPU cap by using the `create-domain` or `set-domain` subcommand to specify the `max-cores` property value.

You must set the cap if you want to configure hard partitioning on your Oracle VM Server for SPARC system.

## ▼ How to Create a New Domain With CPU Whole Cores

---

**Note** - You only need to stop and unbind the domain if you optionally set the max-cores constraint.

---

### 1. Create the domain.

```
primary# ldm create domain-name
```

### 2. Set the number of CPU whole cores for the domain.

```
primary# ldm set-core number-of-CPU-cores domain
```

### 3. (Optional) Set the max-cores property for the domain.

```
primary# ldm set-domain max-cores=max-number-of-CPU-cores domain
```

### 4. Configure the domain.

During this configuration, ensure that you use the `ldm add-core`, `ldm set-core`, or `ldm rm-core` command.

### 5. Bind and start the domain.

```
primary# ldm bind domain-name
primary# ldm start domain-name
```

**Example 13-3** Creating a New Domain With Two CPU Whole Cores

This example creates a domain, `ldg1`, with two CPU whole cores. The first command creates the `ldg1` domain. The second command configures the `ldg1` domain with two CPU whole cores.

At this point, you can perform further configuration on the domain, subject to the restrictions described in Step 3 in “[How to Create a New Domain With CPU Whole Cores](#)” on page 286.

The third and fourth commands show how to bind and start the `ldg1` domain, at which time you can use the `ldg1` domain.

```
primary# ldm create ldg1
primary# ldm set-core 2 ldg1
...
primary# ldm bind ldg1
primary# ldm start ldg1
```

## ▼ How to Configure an Existing Domain With CPU Whole Cores

If a domain already exists and is configured to use CPU threads, you can change its configuration to use CPU whole cores.

- 1. (Optional) Stop and unbind the domain.**

This step is required only if you also set the max-cores constraint.

```
primary# ldm stop domain-name
primary# ldm unbind domain-name
```

- 2. Set the number of CPU whole cores for the domain.**

```
primary# ldm set-core number-of-CPU-cores domain
```

- 3. (Optional) Set the max-cores property for the domain.**

```
primary# ldm set-domain max-cores=max-number-of-CPU-cores domain
```

- 4. (Optional) Rebind and restart the domain.**

This step is required only if you also set the max-cores constraint.

```
primary# ldm bind domain-name
primary# ldm start domain-name
```

**Example 13-4** Configuring an Existing Domain With Four CPU Whole Cores

This example updates the configuration of an existing domain, `ldg1` by configuring it with four CPU whole cores.

```
primary# ldm set-core 4 ldg1
```

## ▼ How to Configure the Primary Domain With CPU Whole Cores

If the primary domain is configured to use CPU threads, you can change its configuration to use CPU whole cores.

1. **(Optional) Place the primary domain in delayed reconfiguration mode.**

You need to initiate a delayed reconfiguration only if you want to modify the `max-cores` property.

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

2. **Set the number of CPU whole cores for the primary domain.**

```
primary# ldm set-core number-of-CPU-cores primary
```

3. **(Optional) Set the max-cores property for the primary domain.**

```
primary# ldm set-domain max-cores=max-number-of-CPU-cores primary
```

4. **(Optional) Reboot the primary domain.**

Use the appropriate procedure to reboot the primary domain depending on the system configuration. See “[Rebooting the Root Domain With PCIe Endpoints Configured](#)” on page 78.

You need to reboot the domain only if you want to modify the `max-cores` property.

**Example 13-5** Configuring the Control Domain With Two CPU Whole Cores

This example configures CPU whole cores on the primary domain. The first command initiates delayed reconfiguration mode on the primary domain. The second command configures the primary domain with two CPU whole cores. The third command sets the `max-cores` property to 2, and the fourth command reboots the primary domain.

```
primary# ldm start-reconf primary
primary# ldm set-core 2 primary
primary# ldm set-domain max-cores=2 primary
primary# shutdown -i 5
```

The optional Steps 1 and 4 are required only if you want to modify the `max-cores` property.

## Interaction of Hard Partitioned Systems With Other Oracle VM Server for SPARC Features

This section describes how hard partitioned systems interact with other Oracle VM Server for SPARC features.

## CPU Dynamic Reconfiguration

You can use CPU dynamic reconfiguration with domains that are configured with CPU whole cores. However, you can add or remove only entire CPU cores, not individual CPU threads. The hard partitioning state of the system is maintained by the CPU dynamic reconfiguration feature. In addition, if CPU cores are dynamically added to a domain, the maximum is enforced. Therefore, the CPU DR command would fail if it attempted to exceed the maximum number of CPUs.

---

**Note** - The `max-cores` property cannot be altered unless the domain is stopped and unbound. So, to increase the maximum number of cores from the value specified at the time the whole-core constraint was set, you must first stop and unbind the domain.

---

Use the following commands to dynamically add to or remove CPU whole cores from a bound or active domain and to dynamically set the number of CPU whole cores for a bound or active domain:

```
ldm add-core number-of-CPU-cores domain
ldm rm-core number-of-CPU-cores domain
ldm set-core number-of-CPU-cores domain
```

---

**Note** - If the domain is not active, these commands also adjust the maximum number of CPU cores for the domain. If the domain is bound or active, these commands do not affect the maximum number of CPU cores for the domain.

---

### EXAMPLE 13-6 Dynamically Adding Two CPU Whole Cores to a Domain

This example shows how to dynamically add two CPU whole cores to the `ldg1` domain. The `ldg1` domain is an active domain that has been configured with CPU whole cores. The first command shows that the `ldg1` domain is active. The second command shows that the `ldg1` domain is configured with CPU whole cores and a maximum of four CPU cores. The third and fifth commands show the CPU cores that are assigned to the domain before and after the addition of two CPU whole cores. The fourth command dynamically adds two CPU whole cores to the `ldg1` domain.

```
primary# ldm list ldg1
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
ldg1 active -n---- 5000 16 2G 0.4% 5d 17h 49m
primary# ldm list -o resmgmt ldg1
NAME
ldg1
CONRAINT
```

```
whole-core
 max-cores=4
primary# ldm list -o core ldg1
NAME
ldg1

CORE
CID PCPUSER
1 (8, 9, 10, 11, 12, 13, 14, 15)
2 (16, 17, 18, 19, 20, 21, 22, 23)
primary# ldm add-core 2 ldg1
primary# ldm list -o core ldg1
NAME
ldg1

CORE
CID PCPUSER
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)
```

## CPU Dynamic Resource Management

Dynamic resource management (DRM) can be used to automatically manage CPU resources on some domains. If DRM is used, the DRM policies do not apply to domains that are configured with CPU whole cores.

A DRM policy can include a domain that is configured with CPU whole cores. However, when such a policy is activated, it is automatically disabled for that domain. The domain remains configured with CPU whole cores unless and until the domain is later reconfigured with CPU threads instead of CPU whole cores. When the domain is configured to use CPU threads, the DRM policy is automatically re-enabled for that domain.

## Power Management

You can set a separate power management (PM) policy for each hard-partitioned domain.

## Domain Reboot or Rebind

A domain that is configured with CPU whole cores remains configured with CPU whole cores when the domain is restarted, or if the entire system is restarted. A domain uses the same

physical CPU cores for the entire time it remains bound. For example, if a domain is rebooted, it uses the same physical CPU cores both before and after the reboot. Or, if the entire system is powered off while a domain is bound, that domain will be configured with the same physical CPU cores when the system is powered on again. If you unbind a domain and then rebind it, or if the entire system is restarted with a new configuration, the domain might use different physical CPU cores.

## Assigning Physical Resources to Domains

The Logical Domains Manager automatically selects the physical resources to be assigned to a domain. The Oracle VM Server for SPARC 3.2 software also enables expert administrators to explicitly choose the physical resources to assign to or remove from a domain.

Resources that you explicitly assign are called *named resources*. Resources that are automatically assigned are called *anonymous resources*.



---

**Caution** - Do not assign named resources unless you are an expert administrator.

---

You can explicitly assign physical resources to the control domain and to guest domains. Because the control domain remains active, the control domain might optionally be in a delayed reconfiguration before you make physical resource assignments. Or, a delayed reconfiguration is automatically triggered when you make physical assignments. See “[Managing Physical Resources on the Control Domain](#)” on page 294. For information about physical resource restrictions, see “[Restrictions for Managing Physical Resources on Domains](#)” on page 294.

You can explicitly assign the following physical resources to the control domain and to guest domains:

- **Physical CPUs.** Assign the physical core IDs to the domain by setting the `cid` property.

The `cid` property should be used *only* by an administrator who is knowledgeable about the topology of the system to be configured. This advanced configuration feature enforces specific allocation rules and might affect the overall performance of the system.

You can set this property by running any of the following commands:

```
ldm add-core cid=core-ID[,core-ID[,...]] domain-name
ldm set-core cid=core-ID[,core-ID[,...]] domain-name
ldm rm-core [-f] cid=core-ID[,core-ID[,...]] domain-name
```

If you specify a core ID as the value of the `cid` property, *core-ID* is explicitly assigned to or removed from the domain.

---

**Note** - You cannot use the `ldm add-core` command to add named core resources to a domain that already uses anonymous core resources.

---

- **Physical memory.** Assign a set of contiguous physical memory regions to a domain by setting the `mblock` property. Each physical memory region is specified as a physical memory start address and a size.

The `mblock` property should be used *only* by an administrator who is knowledgeable about the topology of the system to be configured. This advanced configuration feature enforces specific allocation rules and might affect the overall performance of the system.

You can set this property by running any of the following commands:

```
ldm add-mem mblock=PA-start:size[,PA-start:size[,...]] domain-name
```

```
ldm set-mem mblock=PA-start:size[,PA-start:size[,...]] domain-name
```

```
ldm rm-mem mblock=PA-start:size[,PA-start:size[,...]] domain-name
```

To assign a memory block to or remove it from a domain, set the `mblock` property. A valid value includes a physical memory starting address (*PA-start*) and a memory block size (*size*), separated by a colon (:).

---

**Note** - You cannot use dynamic reconfiguration (DR) to move memory or core resources between running domains when you set the `mblock` or `cid` property. To move resources between domains, ensure that the domains are in a bound or inactive state. For information about managing physical resources on the control domain, see “[Managing Physical Resources on the Control Domain](#)” on page 294.

---

---

**Note** - If you migrate a domain, any named resources that you assigned by using the `cid` and `mblock` properties are dropped. Instead, the domain uses anonymous resources on the target system.

---

You can use the `ldm list-constraints` command to view the resource constraints for domains. The `physical-bindings` constraint specifies which resource types have been physically assigned to a domain. When a domain is created, the `physical-bindings` constraint is unset until a physical resource is assigned to that domain.

The `physical-bindings` constraint is set to particular values in the following cases:

- `memory` when the `mblock` property is specified
- `core` when the `cid` property is specified
- `core, memory` when both the `cid` and `mblock` properties are specified

## ▼ How to Remove the physical-bindings Constraint

To remove the physical-bindings constraint for a guest domain, you must first remove all physically bound resources.

### 1. Unbind the domain.

```
primary# ldm unbind domain-name
```

### 2. Remove the named resources.

- To remove named cores:

```
primary# ldm set-core cid=core-ID domain-name
```

- To remove named memory:

```
primary# ldm set-mem mblock=PA-start:size domain-name
```

### 3. Add CPU or memory resources.

- To add a CPU resource:

```
primary# ldm add-vcpu number domain-name
```

- To add a memory resource:

```
primary# ldm add-mem size[unit] domain-name
```

### 4. Rebind the domain.

```
primary# ldm bind domain-name
```

## ▼ How to Remove All Non-Physically Bound Resources

To constrain guest domains that do not have the physical-bindings constraint, you must first remove all non-physically bound resources.

### 1. Unbind the domain.

```
primary# ldm unbind domain-name
```

### 2. Set the number of resources to 0.

- To set the CPU resource:

```
primary# ldm set-core 0 domain-name
```

- To set the memory resource:

```
primary# ldm set-mem 0 domain-name
```

**3. Add CPU or memory resources that are physically bound.**

- To add a CPU resource:

```
primary# ldm add-core cid=core-ID domain-name
```

- To add a memory resource:

```
primary# ldm add-mem mblock=PA-start:size domain-name
```

**4. Rebind the domain.**

```
primary# ldm bind domain-name
```

## Managing Physical Resources on the Control Domain

To constrain or remove the physical-bindings constraint from the control domain, follow the appropriate steps in the previous section. However, instead of unbinding the domain, place the control domain in a delayed reconfiguration.

A change of constraint between anonymous resources and physically bound named resources auto-triggers a delayed reconfiguration. You can still explicitly enter a delayed reconfiguration by using the `ldm start-reconf primary` command.

As with any delayed reconfiguration change, you must perform a reboot of the domain, in this case the control domain, to complete the process.

---

**Note -** When the control domain is in delayed reconfiguration mode, you can perform unlimited memory assignments by using the `ldm add-mem` and `ldm rm-mem` commands on the control domain. However, you can perform only *one* core assignment to the control domain by using the `ldm set-core` command.

---

## Restrictions for Managing Physical Resources on Domains

The following restrictions apply to the assignment of physical resources:

- You cannot make physical and non-physical memory bindings, or physical and non-physical core bindings, in the same domain.

- You can have non-physical memory and physical core bindings, or non-physical core and physical memory bindings, in the same domain.
- When you add a physical resource to a domain, the corresponding resource type becomes constrained as a physical binding.
- Attempts to add anonymous CPUs to or remove them from a domain where `physical-bindings=core` will fail.
- For unbound resources, the allocation and checking of the resources can occur *only* when you run the `ldm bind` command.
- When removing physical memory from a domain, you must remove the *exact* physical memory block that was previously added.
- Physical memory ranges must *not* overlap.
- You can use only the `ldm add-core cid=` or `ldm set-core cid=` command to assign a physical resource to a domain.
- If you use the `ldm add-mem mblock=` or `ldm set-mem mblock=` command to assign multiple physical memory blocks, the addresses and sizes are checked immediately for collisions with other bindings.
- A domain that has partial cores assigned to it can use the whole-core semantics if the remaining CPUs of those cores are free and available.

## Using Memory Dynamic Reconfiguration

Memory dynamic reconfiguration (DR) is capacity-based and enables you to add an arbitrary amount of memory to or remove it from an active logical domain.

The requirements and restrictions for using the memory DR feature are as follows:

- 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 297.
- 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 299.

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.

Under certain circumstances, the Logical Domains Manager rounds up the requested memory allocation to either the next largest 8-Kbyte or 4-Mbyte multiple. The following example shows sample output of the `ldm list-domain -l` command, where the constraint value is smaller than the actual allocated size:

```
Memory:
 Constraints: 1965 M
```

|           |             |       |
|-----------|-------------|-------|
| raddr     | paddr5      | size  |
| 0x1000000 | 0x291000000 | 1968M |

## 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 `ldm remove-memory` command by running the `ldm list -l` command for the specified domain.

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 request to dynamically add memory to or remove memory from a domain might only be partially fulfilled. This result depends on the availability of suitable memory to add or remove, respectively.

---

**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

You can use the memory DR feature 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. For those domains, the request 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. Therefore, 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. Therefore, 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 298 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. Therefore, any unaligned memory in an active domain cannot be removed by using memory DR.

Also, 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 command initiates a delayed reconfiguration operation on the control domain.

```
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.
```

The following command adds one of the 128-Mbyte memory blocks to the control domain.

```
primary# 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.

primary# ldm list
NAME STATE FLAGS CONS VCPU MEMORY UTIL UPTIME
primary active -ndcv- SP 8 2688M 0.1% 23d 8h 8m

primary# 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
```

The following command adds the other 128-Mbyte memory block to the `ldom1` domain.

```
primary# ldm add-mem 128M ldom1
primary# 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 13-7 Memory DR Operations on Active Domains

This example shows how to dynamically add memory to and remove it from an active domain, ldom1.

The ldm list output shows the memory for each domain in the Memory field.

```
primary# 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
```

The following 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.

```
primary# ldm add-mem 200M ldom1
The size of memory must be a multiple of 256MB.
```

```
primary# 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.
```

```
primary# 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
```

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 because the amount of memory is rounded down to the next 256-Mbyte boundary.

```
primary# 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.
```

```
primary# 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 13-8** 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.

```
primary# 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

primary# ldm add-mem 100M ldom2

primary# 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

primary# 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.

primary# 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

primary# ldm rm-mem 100M ldom2
primary# 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

primary# 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.
```

```
primary# 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 13-9** Setting Domain Memory Sizes

This example shows how to use the `ldm set-mem` 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.

```
primary# 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
```

The following `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.

```
primary# 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.
```

```
primary# 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.
```

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.

```
primary# ldm set-mem 690M ldom2
primary# 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

primary# 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.

```
primary# 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 Resource Groups

A *resource group* provides an alternate way to view the resources in a system. Resources are grouped based on the underlying physical relationships between processor cores, memory, and I/O buses. Different platforms, and even different platform configurations within the same server family, such as SPARC T5-2 and SPARC T5-8, can have different resource groups that reflect the differences in the hardware. Use the `ldm list-resrc-group` command to view resource group information.

The membership of resource groups is statically defined by the hardware configuration. You can use the `ldm remove-core` and `ldm remove-memory` commands to operate on resources from a particular resource group.

- The `remove-core` subcommand specifies the number of CPU cores to remove from a domain. When you specify a resource group by using the `-g` option, the cores that are selected for removal all come from that resource group.
- The `remove-memory` subcommand removes the specified amount of memory from a logical domain. When you specify a resource group by using the `-g` option, the memory that is selected for removal all comes from that resource group.

For information about these commands, see the [ldm\(1M\)](#) man page.

Resource group information is not supported on UltraSPARC T2, UltraSPARC T2 Plus, SPARC T3, and SPARC T4 platforms. The `ldm list-resrc-group` command does not show any information on those platforms and the `-g` variants of the `ldm remove-core` and `ldm remove-memory` commands are not functional.

For examples, see “[Listing Resource Group Information](#)” on page 312.

## Using Power Management

To enable power management (PM), you first need to set the PM policy in at least version 3.0 of the ILOM 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 PM features and ILOM features, see the following:

- [Chapter 19, “Using Power Management”](#)
- “Monitoring Power Consumption” in the *Oracle Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide*
- [Oracle Integrated Lights Out Manager \(ILOM\) 3.0 Feature Updates and Release Notes](#)

## Using Dynamic Resource Management

You can use policies to determine how to automatically perform DR activities. At this time, you can create policies *only* to govern the dynamic resource management of virtual CPUs.



**Caution** - The following restrictions affect CPU dynamic resource management (DRM):

- On UltraSPARC T2 and UltraSPARC T2 Plus platforms, DRM cannot be enabled when the PM elastic policy is set.
- On UltraSPARC T2 and UltraSPARC T2 Plus platforms, any change from the performance policy to the elastic policy is delayed while DRM is enabled.
- Ensure that you disable CPU DRM prior to performing a domain migration operation, or you will see an error message.
- DRM policies do not apply to domains that are configured with the `whole-core` constraint. If you attempt to use DRM on a domain that has the `whole-core` constraint set, you will see an error message.
- When the PM elastic policy is set, you can use DRM only when the firmware supports normalized utilization (8.2.0).

---

A *resource management policy* specifies the conditions under which 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 domain-name...
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 domain-name...
ldm remove-policy [name=]policy-name... domain-name
```

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 a 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** Adding Resource Management Policies  
**13-10**

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.

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.

```
primary# 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.
- Uses the default values for those properties that are not specified, such as `enable` and `sample-rate`. See the [ldm\(1M\)](#) man page.

```
primary# 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 appears 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.

To view syntax usage for all `ldm` subcommands, use the following command:

```
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. The following values can appear in each of the six columns from left to right.

**Column 1 – Starting or stopping domains**

- s – Starting or stopping

**Column 2 – Domain status**

- n – Normal
- t – Transition
- d – Degraded domain that cannot be started due to missing resources

**Column 3 – Reconfiguration status**

- d – Delayed reconfiguration
- r – Memory dynamic reconfiguration

**Column 4 – Control domain**

- c – Control domain

**Column 5 – Service domain**

- v – Virtual I/O service domain

**Column 6 – Migration status**

- s – Source domain in a migration
- t – Target domain in a migration
- e – Error occurred during a migration

## Utilization Statistic Definition

The per virtual CPU utilization statistic (UTIL) is shown through 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. The normalized utilization statistic (NORM) is the percentage of time the virtual CPU spends executing on behalf of the guest OS. This value takes

into account such operations as cycle skip. Normalized virtualization is only available when your system runs at least version 8.2.0 of the system firmware.

When PM does not perform cycle skip operations, 100% utilization equals 100% normalized utilization. When PM adjusts the cycle skip to four eighths, 100% utilization equals 50% utilization, which means that the CPU effectively has only half the possible number of cycles available. So, a fully utilized CPU has a 50% normalized utilization. Use the `ldm list` or `ldm list -l` command to show normalized utilization for both virtual CPUs and the guest OS.

## Viewing Various Lists

- To view the current software versions installed:

```
primary# ldm -V
```

- To generate a short list for all domains:

```
primary# ldm list
```

- To generate a long list for all domains:

```
primary# ldm list -l
```

- To generate an extended list of all domains:

```
primary# ldm list -e
```

- To generate a parseable, machine-readable list of all domains:

```
primary# ldm list -p
```

- You can 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...] domain-name
```

- **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-name** – 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 (vldc) service, virtual logical domain channel client (vldcc), virtual data plane channel client (vdgcc), and 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.

- To list CPU information for the control domain:

```
primary# ldm list -o cpu primary
```

- To list domain information for a guest domain:

```
primary# ldm list -o domain ldm2
```

- To list memory and network information for a guest domain:

```
primary# ldm list -o network,memory ldm1
```

- To list DRM policy information for a guest domain:

```
primary# ldm list -o resmgmt,stats ldm1
```

- To show a variable and its value for a domain:

```
primary# ldm list-variable variable-name domain-name
```

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
```

- To list the resources that are bound to a domain:

```
primary# ldm list-bindings domain-name
```

- To list logical domain configurations that have been stored on the SP:

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 Domain Configurations](#)” on page 317. For more examples, see the `ldm(1M)` man page.

```
primary# ldm list-config
factory-default
3guests
foo [next poweron]
primary
reconfig-primary
```

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 power cycle.
- [degraded] – Configuration is a degraded version of the previously booted configuration.
- To list all server resources, bound and unbound:

```
primary# ldm list-devices -a
```

- To list the amount of memory available to be allocated:

```
primary# ldm list-devices mem
MEMORY
PA SIZE
0x14e000000 2848M
```

- To determine which portions of memory are unavailable for logical domains:

```
primary# ldm list-devices -a mem
MEMORY
PA SIZE BOUND
0x0 57M _sys_
0x3900000 32M _sys_
0x5900000 94M _sys_
0xb700000 393M _sys_
0x24000000 192M _sys_
0x30000000 255G primary
0x3ff0000000 64M _sys_
0x3ff4000000 64M _sys_
0x3ff8000000 128M _sys_
0x80000000000 2G ldg1
0x80080000000 2G ldg2
0x80100000000 2G ldg3
0x80180000000 2G ldg4
0x80200000000 103G primary
0x81bc0000000 145G primary
```

- To 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.

- To list constraints for one domain:

```
ldm list-constraints domain-name
```

- To list constraints in XML format for a particular domain:

```
ldm list-constraints -x domain-name
```

- To list constraints for all domains in a parseable format:

```
ldm list-constraints -p
```

## **Listing Resource Group Information**

You can use the `ldm list-rsrc-group` command to show information about resource groups.

The following command shows information about all the resource groups:

```
primary# ldm list-rsrc-group
NAME CORE MEMORY IO
/SYS/CMU4 12 256G 4
/SYS/CMU5 12 256G 4
/SYS/CMU6 12 128G 4
/SYS/CMU7 12 128G 4
```

Like the other `ldm list-*` commands, you can specify options to show parseable output, detailed output, and information about particular resource groups and domains. For more information, see the [ldm\(1M\)](#) man page.

The following example uses the `-l` option to show detailed information about the `/SYS/CMU5` resource group.

```
primary# ldm list-rsrc-group -l /SYS/CMU5
NAME CORE MEMORY IO
/SYS/CMU5 12 256G 4

CORE
CID BOUND
192, 194, 196, 198, 200, 202, 208, 210 primary
```

```

212, 214, 216, 218 primary

MEMORY

PA SIZE BOUND
0xc0000000000 228M ldg1
0xc0030000000 127G primary
0xc1ffc000000 64M _sys_
0xd0000000000 130816M primary
0xd1ffc000000 64M _sys_

IO

DEVICE PSEUDONYM BOUND
pci@900 pci_24 primary
pci@940 pci_25 primary
pci@980 pci_26 primary
pci@9c0 pci_27 primary

```

## Using Perf-Counter Properties

The performance register access control feature enables you to get, set and unset a domain's access rights to certain groups of performance registers.

Use the `ldm add-domain` and `ldm set-domain` commands to specify a value for the `perf-counters` property. If no `perf-counters` value is specified, the value is `htstrand`. See the [ldm\(1M\)](#) man page.

You can specify the following values for the `perf-counters` property:

|                       |                                                                                                                                                                                                                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>global</code>   | Grants the domain access to the global performance counters that its allocated resources can access. Only one domain at a time can have access to the global performance counters. You can specify this value alone or with either the <code>strand</code> or <code>htstrand</code> value. |
| <code>strand</code>   | Grants the domain access to the strand performance counters that exist on the CPUs that are allocated to the domain. You cannot specify this value and the <code>htstrand</code> value together.                                                                                           |
| <code>htstrand</code> | Behaves the same as the <code>strand</code> value and enables instrumentation of hyperprivilege mode events on the CPUs that are allocated to the domain. You cannot specify this value and the <code>strand</code> value together.                                                        |

To disable all access to any of the performance counters, specify `perf-counters=`.

If the hypervisor does not have the performance access capability, attempting to set the `perf-counters` property fails.

The `ldm list -o domain` and `ldm list -e` commands show the value of the `perf-counters` property. If the performance access capability is not supported, the `perf-counters` value is not shown in the output.

**EXAMPLE 13-11** Creating a Domain and Specifying Its Performance Register Access

Create the new `ldg0` domain with access to the `global` register set:

```
primary# ldm add-domain perf-counters=global ldg0
```

**EXAMPLE 13-12** Specifying the Performance Register Access for a Domain

Specify that the `ldg0` domain has access to the `global` and `strand` register sets:

```
primary# ldm set-domain perf-counters=global,strand ldg0
```

**EXAMPLE 13-13** Specifying that a Domain Does Not Have Access to Any Register Sets

Specify that the `ldg0` domain does not have access to any of the register sets:

```
primary# ldm set-domain perf-counters= ldg0
```

**EXAMPLE 13-14** Viewing Performance Access Information

The following examples show how to view performance access information by using the `ldm list -o domain` command.

- The following `ldm list -o domain` command shows that the `global` and `htstrand` performance values are specified on the `ldg0` domain:

```
primary# ldm list -o domain ldg0
NAME STATE FLAGS UTIL
NORM
ldg0 active -n---- 0.0% 0.0%
SOFTSTATE
Solaris running

UUID
062200af-2de2-e05f-b271-f6200fd3eee3

HOSTID
0x84fb315d
```

```

CONTROL
 failure-policy=ignore
 extended-mapin-space=on
 cpu-arch=native
 rc-add-policy=
 shutdown-group=15
 perf-counters=global,htstrand

DEPENDENCY
 master=

PPRIORITY 4000

VARIABLES
 auto-boot?=false
 boot-device=/virtual-devices@100/channel-devices@200/disk@0:a
 /virtualdevices@100/channel@200/disk@0
 network-boot-arguments=dhcp,hostname=solaris,
 file=http://10.129.241.238:5555/cgibin/wanboot-cgi
 pm_boot_policy=disabled=0;ttfc=2000;ttmr=0;

■ The following ldm list -p -o domain command shows the same information as in the
previous example but in the parseable form:

primary# ldm list -p -o domain ldg0
VERSION 1.12
DOMAIN|name=ldg0|state=active|flags=normal|util=|norm_util=
UUID|uuid=4e8749b9-281b-e2b1-d0e2-ef4dc2ce5ce6
HOSTID|hostid=0x84f97452
CONTROL|failure-policy=reset|extended-mapin-space=on|cpu-arch=native|rc-add-policy=|
shutdown-group=15|perf-counters=global,htstrand
DEPENDENCY|master=
VARIABLES
|auto-boot?=false
|boot-device=/virtual-devices@100/channel-devices@200/disk@0
|pm_boot_policy=disabled=0;ttfc=2500000;ttmr=0;

```



## Managing Domain Configurations

---

This chapter contains information about managing domain configurations.

This chapter covers the following topics:

- “[Managing Domain Configurations](#)” on page 317
- “[Available Configuration Recovery Methods](#)” on page 318

## Managing Domain Configurations

A domain *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.

Saving a configuration on the SP makes it persist across system power cycles. You can save several configurations and specify which configuration to boot on the next power-on attempt.

When you power up a system, the SP boots the selected 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. You can also explicitly request another configuration by using the `ldm set-spconfig` command or the appropriate ILOM command.



**Caution** - Always save your stable configuration to the SP and save it as XML. Saving the configuration in these ways enable you to recover your system configuration after a power failure and save it for later use. See “[Saving Domain Configurations](#)” on page 321.

A local copy of the SP configuration and the Logical Domains constraint database is saved on the control domain whenever you save a configuration to the SP. This local copy is called a *bootset*. The bootset is used to load the corresponding Logical Domains constraints database when the system undergoes a power cycle.

On SPARC T5, SPARC M5, and SPARC M6 systems, the bootsets on the control domain are the master copies of the configurations. On startup, the Logical Domains Manager

automatically synchronizes all configurations with the SP, which ensures that the configurations on the SP are always identical to the bootsets that are stored on the control domain.

---

**Note** - Because the bootsets contain critical system data, ensure that the control domain's file system uses technology such as disk mirroring or RAID to reduce the impact of disk failures.

---

A [physical domain](#) is the scope of resources that are managed by a single Oracle VM Server for SPARC instance. A physical domain might be a complete physical system as is the case of supported SPARC T-Series platforms. Or, it might be either the entire system or a subset of the system as is the case of supported SPARC M-Series platforms.

## Available Configuration Recovery Methods

Oracle VM Server for SPARC supports the following configuration recovery methods:

- The autosave method, used when the configuration is not available on the SP.  
This situation might occur in one of the following circumstances:
  - The hardware that holds the saved configurations has been replaced
  - The configuration is not up to date because you neglected to save the latest configuration changes to the SP or an unexpected power cycle occurred
- The `ldm add-domain` method, used if a subset of the domains need to have their configurations restored
- The `ldm init-system` method, which should be used only as a last resort. Use this method only when both the configuration on the SP and the autosave information from the control domain are lost.

## Restoring Configurations By Using Autosave

A copy of the current configuration is automatically saved on the control domain whenever the domain configuration is changed. This autosave operation does not explicitly save the configuration to the SP.

The autosave operation occurs immediately, even in the following situations:

- When the new configuration is not explicitly saved on the SP
- When the 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 after a system power cycle. 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, and ASR 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 that you must perform a power cycle after using this command to complete the manual recovery.

---

**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 Oracle VM Server for SPARC With the Service Processor](#)” on page 338. You can also use the `ldm set-spconfig` command, which is described on the [ldm\(1M\)](#) man page.

## 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. This 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. You 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. You 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 power cycle. To make this configuration available, you must perform another power cycle. This configuration is updated with the newer configuration that is saved on the control domain. This action does not affect the currently running configuration. It affects only the configuration that will be used during the next power cycle. A message is also logged that states that a newer

configuration has been saved on the SP and that it will be booted at the next system power cycle. These messages are logged in the `ldmd` SMF log file.

## ▼ How to Modify the Autorecovery Policy

1. **Log in to the control domain.**
2. **Become an administrator.**
  - For Oracle Solaris 10, see “Configuring RBAC (Task Map)” in “System Administration Guide: Security Services”.
  - For Oracle Solaris 11.2, see Chapter 1, “About Using Rights to Control Users and Processes,” in “Securing Users and Processes in Oracle Solaris 11.2”.
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 14-1 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 ldm
svcadm enable ldm
```

## Saving Domain Configurations

You can save a domain configuration for a single domain or for all the domains on a system.

With the exception of the named physical resources, the following method does not preserve actual bindings. However, the method does preserve the constraints used to create those bindings. After saving and restoring the configuration, the domains have the same virtual resources but are not necessarily bound to the same physical resources. Named physical resources are bound as specified by the administrator.

- To save the configuration for a single domain, create an XML file containing the domain's constraints.

```
ldm list-constraints -x domain-name >domain-name.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
```

## Restoring Domain Configurations

This section describes how to restore a domain configuration from an XML file for guest domains and for the control (primary) domain.

- To restore a domain configuration for guest domains, you use the `ldm add-domain -i` command, as described in [“How to Restore a Domain Configuration From an XML File \(`ldm add-domain`\)”](#) on page 322. Although you can save the primary domain's constraints to an XML file, you cannot use the file as input to this command.
- To restore a domain configuration for the primary domain, you 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 might be left inactive when the configuration is complete. See “[How to Restore a Domain Configuration From an XML File \(`ldm init-system`\)](#)” on page 322.

## ▼ How to Restore a Domain Configuration From an XML File (`ldm add-domain`)

This procedure works for guest domains but not for the control (primary) domain. If you want to restore the configuration for the primary domain, or for other domains that are described in the XML file, see “[How to Restore a Domain Configuration From an XML File \(`ldm init-system`\)](#)” on page 322.

1. **Create the domain by using the XML file that you created as input.**

```
ldm add-domain -i domain-name.xml
```

2. **Bind the domain.**

```
ldm bind-domain [-fq] domain-name
```

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 domain-name
```

**Example 14-2** 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
```

## ▼ How to 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.



**Caution** - The `ldm init-system` command might not correctly restore a configuration in which physical I/O commands have been used. Such commands are `ldm add-io`, `ldm set-io`, `ldm remove-io`, `ldm create-vf`, and `ldm destroy-vf`. For more information, see “[“`ldm init-system` Command Might Not Correctly Restore a Domain Configuration on Which Physical I/O Changes Have Been Made”](#) in “[Oracle VM Server for SPARC 3.2 Release Notes](#)”.

**Before You Begin** You should have created an XML configuration file by running the `ldm list-constraints -x` command. The file should describe one or more domain configurations.

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 “[How to Restore the Factory Default Configuration](#)” in “[Oracle VM Server for SPARC 3.2 Installation Guide](#)”.

3. **Become an administrator.**
  - [For Oracle Solaris 10, see “Configuring RBAC \(Task Map\)” in “System Administration Guide: Security Services”.](#)
  - [For Oracle Solaris 11.2, see Chapter 1, “About Using Rights to Control Users and Processes,” in “Securing Users and Processes in Oracle Solaris 11.2”.](#)
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 regardless 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.

The primary domain is reconfigured as specified in the file. Any non-primary domains that have configurations in the XML file are reconfigured but left inactive.

**Example 14-3** 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
```

## Addressing Service Processor Connection Problems

If an attempt to use the `ldm` command to manage domain configurations fails because of an error communicating with the SP, perform the following recovery step that applies to your platform:

- **SPARC T3, SPARC T4, SPARC T5, SPARC M5, and SPARC M6 systems.** Restart the `ldmd` service.

```
primary# svcadm enable ldmd
```

If restarting the `ldmd` daemon fails to restore communications, restart the SP.

## Handling Hardware Errors

---

This chapter contains information about how Oracle VM Server for SPARC handles hardware errors.

This chapter covers the following topics:

- “Hardware Error-Handling Overview” on page 325
- “Using FMA to Blacklist or Unconfigure Faulty Resources” on page 325
- “Recovering Domains After Detecting Faulty or Missing Resources” on page 327
- “Marking Domains as Degraded” on page 330
- “Marking I/O Resources as Evacuated” on page 331

### Hardware Error-Handling Overview

The Oracle VM Server for SPARC software adds the following RAS capabilities for the SPARC enterprise-class platforms starting with SPARC T5 and SPARC M5:

- **Fault Management Architecture (FMA) blacklisting.** When FMA detects faulty CPU or memory resources, Oracle VM Server for SPARC places them on a blacklist. A faulty resource that is on the blacklist cannot be reassigned to any domains until FMA marks it as being repaired.
- **Recovery mode.** Automatically recover domain configurations that cannot be booted because of faulty or missing resources.

Though the Fujitsu M10 platform does not support this blacklisting of faulty resources, the Fujitsu M10 platform auto-replacement feature provides similar functionality.

### Using FMA to Blacklist or Unconfigure Faulty Resources

FMA contacts the Logical Domains Manager when it detects a faulty resource. Then, the Logical Domains Manager attempts to stop using that resource in all running domains. To

ensure that a faulty resource cannot be assigned to a domain in the future, FMA adds the resource to a blacklist.

The Logical Domains Manager supports blacklisting only for CPU and memory resources, not for I/O resources.

If a faulty resource is not in use, the Logical Domains Manager removes it from the available resource list, which you can see in the `ldm list-devices` output. At this time, this resource is internally marked as “blacklisted” so that it cannot be re-assigned to a domain in the future.

If the faulty resource is in use, the Logical Domains Manager attempts to evacuate the resource. To avoid a service interruption on the running domains, the Logical Domains Manager first attempts to use CPU or memory dynamic reconfiguration to evacuate the faulty resource. The Logical Domains Manager remaps a faulted core if a core is free to use as a target. If this “live evacuation” succeeds, the faulty resource is internally marked as blacklisted and is not shown in the `ldm list-devices` output so that it will not be assigned to a domain in the future.

If the live evacuation fails, the Logical Domains Manager internally marks the faulty resource as “evacuation pending.” The resource is shown as normal in the `ldm list-devices` output because the resource is still in use on the running domains until the affected guest domains are rebooted or stopped.

When the affected guest domain is stopped or rebooted, the Logical Domains Manager attempts to evacuate the faulty resources and internally mark them as blacklisted so that the resource cannot be assigned in the future. Such a device is not shown in the `ldm` output. After the pending evacuation completes, the Logical Domains Manager attempts to start the guest domain. However, if the guest domain cannot be started because sufficient resources are not available, the guest domain is marked as “degraded” and the following warning message is logged for the user intervention to perform the manual recovery.

```
primary# ldm ls
NAME STATE FLAGS CONS VCPU MEMORY UTIL NORM UPTIME
primary active -n-cv- UART 368 2079488M 0.1% 0.0% 16h 57m
gd0 bound -d---- 5000 8

warning: Could not restart domain gd0 after completing pending evacuation.
The domain has been marked degraded and should be examined to see
if manual recovery is possible.
```

When the system is power-cycled, FMA repeats the evacuation requests for resources that are still faulty and the Logical Domains Manager handles those requests by evacuating the faulty resources and internally marking them as blacklisted.

Prior to support for FMA blacklisting, a guest domain that panicked because of a faulty resource might result in a never-ending panic-reboot loop. By using resource evacuation and blacklisting when the guest domain is rebooted, you can avoid this panic-reboot loop and prevent future attempts to use a faulty resource.

## Recovering Domains After Detecting Faulty or Missing Resources

If a SPARC T5, SPARC M5, SPARC M6, or Fujitsu M10 system detects a faulty or missing resource at power on, the Logical Domains Manager attempts to recover the configured domains by using the remaining available resources. While the recovery takes place, the system (or physical domain on SPARC M-Series) is said to be in *recovery mode*. A recovery is attempted only if recovery mode is enabled. See “[Enabling Recovery Mode](#)” on page 330.

At power on, the system firmware reverts to the factory-default configuration if the last selected power-on configuration cannot be booted in any of the following circumstances:

- The I/O topology within each PCIe switch in the configuration does not match the I/O topology of the last selected power-on configuration
- CPU resources or memory resources of the last selected power-on configuration are no longer present in the system

If recovery mode is enabled, the Logical Domains Manager recovers all active and bound domains from the last selected power-on configuration. The resulting running configuration is called the *degraded configuration*. The degraded configuration is saved to the SP and remains the active configuration until either a new SP configuration is saved or the physical domain is power-cycled.

---

**Note** - The physical domain does not require a power cycle to activate the degraded configuration after recovery as it is already the running configuration.

---

If the physical domain is power-cycled, the system firmware first attempts to boot the last original power-on configuration. That way, if the missing or faulty hardware was replaced in the meantime, the system can boot the original normal configuration. If the last selected power-on configuration is not bootable, the firmware next attempts to boot the associated degraded configuration if it exists. If the degraded configuration is not bootable or does not exist, the factory-default configuration is booted and recovery mode is invoked.

The recovery operation works in the following order:

- **Control domain.** The Logical Domains Manager recovers the control domain by restoring its CPU, memory, and I/O configuration as well as its virtual I/O services.  
If the amount of CPU or memory required for all recoverable domains is larger than the remaining available amounts, the number of CPUs or cores or memory is reduced in proportion to the size of the other domains. For example, in a four-domain system where each domain has 25% of the CPUs and memory assigned, the resulting degraded configuration still assigns 25% of the CPUs and memory to each domain. If the primary domain originally had up to two cores (16 virtual CPUs) and eight Gbytes of memory, the control domain size is not reduced.

Root complexes and PCIe devices that are assigned to other domains are removed from the control domain. The virtual functions on root complexes that are owned by the control

domain are re-created. Any missing root complex, PCIe device, physical function, or virtual function that is assigned to the control domain is marked as evacuated. The Logical Domains Manager then reboots the control domain to make the changes active.

- **Root domains.** After the control domain has been rebooted, the Logical Domains Manager recovers the root domains. The amount of CPU and memory is reduced in proportion to the other recoverable domains, if needed. If a root complex is no longer physically present in the system, it is marked as evacuated. This root complex is not configured into the domain during the recovery operation. A root domain is recovered as long as at least one of the root complexes that is assigned to the root domain is available. If none of its root complexes are available, the root domain is not recovered. The Logical Domains Manager boots the root domain and re-creates the virtual functions on the physical functions that are owned by the root domain. Any missing PCIe slots, physical functions, and virtual functions are marked as evacuated. Any virtual I/O services that are provided by the domain are re-created, if possible.

---

**Note** - Configurations in which a non-primary root domain loans out PCIe slots cannot be recovered at this time. So, you must manually move those slots to an I/O domain after the recovery completes.

- **I/O domains.** Logical Domains Manager recovers any I/O domains. Any PCIe slots and virtual functions that are missing from the system are marked as evacuated. If none of the required I/O devices are present, the domain is not recovered and its CPU and memory resources are available for use by other domains. Any virtual I/O services that are provided by the domain are re-created, if possible.
- **Guest domains.** A guest domain is recovered *only* if at least one of the service domains that serves the domain has been recovered. If the guest domain cannot be recovered, its CPU and memory resources are available for use by other guest domains.

When possible, the same number of CPUs and amount of memory are allocated to a domain as specified by the original configuration. If that number of CPUs or amount of memory are not available, these resources are reduced proportionally to consume the remaining available resources.

---

**Note** - When a system is in recovery mode, you can only perform `ldm list-*` commands. All other `ldm` commands are disabled until the recovery operation completes.

The Logical Domains Manager only attempts to recover bound and active domains. The existing resource configuration of any unbound domain is copied to the new configuration as-is.

During a recovery operation, fewer resources might be available than in the previously booted configuration. As a result, the Logical Domains Manager might only be able to recover some of the previously configured domains. Also, a recovered domain might not include all of the resources from its original configuration. For example, a recovered bound domain might have fewer I/O resources than it had in its previous configuration. A domain might not be recovered if its I/O devices are no longer present or if its parent service domain could not be recovered.

Recovery mode records its steps to the Logical Domains Manager SMF log, `/var/svc/log/ldoms-ldmd:default.log`. A message is written to the system console when Logical Domains Manager starts a recovery, reboots the control domain, and when the recovery completes.



**Caution** - A recovered domain is not guaranteed to be completely operable. The domain might not include a resource that is essential to run the OS instance or an application. For example, a recovered domain might only have a network resource and no disk resource. Or, a recovered domain might be missing a file system that is required to run an application. Using multipathed I/O for a domain reduces the impact of missing I/O resources.

## Recovery Mode Hardware and Software Requirements

- **Hardware Requirements** – The recovery mode feature is supported on the SPARC T5, SPARC M5, SPARC M6, and Fujitsu M10 platforms.
- **Firmware Requirements** – At least version 9.1.0.a of the system firmware for the SPARC T5, SPARC M5, and SPARC M6 systems. At least version XCP2230 of the system firmware for Fujitsu M10 servers
- **Software Requirements** – All domains must be running at least the Oracle Solaris 11.1.10.5.0 OS or the Oracle Solaris 10 1/13 OS plus the required patches in “[Fully Qualified Oracle Solaris OS Versions](#)” in “[Oracle VM Server for SPARC 3.2 Installation Guide](#)”.

## Degraded Configuration

Each physical domain can have only one degraded configuration saved to the SP. If a degraded configuration already exists, it is replaced by the newly created degraded configuration.

You cannot interact directly with degraded configurations. The system firmware transparently boots the degraded version of the next power-on configuration, if necessary. This transparency enables the system to boot the original configuration after a power cycle when the missing resources reappear. When the active configuration is a degraded configuration, it is marked as [degraded] in the `ldm list-spconfig` output.

The autosave functionality is disabled while the active configuration is a degraded configuration. If you save a new configuration to the SP while a degraded configuration is active, the new configuration is a normal non-degraded configuration.

---

**Note** - A previously missing resource that reappears on a subsequent power cycle has no effect on the contents of a normal configuration. However, if you subsequently select the configuration that triggered recovery mode, the SP boots the original, non-degraded configuration now that all its hardware is available.

---

## Enabling Recovery Mode

The `ldmd/recovery_mode` SMF property controls recovery mode behavior.

To configure the Logical Domains Manager to automatically begin the recovery process when the system enters recovery mode, you must first enable recovery mode. To enable recovery mode, set the `ldmd/recovery_mode` property value to `auto` and refresh the `ldmd` SMF service.

```
primary# svccfg -s ldmd setprop ldmd/recovery_mode = astring: auto
primary# svcadm refresh ldmd
primary# svcadm restart ldmd
```

By default, the `ldmd/recovery_mode` property is not present. When this property is not present or is set to `never`, the Logical Domains Manager exits recovery mode without taking any action and the physical domain runs the factory-default configuration.

---

**Note** - If the system firmware requests recovery mode while it is not enabled, issue the following commands to enable recovery mode after the request is made:

```
primary# svccfg -s ldmd setprop ldmd/recovery_mode = astring: auto
primary# svcadm refresh ldmd
primary# svcadm restart ldmd
```

Recovery mode is initiated immediately in this scenario only if no changes were made to the system, that is, if it is still in the factory-default configuration.

---

## Marking Domains as Degraded

A domain is marked as degraded if the FMA blacklisting of a resource leaves a domain with insufficient resources to start. The domain then remains in the bound state, which prevents the remaining resources that are assigned to the domain from being reallocated to other domains.

## Marking I/O Resources as Evacuated

An I/O resource that is detected as missing by recovery mode is marked as evacuated by showing an asterisk (\*) in `lsm` list output.



## Performing Other Administration Tasks

---

This chapter contains information about using the Oracle VM Server for SPARC software and tasks that are not described in the preceding chapters.

This chapter covers the following topics:

- “Entering Names in the CLI” on page 333
- “Updating Property Values in the /etc/system File” on page 334
- “Connecting to a Guest Console Over the Network” on page 335
- “Using Console Groups” on page 335
- “Stopping a Heavily Loaded Domain Can Time Out” on page 336
- “Operating the Oracle Solaris OS With Oracle VM Server for SPARC” on page 337
- “Using Oracle VM Server for SPARC With the Service Processor” on page 338
- “Configuring Domain Dependencies” on page 339
- “Determining Where Errors Occur by Mapping CPU and Memory Addresses” on page 343
- “Using Universally Unique Identifiers” on page 345
- “Virtual Domain Information Command and API” on page 346
- “Using Logical Domain Channels” on page 347
- “Booting a Large Number of Domains” on page 349
- “Cleanly Shutting Down and Power Cycling an Oracle VM Server for SPARC System” on page 350
- “Logical Domains Variable Persistence” on page 351
- “Adjusting the Interrupt Limit” on page 352
- “Listing Domain I/O Dependencies” on page 354

### 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 (*domain-name*), 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 - \_ +#:.:;~().

## Updating Property Values in the /etc/system File

Do not make manual changes to the /etc/system file. This file is automatically generated upon reboot when files in the /etc/system.d directory specify tuning property values.

### ▼ How to Add or Modify a Tuning Property Value

#### 1. Search for the tuning property name in the existing /etc/system file and in the /etc/system.d files.

For example, to specify a value for the vds:vd\_volume\_force\_slice property, determine whether the property is already set.

```
grep 'vds:vd_volume_force_slice' /etc/system /etc/system.d/*
```

#### 2. Update the property value:

- If the property is found in one of the /etc/system.d files, update the property value in the existing file.
- If the property is found in the /etc/system file or it is not found, create a file in the /etc/system.d directory with a name such as the following example:

```
/etc/system.d/com.company-name:ldoms-config
```

## Connecting to a Guest Console Over the 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 hostname 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](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`, 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.

### ▼ How to 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 reassign 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
```

The `ldm stop` command uses the `shutdown` command to stop a domain. The execution of the shutdown sequence usually takes much longer than a quick stop, which can be performed by running the `ldm stop -q` command. See the [ldm\(1M\)](#) man page.

A long shutdown sequence might generate the following timeout message:

*domain-name* stop timed out. The domain might still be in the process of shutting down.  
Either let it continue, or specify -f to force it to stop.

While this shutdown sequence runs, the s flag is also shown for the domain.

## 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 the 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 by using the Oracle Solaris OS halt command.

### Performing a Power Cycle of a Server

Whenever performing any maintenance on a system running Oracle VM Server for SPARC software that requires you to perform a power cycle of the server, you must save your current logical domain configurations to the SP first.

To save your current domain configurations to the SP, use the following command:

```
ldm add-config config-name
```

### Result of Oracle Solaris OS Breaks

You can initiate Oracle Solaris OS breaks as follows:

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.

When you initiate such a break, the Oracle Solaris OS issues the following prompt:

`c)ontinue, s)ync, r)eboot, h)alt?`

Type the letter that represents what you want the system to do after these types of breaks.

## Results From Rebooting the Control Domain

The following table shows the expected behavior of rebooting the control (**primary**) domain.

**TABLE 16-1** Expected Behavior of Rebooting the Control Domain

| Command                    | Other Domain Configured? | Behavior                                                                        |
|----------------------------|--------------------------|---------------------------------------------------------------------------------|
| <code>reboot</code>        | Not configured           | Reboots the control domain without graceful shutdown, no power off.             |
|                            | Configured               | Reboots the control domain without graceful shutdown, no power off.             |
| <code>shutdown -i 5</code> | Not configured           | Host powered off after graceful shutdown, stays off until powered on at the SP. |
|                            | Configured               | Reboots with graceful shutdown, no power off.                                   |

For information about the consequences of rebooting a domain that has the root domain role, see “[Rebooting the Root Domain With PCIe Endpoints Configured](#)” on page 78.

## Using Oracle VM Server for SPARC With the Service Processor

The section describes information related to 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 at <http://www.oracle.com/technetwork/documentation/sparc-tseries-servers-252697.html>.

An additional `config` option is available to the existing ILOM command:

```
-> set /HOST/bootmode config=config-name
```

This 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 regardless of whether the host is powered on or off. It takes effect on the next host reset or power on.

To reset the logical domain configuration, you set the option to `factory-default`.

```
-> 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 perform a power cycle of 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=alpha,beta,gamma,delta pine
```

The `alpha`, `beta`, `gamma`, and `delta` master domains all specify a failure policy of `stop`.

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, each master domain must have the same failure policy. So, the first master domain to fail triggers its defined failure policy on all of its slave domains.

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
- `panic` panics any slave domains (similar to running the `ldm panic` command)

- reset immediately stops and then restarts any slave domains (similar to running the `ldm stop -f` command and then the `ldm start` command)
- stop stops any slave domains (similar to running the `ldm stop -f` command)

In this example, the master domains specify their failure policy as follows:

```
primary# ldm set-domain failure-policy=ignore apple
primary# ldm set-domain failure-policy=panic lemon
primary# ldm set-domain failure-policy=reset orange
primary# ldm set-domain failure-policy=stop peach
primary# ldm set-domain failure-policy=stop alpha
primary# ldm set-domain failure-policy=stop beta
primary# ldm set-domain failure-policy=stop gamma
primary# ldm set-domain failure-policy=stop delta
```

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 342.

---

For domain dependency XML examples, see [Example 22-6](#).

## Domain Dependency Examples

The following examples show how to configure domain dependencies.

### EXAMPLE 16-1 Configuring a Failure Policy by Using 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=reset` to specify that slave domains reset 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.

```
primary# ldm add-domain failure-policy=reset twizzle
primary# ldm set-domain failure-policy=reset primary
```

```
primary# ldm add-domain master=twizzle,primary chocklaw
```

**EXAMPLE 16-2** Modifying a Domain to Assign a Master Domain

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.

```
primary# ldm set-domain master=primary orange
primary# ldm set-domain master=orange,primary tangerine
primary# 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=ignore

DEPENDENCY
master=primary

NAME STATE FLAGS UTIL
tangerine bound -----

HOSTID
0x84f948e9

CONTROL
failure-policy=ignore

DEPENDENCY
master=orange,primary
```

**EXAMPLE 16-3** Showing a Parseable Domain Listing

The following shows an example listing with parseable output:

```
primary# ldm list -o domain -p
```

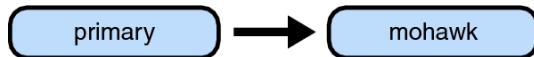
## Dependency Cycles

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 dependency chain shown in the following diagram.

**FIGURE 16-1** Single Domain Dependency

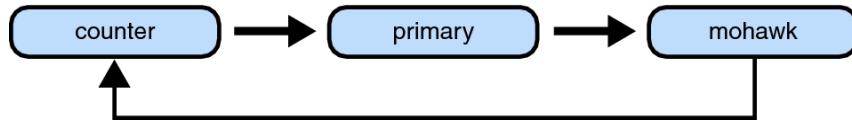


The second command creates a slave domain called `primary` that specifies its master domain as `counter`. So, `mohawk` depends on `primary`, which depends on `counter` in the dependency chain shown in the following diagram.

**FIGURE 16-2** Multiple Domain Dependency



The third command attempts to create a dependency between the `counter` and `mohawk` domains, which would produce the dependency cycle shown in the following diagram.

**FIGURE 16-3** 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

You can find the domain and the virtual CPU number within the domain that correspond to a given physical CPU number.

First, generate a long parseable list for all domains by using the following command:

```
primary# ldm list -l -p
```

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

You can find the domain and the real memory address within the domain that correspond to a given physical memory address (PA).

First, generate a long parseable list for all domains.

```
primary# ldm list -l -p
```

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,  $pa \leq PA \leq (pa + size - 1)$ . *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.

## Example of CPU and Memory Mapping

### EXAMPLE 16-4 Determining the Configuration of Domains

The following command produces a long parseable list of logical domains configurations.

```
primary# ldm list -l -p
VERSION 1.6
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=0x8000000|pa=0x8000000|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=0x8000000|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=0x8000000|pa=0x78000000|size=1073741824
...

```

**EXAMPLE 16-5** Determining the Virtual CPU That Corresponds to a Physical CPU Number

The logical domain configuration is shown in [Example 16-4](#). This example describes how 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=0x8000000|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  $0x8000000 + (0x7e816000 - 0x78000000) = 0xe816000$ .

## Using Universally Unique Identifiers

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 1.6
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.

## Using Logical Domain Channels

Oracle VM Server for SPARC uses logical domain channels (LDCs) to implement all communications such as console, virtual I/O, and control traffic. An LDC is the method used to enable communications between two endpoints. Although typically each endpoint is in a different domain, the endpoints can be in the same domain to enable loopback communications.

The Oracle VM Server for SPARC 3.2 software and system firmware provide a large pool of LDC endpoints that you can use for the control domain and guest domains. This LDC endpoint pool is only available for the SPARC T4, SPARC T5, SPARC M5, SPARC M6, and Fujitsu M10 platforms. The number of LDCs in the pool is based on the platform type as follows:

- **SPARC T4** – 1984 LDC endpoints per guest domain, 98304 LDC endpoints total
- **SPARC T5** – 1984 LDC endpoints per guest domain, 98304 LDC endpoints total
- **SPARC M5** – 1984 LDC endpoints per guest domain, 98304 LDC endpoints per physical domain
- **SPARC M6** – 1984 LDC endpoints per guest domain, 98304 LDC endpoints per physical domain
- **Fujitsu M10** – 1984 LDC endpoints per guest domain, 196608 LDC endpoints total

The required system firmware to support the LDC endpoint pool is 8.5.2 for SPARC T4 and 9.2.1 for SPARC T5, SPARC M5, and SPARC M6, and XCP2240 for Fujitsu M10.

The following LDC endpoint limits still apply if you run an older version of the system firmware on a supported platform or on an UltraSPARC T2, UltraSPARC T2 Plus, or SPARC T3 platform:

- **UltraSPARC T2 system** – 512 LDC endpoints
- **UltraSPARC T2 Plus, SPARC T3, SPARC T4, SPARC T5, SPARC M5, SPARC M6, and Fujitsu M10 platforms** – 768 LDC endpoints

This limitation might be an issue on the control domain because of the potentially large number of LDC endpoints that are used for both virtual I/O data communications and the Logical Domains Manager control of the other domains.

If you attempt to add a service or bind a domain so that the number of LDC endpoints exceeds the limit on any single domain, the operation fails with an error message similar to the following:

```
13 additional LDCs are required on guest primary to meet this request,
but only 9 LDCs are available
```

The following guidelines enable you to plan properly for using LDC endpoints and explain why you might experience an overflow of the LDC capabilities of the control domain:

- The control domain uses approximately 15 LDC endpoints for various communication purposes with the hypervisor, Fault Management Architecture (FMA), and the system

processor (SP), independent of the number of other domains configured. The number of LDC endpoints used by the control domain depends on the platform and on the version of the software that is used.

- The Logical Domains Manager allocates an LDC endpoint to the control domain for every domain, including itself, for control traffic.
- Each virtual I/O service on the control domain uses one LDC endpoint for every connected client of that service. Each domain needs at least a virtual network, a virtual disk, and a virtual console.

The following equation incorporates these guidelines to determine the number of LDC endpoints that are required by the control domain:

$$\begin{aligned} & 15 + \text{number-of-domains} + (\text{number-of-domains} \times \text{number-of-virtual-services}) \\ & = \text{total-LDC-endpoints} \end{aligned}$$

*number-of-domains* is the total number of domains including the control domain and *number-of-virtual-services* is the total number of virtual I/O devices that are serviced by this domain.

The following example shows how to use the equation to determine the number of LDC endpoints when there is a control domain and eight additional domains:

$$15 + 9 + (8 \times 3) = 48 \text{ LDC endpoints}$$

The following example has 45 guest domains and each domain includes five virtual disks, two virtual networks, and a virtual console. The calculation yields the following result:

$$15 + 46 + 45 \times 8 = 421 \text{ LDC endpoints}$$

Depending upon the number of supported LDC endpoints of your platform, the Logical Domains Manager will either accept or reject the configuration.

If you run out of LDC endpoints on the control domain, consider creating service domains or I/O domains to provide virtual I/O services to the guest domains. This action enables the LDC endpoints to be created on the I/O domains and the service domains instead of on the control domain.

A guest domain can also run out of LDC endpoints. This situation might be caused by the `inter-vnet-link` property being set to `on`, which assigns additional LDC endpoints to guest domains to connect directly to each other.

The following equation determines the number of LDC endpoints that are required by a guest domain when `inter-vnet-link=off`:

$$2 + \text{number-of-vnets} + \text{number-of-vdisks} = \text{total-LDC-endpoints}$$

2 represents the virtual console and control traffic, *number-of-vnets* is the total number of virtual network devices assigned to the guest domain, and *number-of-vdisks* is the total number of virtual disks assigned to the guest domain.

The following example shows how to use the equation to determine the number of LDC endpoints per guest domain when `inter-vnet-link=off` and you have two virtual disks and two virtual networks:

$$2 + 2 + 2 = 6 \text{ LDC endpoints}$$

The following equation determines the number of LDC endpoints that are required by a guest domain when `inter-vnet-link=on`:

$$2 + [(number-of-vnets-from-vswX \times number-of-vnets-in-vswX)] \dots + number-of-vdisks = total-LDC-endpoints$$

2 represents the virtual console and control traffic, `number-of-vnets-from-vswX` is the total number of virtual network devices assigned to the guest domain from the `vswX` virtual switch, `number-of-vnets-in-vswX` is the total number of virtual network devices on the `vswX` virtual switch, and `number-of-virtual-disks` is the total number of virtual disks assigned to the guest domain.

The following example shows how to use the equation to determine the number of LDC endpoints per guest domain when `inter-vnet-link=on` and you have two virtual disks and two virtual switches. The first virtual switch has eight virtual networks and assigns four of them to the domain. The second virtual switch assigns all eight of its virtual networks to the domain:

$$2 + (4 \times 8) + (8 \times 8) + 2 = 100 \text{ LDC endpoints}$$

To resolve the issue of running out of LDC endpoints on a guest domain, consider using the `ldm add-vsw` or `ldm set-vsw` command to set the `inter-vnet-link` property to `off`. This action reduces the number of LDC endpoints in the domain or domains that have the virtual network devices. However, the `off` property value does not affect the service domain that has the virtual switch because the service domain still requires an LDC connection to each virtual network device. When this property is set to `off`, LDC channels are not used for inter-vnet communications. Instead, an LDC channel is assigned only for communication between virtual network devices and virtual switch devices. See the [ldm\(1M\)](#) man page.

---

**Note** - Although disabling the assignment of inter-vnet links reduces the number of LDC endpoints, it might negatively affect guest-to-guest network performance. This degradation would occur because all guest-to-guest communications traffic goes through the virtual switch rather than directly from one guest domain to another guest domain.

---

## Booting a Large Number of Domains

You can boot the following number of domains depending on your platform:

- Up to 256 on Fujitsu M10 servers per physical partition

- Up to 128 on SPARC M6 systems per physical domain
- Up to 128 on SPARC M5 systems per physical domain
- Up to 128 on SPARC T5 systems
- Up to 128 on SPARC T4 servers
- Up to 128 on SPARC T3 servers
- Up to 128 on UltraSPARC T2 Plus servers
- Up to 64 on UltraSPARC T2 servers

If unallocated virtual CPUs are available, assign them to the service domain to help process the virtual I/O requests. Allocate 4 to 8 virtual CPUs to the service domain when creating more than 32 domains. In cases where maximum domain configurations have only a single CPU in the service domain, do not put unnecessary stress on the single CPU when configuring and using the domain. The virtual switch (vsw) services should be spread across all the network adapters available in the machine. For example, if booting 128 domains on a Sun SPARC Enterprise T5240 server, create 4 vsw services, each serving 32 virtual net (vnet) instances. Assigning more than 32 vnet instances per vsw service could cause hard hangs in the service domain.

To run the maximum configurations, a machine needs an adequate amount of memory to support the guest domains. The amount of memory is dependent on your platform and your OS. See the documentation for your platform, “[Oracle Solaris 10 8/11 Installation Guide: Planning for Installation and Upgrade](#)”, “[Installing Oracle Solaris 11 Systems](#)”, and “[Installing Oracle Solaris 11.1 Systems](#)”.

Memory and swap space usage increases in a guest domain when the vsw services used by the domain provide services to many virtual networks in multiple domains. This increase is due to the peer-to-peer links between all the vnet instances connected to the vsw. The service domain benefits from having extra memory. The recommended minimum is four Gbytes when running more than 64 domains. Start domains in groups of 10 or fewer and wait for them to boot before starting the next batch. The same advice applies to installing operating systems on domains. You can reduce the number of links by disabling inter-vnet links. See [Inter-Vnet LDC Channels](#).

## Cleanly Shutting Down and Power Cycling an Oracle VM Server for SPARC System

If you have made any configuration changes since last saving a configuration to the SC, before you attempt to power off or power cycle an Oracle VM Server for SPARC system, make sure that you save the latest configuration that you want to keep.

## ▼ How to Power Off a System With Multiple Active Domains

1. Shut down, stop, and unbind all the non-I/O domains.
2. Shut down, stop, and unbind any active I/O domains.
3. Change the domain into init state 5.

```
primary# shutdown -i 5
```

Instead of using the `shutdown` command, you can also use the `init 5` command.

## ▼ How to Power Cycle the System

1. Power off the system.

See “[How to Power Off a System With Multiple Active Domains](#)” on page 351.

2. Use the SP to power on the system.

## Logical Domains Variable Persistence

Variable updates persist across a reboot but not across a power cycle unless the variable updates are either initiated from OpenBoot firmware on the control domain or followed by saving the configuration to the SC.

Note the following conditions:

- When the control domain reboots, if there are no bound guest domains and no delayed reconfiguration in progress, the SC performs a power cycle of the system.
- When the control domain reboots, if guest domains are bound or active (or the control domain is in the middle of a delayed reconfiguration), the SC does not perform a power cycle of the system.

Logical Domains variables for a domain can be specified using any of the following methods:

- At the OpenBoot prompt.
- Using the Oracle Solaris OS `eeprom(1M)` command.
- Using the Logical Domains Manager CLI (`ldm`).
- In a limited fashion, from the system controller (SC) using the `bootmode` command. This method can be used for only certain variables, and only when in the `factory-default` configuration.

Variable updates that are made by using any of these methods should always persist across reboots of the domain. The variable updates also always apply to any subsequent domain configurations that were saved to the SC.

In Oracle VM Server for SPARC 3.2 software, variable updates do not persist as expected in a few cases:

- All methods of updating a variable persist across reboots of that domain. However, they do not persist across a power cycle of the system unless a subsequent logical domain configuration is saved to the SC.

However in the control domain, updates made using either OpenBoot firmware commands or the `eeprom` command *do* persist across a power cycle of the system even without subsequently saving a new logical domain configuration to the SC. The `eeprom` command supports this behavior on SPARC T5, SPARC M5, and SPARC M6 systems, and on SPARC T3 and SPARC T4 systems that run at least version 8.2.1 of the system firmware.

If you are concerned about Logical Domains variable changes, do one of the following:

- Bring the system to the `ok` prompt and update the variables.
- Update the variables while the Logical Domains Manager is disabled:

```
svcadm disable ldmd
update variables
svcadm enable ldmd
```

- When running Live Upgrade, perform the following steps:

```
svcadm disable -t ldmd
luactivate be3
init 6
```

If you modify the time or date on a logical domain, for example, using the `ntpdate` command, the change persists across reboots of the domain but not across a power cycle of the host. To ensure that time changes persist, save the configuration with the time change to the SP and boot from that configuration.

The following Bug IDs have been filed to resolve these issues: 15375997, 15387338, 15387606, and 15415199.

## Adjusting the Interrupt Limit

Hardware provides a finite number of interrupts, so Oracle Solaris limits the number of interrupts that each device can use. The default limit should match the needs of a typical system configuration but you might need to adjust this value for certain system configurations.

When you enable I/O virtualization on a PCIe bus, interrupt hardware resources are assigned to each I/O domain. Each domain is allotted a finite number of those resources, which might lead

to some interrupt allocation issues. This situation affects only the UltraSPARC T2, UltraSPARC T2 Plus, SPARC T3, SPARC T4, SPARC T5, SPARC M5, and SPARC M6 platforms.

The following warning on the Oracle Solaris console means the interrupt supply was exhausted while attaching I/O device drivers:

```
WARNING: ddi_intr_alloc: cannot fit into interrupt pool
```

Specifically, the limit might need adjustment if the system is partitioned into multiple logical domains and if too many I/O devices are assigned to any guest domain. Oracle VM Server for SPARC divides the total number of interrupts into smaller sets and assigns them to guest domains. If too many I/O devices are assigned to a guest domain, its interrupt supply might be too small to provide each device with the default limit of interrupts. Thus, the guest domain exhausts its interrupt supply before it completely attaches all the drivers.

Some drivers provide an optional callback routine that permits the Oracle Solaris OS to automatically adjust their interrupts. The default limit does not apply to these drivers.

Use the `::irm pools` and `::irmreqs` MDB macros to determine how interrupts are used. The `::irm pools` macro shows the overall interrupt supply divided into pools. The `::irmreqs` macro shows which devices are mapped to each pool. For each device, `::irmreqs` shows whether the default limit is enforced by an optional callback routine, how many interrupts each driver requested, and how many interrupts each driver has.

Although these macros do not show information about drivers that failed to attach, you can use the information to calculate the extent to which you can adjust the default limit. You can force any device that uses more than one interrupt without providing a callback routine to use fewer interrupts by adjusting the default limit. For such devices, reduce the default limit to free interrupts that can be used by other devices.

To adjust the default limit, set the `ddi_msix_alloc_limit` property to a value from 1 to 8 in the `/etc/system` file. Then, reboot the system for the change to take effect.

For information about correctly creating or updating `/etc/system` property values, see [“Updating Property Values in the /etc/system File” on page 334](#).

To maximize performance, start by assigning larger values and decrease the values in small increments until the system boots successfully without any warnings. Use the `::irm pools` and `::irmreqs` macros to measure the adjustment's impact on all attached drivers.

For example, suppose the following warnings are issued while booting the Oracle Solaris OS in a guest domain:

```
WARNING: emlxs3: interrupt pool too full.
WARNING: ddi_intr_alloc: cannot fit into interrupt pool
```

The `::irm pools` and `::irmreqs` macros show the following information:

```
echo "::irm pools" | mdb -k
ADDR OWNER TYPE SIZE REQUESTED RESERVED
```

```
00000400016be970 px#0 MSI/X 36 36

echo "00000400016be970::irmreqs" | mdb -k
ADDR OWNER TYPE CALLBACK NINTRS NREQ NAVAIL
00001000143acaa8 emlxs#0 MSI-X No 32 8 8
00001000170199f8 emlxs#1 MSI-X No 32 8 8
000010001400ca28 emlxs#2 MSI-X No 32 8 8
0000100016151328 igb#3 MSI-X No 10 3 3
0000100019549d30 igb#2 MSI-X No 10 3 3
0000040000e0f878 igb#1 MSI-X No 10 3 3
000010001955a5c8 igb#0 MSI-X No 10 3 3
```

The default limit in this example is 8 interrupts per device, which is not enough interrupts to accommodate attaching the final emlxs3 device to the system. Assuming that all emlxs instances behave in the same way, emlxs3 probably requested 8 interrupts.

By subtracting the 12 interrupts used by all of the igb devices from the total pool size of 36 interrupts, 24 interrupts are available for the emlxs devices. Dividing the 24 interrupts by 4 suggests that 6 interrupts per device would enable all emlxs devices to attach with equal performance. So, the following adjustment is added to the /etc/system file:

```
set ddi_msix_alloc_limit = 6
```

For information about correctly creating or updating /etc/system property values, see [“Updating Property Values in the /etc/system File” on page 334](#).

When the system successfully boots without warnings, the ::irm pools and ::irmreqs macros show the following updated information:

```
primary# echo "::irm pools" | mdb -k
ADDR OWNER TYPE SIZE REQUESTED RESERVED
00000400018ca868 px#0 MSI/X 36 36 36

echo "00000400018ca868::irmreqs" | mdb -k
ADDR OWNER TYPE CALLBACK NINTRS NREQ NAVAIL
0000100016143218 emlxs#0 MSI-X No 32 8 6
0000100014269920 emlxs#1 MSI-X No 32 8 6
000010001540be30 emlxs#2 MSI-X No 32 8 6
00001000140cbe10 emlxs#3 MSI-X No 32 8 6
00001000141210c0 igb#3 MSI-X No 10 3 3
0000100017549d38 igb#2 MSI-X No 10 3 3
0000040001ceac40 igb#1 MSI-X No 10 3 3
000010001acc3480 igb#0 MSI-X No 10 3 3
```

## **Listing Domain I/O Dependencies**

I/O operations for a domain are often provided by another domain such as a service domain or an I/O domain. For example, a service domain can export a virtual device or a root domain can provide direct access to a physical device.

Be aware of these implicit I/O dependencies, as an outage in a service domain or a root domain will result in a service interruption of the dependent domain, as well.

You can use the `ldm list-dependencies` command to view the I/O dependencies between domains. In addition to listing the dependencies of a domain, you can invert the output to show the dependents of a particular domain.

The following list shows the types of I/O dependencies that you can view by using the `ldm list-dependencies` command:

|       |                                                                                                                               |
|-------|-------------------------------------------------------------------------------------------------------------------------------|
| VDISK | Dependency created when a virtual disk is connected to a virtual disk backend that has been exported by a virtual disk server |
| VNET  | Dependency created when a virtual network device is connected to a virtual switch                                             |
| IOV   | Dependency created when an SR-IOV virtual function is associated with an SR-IOV physical function                             |

The following `ldm list-dependencies` commands show some of the ways in which you can view domain dependency information:

- To show detailed domain dependency information, use the `-l` option.

```
primary# ldm list-dependencies -l
DOMAIN DEPENDENCY TYPE DEVICE
primary
svcdom
ldg0 primary VDISK primary-vds0/vdisk0
 VNET primary-vsw0/vnet0
 svcdom VDISK svcdom-vds0/vdisk1
 VNET svcdom-vsw0/vnet1
ldg1 primary VDISK primary-vds0/vdisk0
 VNET primary-vsw0/vnet0
 IOV /SYS/MB/NET0/IOVNET.PF0.VF0
 svcdom VDISK svcdom-vds0/vdisk1
 VNET svcdom-vsw0/vnet1
 IOV /SYS/MB/NET2/IOVNET.PF0.VF0
```

- To show detailed information about dependents grouped by their dependencies, use both the `-l` and `-r` options.

```
primary# ldm list-dependencies -r -l
DOMAIN DEPENDENT TYPE DEVICE
primary ldg0 VDISK primary-vds0/vdisk0
 VNET primary-vsw0/vnet0
 ldg1 VDISK primary-vds0/vdisk0
 VNET primary-vsw0/vnet0
```

[Listing Domain I/O Dependencies](#)

---

|        |      |       |                             |
|--------|------|-------|-----------------------------|
| svcdom | ldg0 | IOV   | /SYS/MB/NET0/IOVNET.PF0.VF0 |
|        |      | VDISK | svcdom-vds0/vdisk1          |
|        |      | VNET  | svcdom-vsw0/vnet1           |
|        | ldg1 | VDISK | svcdom-vds0/vdisk1          |
|        |      | VNET  | svcdom-vsw0/vnet1           |
|        |      | IOV   | /SYS/MB/NET2/IOVNET.PF0.VF0 |

## **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 3.2 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 359
- “[Back-End Devices](#)” on page 361
- “[Installing the Oracle VM Server for SPARC P2V Tool](#)” on page 362
- “[Using the `ldmp2v` Command](#)” on page 365
- “[Known Issues With the Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool](#)” on page 372

### Oracle VM Server for SPARC P2V Tool Overview

The Oracle VM Server for SPARC Physical-to-Virtual (P2V) Conversion tool automatically converts an existing physical system to a virtual system that runs the Oracle Solaris 10 OS in a logical domain on a chip multithreading (CMT) system. You can run the `ldmp2v` command from a control domain that runs the Oracle Solaris 10 OS or the Oracle Solaris 11 OS to convert one of the following source systems to a logical domain:

- Any `sun4u` SPARC based system that runs at least the Solaris 8, Solaris 9, or Oracle Solaris 10 OS
- Any `sun4v` system that runs the Oracle Solaris 10 OS but does not run in a logical domain

---

**Note** - The `ldmp2v` command does not support any SPARC based system that runs the Oracle Solaris 10 OS with a ZFS root or the Oracle Solaris 11 OS.

---

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 Oracle 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.

## 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 `flarccreate` 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. You can then upgrade the system to the Oracle Solaris 10 OS by using the normal Oracle Solaris upgrade process. Modifications include updating the /etc/vfstab file to account for new disk names. Any Oracle 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 runs only 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 Oracle 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. On Oracle Solaris 10 systems, you can also use the Oracle Solaris JumpStart feature to perform a fully automated 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 (Oracle 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 Oracle Solaris Volume Manager or `/dev/vx/dsk/l domdg/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.

---

**Note** - The `ldmp2v` is installed on an Oracle Solaris 10 system from the `SUNWldmp2v` package, while the `ldmp2v` is installed by default on an Oracle Solaris 11 system from the `ldomsmanager` package.

---

## Prerequisites for using the SPARC P2V Tool

Before you can run the Oracle VM Server for SPARC P2V tool, ensure that the following conditions are met:

- The following Flash Archive 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 113434-06
- The target system runs at least Logical Domains 1.1 on the following:
  - Oracle Solaris 10 10/08 OS
  - Oracle Solaris 10 5/08 OS with the appropriate Logical Domains 1.1 patches
- Guest domains run at least the Oracle Solaris 10 5/08 OS
- The 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 of Using the SPARC P2V Tool

The Oracle VM Server for SPARC P2V tool has the following limitations:

- Only UFS file systems are supported.
- Only plain disks (`/dev/dsk/c0t0d0s0`), Oracle 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 Oracle Solaris 10 OS. However, you can also use the P2V tool to convert Solaris 8 and Solaris 9 operating systems that use VxVM.
- Oracle Solaris 10 systems that have zones can be converted if the zones are detached by using the `zoneadm detach` command prior to running the `ldmp2v collect` operation. After the P2V conversion completes, use the `zoneadm attach` command to reattach the zones that have been created on the guest domain. For information about performing these steps on a guest domain, see [Chapter 23, “Moving and Migrating Non-Global Zones \(Tasks\),”](#) in [“System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones”](#).

**Note** - The P2V tool does not update any zone configuration, such as the zone path or network interface, nor does the tool move or configure the storage for the zone path. You must manually update the zone configuration and move the zone path on the guest domain. See [Chapter 23, “Moving and Migrating Non-Global Zones \(Tasks\),” in “System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones”](#).

---

## ▼ How to Install the Oracle VM Server for SPARC P2V Tool

This procedure describes how to install the `ldmp2v` command on an Oracle Solaris 10 system by using the `SUNWldmp2v` package.

On an Oracle Solaris 11 system, the `ldmp2v` command is installed by default when you install the `ldomsmanager` package.

1. **From the Oracle VM Server for SPARC download page at <http://www.oracle.com/virtualization/index.html>, download the P2V software package, `SUNWldmp2v`.**  
The `SUNWldmp2v` package is included in the Oracle VM Server for SPARC zip file.
2. **Become an administrator.**
  - **For Oracle Solaris 10, see “Configuring RBAC (Task Map)” in “System Administration Guide: Security Services”.**
  - **For Oracle Solaris 11.2, see [Chapter 1, “About Using Rights to Control Users and Processes,” in “Securing Users and Processes in Oracle Solaris 11.2”](#).**
3. **Use the `pkgadd` command to install the `SUNWldmp2v` package.**

```
pkgadd -d . SUNWldmp2v
```
4. **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` – Determines 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 of conversion.

### **EXAMPLE 17-1** 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/c1t1d0s0 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/c1t1d0s3 8258597 4304 8171708 1% /var
swap 4487448 16 4487432 1% /var/run
swap 4487448 16 4487432 1% /tmp
/dev/dsk/c1t0d0s0 1016122 9 955146 1% /u01
vandikhout:/u1/home/dana
 6230996752 1051158977 5179837775 17% /home/dana
```

The following example 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 Archive 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 errors generate 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 the system state. Be sure to review all error messages thoroughly.

---

- **Skip file-system backup step.** If you already create backups of the system 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 17-2 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 disk0 volumia-vol0@primary-vds0
disk1 disk1 volumia-vol1@primary-vds0
```

- The following example shows how to 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 example shows how to 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 17-3** 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. On Oracle Solaris 10 systems, you can use the Oracle Solaris JumpStart feature to perform a fully automated 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 JumpStart, see “[Oracle Solaris 10 1/13 Installation Guide: JumpStart 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 required only 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...
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.



**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.

---

The answers to the `sysid` questions are used only 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
- 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

## Known Issues With the Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool

### **ldmp2v convert Command: VxVM Warning Messages During Boot**

Running Veritas Volume Manager (VxVM) 5.x on the Oracle Solaris 10 OS is the only supported (tested) version for the Oracle VM Server for SPARC P2V tool. Older versions of VxVM, such as 3.x and 4.x running on the Solaris 8 and Solaris 9 operating systems, might also work. In those cases, the first boot after running the `ldmp2v convert` command might show warning messages from the VxVM drivers. You can ignore these messages. You can remove the old VRTS\* packages after the guest domain has booted.

```
Boot device: disk0:a File and args:
SunOS Release 5.10 Version Generic_139555-08 64-bit
Copyright 1983-2009 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
Hostname: normaal
Configuring devices.
/kernel/drv/sparcv9/vxdmp: undefined symbol 'romp'
WARNING: mod_load: cannot load module 'vxdmp'
WARNING: vxdmp: unable to resolve dependency, module 'misc/ted' not found
/kernel/drv/sparcv9/vxdmp: undefined symbol 'romp'
WARNING: mod_load: cannot load module 'vxdmp'
WARNING: vxdmp: unable to resolve dependency, module 'misc/ted' not found
/kernel/drv/sparcv9/vxio: undefined symbol 'romp'
WARNING: mod_load: cannot load module 'vxio'
WARNING: vxio: unable to resolve dependency, module 'drv/vxdmp' not found
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
```

```
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
NOTICE: VxVM not started
```

## Upgrade Option Not Presented When Using `ldmp2v prepare -R`

The Oracle Solaris Installer does not present the Upgrade option when the partition tag of the slice that holds the root (/) file system is not set to root. This situation occurs if the tag is not explicitly set when labeling the guest's boot disk. You can use the `format` command to set the partition tag as follows:

```
AVAILABLE DISK SELECTIONS:
0. c0d0 <SUN-DiskImage-10GB cyl 282 alt 2 hd 96 sec 768>
 /virtual-devices@100/channel-devices@200/disk@0
1. c4t2d0 <SUN146G cyl 14087 alt 2 hd 24 sec 848>
 /pci@400/pci@0/pci@1/scsi@0/sd@2,0
2. c4t3d0 <SUN146G cyl 14087 alt 2 hd 24 sec 848>
 /pci@400/pci@0/pci@1/scsi@0/sd@3,0
Specify disk (enter its number)[0]: 0
selecting c0d0
[disk formatted, no defect list found]
format> p
```

```
PARTITION MENU:
0 - change '0' partition
1 - change '1' partition
2 - change '2' partition
3 - change '3' partition
4 - change '4' partition
5 - change '5' partition
6 - change '6' partition
7 - change '7' partition
select - select a predefined table
modify - modify a predefined partition table
name - name the current table
print - display the current table
label - write partition map and label to the disk
!<cmd> - execute <cmd>, then return
quit
```

```
partition> 0
Part Tag Flag Cylinders Size Blocks
0 unassigned w 0 0 (0/0/0) 0

Enter partition id tag[unassigned]: root
Enter partition permission flags[w]:
Enter new starting cyl[0]: 0
Enter partition size[0b, 0c, 0e, 0.00mb, 0.00gb]: 8g
```

```
partition> label
Ready to label disk, continue? y

partition>
```

## **ldmp2v Command: ufsdump Archiving Method Is No Longer Used**

Restoring `ufsdump` archives on a virtual disk that is backed by a file on a UFS file system might cause the system to hang. In such a case, the `ldmp2v prepare` command will exit. You might encounter this problem when you manually restore `ufsdump` archives in preparation for the `ldmp2v prepare -R /altroot` command when the virtual disk is a file on a UFS file system. For compatibility with previously created `ufsdump` archives, you can still use the `ldmp2v prepare` command to restore `ufsdump` archives on virtual disks that are not backed by a file on a UFS file system. However, the use of `ufsdump` archives is not recommended.

## Oracle VM Server for SPARC Configuration Assistant (Oracle Solaris 10)

---

The Oracle VM Server for SPARC Configuration Assistant (the `ldmconfig` command) 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.

---

**Note** - The `ldmconfig` command is supported only on Oracle Solaris 10 systems.

---

In addition to this chapter, see the [ldmconfig\(1M\)](#) man page.

## Using the Configuration Assistant (`ldmconfig`)

The `ldmconfig` command 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 for Running the Configuration Assistant

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 of the Configuration Assistant

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
- Works only 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` command 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 an 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. This setting 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 Oracle VM Server for SPARC 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 doing so might result in a partially configured system.

---

After the commands have been completed successfully, reboot the system for the changes to take effect.

## Using Power Management

---

This chapter contains information about using power management on Oracle VM Server for SPARC systems.

### Using Power Management

To enable power management (PM), you first need to set the PM policy 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 *Oracle 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 policy governs system power usage at any point in time. The following power policies are supported, assuming that the underlying platform has implemented PM features:

- **Disabled.** Permits the system to use all the power that is available.
- **Performance.** Enables one or more of the following PM features that have a negligible affect on performance:
  - CPU core auto-disabling
  - CPU clock cycle skip
  - CPU dynamic voltage and frequency scaling (DVFS)
  - Coherency link scaling
  - Oracle Solaris Power Aware Dispatcher (PAD)
- **Elastic.** Adapts the system power usage to the current utilization level by using the PM features described in the performance section. For example, the power state of resources is reduced as utilization decreases.

## Power Management Features

The PM features are as follows:

- **CPU core auto-disabling.** When the elastic or performance policy is in effect, the Logical Domains Manager automatically disables a CPU core when all the hardware threads (strands) on that core are not bound to a domain. This feature is available only for the UltraSPARC T2, UltraSPARC T2 Plus, SPARC T3, and SPARC T4 platforms.
- **CPU clock cycle skip.** When the elastic policy is in effect, the Logical Domains Manager automatically adjusts the number of clock cycles that execute instructions on the following CPU resources that are bound to domains:
  - Processors (SPARC T3 or SPARC T4 on domains that run the Oracle Solaris 10 or Oracle Solaris 11 OS)
  - Cores (SPARC M5 only on domains that run the Oracle Solaris 10 OS)
  - Core-pairs (SPARC T5 or SPARC M6 only on domains that run the Oracle Solaris 10 OS)

The Logical Domains Manager also applies cycle skipping if the processor, core, or core-pair has no bound strands.

- **CPU dynamic voltage and frequency scaling (DVFS).** When the elastic policy is in effect, the Logical Domains Manager automatically adjusts the clock frequency of processors that are bound to domains running the Oracle Solaris 10 OS. The Logical Domains Manager also reduces the clock frequency on SPARC T5, SPARC M5, and SPARC M6 processors that have no bound strands. This feature is available only on SPARC T5, SPARC M5 and SPARC M6 systems.
- **Coherency link scaling.** When the elastic policy is in effect, the Logical Domains Manager causes the hypervisor to automatically adjust the number of coherency links that are in use. This feature is only available on SPARC T5-2 systems.
- **Power limit.** You can set a *power limit* on SPARC T3, SPARC T4, SPARC T5, SPARC M5, and SPARC M6 platforms to restrict the power draw of a system. If the power draw is greater than the power limit, PM uses techniques to reduce power. You can use the ILOM service processor (SP) to set the power limit.

See the following documents:

- *Oracle 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 CPUs. If the power draw drops below the power limit, the power state of those resources is permitted to increase. If the system has the elastic policy in effect, an increase in the power state of resources is driven by the utilization level.

- **Solaris Power Aware Dispatcher (PAD).** A guest domain that runs the Oracle Solaris 11.1 OS uses the power-aware dispatcher (PAD) on SPARC T5, SPARC M5, and SPARC M6 systems to minimize power consumption from idle or under-utilized resources. PAD, instead of the Logical Domains Manager, adjusts the CPU clock cycle skip level and DVFS level.

For instructions on configuring the power policy by using the ILOM 3.0 firmware CLI, see “Monitoring Power Consumption” in the *Oracle Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide*.

## Viewing Power-Consumption Data

The Oracle VM Server for SPARC 3.1 software provides the Power Management (PM) Observability Module and the `ldmpower` command that enable you to view CPU thread power-consumption data for your domains.

The PM Observability Module is enabled by default because the `ldmd/`  
`pm_observability_enabled` Service Management Facility (SMF) property is set to `true`. See the [ldmd\(1M\)](#) man page.

The `ldmpower` command has the following options and operands with which you can customize the power-consumption reporting data:

```
ldmpower [-ehiprstvx | -o hours | -m minutes] | -c resource [-l domain-name[,domain-name[,...]]]
[interval [count]]
```

For information about the options, see the [ldmpower\(1M\)](#) man page.

To run this command as a non-privileged user, you must be assigned the LDOMs Power Mgmt Observability rights profile. If you already have been assigned the LDOMs Management or LDOMs Review rights profile, you automatically have permission to run the `ldmpower` command.

For information about how Oracle VM Server for SPARC uses rights, see [“Logical Domains Manager Profile Contents” on page 29](#).

These rights profiles can be assigned directly to users or to a role that is then assigned to users. When one of these profiles is assigned directly to a user, you must use the `pexec` command or a profile shell, such as `pfbash` or `pfksh`, to successfully use the `ldmpower` command to view CPU thread power-consumption data. See [“Delegating the Management of Logical Domains by Using Rights” on page 25](#).

The following examples show how to enable the PM Observability Module and show ways in which to gather power-consumption data for the CPUs that are assigned to your domains.

**EXAMPLE 19-1** Enabling the Power Management Observability Module

The following command enables the PM Observability Module by setting the `ldmd/pm_observability_enabled` property to `true` if the property is currently set to `false`.

```
svccfg -s ldmd setprop ldmd/pm_observability_enabled=true
svcadm refresh ldmd
svcadm restart ldmd
```

**EXAMPLE 19-2** Using a Profile Shell to Obtain CPU Thread Power-Consumption Data by Using Roles and Rights Profiles

- The following example shows how to create the `ldmpower` role with the LDoms Power Mgmt Observability rights profile, which permits you to run the `ldmpower` command.

```
primary# roleadd -P "LDoms Power Mgmt Observability" ldmpower
primary# passwd ldmpower
New Password:
Re-enter new Password:
passwd: password successfully changed for ldmpower
```

This command assigns the `ldmpower` role to the `sam` user.

```
primary# usermod -R ldmpower sam
```

User `sam` assumes the `ldmpower` role and can use the `ldmpower` command. For example:

```
$ id
uid=700299(sam) gid=1(other)
$ su ldmpower
Password:
$ pfexec ldmpower
Processor Power Consumption in Watts
DOMAIN 15_SEC_AVG 30_SEC_AVG 60_SEC_AVG
primary 75 84 86
gdom1 47 24 19
gdom2 10 24 26
```

- The following examples show how to use rights profiles to run the `ldmpower` command.
  - **Oracle Solaris 10:** Assign the rights profile to `username`.

```
primary# usermod -P "All,Basic Solaris User,LDoms Power Mgmt Observability" \
username
```

The following commands show how to verify that the user is `sam` and that the `All`, `Basic Solaris User`, and `LDoms Power Mgmt Observability` rights profiles are in effect.

```
$ id
```

```

uid=702048(sam) gid=1(other)
$ profiles
All
Basic Solaris User
LDoms Power Mgmt Observability
$ pfexec ldmpower
Processor Power Consumption in Watts
DOMAIN 15_SEC_AVG 30_SEC_AVG 60_SEC_AVG
primary 75 84 86
gdom1 47 24 19
gdom2 10 24 26

```

- **Oracle Solaris 11:** Assign the rights profile to a user.

```
primary# usermod -P +"LDoms Power Mgmt Observability" sam
```

The following commands show how to verify that the user is `sam` and that the `All`, `Basic Solaris User`, and `LDoms Power Mgmt Observability` rights profiles are in effect.

```

$ id
uid=702048(sam) gid=1(other)
$ profiles
All
Basic Solaris User
LDoms Power Mgmt Observability
$ pfexec ldmpower
Processor Power Consumption in Watts
DOMAIN 15_SEC_AVG 30_SEC_AVG 60_SEC_AVG
primary 75 84 86
gdom1 47 24 19
gdom2 10 24 26

```

#### **EXAMPLE 19-3** Viewing Processor Power-Consumption Data

The following examples show how to use the `ldmpower` to report processor power-consumption data for your domains.

- The following command shows the 15-second, 30-second, and 60-second rolling average processor power-consumption data for all domains:

```

primary# ldmpower
Processor Power Consumption in Watts
DOMAIN 15_SEC_AVG 30_SEC_AVG 60_SEC_AVG
primary 75 84 86
gdom1 47 24 19
gdom2 10 24 26

```

- The following command shows extrapolated power-consumption data for all the domains: primary, gdom1, and gdom2.

```
primary# ldmpower -x
System Power Consumption in Watts
DOMAIN 15_SEC_AVG 30_SEC_AVG 60_SEC_AVG
primary 585/57.47% 701/68.96% 712/70.22%
gdom1 132/12.97% 94/9.31% 94/9.30%
gdom2 298/29.27% 218/21.47% 205/20.22%
```

- The following command shows the instantaneous processor power-consumption data for the gdom2 and gdom5 domains. It reports the data every ten seconds for five times.

```
primary# ldmpower -itl gdom2,gdom5 10 5
Processor Power Consumption in Watts
DOMAIN TIMESTAMP INSTANT
gdom2 2013.05.17 11:14:45 13
gdom5 2013.05.17 11:14:45 24

gdom2 2013.05.17 11:14:55 18
gdom5 2013.05.17 11:14:55 26

gdom2 2013.05.17 11:15:05 9
gdom5 2013.05.17 11:15:05 16

gdom2 2013.05.17 11:15:15 15
gdom5 2013.05.17 11:15:15 19

gdom2 2013.05.17 11:15:25 12
gdom5 2013.05.17 11:15:25 18
```

- The following command shows the average power-consumption data for the last 12 hours for all domains. Data is shown at one-hour intervals starting from the last requested hourly calculation.

```
primary# ldmpower -eto 12
Per domain MINIMUM and MAXIMUM power consumption ever recorded:
primary 2013.05.17 08:53:06 3 Min Processors
primary 2013.05.17 08:40:44 273 Max Processors
gdom1 2013.05.17 09:56:35 2 Min Processors
gdom1 2013.05.17 08:53:06 134 Max Processors
gdom2 2013.05.17 10:31:55 2 Min Processors
gdom2 2013.05.17 08:56:35 139 Max Processors

primary 2013.05.17 08:53:06 99 Min Memory
primary 2013.05.17 08:40:44 182 Max Memory
gdom1 2013.05.17 09:56:35 13 Min Memory
gdom1 2013.05.17 08:53:06 20 Max Memory
```

|       |                     |    |            |
|-------|---------------------|----|------------|
| gdom2 | 2013.05.17 10:31:55 | 65 | Min Memory |
| gdom2 | 2013.05.17 08:56:35 | 66 | Max Memory |

Processor Power Consumption in Watts  
12 hour's worth of data starting from 2013.05.16 23:17:02

| DOMAIN  | TIMESTAMP           | 1 HOUR AVG |
|---------|---------------------|------------|
| primary | 2013.05.17 09:37:35 | 112        |
| gdom1   | 2013.05.17 09:37:35 | 15         |
| gdom2   | 2013.05.17 09:37:35 | 26         |
| primary | 2013.05.17 10:37:35 | 96         |
| gdom1   | 2013.05.17 10:37:35 | 12         |
| gdom2   | 2013.05.17 10:37:35 | 21         |
| primary | 2013.05.17 11:37:35 | 85         |
| gdom1   | 2013.05.17 11:37:35 | 11         |
| gdom2   | 2013.05.17 11:37:35 | 23         |
| ...     |                     |            |



## 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 2.2 software.

This chapter covers the following topics:

- “Oracle VM Server for SPARC Management Information Base Overview” on page 387
- “Installing and Configuring the Oracle VM Server for SPARC MIB Software” on page 392
- “Managing Security” on page 395
- “Monitoring Domains” on page 397
- “Using SNMP Traps” on page 417
- “Starting and Stopping Domains” on page 425

## Oracle VM Server for SPARC Management Information Base Overview

This section covers the following topics:

- “Related Products and Features” on page 388
- “Software Components” on page 388
- “System Management Agent” on page 389
- “Logical Domains Manager and the Oracle VM Server for SPARC MIB” on page 390
- “Oracle VM Server for SPARC MIB Object Tree” on page 391

## Related Products and Features

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)
- Oracle Solaris SNMP Agent
- 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)

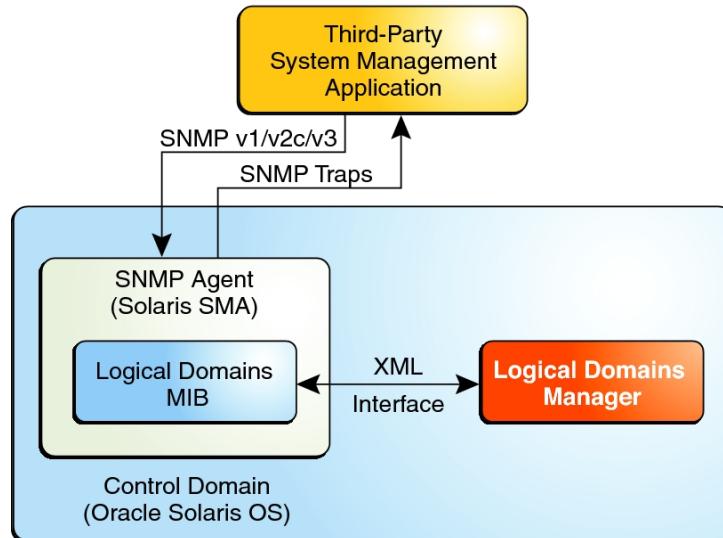
## Software Components

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 Oracle Solaris SNMP agent 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 Oracle Solaris SNMP agent, 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 389 and “[Logical Domains Manager and the Oracle VM Server for SPARC MIB](#)” on page 390.

**FIGURE 20-1** Oracle VM Server for SPARC MIB Interaction With Oracle Solaris SNMP Agent, Logical Domains Manager, and a Third-Party System Management Application



## System Management Agent

The Oracle Solaris SNMP agent 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 Oracle Solaris SNMP agent on the control domain.

The Oracle Solaris SNMP agent 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 20-1](#).

## 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 Oracle Solaris SNMP agent with the Oracle VM Server for SPARC MIB is started and monitored by the `init` process and the Service Management Facility (SMF). If the Oracle Solaris SNMP agent fails and exits, SMF restarts the process automatically, and then the new process dynamically restarts the Oracle VM Server for SPARC MIB module.

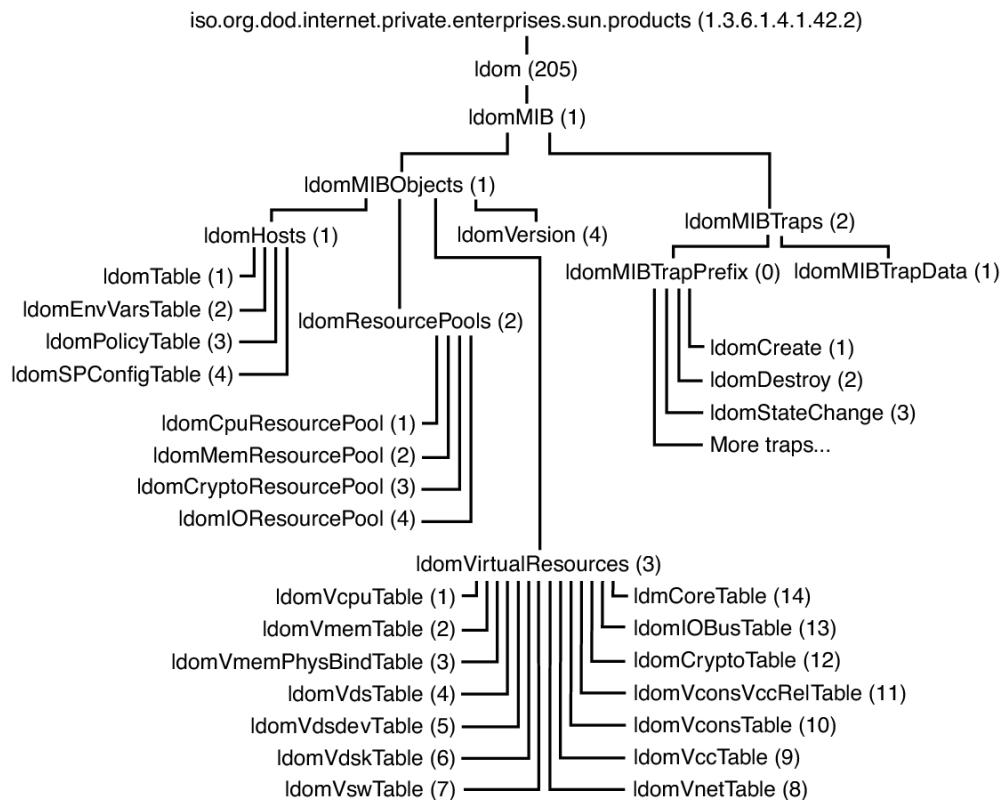
## Oracle VM Server for SPARC MIB Object Tree

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 20-2** 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 on Oracle Solaris 10 and Oracle Solaris 11 systems. For information about administering SNMP, see the `snmpd.conf(4)` or `snmpd.conf(5)` man page.

### Installing and Configuring the Oracle VM Server for SPARC MIB Software

The following table lists 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 on an Oracle Solaris 10 system. Or, use the <code>pkg install</code> command to install the <code>system/ldoms/mib</code> package on an Oracle Solaris 11 system. | <a href="#">“How to Install the Oracle VM Server for SPARC MIB Software Package” on page 392</a>                       |
| Load the Oracle VM Server for SPARC MIB module into the Oracle Solaris SNMP agent to query the Oracle VM Server for SPARC MIB. | Modify the SNMP configuration file to load the <code>ldomMIB.so</code> module.                                                                                                                                                                        | <a href="#">“How to Load the Oracle VM Server for SPARC MIB Module Into the Oracle Solaris SNMP Agent” on page 393</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</code> package from an Oracle Solaris 10 system. Or, use the <code>pkg remove</code> command to remove the <code>system/ldoms/mib</code> package from an Oracle Solaris 11 system.   | <a href="#">“How to Remove the Oracle VM Server for SPARC MIB Software Package” on page 394</a>                        |

#### ▼ How to 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 on Oracle Solaris 10 and Oracle Solaris 11 systems. The Oracle VM Server for SPARC MIB software package is included as part of the Oracle VM Server for SPARC 3.2 software.

The Oracle VM Server for SPARC MIB package includes the following files:

- `/opt/SUNWldmib/lib/mibs/SUN-LDOM-MIB.mib`
- `/opt/SUNWldmib/lib/ldomMIB.so`

**Before You Begin** Download and install the Oracle VM Server for SPARC 3.2 software. See [Chapter 2, “Installing and Enabling Software,” in “Oracle VM Server for SPARC 3.2 Installation Guide”](#).

**1. Determine whether your system runs the Oracle Solaris 10 OS or the Oracle Solaris 11 OS.**

```
uname -r
```

**2. Install the Oracle VM Server for SPARC MIB software on the primary domain.**

- **Oracle Solaris 10: Install the Oracle VM Server for SPARC MIB software package, SUNWldmib.**

```
pkgadd -d . SUNWldmib.v
```

- **Oracle Solaris 11: Install the Oracle VM Server for SPARC MIB software package, system/ldoms/mib.**

```
pkg install -v -g IPS-package-directory/ldoms.repo mib
```

**Next Steps** After you install this package, you can configure your system to dynamically load the Oracle VM Server for SPARC MIB module. See [“How to Load the Oracle VM Server for SPARC MIB Module Into the Oracle Solaris SNMP Agent” on page 393](#).

**▼ How to Load the Oracle VM Server for SPARC MIB Module Into the Oracle Solaris SNMP Agent**

The Oracle VM Server for SPARC MIB module, `ldomMIB.so`, must be loaded into the Oracle Solaris SNMP agent 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 Oracle Solaris SNMP agent are provided in *Solaris System Management Agent Developer's Guide*. For more information about the Oracle Solaris SNMP agent, see *Solaris System Management Administration Guide*.

**1. Determine whether your system runs the Oracle Solaris 10 OS or the Oracle Solaris 11 OS.**

```
uname -r
```

**2. Update the SNMP configuration file.**

- **Oracle Solaris 10:**

Append the following line to the /etc/sma/snmp/snmpd.conf configuration file:

```
dlmod ldomMIB /opt/SUNWldmib/lib/ldomMIB.so
```

■ **Oracle Solaris 11:**

Append the following line to the /etc/net-snmp/snmp/snmpd.conf configuration file:

```
dlmod ldomMIB /opt/SUNWldmib/lib/ldomMIB.so
```

**3. Restart the SMF service.**

■ **Oracle Solaris 10:**

```
svcadm restart svc:/application/management/sma:default
```

■ **Oracle Solaris 11:**

```
svcadm restart svc:/application/management/net-snmp:default
```

▼ **How to 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 and unload the Oracle VM Server for SPARC MIB module from an Oracle Solaris 10 or Oracle Solaris 11 system.

**1. Stop the SMF service.**

■ **Oracle Solaris 10:**

```
svcadm disable svc:/application/management/sma:default
```

■ **Oracle Solaris 11:**

```
svcadm disable svc:/application/management/net-snmp:default
```

**2. Remove the Oracle VM Server for SPARC MIB software package from the primary domain.**

■ **Oracle Solaris 10:**

```
pkgrm SUNWldmib
```

■ **Oracle Solaris 11:**

```
pkg uninstall system/ldoms/mib
```

### 3. Update the SNMP configuration file.

- **Oracle Solaris 10:**

Remove the line that you added to the /etc/sma/snmp/snmpd.conf file during installation.

```
dlmod ldomMIB /opt/SUNWldmib/lib/ldomMIB.so
```

- **Oracle Solaris 11:**

Remove the line that you added to the /etc/net-snmp/snmp/snmpd.conf file during installation.

```
dlmod ldomMIB /opt/SUNWldmib/lib/ldomMIB.so
```

### 4. Restart the SMF service.

- **Oracle Solaris 10:**

```
svcadm restart svc:/application/management/sma:default
```

- **Oracle Solaris 11:**

```
svcadm restart svc:/application/management/net-snmp: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 Oracle Solaris SNMP agent. 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 Oracle Solaris SNMP agent in SNMP with the Oracle VM Server for SPARC MIB. This type of user does not interact with or conflict with users that you might have configured by using the rights feature of Oracle Solaris for the Logical Domains Manager.

---

## ▼ How to Create the Initial snmpv3 User

This procedure describes how to create the initial snmpv3 user on an Oracle Solaris 10 or Oracle Solaris 11 system.

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 Oracle Solaris SNMP agent.**

■ **Oracle Solaris 10:**

```
svcadm disable -t svc:/application/management/sma:default
```

■ **Oracle Solaris 11:**

```
svcadm disable svc:/application/management/net-snmp:default
```

**2. Create the initial user.**

This step 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.

---

■ **Oracle Solaris 10:**

```
/usr/sfw/bin/net-snmp-config --create-snmpv3-user -a my-password initial-user
```

■ **Oracle Solaris 11:**

```
/usr/bin/net-snmp-config --create-snmpv3-user -a my-password initial-user
```

**3. Start the Oracle Solaris SNMP agent.**

■ **Oracle Solaris 10:**

```
svcadm enable svc:/application/management/sma:default
```

■ **Oracle Solaris 11:**

```
svcadm enable svc:/application/management/net-snmp: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 397
- “[Querying the Oracle VM Server for SPARC MIB](#)” on page 398
- “[Retrieving Oracle VM Server for SPARC MIB Information](#)” on page 400

## Setting Environment Variables

Before you can query the Oracle VM Server for SPARC MIB, you must set the PATH, MIBDIRS, and MIBS environment variables for the shell that you use. The values are different for Oracle Solaris 10 and Oracle Solaris 11.

- Oracle Solaris 10:

- **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
```

- Oracle Solaris 11:

- **For C shell users:**

```
% setenv PATH /usr/bin:$PATH
% setenv MIBDIRS /opt/SUNWldmib/lib/mibs:/etc/net-snmp/snmp/mibs
% setenv MIBS +SUN-LDOM-MIB
```

- **For Bourne and Korn shell users:**

```
$ PATH=/usr/bin:$PATH; export PATH
$ MIBDIRS=/opt/SUNWldmib/lib/mibs:/etc/net-snmp/snmp/mibs; export MIBDIRS
$ MIBS+=+SUN-LDOM-MIB; export MIBS
```

## Querying the Oracle VM Server for SPARC MIB

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.

- To retrieve a single MIB object:

```
snmpget -v version -c community-string host MIB-object
```

- To 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
```

---

**Note** - You receive empty SNMP tables if you query the Oracle VM Server for SPARC MIB 2.1 software using the `snmptable` command with the `-v2c` or `-v3` option. The `snmptable` command with the `-v1` option works as expected.

To work around this issue, use the `-CB` option to use only GETNEXT, not GETBULK, requests to retrieve data. See “[Querying the Oracle VM Server for SPARC MIB](#)” on page 398.

---

### EXAMPLE 20-1 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 20-2 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: 8
SUN-LDOM-MIB::ldomNumVCpu.2 = INTEGER: 4
SUN-LDOM-MIB::ldomMemSize.1 = INTEGER: 3360
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: 1
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: 5f8817d4-5d2e-6f7d-c4af-91b5b34b5723
SUN-LDOM-MIB::ldomUUID.2 = STRING: 11284146-87ca-4877-8d80-cd0f60d5ec26
SUN-LDOM-MIB::ldomMacAddress.1 = STRING: 00:14:4f:46:47:d6
SUN-LDOM-MIB::ldomMacAddress.2 = STRING: 00:14:4f:f8:d5:6c
SUN-LDOM-MIB::ldomHostID.1 = STRING: 0x844647d6
SUN-LDOM-MIB::ldomHostID.2 = STRING: 0x84f8d56c
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:
SUN-LDOM-MIB::ldomExtMapinSpace.1 = STRING: off
SUN-LDOM-MIB::ldomExtMapinSpace.2 = STRING: off
SUN-LDOM-MIB::ldomWholeCore.1 = INTEGER: 0
SUN-LDOM-MIB::ldomWholeCore.2 = INTEGER: 0
SUN-LDOM-MIB::ldomCpuArch.1 = STRING: native
SUN-LDOM-MIB::ldomCpuArch.2 = STRING: native
SUN-LDOM-MIB::ldomShutdownGroup.1 = INTEGER: 0
SUN-LDOM-MIB::ldomShutdownGroup.2 = INTEGER: 15
SUN-LDOM-MIB::ldomPerfCounters.1 = STRING: htstrand
SUN-LDOM-MIB::ldomPerfCounters.2 = STRING: global,htstrand
```

- 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-LDOMMIB::ldomTable
```

**EXAMPLE 20-3** 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 20-1** Domain Table (ldomTable)

| Name           | Data Type      | Access         | Description                                                                                                                                                               |
|----------------|----------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomIndex      | Integer        | Not accessible | Integer that is used as an index of this table                                                                                                                            |
| ldomName       | Display string | Read-only      | Name of the domain                                                                                                                                                        |
| ldomAdminState | 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> |
| 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> </ul>                          |

| Name              | Data Type      | Access    | Description                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------|----------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                   |                |           | <ul style="list-style-type: none"> <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 | <p>One of the following memory units:</p> <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>                                                                                                                                                                                                                       |
| 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 <code>ignore</code> , <code>panic</code> , <code>reset</code> , or <code>stop</code>                                                                                                                                                                                                                                                                                                  |
| ldomMaster        | Display string | Read-only | Name of up to four master domains for a slave domain                                                                                                                                                                                                                                                                                                                                                                                      |
| ldomExtMapinSpace | Display string | Read-only | Extended mapin space for a domain. The extended mapin space refers to the additional LDC shared memory space. This memory space is required to support a large number of virtual I/O devices that use direct-mapped shared memory. This space is also used by virtual network devices to improve performance and scalability. The default value is <code>off</code> .                                                                     |
| ldomThreading     | Display string | Read-only | <p>Specifies the high instructions per cycle (IPC) threading control for a domain. The dynamic threading provides control over the number of hardware threads activated per core. Valid values are:</p> <ul style="list-style-type: none"> <li>■ <code>max-throughput</code>, which means that all strands per core are active (default value)</li> <li>■ <code>max-ipc</code>, which means that one strand per core is active</li> </ul> |

| Name              | Data Type      | Access    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|-------------------|----------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ldomWholeCore     | Integer        | Read-only | Constrains the domain to use whole-cores only. If the whole-core constraint is not enabled, the value is 0. Otherwise, the value shows the number of max-cores.                                                                                                                                                                                                                                                                                                      |
| ldomCpuArch       | Display string | Read-only | CPU architecture for a domain. The CPU architecture specifies whether the domain can be migrated to another sun4v CPU architecture. Valid values are: <ul style="list-style-type: none"> <li>■ native, which means that the domain is permitted to be migrated only to platforms of the same sun4v CPU architecture (default value)</li> <li>■ generic, which means that the domain is permitted to be migrated to all compatible sun4v CPU architectures</li> </ul> |
| ldomShutdownGroup | Integer        | Read-only | Shutdown-group number for a guest domain. On a SPARC64-X system, an SP-initiated ordered shutdown will shut down domains in descending order of their shutdown-group numbers, from 15 to 0. The default value is 15.                                                                                                                                                                                                                                                 |
| ldomPerfCounters  | String         | Read-only | Performance register access information for a guest domain. Values can be global (on one domain at a time only) and optionally one of the following: htstrand or strand. The default value is htstrand.                                                                                                                                                                                                                                                              |

## Environment Variables Table (ldomEnvVarsTable)

ldomEnvVarsTable describes the OpenBoot PROM environment variables that all domains use.

**TABLE 20-2** 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 20-3** Domain Policy Table (`lDomPolicyTable`)

| Name                                 | Data Type      | Access         | Description                                                                                                                                                                        |
|--------------------------------------|----------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>lDomPolicyLdomIndex</code>     | Integer        | Read-only      | Integer that is used as an index into <code>lDomTable</code> that represents the domain that contains the DRM policy                                                               |
| <code>lDomPolicyIndex</code>         | Integer        | Not accessible | Integer that is used to index the DRM policy in this table                                                                                                                         |
| <code>lDomPolicyName</code>          | Display string | Read-only      | Policy name                                                                                                                                                                        |
| <code>lDomPolicyStatus</code>        | Display string | Read-only      | Policy status                                                                                                                                                                      |
| <code>lDomPolicyPriority</code>      | Integer        | Read-only      | Priority that is used to determine which DRM policy is selected when policies overlap                                                                                              |
| <code>lDomPolicyVcpuMin</code>       | Integer        | Read-only      | Minimum number of virtual CPUs for a domain                                                                                                                                        |
| <code>lDomPolicyVcpuMax</code>       | Integer        | Read-only      | Maximum number of virtual CPUs for a domain. A value of <code>unlimited</code> uses the maximum integer value of 2147483647.                                                       |
| <code>lDomPolicyUtilLower</code>     | Integer        | Read-only      | Lower utilization level at which policy analysis is triggered                                                                                                                      |
| <code>lDomPolicyUtilUpper</code>     | Integer        | Read-only      | Upper utilization level at which policy analysis is triggered                                                                                                                      |
| <code>lDomPolicyTodBegin</code>      | Display string | Read-only      | Effective start time of a policy with a format of <code>hh:mm:[ss]</code>                                                                                                          |
| <code>lDomPolicyTodEnd</code>        | Display string | Read-only      | Effective stop time of a policy with a format of <code>hh:mm:[ss]</code>                                                                                                           |
| <code>lDomPolicySampleRate</code>    | Integer        | Read-only      | Resource cycle time in seconds                                                                                                                                                     |
| <code>lDomPolicyElasticMargin</code> | Integer        | Read-only      | Amount of buffer between <code>util-lower</code> property ( <code>lDomPolicyUtilLower</code> ) and the number of free virtual CPUs to avoid oscillations at low virtual CPU counts |
| <code>lDomPolicyAttack</code>        | Integer        | Read-only      | Maximum amount of a resource to be added during any one resource-control cycle. A value of <code>unlimited</code> uses the maximum integer value of 2147483647.                    |
| <code>lDomPolicyDecay</code>         | 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 20-4** 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 20-5** 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 ldomCpuRpReserved<br>Unit | Integer   | Read-only | One of the following CPU allocation units:<br><ul style="list-style-type: none"> <li>■ 1 is MHz</li> <li>■ 2 is GHz</li> </ul> The default value is MHz. |

**TABLE 20-6** Scalar Variables for Memory Resource Pool

| Name                                                                         | Data Type | Access    | Description                                                                                                                                                         |
|------------------------------------------------------------------------------|-----------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ldomMemRpCapacity</code>                                               | Integer   | Read-only | Maximum reservation allowed by the resource pool in <code>MemRpCapacityUnits</code>                                                                                 |
| <code>ldomMemRpReserved</code>                                               | Integer   | Read-only | Amount of memory, in <code>MemRpReservedUnits</code> , that is currently reserved from the resource pool                                                            |
| <code>ldomMemRpCapacityUnit</code><br>and <code>ldomMemRpReservedUnit</code> | Integer   | Read-only | One of the following memory allocation units:<br><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.                                   |           |           |                                                                                                                                                                     |

**TABLE 20-7** Scalar Variables for Cryptographic Resource Pool

| Name                              | Data Type | Access    | Description                                                                     |
|-----------------------------------|-----------|-----------|---------------------------------------------------------------------------------|
| <code>ldomCryptoRpCapacity</code> | Integer   | Read-only | Maximum reservation allowed by the resource pool                                |
| <code>ldomCryptoRpReserved</code> | Integer   | Read-only | Number of cryptographic units that is currently reserved from the resource pool |

**TABLE 20-8** 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 20-9** 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)                                                                                   |
| <code>ldomVcpuOperationalStatus</code> | Integer        | Read-only      | One of the following CPU statuses:                                                                                    |

## Monitoring Domains

---

| Name                  | Data Type      | Access    | Description                                                                                                                                                                                                                                                               |
|-----------------------|----------------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                       |                |           | 1=Unknown                                                                                                                                                                                                                                                                 |
|                       |                |           | 2=Other                                                                                                                                                                                                                                                                   |
|                       |                |           | 3=OK                                                                                                                                                                                                                                                                      |
|                       |                |           | 4=Degraded                                                                                                                                                                                                                                                                |
|                       |                |           | 5=Stressed                                                                                                                                                                                                                                                                |
|                       |                |           | 6=Predictive failure                                                                                                                                                                                                                                                      |
|                       |                |           | 7=Error                                                                                                                                                                                                                                                                   |
|                       |                |           | 8=Nonrecoverable error                                                                                                                                                                                                                                                    |
|                       |                |           | 9=Starting                                                                                                                                                                                                                                                                |
|                       |                |           | 10=Stopping                                                                                                                                                                                                                                                               |
|                       |                |           | 11=Stopped                                                                                                                                                                                                                                                                |
|                       |                |           | 12=In service                                                                                                                                                                                                                                                             |
|                       |                |           | 13=No contact                                                                                                                                                                                                                                                             |
|                       |                |           | 14=Lost communication                                                                                                                                                                                                                                                     |
|                       |                |           | 15=Aborted                                                                                                                                                                                                                                                                |
|                       |                |           | 16=Dormant                                                                                                                                                                                                                                                                |
|                       |                |           | 17=Supporting entity in error                                                                                                                                                                                                                                             |
|                       |                |           | 18=Completed                                                                                                                                                                                                                                                              |
|                       |                |           | 19=Power mode                                                                                                                                                                                                                                                             |
|                       |                |           | The default value is 1 (Unknown) because the Logical Domains Manager does not provide the CPU state.                                                                                                                                                                      |
| ldomVcpuPhysBind      | Display string | Read-only | Physical binding (PID). Contains the identifier of a hardware thread (strand) that is assigned to this virtual CPU. This identifier uniquely identifies the core and the chip.                                                                                            |
| ldomVcpuPhysBindUsage | Integer        | Read-only | Indicates how much (in MHz) of the total capacity of the thread 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 thread), 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. Therefore, an auxiliary table is used to hold each memory segment instead of a display string. A display string has a 255-character limit.

### Virtual Memory Table (`lDomVmemTable`)

`lDomVmemTable` describes the properties of virtual memory that domains use.

**TABLE 20-10** Virtual Memory Table (`lDomVmemTable`)

| Name                                | Data Type | Access         | Description                                                                                                              |
|-------------------------------------|-----------|----------------|--------------------------------------------------------------------------------------------------------------------------|
| <code>lDomVmemLdomIndex</code>      | Integer   | Read-only      | Integer that is used as an index into <code>lDomTable</code> that represents the domain that contains the virtual memory |
| <code>lDomVmemIndex</code>          | Integer   | Not accessible | Integer that is used to index the virtual memory in this table                                                           |
| <code>lDomVmemNumberOfBlocks</code> | 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 20-11** Virtual Memory Physical Binding Table (`lDomVmemPhysBindTable`)

| Name                                   | Data Type | Access    | Description                                                                                                                        |
|----------------------------------------|-----------|-----------|------------------------------------------------------------------------------------------------------------------------------------|
| <code>lDomVmemPhysBindLdomIndex</code> | Integer   | Read-only | Integer that is used as an index into <code>lDomTable</code> that represents the domain that contains the physical memory segments |

| Name             | Data Type      | Access    | Description                                                                                                                            |
|------------------|----------------|-----------|----------------------------------------------------------------------------------------------------------------------------------------|
| lDomVmemPhysBind | Display string | Read-only | List of physical memory that is mapped to this virtual memory block in the following format:<br>{physical-address, real-address, size} |

## 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 20-12** 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              |
| 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 ldm add-vds 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 20-13** Virtual Disk Service Device Table (`ldomVdsdevTable`)

| Name                              | Data Type      | Access         | Description                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------------------|----------------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ldomVdsdevVdsIndex</code>   | Integer        | Read-only      | Integer that is used to index into <code>ldomVdsTable</code> that represents the virtual disk service that contains the virtual disk device                                                                                                                                                                                                                       |
| <code>ldomVdsdevIndex</code>      | Integer        | Not accessible | Integer that is used to index the virtual disk service device in this table                                                                                                                                                                                                                                                                                       |
| <code>ldomVdsdevVolumeName</code> | 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. |
| <code>ldomVdsdevDevPath</code>    | 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.                                                                                                                                                                                                                             |
| <code>ldomVdsdevOptions</code>    | 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>                                                                                                                                                                                                                                             |
| <code>ldomVdsdevMPGroup</code>    | Display string | Read-only      | Multipath group name for the disk device                                                                                                                                                                                                                                                                                                                          |

### Virtual Disk Table (`ldomVdiskTable`)

`ldomVdiskTable` describes the virtual disks for all domains.

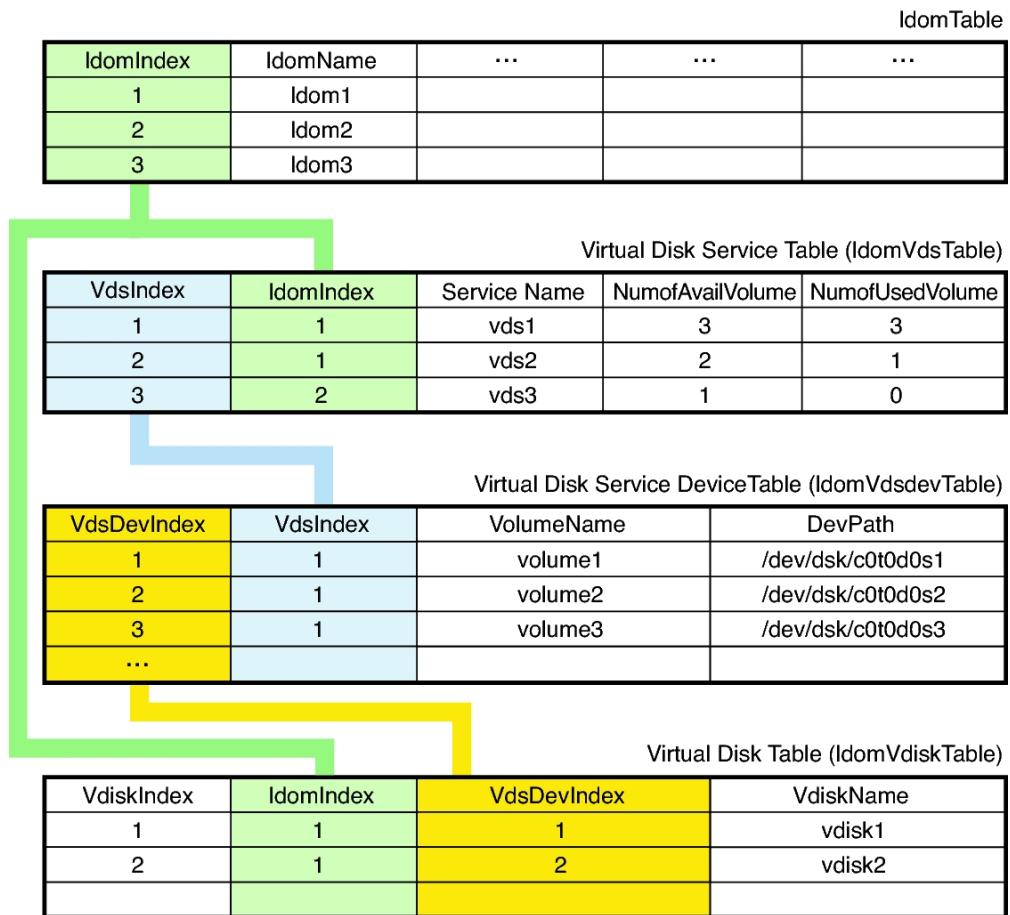
**TABLE 20-14** Virtual Disk Table (`ldomVdiskTable`)

| Name                              | Data Type      | Access         | Description                                                                                                                   |
|-----------------------------------|----------------|----------------|-------------------------------------------------------------------------------------------------------------------------------|
| <code>ldomVdiskLdomIndex</code>   | Integer        | Read-only      | Integer that is used as an index into <code>ldomTable</code> that represents the domain that contains the virtual disk device |
| <code>ldomVdiskVdsDevIndex</code> | Integer        | Read-only      | Integer that is used to index into <code>ldomVdsdevTable</code> that represents the virtual disk service device               |
| <code>ldomVdiskIndex</code>       | Integer        | Not accessible | Integer that is used to index the virtual disk in this table                                                                  |
| <code>ldomVdiskName</code>        | 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.     |
| <code>ldomVdiskTimeout</code>     | Integer        | Read-only      | Timeout, in seconds, for establishing a connection between a virtual disk client and a virtual disk server                    |
| <code>ldomVdiskID</code>          | 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 20-3** 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 20-15** Virtual Switch Service Table (`lDomVswTable`)

| Name                              | Data Type      | Access         | Description                                                                                                                             |
|-----------------------------------|----------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| <code>lDomVswLdomIndex</code>     | Integer        | Read-only      | Integer that is used as an index into <code>lDomTable</code> that represents the domain that contains the virtual switch service        |
| <code>lDomVswIndex</code>         | Integer        | Not accessible | Integer that is used to index the virtual switch device in this table                                                                   |
| <code>lDomVswServiceName</code>   | Display string | Read-only      | Virtual switch service name                                                                                                             |
| <code>lDomVswMacAddress</code>    | Display string | Read-only      | MAC address used by the virtual switch                                                                                                  |
| <code>lDomVswPhysDevPath</code>   | 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. |
| <code>lDomVswMode</code>          | Display string | Read-only      | Value is <code>mode=sc</code> for running cluster nodes                                                                                 |
| <code>lDomVswDefaultVlanID</code> | Display string | Read-only      | Default VLAN ID for the virtual switch                                                                                                  |
| <code>lDomVswPortVlanID</code>    | Display string | Read-only      | Port VLAN ID for the virtual switch                                                                                                     |
| <code>lDomVswVlanID</code>        | Display string | Read-only      | VLAN ID for the virtual switch                                                                                                          |
| <code>lDomVswLinkprop</code>      | Display string | Read-only      | Value is <code>linkprop=phys-state</code> to report the link status based on the physical network device                                |
| <code>lDomVswMtu</code>           | Integer        | Read-only      | Maximum transmission unit (MTU) for a virtual switch device                                                                             |
| <code>lDomVswID</code>            | Display string | Read-only      | Identifier of the virtual switch device                                                                                                 |
| <code>lDomVswInterVnetLink</code> | Display string | Read-only      | State of LDC channel assignment for inter-vnet communications. Value is either <code>on</code> or <code>off</code> .                    |

## Virtual Network Device Table (`lDomVnetTable`)

`lDomVnetTable` describes the virtual network devices for all domains.

**TABLE 20-16** Virtual Network Device Table (`lDomVnetTable`)

| Name                               | Data Type      | Access         | Description                                                                                                                                       |
|------------------------------------|----------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>lDomVnetLdomIndex</code>     | Integer        | Read-only      | Integer that is used as an index into <code>lDomTable</code> that represents the domain that contains the virtual network device                  |
| <code>lDomVnetVswIndex</code>      | Integer        | Read-only      | Integer that is used to index into the virtual switch service table                                                                               |
| <code>lDomVnetIndex</code>         | Integer        | Not accessible | Integer that is used to index the virtual network device in this table                                                                            |
| <code>lDomVnetDevName</code>       | Display string | Read-only      | Virtual network device name. The property value is the <code>net-dev</code> property specified by the <code>ldm add-vnet</code> command.          |
| <code>lDomVnetDevMacAddress</code> | Display string | Read-only      | MAC address for this network device. The property value is the <code>mac-addr</code> property specified by the <code>ldm add-vnet</code> command. |
| <code>lDomVnetMode</code>          | Display string | Read-only      | Value is <code>mode=hybrid</code> to use NIU hybrid I/O on the virtual network device                                                             |
| <code>lDomVnetPortVlanID</code>    | Display string | Read-only      | Port VLAN ID for the virtual network device                                                                                                       |
| <code>lDomVnetVlanID</code>        | Display string | Read-only      | VLAN ID for the virtual network device                                                                                                            |
| <code>lDomVnetLinkprop</code>      | Display string | Read-only      | Value is <code>linkprop=phys-state</code> to report the link status based on the physical network device                                          |
| <code>lDomVnetMtu</code>           | Integer        | Read-only      | MTU for a virtual network device                                                                                                                  |
| <code>lDomVnetID</code>            | 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 20-17** Virtual Console Concentrator Table (`ldomVccTable`)

| Name                              | Data Type      | Access         | Description                                                                                                                                                                                                |
|-----------------------------------|----------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ldomVccLdomIndex</code>     | Integer        | Read-only      | Integer that is used as an index into <code>ldomTable</code> that represents the domain that contains the virtual console service                                                                          |
| <code>ldomVccIndex</code>         | Integer        | Not accessible | Integer that is used to index the virtual console concentrator in this table                                                                                                                               |
| <code>ldomVccName</code>          | 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.                                                                            |
| <code>ldomVccPortRangeLow</code>  | 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 <i>port-range</i> specified by the <code>ldm add-vcc</code> command.  |
| <code>ldomVccPortRangeHigh</code> | 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 <i>port-range</i> 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. This table also shows whether console logging is enabled or disabled on each domain.

**TABLE 20-18** Virtual Console Group Table (`ldomVconsTable`)

| Name                             | Data Type      | Access         | Description                                                                                                                                                                              |
|----------------------------------|----------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ldomVconsIndex</code>      | Integer        | Not accessible | Integer that is used to index a virtual group in this table                                                                                                                              |
| <code>ldomVconsGroupName</code>  | Display string | Read-only      | Group name to which to attach the virtual console. The property value is the <i>group</i> specified by the <code>ldm set-vcons</code> command.                                           |
| <code>ldomVconsLog</code>        | Display string | Read-only      | Console logging status. The property value is the <i>on</i> or <i>off</i> as specified by the <code>ldm set-vcons</code> command.                                                        |
| <code>ldomVconsPortNumber</code> | Integer        | Read-only      | When a group contains more than one domain, this property shows the console logging status of the domain that has most recently been modified by the <code>ldm set-vcons</code> command. |
|                                  |                |                | 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`)

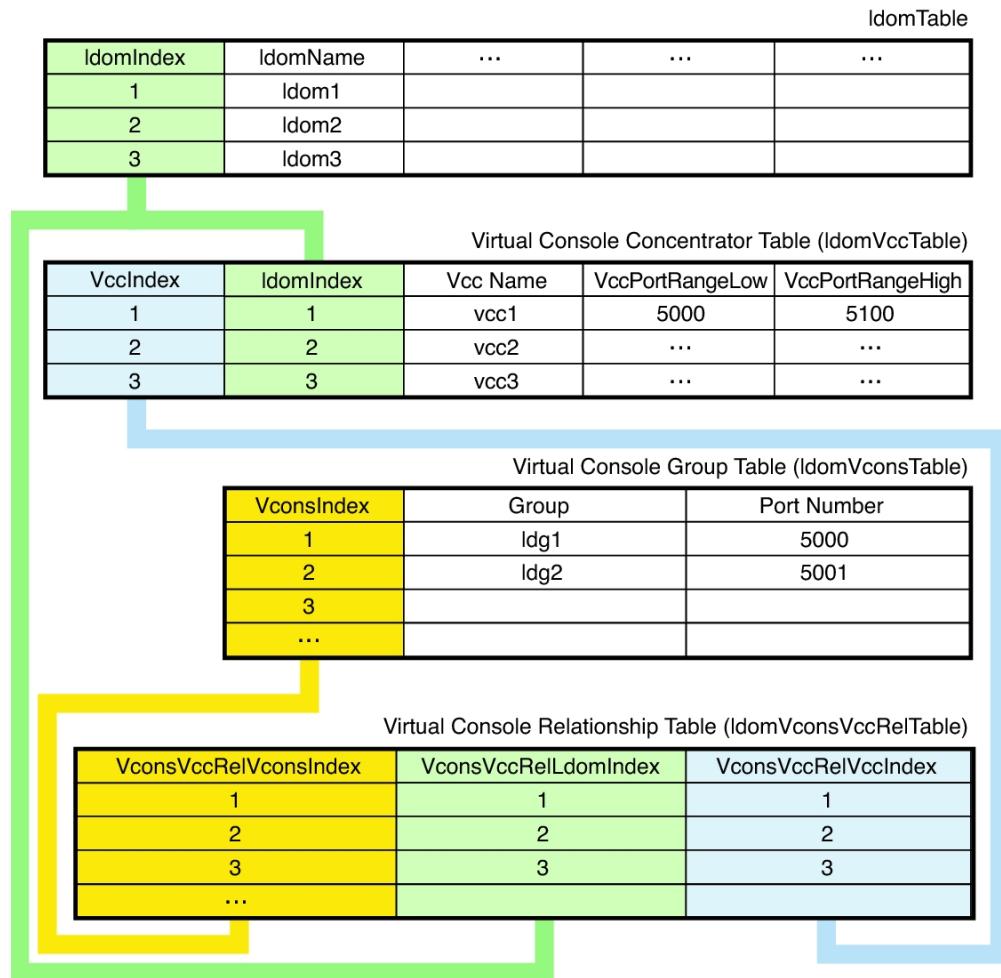
`lDomVconsVccRelTable` contains index values to show the inter-table relationships among a domain, a virtual console concentrator, and console groups.

**TABLE 20-19** Virtual Console Relationship Table (`lDomVconsVccRelTable`)

| Name                                   | Data Type | Access    | Description                                                         |
|----------------------------------------|-----------|-----------|---------------------------------------------------------------------|
| <code>lDomVconsVccRelVconsIndex</code> | Integer   | Read-only | Value of <code>lDomVconsIndex</code> in <code>lDomVconsTable</code> |
| <code>lDomVconsVccRelLdomIndex</code>  | Integer   | Read-only | Value of <code>lDomIndex</code> in <code>lDomTable</code>           |
| <code>lDomVconsVccRelVccIndex</code>   | Integer   | Read-only | Value of <code>lDomVccIndex</code> in <code>lDomVccTable</code>     |

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 20-4** Relationship Among Virtual Console Tables and the Domain Table

## Cryptographic Units Table (ldomCryptoTable)

**ldomCryptoTable** describes the cryptographic units that all domains use. A cryptographic unit is sometimes referred to as a modular arithmetic unit (MAU).

**TABLE 20-20** Cryptographic Units Table (`lDomCryptoTable`)

| Name                             | Data Type      | Access         | Description                                                                                                                  |
|----------------------------------|----------------|----------------|------------------------------------------------------------------------------------------------------------------------------|
| <code>lDomCryptoLdomIndex</code> | Integer        | Read-only      | Integer that is used as an index into <code>lDomTable</code> that represents the domain that contains the cryptographic unit |
| <code>lDomCryptoIndex</code>     | Integer        | Not accessible | Integer that is used to index the cryptographic unit in this table                                                           |
| <code>lDomCryptoCpuSet</code>    | 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 20-21** I/O Bus Table (`lDomIOBusTable`)

| Name                            | Data Type      | Access         | Description                                                                                                       |
|---------------------------------|----------------|----------------|-------------------------------------------------------------------------------------------------------------------|
| <code>lDomIOBusLdomIndex</code> | Integer        | Read-only      | Integer that is used as an index into <code>lDomTable</code> that represents the domain that contains the I/O bus |
| <code>lDomIOBusIndex</code>     | Integer        | Not accessible | Integer that is used to index the I/O bus in this table                                                           |
| <code>lDomIOBusName</code>      | Display string | Read-only      | Physical I/O device name                                                                                          |
| <code>lDomIOBusPath</code>      | Display string | Read-only      | Physical I/O device path                                                                                          |
| <code>lDomIOBusOptions</code>   | 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 20-22** Core Table (`lDomCoreTable`)

| Name                           | Data Type      | Access         | Description                                                                                                    |
|--------------------------------|----------------|----------------|----------------------------------------------------------------------------------------------------------------|
| <code>lDomCoreLdomIndex</code> | Integer        | Read-only      | Integer that is used as an index into <code>lDomTable</code> that represents the domain that contains the core |
| <code>lDomCoreIndex</code>     | Integer        | Not accessible | Integer that is used to index a core in this table                                                             |
| <code>lDomCoreID</code>        | Display string | Read-only      | Identifier of a core (core ID)                                                                                 |
| <code>lDomCoreCpuSet</code>    | Display string | Read-only      | List of CPUs that is mapped to the core <code>cpuset</code>                                                    |

## Scalar Variables for Domain Version Information

The Logical Domains Manager protocol supports domain versions, which consists of a major number and a minor number. The Oracle VM Server for SPARC MIB has scalar variables to describe the domain version information.

**TABLE 20-23** Scalar Variables for Domain 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 1.6
...
$ 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.

The Oracle VM Server for SPARC MIB provides the same SNMP traps for both Oracle Solaris 10 and Oracle Solaris 11. However, the `snmptrapd` daemon no longer automatically accepts all incoming traps for Oracle Solaris 11. Instead, the daemon must be configured with authorized SNMP v1 and v2c community strings, with SNMPv3 users, or both. Unauthorized traps or notifications are dropped. See the `snmptrapd.conf(4)` or `snmptrapd.conf(5)` man page.

## Using Oracle VM Server for SPARC MIB Module Traps

The Oracle Solaris 11 MIB provides the same SNMP traps as are provided by the Oracle Solaris 10 MIB. However, the net-snmp versions are different and they must be configured in different ways. In Oracle Solaris 10 MIB, snmptrapd accepts all incoming notifications and automatically logs them. In Oracle Solaris 11 MIB, access control checks are applied to incoming notifications. If snmptrapd runs without a suitable configuration file, or with equivalent access control settings, such traps are not processed. See the `snmptrapd.conf(4)` or `snmptrapd.conf(5)` man page.

### ▼ How to Send Traps

#### 1. Configure the trap.

##### ■ Oracle Solaris 10:

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)` or `snmpd.conf(5)` man page.

##### ■ Oracle Solaris 11:

Edit the `/etc/net-snmp/snmp/snmpd.conf` SNMP configuration file to add the directives to define the trap, inform version, and destination.

You must use the `pfedit` command to edit the `/etc/net-snmp/snmp/snmpd.conf` file.

```
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)` or `snmpd.conf(5)` man page.

For example, the following directives use the `public` string as the community string when sending traps and indicate that the `v1` traps are sent to the `localhost` destination:

```
trapcommunity public
trapsink localhost
```

**2. Configure access control settings by creating or editing the /usr/etc/snmp/snmptrapd.conf SNMP trapd configuration file.**

You must use the pfedit command to edit the /etc/net-snmp/snmp/snmpd.conf file.

The following example shows who is authorized to send traps (public) and how incoming traps should be processed (log, execute, net). See the snmptrapd.conf(4) or snmptrapd.conf(5) man page.

```
authCommunity log,execute,net public
```

**3. To receive SNMP trap messages, start the SNMP trap daemon utility, snmptrapd.**

**Example 20-4** 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. Update the Oracle Solaris 10 /etc/sma/snmp/snmpd.conf file or the Oracle Solaris 11 /etc/net-snmp/snmp/snmpd.conf file with the following directives:

```
trapcommunity public
trapsink localhost
trap2sink localhost
```

## ▼ How to Receive Traps

- Start the SNMP trap daemon utility.

- **Oracle Solaris 10:**

For information about the output format options, see the snmptrapd(1M) man page.

The snmptrapd utility is an SNMP application that receives and logs SNMP TRAP messages. For example, the following snmptrapd command shows that a new domain was created (ldomTrapDesc = Ldom Created) with a name of ldg2 (ldomName = ldg2).

```
/usr/sfw/sbin/snmptrapd -P -F \
"TRAP from %B on %m/%l/%y at %h:%j:%k Enterprise=%N Type=%w SubType=%q\n
with 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
```

Note that the -F option argument string is broken on to two lines for readability purposes.

- **Oracle Solaris 11:**

For information about the output format options, see the `snmptrapd(1M)` man page.

The `snmptrapd` utility is an SNMP application that receives and logs SNMP TRAP messages. For example, the following `snmptrapd` command shows that a new domain was created (`lDomTrapDesc = Ldom Created`) with a name of `ldg2` (`lDomName = ldg2`).

```
/usr/sbin/snmptrapd -f -Le -F \
"TRAP from %B on %m/%l/%y at %h:%j:%k Enterprise=%N Type=%w SubType=%q\n
with Varbinds: %v\nSecurity info:%P\n\n" localhost:162
NET-SNMP version 5.4.1
TRAP from localhost on 6/27/2012 at 12:13:48
Enterprise=SUN-LDOM-MIB::lDomMIBTraps Type=6 SubType=SUN-LDOM-MIB::lDomCreate
with Varbinds: SUN-LDOM-MIB::lDomIndexNotif = INTEGER: 3
SUN-LDOM-MIB::lDomName = STRING: ldg2 SUN-LDOM-MIB::lDomTrapDesc = STRING:
Ldom Created
Security info:TRAP, SNMP v1, community public
```

Note that the `-F` option argument string is broken on to two lines for readability purposes.

## 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 20-24** Domain Creation Trap (`lDomCreate`)

| Name                        | Data Type      | Description                       |
|-----------------------------|----------------|-----------------------------------|
| <code>lDomIndexNotif</code> | Integer        | Index into <code>lDomTable</code> |
| <code>lDomName</code>       | Display string | Name of the domain                |
| <code>lDomTrapDesc</code>   | Display string | Description of the trap           |

### Domain Destroy (`lDomDestroy`)

This trap notifies you when any domains are destroyed.

**TABLE 20-25** Domain Destroy Trap (`lDomDestroy`)

| Name                        | Data Type      | Description                       |
|-----------------------------|----------------|-----------------------------------|
| <code>lDomIndexNotif</code> | Integer        | Index into <code>lDomTable</code> |
| <code>lDomName</code>       | Display string | Name of the domain                |
| <code>lDomTrapDesc</code>   | Display string | Description of the trap           |

## Domain State Change (`lDomStateChange`)

This trap notifies you of any domain operating state changes.

**TABLE 20-26** Domain State Change Trap (`lDomStateChange`)

| Name                        | Data Type      | Description                       |
|-----------------------------|----------------|-----------------------------------|
| <code>lDomIndexNotif</code> | Integer        | Index into <code>lDomTable</code> |
| <code>lDomName</code>       | Display string | Name of the domain                |
| <code>lDomOperState</code>  | Integer        | New state of the domain           |
| <code>lDomStatePrev</code>  | Integer        | Previous state of the domain      |
| <code>lDomTrapDesc</code>   | 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 20-27** Domain Virtual CPU Change Trap (`lDomVCpuChange`)

| Name                         | Data Type      | Description                                      |
|------------------------------|----------------|--------------------------------------------------|
| <code>lDomIndexNotif</code>  | Integer        | Index into <code>lDomTable</code>                |
| <code>lDomName</code>        | Display string | Name of the domain that contains the virtual CPU |
| <code>lDomNumVCPU</code>     | Integer        | New number of virtual CPUs for the domain        |
| <code>lDomNumVCPUPrev</code> | Integer        | Previous number of virtual CPUs for the domain   |
| <code>lDomTrapDesc</code>    | 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 20-28** Domain Virtual Memory Change Trap (`lDomVMemChange`)

| Name                         | Data Type      | Description                                                                                                                        |
|------------------------------|----------------|------------------------------------------------------------------------------------------------------------------------------------|
| <code>lDomIndexNotif</code>  | Integer        | Index into <code>lDomTable</code>                                                                                                  |
| <code>lDomName</code>        | Display string | Name of the domain that contains the virtual memory                                                                                |
| <code>lDomMemSize</code>     | Integer        | Amount of virtual memory for the domain                                                                                            |
| <code>lDomMemSizePrev</code> | Integer        | Previous amount of virtual memory for the domain                                                                                   |
| <code>lDomMemUnit</code>     | Integer        | Memory unit for virtual memory, which is one of the following:<br><br>■ 1 is KB<br>■ 2 is MB<br>■ 3 is GB<br>■ 4 is bytes          |
|                              |                | If not specified, the unit value is bytes.                                                                                         |
| <code>lDomMemUnitPrev</code> | Integer        | Memory unit for previous virtual memory, which is one of the following:<br><br>■ 1 is KB<br>■ 2 is MB<br>■ 3 is GB<br>■ 4 is bytes |
|                              |                | If not specified, the unit value is bytes.                                                                                         |
| <code>lDomTrapDesc</code>    | Display string | Description of the trap                                                                                                            |

## Virtual Disk Service Change (`lDomVdsChange`)

This trap notifies you when a domain's virtual disk service changes.

**TABLE 20-29** Domain Virtual Disk Service Change Trap (`lDomVdsChange`)

| Name                            | Data Type      | Description                                                                                                                                |
|---------------------------------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <code>lDomIndexNotif</code>     | Integer        | Index into <code>lDomTable</code>                                                                                                          |
| <code>lDomName</code>           | Display string | Name of the domain that contains the virtual disk service                                                                                  |
| <code>lDomVdsServiceName</code> | Display string | Name of the virtual disk service that has changed                                                                                          |
| <code>lDomChangeFlag</code>     | Integer        | Indicates one of the following changes that occurred to the virtual disk service:<br><br>■ 1 is Added<br>■ 2 is Modified<br>■ 3 is Removed |

| Name         | Data Type      | Description             |
|--------------|----------------|-------------------------|
| ldomTrapDesc | Display string | Description of the trap |

## Virtual Disk Change (ldomVdiskChange)

This trap notifies you when a domain's virtual disk changes.

**TABLE 20-30** 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 20-31** 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                                                                                                                                         |
| 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 20-32** Virtual Network Change Trap (**lDomVnetChange**)

| Name            | Data Type      | Description                                                                                                                                                                                       |
|-----------------|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lDomIndexNotif  | Integer        | Index into <code>lDomTable</code>                                                                                                                                                                 |
| 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 20-33** Virtual Console Concentrator Change Trap (**lDomVccChange**)

| Name           | Data Type      | Description                                                                                                                                                                                               |
|----------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lDomIndexNotif | Integer        | Index into <code>lDomTable</code>                                                                                                                                                                         |
| lDomName       | Display string | Name of the domain that contains the virtual console concentrator                                                                                                                                         |
| 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 20-34** Virtual Console Group Change Trap (`lDomVconsChange`)

| Name                            | Data Type      | Description                                                                                                                                                                                    |
|---------------------------------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>lDomIndexNotif</code>     | Integer        | Index into <code>lDomTable</code>                                                                                                                                                              |
| <code>lDomName</code>           | Display string | Name of the domain that contains the virtual console group                                                                                                                                     |
| <code>lDomVconsGroupName</code> | Display string | Name of the virtual console group that has changed                                                                                                                                             |
| <code>lDomChangeFlag</code>     | 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> |
| <code>lDomTrapDesc</code>       | 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 20-1](#).

### ▼ How to 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: 8
SUN-LDOM-MIB::lDomNumVCpu.2 = INTEGER: 4
```

```
SUN-LDOM-MIB::ldomMemSize.1 = INTEGER: 3360
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: 1
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: 5f8817d4-5d2e-6f7d-c4af-91b5b34b5723
SUN-LDOM-MIB::ldomUUID.2 = STRING: 11284146-87ca-4877-8d80-cd0f60d5ec26
SUN-LDOM-MIB::ldomMacAddress.1 = STRING: 00:14:4f:46:47:d6
SUN-LDOM-MIB::ldomMacAddress.2 = STRING: 00:14:4f:f8:d5:6c
SUN-LDOM-MIB::ldomHostID.1 = STRING: 0x844647d6
SUN-LDOM-MIB::ldomHostID.2 = STRING: 0x84f8d56c
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:
SUN-LDOM-MIB::ldomExtMapInSpace.1 = STRING: off
SUN-LDOM-MIB::ldomExtMapInSpace.2 = STRING: off
SUN-LDOM-MIB::ldomWholeCore.1 = INTEGER: 0
SUN-LDOM-MIB::ldomWholeCore.2 = INTEGER: 0
SUN-LDOM-MIB::ldomCpuArch.1 = STRING: native
SUN-LDOM-MIB::ldomCpuArch.2 = STRING: native
SUN-LDOM-MIB::ldomShutdownGroup.1 = INTEGER: 0
SUN-LDOM-MIB::ldomShutdownGroup.2 = INTEGER: 15
SUN-LDOM-MIB::ldomPerfCounters.1 = STRING: htstrand
SUN-LDOM-MIB::ldomPerfCounters.2 = STRING: global,htstrand
```

**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 by using one of the following commands:**

- `# ldm list domain-name`
- `# snmpget -v version -c community-string hostname SUN-LDOM-MIB::ldomOperState.n`

**Example 20-5 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
```

You can also use the `snmpget` command to retrieve the `LdomMibTest_1` domain's state instead of using the `ldm list` 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.

## ▼ How to 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: 8
SUN-LDOM-MIB::ldomNumVCpu.2 = INTEGER: 4
SUN-LDOM-MIB::ldomMemSize.1 = INTEGER: 3360
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: 1
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: 5f8817d4-5d2e-6f7d-c4af-91b5b34b5723
SUN-LDOM-MIB::ldomUUID.2 = STRING: 11284146-87ca-4877-8d80-cd0f60d5ec26
SUN-LDOM-MIB::ldomMacAddress.1 = STRING: 00:14:4f:46:47:d6
SUN-LDOM-MIB::ldomMacAddress.2 = STRING: 00:14:4f:f8:d5:6c
SUN-LDOM-MIB::ldomHostID.1 = STRING: 0x844647d6
SUN-LDOM-MIB::ldomHostID.2 = STRING: 0x84f8d56c
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:
SUN-LDOM-MIB::ldomExtMapinSpace.1 = STRING: off
SUN-LDOM-MIB::ldomExtMapinSpace.2 = STRING: off
SUN-LDOM-MIB::ldomWholeCore.1 = INTEGER: 0
SUN-LDOM-MIB::ldomWholeCore.2 = INTEGER: 0
SUN-LDOM-MIB::ldomCpuArch.1 = STRING: native
SUN-LDOM-MIB::ldomCpuArch.2 = STRING: native
SUN-LDOM-MIB::ldomShutdownGroup.1 = INTEGER: 0
```

```
SUN-LDOM-MIB::ldomShutdownGroup.2 = INTEGER: 15
SUN-LDOM-MIB::ldomPerfCounters.1 = STRING: htstrand
SUN-LDOM-MIB::ldomPerfCounters.2 = STRING: global,htstrand
```

**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 by using one of the following commands:**

- `# ldm list domain-name`
- `# snmpget -v version -c community-string hostname SUN-LDOM-MIB::ldomOperState.n`

**Example 20-6** 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

---

This chapter provides information about discovering the Logical Domains Manager running on systems on a subnet.

### Discovering Systems Running the Logical Domains Manager

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.

### 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;
```

## ▼ How to 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 429.

### 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 information:

- 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 433
- “[XML Protocol](#)” on page 434
- “[Event Messages](#)” on page 439
- “[Logical Domains Manager Actions](#)” on page 444
- “[Logical Domains Manager Resources and Properties](#)” on page 446
- “[XML Schemas](#)” on page 464

### 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 disable 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, Oracle VM Server for SPARC-defined XML messages are sent next. The two general types of XML messages are:

- Request and response messages, which 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, which 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 Oracle VM Server for SPARC has two different formats:

- A format for sending commands into the Logical Domains Manager
- A 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. The following example shows the structure of a basic XML command.

### EXAMPLE 22-1 Format of a Single Command Operating on a Single Object

```
<LDM_interface version="1.3">
 <cmd>
 <action>Place command here</action>
 <options>Place options for certain commands here</options>
 <arguments>Place arguments for certain commands here</arguments>
 <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 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>
```

### **<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 22-1](#).

## 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 <options> tag, which is used for options and flags that are associated with some commands. The following commands use options:

- The `ldm remove-domain` command can use the `-a` option.
- The `ldm bind-domain` command can use the `-f` option.
- The `ldm add-vdsdev` command can use the `-f` option.
- The `ldm cancel-operation` command can use the `migration` or `reconf` option.
- The `ldm add-spconfig` command can use the `-r autosave-name` option.
- The `ldm remove-spconfig` command can use the `-r` option.
- The `ldm list-spconfig` command can use the `-r [autosave-name]` option.
- The `ldm stop-domain` command can use the following tags to set the command arguments:
  - `<force>` represents the `-f` option.
  - `<halt>` represents the `-h` option.
  - `<message>` represents the `-m` option.
  - `<quick>` represents the `-q` option.
  - `<reboot>` represents the `-r` option.
  - `<timeout>` represents the `-t` option.

Note that the tags must not have any content value. However, the `-t` and `-m` options must have a non-null value, for example, `<timeout>10</timeout>` or `<message>Shutting down now</message>`.

The following XML example fragment shows how to pass a reboot request with a reboot message to the `ldm stop-domain` command:

```
<action>stop-domain</action>
<arguments>
 <reboot/>
 <message>my reboot message</message>
</arguments>
```

## 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 Oracle VM Server for SPARC) and <Content> and <Section> sections.

For Oracle VM Server for SPARC, 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 need to identify only 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 need to provide a <Section> section describing the resources needed for the command, placed outside 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.

Two additional OVF types enable the use of the OVF specification's schema to properly define all types of objects:

- <gprop:GenericProperty> tag
- <Binding> tag

The <gprop:GenericProperty> tag handles 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 ldm list-bindings command 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. The following example shows the structure of a response to a basic XML request.

### EXAMPLE 22-2 Format of a Response to a Single Command Operating on a Single Object

```
<LDM_interface version="1.3">
```

```
<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 whether 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 that the command ran against caused a failure.

## Object Response

Finally, each `<data>` section in a `<cmd>` section also has a `<response>` section. This section shows whether 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 437”. 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 441 for complete details.

## Registration and Unregistration

Use an `<LDM_interface>` message to register for events. See “[The `<LDM\_interface>` Tag](#) on page 435”. The `<action>` tag details the type of event for which to register or unregister and the `<data>` section is left empty.

**EXAMPLE 22-3** Example Event Registration Request Message

```
<LDM_interface version="1.3">
<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 22-4** Example Event Registration Response Message

```
<LDM_interface version="1.3">
<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.

## **<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 22-5** Example `<LDM_event>` Notification

```
<LDM_event version='1.1'>
<cmd>
 <action>Event command here</action>
 <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

You can subscribe to the following event types:

- Domain events
- Hardware events
- Progress events
- Resource events

All the events correspond to `ldm` subcommands.

## Domain Events

Domain events describe which actions can be performed directly to a domain. The following domain events can be specified in the `<action>` tag in the `<LDM_event>` message:

- `add-domain`
- `bind-domain`
- `domain-reset`
- `migrate-domain`
- `panic-domain`
- `remove-domain`
- `start-domain`
- `stop-domain`
- `unbind-domain`

These events always contain *only* a `<Content>` tag in the OVF `<data>` section that describes the domain to which the event happened. To register for the domain events, send an

<LDM\_interface> message with the <action> tag set to reg-domain-events. To unregister for these events, send 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 are those to the service processor (SP) when you add, remove, or set 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. To unregister for these events, send 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. To unregister for these events, send 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 providing a service name in the OVF <data> section.

The following events can be specified in the <action> tag in the <LDM\_event> message:

- add-vdiskserverdevice
- remove-vdiskserverdevice
- set-vdiskserverdevice
- remove-vdiskserver
- set-vconscon
- remove-vconscon
- set-vswitch
- remove-vswitch
- remove-vdpcs

The following resource events always contain *only* the <Content> tag in the OVF <data> section that describes the domain to which the event happened:

- 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:

- add-domain
- add-io
- add-mau
- add-memory
- add-spconfig

- add-variable
- add-vconscon
- add-vcpu
- add-vdisk
- add-vdiskserver
- add-vdiskserverdevice
- add-vdpcc
- add-vdpcs
- add-vnet
- add-vswitch
- bind-domain
- cancel-operation
- list-bindings
- list-constraints
- list-devices
- list-domain
- list-services
- list-spconfig
- list-variable
- migrate-domain
- reg-all-events
- reg-domain-events
- reg-hardware-events
- reg-resource-events
- remove-domain
- remove-io
- remove-mau
- remove-memory
- remove-reconf
- remove-spconfig
- remove-variable
- remove-vconscon
- remove-vcpu
- remove-vdisk
- remove-vdiskserver
- remove-vdiskserverdevice

- remove-vdpcc
- remove-vdpcs
- remove-vnet
- remove-vswitch
- set-domain
- set-mau
- set-memory
- set-spconfig
- set-variable
- set-vconscon
- set-vconsole
- set-vcpu
- set-vnet
- set-vswitch
- start-domain
- stop-domain
- unbind-domain
- unreg-all-events
- unreg-domain-events
- unreg-hardware-events
- unreg-resource-events

## Logical Domains Manager Resources and Properties

This section provides examples of 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 except domain migration output. See “[Domain Migration](#)” on page 464. Each resource is defined in a <Section> OVF section and is specified by a <rasd:OtherResourceType> tag.

### Domain Information (**ldom\_info**) Resource

The following example shows the optional properties of the **ldom\_info** resource:

**EXAMPLE 22-6** Example ldom\_info XML Output

The following example shows values specified for several ldom\_info properties such as uuid, hostid, and Address.

```
<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:GenericProperty key="hostid">83d8baf6</gprop:GenericProperty>
 <gprop:GenericProperty key="master">plum</gprop:GenericProperty>
 <gprop:GenericProperty key="failure-policy">reset</gprop:GenericProperty>
 <gprop:GenericProperty key="extended-mapin-space">on</gprop:GenericProperty>
 <gprop:GenericProperty key="progress">45%</gprop:GenericProperty>
 <gprop:GenericProperty key="status">ongoing</gprop:GenericProperty>
 <gprop:GenericProperty key="source">system1</gprop:GenericProperty>
 <gprop:GenericProperty key="rc-add-policy"></gprop:GenericProperty>
 <gprop:GenericProperty key="perf-counters">global</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:GenericProperty key="extended-mapin-space"> tag, which specifies whether extended mapin space is enabled (on) or disabled (off) for the domain. The default value is off.
- <gprop:GenericProperty key="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:GenericProperty key="hostid"> tag, which specifies the host ID to be assigned to the domain.
- <gprop:GenericProperty key="master"> tag, which specifies up to four comma-separated master domain names.

- <gprop:GenericProperty key="progress"> tag, which specifies the percentage of progress made by the command.
- <gprop:GenericProperty key="source"> tag, which specifies the machine reporting on the progress of the command.
- <gprop:GenericProperty key="status"> tag, which specifies the status of the command (done, failed, or ongoing).
- <gprop:GenericProperty key="rc-add-policy"> tag, which specifies whether to enable or disable the direct I/O and SR-IOV I/O virtualization operations on any root complex that might be added to the specified domain. Valid values are iov and no value (rc-add-policy=).
- <gprop:GenericProperty key="perf-counters"> tag, which specifies the performance register sets to access (global, htstrand, strand).

If the platform does not have the performance access capability, the perf-counters property value is ignored.

## CPU (cpu) Resource

The equivalent of the add-vcpu, set-vcpu, and remove-vcpu XML request actions is to set the value of the <gpropGenericProperty key="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.

A cpu resource is always contained within a <Content> section.

### EXAMPLE 22-7 Example cpu XML

The following example shows the XML request equivalent for the ldm add-vcpu -c 1 ldg1 command.

Oracle VM Server for SPARC 3.2 is the last software release to include the -c option with the ldm add-vcpu, ldm set-vcpu, and ldm remove-vcpu commands.

```
<?xml version="1.0"?>
<LDM_interface version="1.3"
 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="ldgl">
<Section xsi:type="ovf:VirtualHardwareSection_Type">
<Item>
<rasd:OtherResourceType>cpu</rasd:OtherResourceType>
<rasd:AllocationUnits>8</rasd:AllocationUnits>
<gprop:GenericProperty key="wcore">1</gprop:GenericProperty>
</Item>
</Section>
</Content>
</Envelope>
</data>
</cmd>
</LDM_interface>

```

**EXAMPLE 22-8** cpu XML Section Output from the `ldm list-bindings` Command

The following example shows the XML output for the <cpu> section by using the `ldm list-bindings` command.

```

<?xml version="1.0"?>
<LDM_interface
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:ovf=".schemas/envelope"
 xmlns:rasd=".schemas/CIM_ResourceAllocationSettingData"
 xmlns:vssd=".schemas/CIM_VirtualSystemSettingData"
 xmlns:gprop=".schemas/GenericProperty"
 xmlns:bind=".schemas/Binding"
 version="1.3"
 xsi:noNamespaceSchemaLocation=".schemas/combined-v3.xsd">
<cmd>
<action>list-bindings</action>
<data version="3.0">
<Envelope>
<References/>
<Content xsi:type="ovf:VirtualSystem_Type" ovf:id="primary">
<Section xsi:type="ovf:ResourceAllocationSection_Type">
<Item>
<rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
<uuid>1e04cdbd-472a-e8b9-ba4c-d3eee86e7725</uuid>
<rasd:Address>00:21:28:f5:11:6a</rasd:Address>
<gprop:GenericProperty key="hostid">0x8486632a</gprop:GenericProperty>
<failure-policy>fff</failure-policy>
<wcore>0</wcore>
<extended-mapin-space>0</extended-mapin-space>
<threading>8</threading>
<cpu-arch>native</cpu-arch>

```

```
<rc-add-policy/>
<gprop:GenericProperty key="state">active</gprop:GenericProperty>
</Item>
</Section>
<Section xsi:type="ovf:VirtualHardwareSection_Type">
<Item>
<rasd:OtherResourceType>cpu</rasd:OtherResourceType>
<rasd:AllocationUnits>8</rasd:AllocationUnits>
<bind:Binding>
<Item>
<rasd:OtherResourceType>cpu</rasd:OtherResourceType>
<gprop:GenericProperty key="vid">0</gprop:GenericProperty>
<gprop:GenericProperty key="pid">0</gprop:GenericProperty>
<gprop:GenericProperty key="cid">0</gprop:GenericProperty>
<gprop:GenericProperty key="strand_percent">100</gprop:GenericProperty>
<gprop:GenericProperty key="util_percent">1.1%</gprop:GenericProperty>
<gprop:GenericProperty key="normalized_utilization">0.1%</
gprop:GenericProperty>
</Item>
</Section>
</Content>
</Envelope>
</data>
</cmd>
</LDM_interface>
```

**EXAMPLE 22-9** cpu XML Section Output from the ldm list-domain Command

The following example shows the XML output for the <cpu> section by using the ldm list-domain command.

```
<?xml version="1.0"?>
<LDM_interface
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:ovf=".schemas/envelope"
 xmlns:rasd=".schemas/CIM_ResourceAllocationSettingData"
 xmlns:vssd=".schemas/CIM_VirtualSystemSettingData"
 xmlns:gprop=".schemas/GenericProperty"
 xmlns:bind=".schemas/Binding"
 version="1.3"
 xsi:noNamespaceSchemaLocation=".schemas/combined-v3.xsd">
<cmd>
<action>list-domain</action>
<data version="3.0">
<Envelope>
<References/>
<Content xsi:type="ovf:VirtualSystem_Type" ovf:id="primary">
<Section xsi:type="ovf:ResourceAllocationSection_Type">
<Item>
<rasd:OtherResourceType>lDom_info</rasd:OtherResourceType>
<gprop:GenericProperty key="state">active</gprop:GenericProperty>
<gprop:GenericProperty key="flags">-n-cv-</gprop:GenericProperty>
<gprop:GenericProperty key="utilization">0.7%</gprop:GenericProperty>
```

```
<gprop:GenericProperty key="uptime">3h</gprop:GenericProperty>
<gprop:GenericProperty key="normalized_utilization">0.1%</gprop:GenericProperty>
</Item>
</Section>
</Content>
</Envelope>
</data>
</cmd>
</LDM_interface>
```

## MAU (mau) Resource

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.

---

**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**      Example mau XML  
**22-10**

```
<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>
```

## Memory (memory) Resource

A memory resource is always contained within a <Content> section. The only property is the <rasd:AllocationUnits> tag, which signifies the amount of memory.

**EXAMPLE**      Example memory XML  
**22-11**

```
<Envelope>
 <References/>
 <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>memory</rasd:OtherResourceType>
 <rasd:AllocationUnits>4G</rasd:AllocationUnits>
 </Item>
 </Section>
 </Content>
</Envelope>
```

## Virtual Disk Server (vds) Resource

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*, which contains the name of the vds resource being described.

**EXAMPLE**      Example vds XML  
**22-12**

```
<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>
```

## Virtual Disk Server Volume (vds\_volume) Resource

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

**EXAMPLE**      Example vds\_volume XML  
**22-13**

```
<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>
 <gprop:GenericProperty key="vol_opts">ro</gprop:GenericProperty>
 <gprop:GenericProperty key="mpgroup">mpgroup-name</gprop:GenericProperty>
 </Item>
 </Section>
</Envelope>
```

## Disk (disk) Resource

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**. The timeout ensures that a connection to any **vds** is established within the specified amount of time.

**EXAMPLE**      Example disk XML  
**22-14**

```
<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>
```

## Virtual Switch (vsw) Resource

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 (the default setting).
- `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 `lsm` 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.

**EXAMPLE**      Example vsw XML  
**22-15**

```
<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>
```

## Network (network) Resource

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 (the default setting).
- **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.

---

**Note** - The NIU Hybrid I/O feature is deprecated in favor of SR-IOV. Oracle VM Server for SPARC 3.2 is the last software release to include this feature.

---

**EXAMPLE**      Example network XML  
22-16

```
<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>
```

## Virtual Console Concentrator (vcc) Resource

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

**EXAMPLE**      Example vcc XML  
22-17

```
<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>
```

```
</Item>
</Section>
</Content>
</Envelope>
```

## Variable (var) Resource

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

**EXAMPLE 22-18** 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>
```

## Physical I/O Device (physio\_device) Resource

A physio\_device resource is always contained within a <Content> section. This resource can be modified by using the add-io, set-io, remove-io, create-vf, destroy-vf, and set-domain subcommands.

**EXAMPLE 22-19** Example physio\_device XML

The following examples show how to perform actions on virtual functions, physical functions, and root complexes.

- The following XML example fragment shows how to use the ldm add-io command to add the /SYS/MB/NET0/IOVNET.PF0.VF0 virtual function to the ldg1 domain.

```
<LDM_interface version="1.3">
 <cmd>
 <action>add-io</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>physio_device</rasd:OtherResourceType>
 <gprop:GenericProperty key="name">
 /SYS/MB/NET0/IOVNET.PF0.VF0</gprop:GenericProperty>
 </Item>
 </Section>
 </Content>
 </Envelope>
 </data>
 </cmd>
 </LDM_interface>
```

- The following XML example fragment shows how to use the `ldm set-io` command to set the `iov_bus_enable_iov` property value to `on` for the `pci_1` root complex.

```
<LDM_interface version="1.3">
 <cmd>
 <action>set-io</action>
 <data version="3.0">
 <Envelope>
 <References/>
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
 <gprop:GenericProperty key="name">pci_1</gprop:GenericProperty>
 <gprop:GenericProperty key="iov_bus_enable_iov">
 on</gprop:GenericProperty>
 </Item>
 </Section>
 </Envelope>
 </data>
 </cmd>
 </LDM_interface>
```

- The following XML example fragment shows how to use the `ldm set-io` command to set the `unicast-slots` property value to 6 for the `/SYS/MB/NET0/IOVNET.PF1` physical function.

```
<LDM_interface version="1.3">
```

```

<cmd>
 <action>set-io</action>
 <data version="3.0">
 <Envelope>
 <References/>
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
 <gprop:GenericProperty key="name">
 /SYS/MB/NET0/IOVNET.PF1</gprop:GenericProperty>
 <gprop:GenericProperty key="unicast-slots">6</gprop:GenericProperty>
 </Item>
 </Section>
 </Envelope>
 </data>
</cmd>
</LDM_interface>

```

- The following XML example fragment shows how to use the ldm create-vf command to create the /SYS/MB/NET0/IOVNET.PF1.VF0 virtual function with the following property values.
  - unicast-slots=6
  - pvid=3
  - mtu=1600

```

<LDM_interface version="1.3">
 <cmd>
 <action>create-vf</action>
 <data version="3.0">
 <Envelope>
 <References/>
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>vf_device</rasd:OtherResourceType>
 <gprop:GenericProperty key="iov_pf_name">
 /SYS/MB/NET0/IOVNET.PF1</gprop:GenericProperty>
 <gprop:GenericProperty key="unicast-slots">6</gprop:GenericProperty>
 <gprop:GenericProperty key="pvid">3</gprop:GenericProperty>
 <gprop:GenericProperty key="mtu">1600</gprop:GenericProperty>
 </Item>
 </Section>
 </Envelope>
 </data>
 </cmd>
</LDM_interface>

```

- The following XML example fragment shows how to use the `ldm create-vf` command to create the number of virtual functions specified by the `iov_pf_repeat_count_str` value (3) with the `/SYS/MB/NET0/IOVNET.PF1` physical function. You cannot specify other property values when you create multiple virtual functions with the `iov_pf_repeat_count_str` property.

```
<LDM_interface version="1.3">
 <cmd>
 <action>create-vf</action>
 <data version="3.0">
 <Envelope>
 <References/>
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>vf_device</rasd:OtherResourceType>
 <gprop:GenericProperty key="iov_pf_name">
 /SYS/MB/NET0/IOVNET.PF1
 </gprop:GenericProperty>
 <gprop:GenericProperty key="iov_pf_repeat_count_str">
 3
 </gprop:GenericProperty>
 </Item>
 </Section>
 </Envelope>
 </data>
 </cmd>
</LDM_interface>
```

## SP Configuration (`spconfig`) Resource

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.

**EXAMPLE 22-20** 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>
```

## DRM Policy Configuration (policy) Resource

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 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

**EXAMPLE**      Example policy XML  
**22-21**

```

<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>
```

## Virtual Data Plane Channel Service (vdpcs) Resource

This resource is of interest only 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.

**EXAMPLE**      Example vdpcs XML  
**22-22**

```
<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">dg1-vdpcs</gprop:GenericProperty>
</Item>
</Section>
</Content>
</Envelope>
```

## Virtual Data Plane Channel Client (vdgcc) Resource

This resource is of interest only 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:

- vdgcc\_name – Name of the virtual data plane channel client (vdgcc)
- service\_name – Name of the virtual data plane channel service vdpcs to which this vdgcc is to be bound

**EXAMPLE**      Example vdpc XML  
**22-23**

```
<Envelope>
<References/>
<Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
 <Section xsi:type="ovf:VirtualHardwareSection_Type">
 <Item>
 <rasd:OtherResourceType>vdpc</rasd:OtherResourceType>
 <gprop:GenericProperty key="vdpc_name">vdpc</gprop:GenericProperty>
 <gprop:GenericProperty
 key="service_name">ldg1-vdpc</gprop:GenericProperty>
 </Item>
 </Section>
</Content>
</Envelope>
```

## Console (console) Resource

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
- enable-log – Enable or disable virtual console logging for this console

**EXAMPLE**      Example console XML  
**22-24**

```
<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>
 <gprop:GenericProperty key="enable-log">on</gprop:GenericProperty>
 </Item>
 </Section>
</Content>
</Envelope>
```

## Domain Migration

This example shows what is contained in the <data> section for a `ldm migrate-domain` command.

- 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, which must be SASL 64-bit encoded.
  - `password` is the password to use for logging into the target machine, which 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.

---

**EXAMPLE**      Example `migrate-domain` <data> Section  
**22-25**

```
<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>
 <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>
```

## XML Schemas

The XML schemas that are used by the Logical Domains Manager are located in the `/opt/SUNWldm/bin/schemas` directory. The file names are as follows:

- `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 glossary 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>auditing</b>	Tracking changes to the system and identifying the user who made the changes.
<b>auditreduce</b>	Command to merge and select audit records from audit trail files (see the <code>auditreduce(1M)</code> man page).
<b>authorization</b>	A way in which to determine who has permission to perform tasks and access data by using Oracle Solaris OS rights.

## B

<b>bge</b>	Broadcom Gigabit Ethernet driver on Broadcom BCM57xx devices.
<b>BSM</b>	Basic Security Module.
<b>bsmconv</b>	Command to enable the BSM (see the <code>bsmconv(1M)</code> man page).
<b>bsmunconv</b>	Command to disable the BSM (see the <code>bsmunconv(1M)</code> man page).

## C

<b>CMT</b>	Chip multithreading.
------------	----------------------

<b>compliance</b>	Determining whether a system's configuration is in compliance with a predefined security profile.
<b>configuration</b>	Name of the 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>CWQ</b>	Control Word Queue; cryptographic unit.

## D

<b>DHCP</b>	Dynamic Host Configuration Protocol.
<b>DIO</b>	Direct I/O.
<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>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 drd(1M) man page).
<b>DRM</b>	Dynamic resource management.
<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 <code>fmd(1M)</code> man page).
<b>fmthard</b>	Command to populate label on hard disks (see the <code>fmthard(1M)</code> man page).
<b>format</b>	Disk partitioning and maintenance utility (see the <code>format(1M)</code> man page).

**G**

<b>Gb</b>	Gigabit.
<b>GLDv3</b>	Generic LAN Driver version 3.
<b>guest domain</b>	Uses services from the I/O and service domains and is managed by the control domain.

**H**

<b>hardening</b>	Modifying Oracle Solaris OS configuration to improve security.
<b>hypervisor</b>	Firmware layer interposed between the operating system and the hardware layer.

**I**

<b>I/O</b>	Input/output devices, such as internal disks and PCIe controllers and their attached adapters and devices.
<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, a dedicated system of hardware and supporting software that enables you to manage your server independently of the operating system.

<b>ioctl</b>	Input/output control call.
<b>IPMP</b>	Internet Protocol Network Multipathing.

## K

<b>k aio</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>l dm</b>	Logical Domains Manager utility (see the <a href="#">l dm(1M)</a> man page).
<b>l dmd</b>	Logical Domains Manager daemon.
<b>lo fi</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>Logical Domains Manager</b>	A CLI to create and manage logical domains and allocate resources to domains.

## M

<b>MAC</b>	Media access control address, which Logical Domains Manager 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>	Command to create and delete replicas of the Solaris Volume Manager metadevice state database (see the <code>metadb(1M)</code> man page).
<b>metaset</b>	Command to configure disk sets (see the <code>metaset(1M)</code> man page).
<b>mhd</b>	Command to perform multihost disk control operations (see the <code>mhd(7i)</code> 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>ndpsldcc</b>	Netra DPS Logical Domain Channel Client. <i>See also</i> <code>vdpc</code> .
<b>ndpsldcs</b>	Netra DPS Logical Domain Channel Service. <i>See also</i> <code>vdpc</code> .
<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.
------------	--------------------------------------------------------------------------------------------------

**OVF** Open Virtualization Format.

## P

<b>P2V</b>	Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool.
<b>PA</b>	Physical address.
<b>PCI</b>	Peripheral component interconnect bus.
<b>PCI-X</b>	PCI Extended bus.
<b>PCIe</b>	PCI EXPRESS bus.
<b>pcpu</b>	Physical CPU.
<b>physical domain</b>	The scope of resources that are managed by a single Oracle VM Server for SPARC instance. A physical domain might be a complete physical system as is the case of supported SPARC T-Series platforms. Or, it might be either the entire system or a subset of the system as is the case of supported SPARC M-Series platforms.
<b>physical function</b>	A PCI function that supports the SR-IOV capabilities as defined in the SR-IOV specification. A physical function contains the SR-IOV capability structure and is used to manage the SR-IOV functionality. Physical functions are fully featured PCIe functions that can be discovered, managed, and manipulated like any other PCIe device. Physical functions have full configuration resources, and can be used to configure or control the PCIe device.
<b>physio</b>	Physical input/output.
<b>PICL</b>	Platform Information and Control Library.
<b>picld</b>	PICL daemon (see the <code>picld(1M)</code> man page).
<b>PM</b>	Power management of virtual CPUs and memory.
<b>praudit</b>	Command to print contents of an audit trail file (see the <code>praudit(1M)</code> man page).
<b>PRI</b>	Priority.

## R

<b>RA</b>	Real address.
<b>RAID</b>	Redundant Array of Inexpensive Disks, which enables you to combine independent disks into a logical unit.

**RPC** Remote Procedure Call.

## S

**SASL** Simple Authentication and Security Layer.

**SAX** Simple API for XML parser, which traverses an XML document. The SAX parser is event-based and used mostly for streaming data.

**service domain** Logical domain that provides devices, such as virtual switches, virtual console connectors, and virtual disk servers, to other logical domains.

**service processor (SP)** The SP, also known as the system controller (SC), monitors and runs the physical machine.

**SMA** System Management Agent.

**SMF** Service Management Facility.

**SMI** Structure of Management Information, which defines and groups managed objects for use by a MIB.

**SNMP** Simple Network Management Protocol.

**SR-IOV** Single root I/O virtualization.

**SSH** Secure Shell.

**ssh** Secure Shell command (see the `ssh(1)` man page).

**sshd** Secure Shell daemon (see the `sshd(1M)` man page).

**SunVTS** Sun Validation Test Suite.

**svcadm** Manipulates service instances (see the `svcadm(1M)` man page).

**system controller (SC)** Also see service processor.

## T

**TLS** Transport Layer Security.

**U**

<b>UDP</b>	User Diagram Protocol.
<b>unicast</b>	Network communication that takes place between a single sender and a single receiver.
<b>uscsi</b>	User SCSI command interface (see the <b>uscsi(7I)</b> man page).
<b>UTP</b>	Unshielded twisted pair.

**V**

<b>var</b>	Variable.
<b>VBSC</b>	Virtual blade system controller.
<b>vcc, vconscon</b>	Virtual console concentrator service with a specific port range to assign to guest domains.
<b>vcons, vconsole</b>	Virtual console for accessing system-level messages. A connection is achieved by connecting to the <b>vconscon</b> service in the control domain at a specific port.
<b>vcpu</b>	Virtual central processing unit. Each core in a server is represented as a virtual CPU.
<b>vdc</b>	Virtual disk client.
<b>vdisk</b>	A virtual disk is a generic block device associated with different types of physical devices, volumes, or files.
<b>vdpcc</b>	Virtual data plane channel client in a Netra DPS environment.
<b>vdpcs</b>	Virtual data plane channel service in a Netra DPS environment.
<b>vds, vdiskserver</b>	Virtual disk server enables you to import virtual disks into a logical domain.
<b>vdsdev, vdiskserverdevice</b>	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.
<b>virtual function</b>	A PCI function that is associated with a physical function. A virtual function is a lightweight PCIe function that shares one or more physical resources with the physical function and with other virtual functions that are associated with the same physical function. Virtual functions are only permitted to have configuration resources for its own behavior.
<b>vldc</b>	Virtual logical domain channel service.
<b>vldcc</b>	Virtual logical domain channel client.

---

<b>vnet</b>	Virtual network device implements a virtual Ethernet device and communicates with other vnet devices in the system by using the virtual network switch (vswitch).
<b>VNIC</b>	Virtual network interface card, which is a virtual instantiation of a physical network device that can be created from the physical network device and assigned to a zone.
<b>vNTS</b>	Virtual network terminal service.
<b>vntsd</b>	Oracle Solaris 10 OS virtual network terminal server daemon for domain consoles (see the <a href="#">vntsd(1M)</a> man page).
<b>volfs</b>	Volume Management file system (see the <a href="#">volfs(7FS)</a> man page).
<b>vsw, vswitch</b>	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.

**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

---

## Numbers and Symbols

/etc/system file  
    updating, 334

## A

accessing  
    Fibre Channel virtual functions from a guest domain, 148  
adding  
    Ethernet virtual functions to an I/O domain, 110  
    Fibre Channel virtual functions to an I/O domain, 146  
    InfiniBand virtual functions to a root domain, 128  
    InfiniBand virtual functions to an I/O domain, 125  
    memory to a domain, 296  
    unaligned memory, 299  
    virtual disks, 170  
adjusting  
    interrupt limit, 352  
allocating  
    CPU resources, 280  
    resources, 279  
    world-wide names for Fibre Channel virtual functions, 137  
applying  
    max-cores constraint, 281  
    whole-core constraint, 281  
assigning  
    endpoint device to an I/O domain, 73  
    MAC addresses, 219  
    automatically, 219  
    manually, 219  
    master domain, 341  
PCIe buses to a root domain, 63  
PCIe endpoint device, 66

physical resources to domains, 291  
rights profiles, 25, 26  
roles, 25  
roles to users, 27  
VLANs to a virtual switch and virtual network device, 234  
audit records  
    reviewing, 37, 39  
    rotating, 39  
auditing  
    disabling, 37, 38  
    enabling, 37, 37  
    reviewing records, 37, 39  
    rotating audit records, 39  
authorization  
    ldm subcommands, 29  
Automatic  
    network configuration profile, 203  
autorecovery policy for domain configurations, 319, 320

## B

back ends, 180  
    *See also* virtual disk backend exporting  
backward compatibility  
    exporting volumes, 180  
blacklisting  
    Fault Management Architecture (FMA), 325  
    faulty hardware resources, 325  
boot disk image  
    cloning, 194  
booting an I/O domain by using an Ethernet SR-IOV virtual functions, 113  
breaks  
    Oracle Solaris OS, 337

bus assignment  
    dynamic, 65  
    static, 64

**C**

cancel-reconf subcommand, 278

CD images  
    exporting, 186

CD or DVD image  
    exporting from service domain to guest domain, 186  
    exporting multiple times, 187

changing  
    changes to PCIe hardware, 80

checking  
    domain configuration, 284

CLI *See* command-line interface

cloning  
    boot disk image, 194

coherency link scaling, 380

combining  
    consoles into a single group, 335

command-line interface, 19

commands  
    `ldmconfig(1M)`, 375, 376

configuration  
    selecting to boot, 21

configuration assistant  
    `ldmconfig`, 22

configuring  
    control domain, 46  
    control domain with CPU whole cores, 288  
    domain dependencies, 339  
    domain with CPU whole cores, 285  
    existing domain with CPU whole cores, 287  
    IPMP in a service domain, 228  
    IPMP in an Oracle VM Server for SPARC environment, 226  
    jumbo frames, 248  
    NAT on Oracle Solaris 10 system, 223  
    NAT on Oracle Solaris 11 system, 224  
    Oracle VM Server for SPARC MIB, 392  
    Oracle VM Server for SPARC MIB software, 392  
    performance register access, 313  
    physical link status updates, 229

SSL certificates for migration, 256, 256

system with hard partitions, 284

virtual disk multipathing, 183

virtual network devices into an IPMP group, 226

virtual switch and service domain for NAT and routing, 222

virtual switch as the primary interface, 49

virtual switch to provide external connectivity to an Oracle Solaris 10 domain, 224

virtual switch to provide external connectivity to an Oracle Solaris 11 domain, 225

ZFS pool in a service domain, 191

connecting  
    to a guest console over the network, 335

console groups  
    using, 335

consoles  
    combining into a single group, 335  
    logging, 40

control domain, 18  
    configuring, 46  
    decreasing memory, 297  
    memory reconfiguration, 296  
    rebooting, 48, 338

control domain CPU and memory resources  
    decreasing, 47

CPU allocation, 280

CPU clock cycle skip, 380

CPU core disable, 380

CPU DR, 283, 289

CPU dynamic resource management, 290

CPU dynamic voltage and frequency scaling (DVFS), 380

CPU mapping, 343

CPU power management, 290

CPU resources  
    allocating, 280

CPU whole cores  
    configuring a domain with, 285  
    configuring an existing domain with, 287  
    configuring the control domain with, 288  
    creating a domain with, 286  
    rebinding system with, 290  
    rebooting system with, 290

creating  
    default services on the control domain, 44

- 
- disk image snapshot, 193, 193  
 disk image snapshot of an unconfigured system, 194  
 domain with CPU whole cores, 286  
 Ethernet virtual functions, 102, 102  
 Fibre Channel virtual functions, 138  
 guest domains, 51  
 I/O domain Ethernet virtual functions, 116  
 InfiniBand virtual functions, 120  
 PVLANS, 238  
 roles, 27  
 root domain from entire PCIe bus, 66  
 snmpv3 user, 395  
 VNICs on Ethernet virtual functions, 115
- D**
- daemons  
 drd, 278  
 ldmd, 19  
 vntsd, 21  
 decreasing  
     control domain CPU and memory resources, 47  
     memory on the control domain, 297  
 default services on the control domain  
     creating, 44  
 DefaultFixed  
     network configuration profile, 203  
 delayed reconfiguration, 278, 297  
 delegating administrative privileges  
     rights profiles, 25  
 dependency cycles, 342  
 destroying, 102  
     *See also* removing  
     Ethernet virtual functions, 102, 106  
     Fibre Channel virtual functions, 142  
     InfiniBand virtual functions, 123  
 determining  
     domain configurations, 344  
     GLDv3 compliance of a network device (Oracle Solaris 10), 222  
 device-specific properties  
     Ethernet SR-IOV, 114  
 direct I/O (DIO)  
     limitations, 76
- managing devices on non-primary root domains, 162  
 planning, 77  
 requirements, 75  
 disabling  
     auditing, 37, 38  
     NIU hybrid I/O, 246  
 disk images  
     creating a snapshot, 193, 193  
     creating snapshot of an unconfigured system, 194  
     storing by using a ZFS file, 192  
     storing by using a ZFS volume, 192  
     storing with ZFS, 191  
 disk slice *See* physical disk slice  
 domain configurations  
     autorecovery policy, 320  
     autorecovery policy for, 319  
     checking, 284  
     degraded, 329  
     determining, 344  
     managing, 317  
     persistent, 21  
     restoring, 318, 322  
     restoring from an XML file with `ldm add-domain`, 324  
     restoring from an XML file with `ldm init-system`, 322  
     restoring with autosave, 318  
     saving, 318, 321  
 domain console  
     controlling access to, 29  
 domain listing  
     parseable, 342  
 domain migration restrictions, 259  
 domain migrations, 262  
     active, 263  
     bound or inactive domain, 270  
     canceling in progress, 272  
     delayed reconfiguration for an active domain, 269  
     failure message, 274  
     from OpenBoot PROM or in kernel debugger, 270  
     monitoring progress, 272  
     non-interactive, 274  
     obtaining status, 274  
     operation, 254  
     operations on other domains, 270

- performing a dry run, 263
- performing non-interactive, 263
- recovering from failed, 273
- requirements for CPUs, 264
- requirements for cryptograph units, 269
- requirements for memory, 266
- requirements for NIU hybrid I/O, 269
- requirements for PCIe endpoint devices, 268, 271
- requirements for physical I/O devices, 267
- requirements for SR-IOV virtual functions, 268, 271
- requirements for virtual I/O devices, 267, 271
- security, 255
- software compatibility, 255
- when active domain has power management elastic policy in effect, 269
- domain resource pool
  - Oracle VM Server for SPARC MIB, 404
- domain resources
  - listing, 307
- domain scalar variables
  - Oracle VM Server for SPARC MIB, 404
- domains
  - configuring a failure policy for dependencies, 340
  - configuring dependencies, 339
  - definition, 16
  - dependencies, 339
  - dependency cycles, 342
  - marked as degraded, 330
  - migrating, 253
  - monitoring with Oracle VM Server for SPARC MIB, 397
  - provisioning by using a clone, 193
  - roles, 18, 18
  - service, 20
  - starting, 425
  - stopping, 427
  - stopping a heavily loaded, 336
  - types of, 18, 18, 19, 19
- DR *See* dynamic reconfiguration (DR)
- DVD images
  - exporting, 186
- dynamic path selection, 184
- dynamic reconfiguration (DR), 277, 297
  - CPUs, 283, 289
  - memory, 295
  - partial memory requests, 296
  - dynamic reconfiguration daemon (drd), 278
  - dynamic resource management, 283
  - CPUs, 290
  - using, 304
- E
  - enabling
    - auditing, 37, 37
    - I/O virtualization, 98
    - I/O virtualization for a PCIe bus, 160
    - NIU hybrid I/O, 246
    - power management observability module, 382
    - recovery mode, 330
    - virtual network terminal server daemon (vntsd), 50
  - environment variables
    - setting, 397
  - errors
    - troubleshooting through CPU and memory address mapping, 343
  - Ethernet SR-IOV
    - device-specific properties, 101, 114
    - limitations, 101
    - network configuration, 113
    - planning, 101
    - requirements, 100
  - evacuated I/O resources, 331
  - exporting
    - back ends
      - comparison, 180
    - back ends, summary, 180
    - CD images, 186
    - CD or DVD image from service domain to guest domain, 186
    - CD or DVD image multiple times, 187
    - disk slice
      - directly, 181
      - indirectly, 181
    - DVD images, 186
    - file as a full disk, 177
    - file or volume as a full disk, 177
    - file or volume as a single-slice disk, 179
    - files, 177
    - files and volumes as virtual disks

- 
- guidelines, 181
  - lofi**, 181
  - ISO image from service domain to guest domain, 188
  - ISO images, 186
  - physical disk as a virtual disk, 175
  - physical disk slice as a virtual disk, 176
  - slice 2, 177
  - virtual disk back end, 170
  - volumes, 177
    - backward compatibility, 180
  - ZFS volume as a full disk, 178
  - ZFS volume as a single-slice disk, 179
  
  - F**
  - failure policy
    - configuring for a domain dependency, 340
  - fault and recovery information
    - providing, 390
  - Fault Management Architecture (FMA)
    - blacklisting, 325
  - faulty hardware resources
    - blacklisting, 325
    - recovering domains with, 327
    - unconfiguring, 325
  - Fibre Channel world-wide names for virtual functions
    - allocating, 137
  - FIPS 140-2 mode for migration, 257
  - FMA *See* Fault Management Architecture (FMA)
  - format
    - virtual disks, 190
  
  - G**
  - GLD compliance (Oracle Solaris 10)
    - of network device, 222
  - guest console
    - connecting to over the network, 335
  - guest domains, 19
    - creating, 51
    - migrating, 273
    - migrating and renaming, 274
    - starting, 51
  - guidelines
    - exporting files and volumes as virtual disks, 181
    - I/O domain creation, 62
  
  - H**
  - hard partitions
    - configuring systems with, 284
  - hardware errors
    - troubleshooting, 325
  - hypervisor
    - definition, 16
    - Logical Domains Manager and, 16
  
  - I**
  - I/O domains, 61, 73, 89
    - booting by assigning an SR-IOV virtual function, 113
    - creating by assigning an endpoint device, 73
    - creating by assigning an SR-IOV virtual function, 89
    - creation guidelines, 62
    - migration limitations, 61
    - PCIe bus, 61
      - using PCIe SR-IOV virtual functions, 89
  - I/O resources
    - marking as evacuated, 331
  - I/O virtualization
    - enabling, 98
    - enabling for a PCIe bus, 160
  - identifying
    - InfiniBand functions, 131
  - InfiniBand SR-IOV
    - requirements, 120
  - installing
    - guest domain when install server is in a VLAN, 235
    - `ldmp2v`, 362
    - Oracle Solaris OS from a DVD, 55
    - Oracle Solaris OS from an ISO file, 57
    - Oracle Solaris OS in guest domains, 54
    - Oracle VM Server for SPARC MIB, 392
    - Oracle VM Server for SPARC MIB software, 392
    - Oracle VM Server for SPARC P2V tool, 364
    - using JumpStart (Oracle Solaris 10), 58
  - inter-vnet LDC channels, 208
  - inter-vnet-links

PVLANs, 236

interrupt limit

adjusting, 352

IPMP

configuring in a service domain, 228

configuring in an Oracle VM Server for SPARC

environment, 226

configuring virtual network devices into a group, 226

ISO images

exporting, 186

exporting from service domain to guest domain, 188

## J

jumbo frames

compatibility with jumbo-unaware versions of the

Oracle Solaris 10 vnet and vsw drivers, 251

configuring, 248

JumpStart

using to install the Oracle Solaris 10 OS on a guest domain, 58

## L

LDC *See* logical domain channels (LDC)

`ldmconfig(1M)` command, 375, 376

`ldmd` *See* Logical Domains Manager daemon

`ldmp2v`

backend devices, 361

collection phase, 360, 365

conversion phase, 361, 368

installing, 362

limitations, 363

Oracle VM Server for SPARC P2V conversion tool, 359

Oracle VM Server for SPARC P2V tool, 21

preparation phase, 360, 366

prerequisites, 363

`ldmp2v(1M)` command, 360

limitations

direct I/O, 76

Ethernet SR-IOV, 101

Fibre Channel virtual functions, 135

non-primary root domains, 159

physical network bandwidth, 213

SR-IOV, 95

link aggregation

using with a virtual switch, 246

link-based IPMP

using, 228

listing

domain resources, 307

InfiniBand virtual functions, 130

PVLAN information, 239

resource constraints, 312

resources as machine-readable output, 307

loading

Oracle VM Server for SPARC MIB module, 392

Oracle VM Server for SPARC MIB module in to

Oracle Solaris SNMP agent, 393

`lofi`

exporting files and volumes as virtual disks, 181

logical domain channels (LDC), 18

inter-vnet, 208

logical domain channels (LDCs), 347

Logical Domains Manager, 16, 18

daemon (`ldmd`), 19

discovery mechanism, 429

Oracle VM Server for SPARC MIB and, 390

XML schema used with, 433

## M

MAC addresses

assigned to domains, 220

assigning, 219

assigning automatically, 219

assigning manually, 219

automatic assignment algorithm, 220

detecting duplicates, 221

machine-readable output

listing resources, 307

managing

direct I/O devices on non-primary root

domains, 162

domain configurations, 317

Oracle VM Server for SPARC MIB security, 395

physical resources on the control domain, 294

resource groups, 303

- SR-IOV virtual functions on non-primary root domains, 163  
 virtual disks, 169
- mapping CPU and memory addresses troubleshooting, 343
- master domain assigning, 341
- max-cores constraint applying, 281
- maximizing virtual network performance, 204, 205
- memory adding to a domain, 296  
 adding unaligned, 299  
 alignment, 297  
   for active domains, 298  
 alignment for bound domains, 298  
 alignment for inactive domains, 298  
 decreasing on the control domain, 297  
 mapping, 344  
 removing from a domain, 296  
 setting sizes for a domain, 302
- memory DR *See* memory dynamic reconfiguration (DR)
- memory dynamic reconfiguration operations on active domains, 300  
 operations on bound domains, 301  
 partial requests, 296
- memory dynamic reconfiguration (DR), 295
- memory reconfiguration control domain, 296
- memory size requirements, 55
- migrating domains, 253  
 guest domain, 273  
 guest domain and renaming, 274  
 using SSL certificates, 273
- migration non-interactive, 274
- migration limitations I/O domain, 61  
 PVLANs, 236
- missing hardware resources recovering domains with, 327
- modifying domain configuration autorecovery policy, 320
- Ethernet SR-IOV virtual function properties, 109  
 Fibre Channel virtual function properties, 145  
 virtual disk options, 171  
 virtual disk timeout option, 171
- monitoring domains with Oracle VM Server for SPARC MIB, 397
- multipathing *See* virtual disk multipathing
- N**
- NAT configuring on Oracle Solaris 10 system, 223  
 configuring on Oracle Solaris 11 system, 224  
 configuring virtual switch and service domain, 222
- network booting I/O domain by using an Ethernet SR-IOV virtual functions, 113
- network configuration Ethernet SR-IOV, 113
- network configuration profile Automatic, 203  
 DefaultFixed, 203
- network device configurations viewing, 210
- network device statistics viewing, 210
- network devices network bandwidth limit, setting, 213  
 using, 222
- network interface name, 216
- NIU hybrid I/O disabling, 246  
 enabling, 246  
 using, 242
- non-interactive domain migration, 274
- non-physically bound resources removing, 293
- non-primary root domain restrictions, 158
- non-primary root domains, 157  
 assigning a PCIe endpoint device, 157  
 assigning a PCIe SR-IOV virtual function, 157  
 limitations, 159
- managing direct I/O devices, 162  
 managing SR-IOV virtual functions, 163

overview, 157

## O

obtaining

domain migration status, 274

Oracle Solaris 10 networking, 200

Oracle Solaris 11 networking, 202

Oracle Solaris 11 networking-specific feature differences, 252

Oracle Solaris OS

breaks, 337

installing on a guest domain, 54

from a DVD, 55

from an ISO file, 57

network interface name (Oracle Solaris 10)

finding, 218

network interface name (Oracle Solaris 11)

finding, 217

operating with Oracle VM Server for SPARC, 337

Oracle Solaris SNMP agent

loading Oracle VM Server for SPARC MIB module in to, 393

Oracle VM Server for SPARC

troubleshooting, 22

using with the service processor, 338

Oracle VM Server for SPARC Management

Information Base (MIB), 387 *See* Oracle VM Server for SPARC MIB

Oracle VM Server for SPARC MIB, 22, 387, 387

configuring, 392

domain resource pool, 404

domain scalar variables, 404

for Oracle VM Server for SPARC, 22

installing, 392

Logical Domains Manager and, 390

object tree, 391

overview, 387

querying, 398

related products and features, 388

software components, 388

starting domains, 425

stopping domains, 425

system management agent, 389

virtual console tables, 412

virtual disk tables, 408

virtual memory tables, 407

virtual network tables, 411

virtual network terminal service (vNTS), 412

XML-based control interface

parsing, 390

Oracle VM Server for SPARC MIB module

loading, 392

loading in to Oracle Solaris SNMP agent, 393

Oracle VM Server for SPARC MIB module traps

receiving, 418

sending, 418

Oracle VM Server for SPARC MIB object

retrieving an

`snmpget`, 398

Oracle VM Server for SPARC MIB object values

retrieving

`snmpable`, 400

`snmpwalk`, 398

Oracle VM Server for SPARC MIB security

managing, 395

Oracle VM Server for SPARC MIB software

configuring, 392

installing, 392

removing, 392, 394

Oracle VM Server for SPARC MIB tables

core table (`lDomCoreTable`), 416

cryptographic units table (`lDomCryptoTable`), 415

domain policy table (`lDomPolicyTable`), 403

domain table (`lDomTable`), 400

environment variables table

(`lDomEnvVarsTable`), 402

I/O bus table (`lDomIOBusTable`), 416

scalar variables for CPU resource pool, 404

scalar variables for cryptographic resource

pool, 405

scalar variables for domain version

information, 417

scalar variables for I/O bus resource pool, 405

service processor configuration table

(`lDomSPConfigTable`), 403

virtual console concentrator table

(`lDomVccTable`), 412

virtual console group table (`lDomVconsTable`), 413

virtual console relationship table

(`lDomVconsVccRelTable`), 414

- 
- virtual CPU table (`lDomVcpuTable`), 405
  - virtual disk service device table  
(`lDomVdsdevTable`), 408
  - virtual disk service table (`lDomVdsTable`), 408
  - virtual disk table (`lDomVdiskTable`), 409
  - virtual memory physical binding table  
(`lDomVmemPhysBindTable`), 407
  - virtual memory table (`lDomVmemTable`), 407
  - virtual network device table (`lDomVnetTable`), 412
  - virtual switch service device table  
(`lDomVswTable`), 411
  - Oracle VM Server for SPARC MIB traps, 420
    - domain creation (`lDomCreate`), 420
    - domain destroy (`lDomDestroy`), 420
    - domain state change (`lDomStateChange`), 421
    - receiving, 419
    - sending, 418
    - virtual console concentrator change  
(`lDomVccChange`), 424
    - virtual console group change  
(`lDomVconsChange`), 424
    - virtual CPU change (`lDomVCpuChange`), 421
    - virtual disk change (`lDomVdiskChange`), 423
    - virtual disk service change (`lDomVdsChange`), 422
    - virtual memory change (`lDomVMemChange`), 421
    - virtual network change (`lDomVnetChange`), 424
    - virtual switch change (`lDomVswChange`), 423
  - Oracle VM Server for SPARC P2V tool
    - installing, 364
    - `ldmp2v`, 21, 359
    - limitations, 363
- P**
- parseable domain listing
    - showing, 342
    - viewing, 342
  - parsing
    - XML-based control interface
      - Oracle VM Server for SPARC MIB, 390
  - PCIe bus, 61
    - changing the hardware, 80
    - enabling I/O virtualization, 98
  - PCIe SR-IOV virtual functions *See* virtual functions
- planning for, 99
  - performance
    - maximizing for virtual networks, 204, 205
    - requirements for maximizing virtual networks, 204
  - performance register access
    - setting, 313
  - physical CPU number
    - determining the corresponding virtual CPU, 345
  - physical devices, 18, 20
  - physical disk, 175
  - physical disk LUN, 175
  - physical disk slice, 176
  - physical disk slices
    - exporting as a virtual disk, 176
  - physical disks
    - exporting as a virtual disk, 175
  - physical link status updates
    - configuring, 229
  - physical machine, 18
  - physical network bandwidth
    - controlling used by a virtual network device, 213
    - limitations, 213
    - setting limit, 214
  - physical resources
    - assigning to domains, 291
    - managing on the control domain, 294
    - restrictions on managing, 294
  - physical-bindings constraint
    - removing, 293
  - planning
    - direct I/O (DIO), 77, 77
    - Ethernet SR-IOV, 101
    - for PCIe SR-IOV virtual functions, 99
  - port VLAN ID (PID), 233
  - power cycle
    - performing on a server, 337
  - power limit, 380
  - power management (PM), 380, 380, 380, 380, 380, 381
    - CPUs, 290
    - features, 380
    - observability module
      - enabling, 382
      - using, 303, 379
    - power-consumption data
      - viewing, 381

primary domain, 18  
private VLANs (PVLANS)  
  using, 236  
processor power-consumption data  
  viewing, 383  
properties  
  Ethernet SR-IOV device-specific, 101  
  Fibre Channel virtual function device-specific properties, 136  
provisioning  
  domain by using a clone, 193  
PVLANS  
  creating, 238  
  inter-vnet-links, 236  
  listing information, 239  
  migration limitations, 236  
  removing, 238  
  requirements, 237  
  restrictions, 236  
  updating, 238

## Q

querying  
  Oracle VM Server for SPARC MIB, 398

## R

rebinding  
  system with CPU whole cores, 290  
rebooting  
  control domain, 48  
  root domains, 78, 155  
  system with CPU whole cores, 290  
receiving  
  Oracle VM Server for SPARC MIB traps, 419  
recovering  
  domains with faulty hardware resources, 327  
  domains with missing hardware resources, 327  
  from failed domain migrations, 273  
recovery mode for domain configurations, 325  
  enabling, 330  
removing, 112  
  *See also* destroying  
    Ethernet virtual functions from an I/O domain, 112

Fibre Channel virtual functions from an I/O domain, 147  
InfiniBand virtual functions to a root domain, 129  
InfiniBand virtual functions to an I/O domain, 126  
memory from a domain, 296  
non-physically bound resources, 293  
Oracle VM Server for SPARC MIB software, 392, 394  
physical-bindings constraint, 293  
PVLANS, 238  
  virtual disks, 171  
requirements  
  direct I/O, 75  
  Ethernet SR-IOV, 100  
  Fibre Channel virtual functions, 135, 135  
  for dynamic SR-IOV, 97, 97  
  for maximizing virtual network performance, 204  
  for static SR-IOV, 96  
  InfiniBand SR-IOV, 120  
  PVLANS, 237  
  SR-IOV, 92  
resource allocation, 279  
resource configuration, 21  
resource constraints  
  listing, 312  
resource groups  
  managing, 303  
resource management  
  dynamic, 283  
resources, 17  
  *See also* virtual devices  
  allocating, 279  
  definition, 17  
  flag definitions in output, 307  
restoring  
  domain configurations, 318, 322  
    from an XML file with `ldm add-domain`, 324  
    from an XML file with `ldm init-system`, 322  
restrictions  
  PVLANS, 236  
retrieving  
  an Oracle VM Server for SPARC MIB object  
    `snmpget`, 398  
  Oracle VM Server for SPARC MIB information, 400  
  Oracle VM Server for SPARC MIB object values

---

**s**  
 snmpable, 400  
 snmpwalk, 398  
 reviewing  
   audit records, 37, 39  
 rights profiles  
   assigning, 25, 26  
 ro option  
   virtual disk back end, 173  
 roles  
   assigning, 25  
   assigning to users, 27  
   creating, 27  
   domains, 18  
 root domains, 19, 63  
   creating, 66  
   creating by assigning PCIe buses, 63  
   rebooting, 78, 155  
 rotating  
   audit records, 39  
 routing  
   configuring virtual switch and service domain, 222

**S**  
 saving  
   domain configurations, 318, 321  
 SCSI and virtual disk, 190  
 sending  
   Oracle VM Server for SPARC MIB traps, 418  
 server  
   performing power cycle on, 337  
 service domains, 18, 20  
   configuring a ZFS pool, 191  
 service processor (SP)  
   monitoring and running physical machines, 18  
   using Oracle VM Server for SPARC with, 338  
 setting  
   environment variables, 397  
   memory sizes for a domain, 302  
   physical network bandwidth limit, 214  
   power limit, 380  
 slice 2  
   exporting, 177  
 slice option  
   virtual disk back end, 174  
 SNMP traps

providing, 390  
 using, 417  
**s**nmpget  
   retrieving an  
     Oracle VM Server for SPARC MIB object, 398  
**s**nmpable  
   retrieving Oracle VM Server for SPARC MIB  
     object values, 400  
**s**nmpv3 user  
   creating, 395  
**s**nmpwalk  
   retrieving Oracle VM Server for SPARC MIB  
     object values, 398  
 Solaris power aware dispatcher (PAD), 381  
 Solaris Volume Manager  
   using, 198  
   using with virtual disks, 196  
 SR-IOV, 89  
   dynamic, 97  
   Ethernet device-specific properties, 101  
   function types, 90  
   limitations, 95  
   requirements, 92  
   requirements for dynamic, 97, 97  
   requirements for static, 96  
   static, 96  
 SR-IOV virtual functions *See* virtual functions  
 SSL certificates  
   migrating, 273  
 SSL certificates for migration  
   configuring, 256, 256  
 starting  
   domains, 425  
   domains with Oracle VM Server for SPARC  
     MIB, 425  
   guest domains, 51  
 stopping  
   domains, 427  
   domains with Oracle VM Server for SPARC  
     MIB, 425  
   heavily loaded domain, 336  
 storing  
   disk image by using a ZFS file, 192  
   disk image by using a ZFS volume, 192  
   disk images with ZFS, 191  
 SUNWldm package, 19

system controller *See* service processor (SP)

system management agent

    Oracle VM Server for SPARC MIB, 389

## T

tables *See* Oracle VM Server for SPARC MIB tables

timeout option

    virtual disks, 171

traps *See* Oracle VM Server for SPARC MIB traps

troubleshooting

    mapping CPU and memory addresses, 343

    Oracle VM Server for SPARC, 22

## U

unconfiguring

    faulty hardware resources, 325

universally unique identifiers (UUID), 345

updating

    /etc/system file, 334

    PVLANS, 238

using

    link-based IPMP, 228

utilization statistics, 308

## V

viewing

    network device configurations, 210

    network device statistics, 210

    parseable domain listing, 342

    power-consumption data, 381

    processor power-consumption data, 383

**virtinfo**

    virtual domain information, 346

virtual console tables

    Oracle VM Server for SPARC MIB, 412

virtual CPU

    determining the corresponding physical CPU number, 345

virtual device identifier, 216

virtual devices

    I/O, 20

    virtual console concentrator (vcc), 20

virtual disk client (vdc), 20

virtual disk service (vds), 20

virtual network (vnet), 20

virtual switch (vsw), 20

virtual disk tables

    Oracle VM Server for SPARC MIB, 408

virtual disks, 167

    adding, 170

    appearance, 172

    back end, 174

    back end excl option, 173

    back end exporting, 170

    back end exporting as a full disk, 172

    back end exporting as a single-slice disk, 172

    back end options, 173

    back end ro option, 173

    back end slice option, 174

    configuring multipathing, 183

    device name, 169

    disk identifier, 169

    exporting from a physical disk, 175

    exporting from a physical disk slice, 176

    format command and, 190

    managing, 169

    modifying options, 171

    modifying timeout option, 171

    multipathing, 181, 183

    removing, 171

    SCSI and, 190

    timeout, 183, 189

    using with Solaris Volume Manager, 196

    using with volume managers, 195

    using with VxVM, 197

    using with ZFS, 191, 197

virtual domain information

    API, 346

**virtinfo**, 346

virtual function

    Ethernet network booting an I/O domain by using an, 113

virtual functions, 99

    accessing Fibre Channel from a guest domain, 148

    adding Ethernet to an I/O domain, 110

    adding Fibre Channel to an I/O domain, 146

    adding InfiniBand to a root domain, 128

- adding InfiniBand to an I/O domain, 125
  - creating an I/O domain, 116
  - creating Ethernet, 102, 102
  - creating Ethernet VNICs on, 115
  - creating Fibre Channel, 138
  - creating InfiniBand, 120
  - destroying Ethernet, 102, 106
  - destroying Fibre Channel, 142
  - destroying InfiniBand, 123
  - device-specific Fibre Channel properties, 136
  - Ethernet, 100, 102
  - Fibre Channel, 134
  - Fibre Channel limitations, 135
  - Fibre Channel requirements, 135, 135
  - InfiniBand, 120
    - listing InfiniBand, 130
    - modifying Ethernet properties, 109
    - modifying Fibre Channel properties, 145
    - removing Fibre Channel from an I/O domain, 147
    - removing from an I/O domain, 112
    - removing InfiniBand to a root domain, 129
    - removing InfiniBand to an I/O domain, 126
    - using to create an I/O domain, 116
  - virtual input/output, 19
  - virtual machine, 18
  - virtual memory tables
    - Oracle VM Server for SPARC MIB, 407
  - virtual network, 200
    - maximizing performance, 204, 205
  - virtual network devices, 207
    - controlling amount of physical network bandwidth, 213
  - virtual network tables
    - Oracle VM Server for SPARC MIB, 411
  - virtual network terminal server daemon (`vntsd`), 21
    - enabling, 50
  - virtual network terminal service (`vNTS`)
    - Oracle VM Server for SPARC MIB, 412
  - virtual switch, 205
    - configuring as the primary interface, 49
    - configuring to provide external connectivity to an Oracle Solaris 10 domain, 224
    - configuring to provide external connectivity to an Oracle Solaris 11 domain, 225
  - VLAN
    - assigning to a virtual switch and virtual network device, 234
    - installing a guest domain when install server is in a VLAN, 235
    - VLAN ID (VID), 233
    - VLAN tagging
      - using, 232
    - VNICs
      - creating SR-IOV virtual functions, 115
    - volume managers
      - using with virtual disks, 195
    - VxVM
      - using, 198
      - using with virtual disks, 197
- W**
- whole-core constraint
    - applying, 281
- X**
- XML
    - <LDM\_event> messages, 440
    - actions, Logical Domains Manager, 444
    - command response, 439
    - domain migration, 464
    - Logical Domains Manager resources and properties, 446
    - object response, 439
    - overall response, 438
    - request and response messages, 434
    - request messages, 435
    - response messages, 437
    - schemas, 464
  - XML events
    - all, 444
    - domain, 441
    - hardware, 442
    - messages, 439
    - progress, 442
    - registration and unregistration, 439
    - resource, 443
    - types, 441
  - XML protocol, 434

XML resources  
  console, 463  
  cpu, 448  
  disk, 453  
  ldom\_info, 446  
  mau, 451  
  memory, 451  
  network, 455  
  physio\_device, 457  
  policy, 461  
  spconfig, 460  
  var, 457  
  vcc, 456  
  vdpc, 462  
  vdpcs, 462  
  vds, 452  
  vds\_volume, 452  
  vsw, 454  
XML schemas, 464  
  Logical Domains Manager used with, 433  
XML tags  
  <cmd>, 436  
  <data>, 437  
  <LDM\_interface>, 435  
XML transport frames, 433  
XML-based control interface  
  Oracle VM Server for SPARC MIB  
    parsing, 390, 390  
XMPP  
  local connections, 434  
  server, 434

## Z

ZFS  
  storing disk images with, 191  
  using with virtual disks, 197  
  virtual disks and, 191  
ZFS file  
  storing a disk image by using a, 192  
ZFS pool  
  configuring in a service domain, 191  
ZFS volume  
  exporting as a full disk, 178