**Oracle® VM Server for SPARC 2.0
Administration Guide**

ORACLE®

110426@25097

Contents

Preface

The *Oracle VM Server for SPARC 2.0 Administration Guide* provides detailed information and procedures that describe the overview, security considerations, installation, configuration, modification, and execution of common tasks for the Oracle VM Server for SPARC 2.0 software on supported servers, blades, and server modules. See "Supported Platforms" in *Oracle VM Server for SPARC 2.0 Release Notes*.

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

Related Documentation

The following table shows the documentation that is available for the Oracle VM Server for SPARC 2.0 release. These documents are available in HTML and PDF formats unless indicated.

TABLE P–1   Related Documentation

Application	Title	Part Number
Oracle VM Server for SPARC 2.0 Software	*Oracle VM Server for SPARC 2.0 Administration Guide*	821-1485
	Oracle VM Server for SPARC 2.0 Release Notes	821-1487
	Oracle VM Server for SPARC 2.0 Reference Manual Oracle Solaris 10 Reference Manual Collection ■ `drd(1M)` man page ■ `vntsd(1M)` man page	821-1486
Logical Domains Software Basics	*Beginners Guide to LDoms: Understanding and Deploying Logical Domains Software* (PDF)	820-0832
Logical Domains Management Information Base (MIB)	*Logical Domains (LDoms) MIB 1.0.1 Administration Guide*	820-2319-10
	Logical Domains (LDoms) MIB 1.0.1 Release Notes	820-2320-10
Oracle Solaris OS: Installation and Configuration	Oracle Solaris 10 9/10 Release and Installation Collection	N/A

You can find documentation that relates to your server, software, or the Oracle Solaris OS at http://docs.sun.com. Use the Search box to find the documents and the information that you need.

Documentation, Support, and Training

See the following web sites for additional resources:

- Documentation (http://docs.sun.com)
- Support (http://www.oracle.com/us/support/systems/index.html)
- Training (http://education.oracle.com) – Click the Sun link in the left navigation bar.

Oracle Welcomes Your Comments

Oracle welcomes your comments and suggestions on the quality and usefulness of its documentation. If you find any errors or have any other suggestions for improvement, go to http://docs.sun.com and click Feedback. Indicate the title and part number of the documentation along with the chapter, section, and page number, if available. Please let us know if you want a reply.

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

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

Typographic Conventions

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

TABLE P–2   Typographic Conventions

Typeface	Meaning	Example
AaBbCc123	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

TABLE P–2 Typographic Conventions *(Continued)*

Typeface	Meaning	Example
AaBbCc123	What you type, contrasted with onscreen computer output	machine_name% **su** Password:
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is rm *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

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

TABLE P–3 Shell Prompts

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

1

Overview of the Oracle VM Server for SPARC Software

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

The Oracle VM Server for SPARC software depends on particular Oracle Solaris OS versions, required software patches, and particular versions of system firmware. For more information, see "Required and Recommended Oracle Solaris OS" in *Oracle VM Server for SPARC 2.0 Release Notes*.

This chapter covers the following topics:

Hypervisor and Logical Domains

This section provides an overview of the SPARC hypervisor that supports Logical Domains.

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

A *logical domain* is a virtual machine comprised of a discrete logical grouping of resources. A logical domain has its own operating system and identity within a single computer system. Each logical domain can be created, destroyed, reconfigured, and rebooted independently, without requiring you to powercycle the server. You can run a variety of applications software in different logical domains and keep them independent for performance and security purposes.

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

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

FIGURE 1–1 Hypervisor Supporting Two Domains
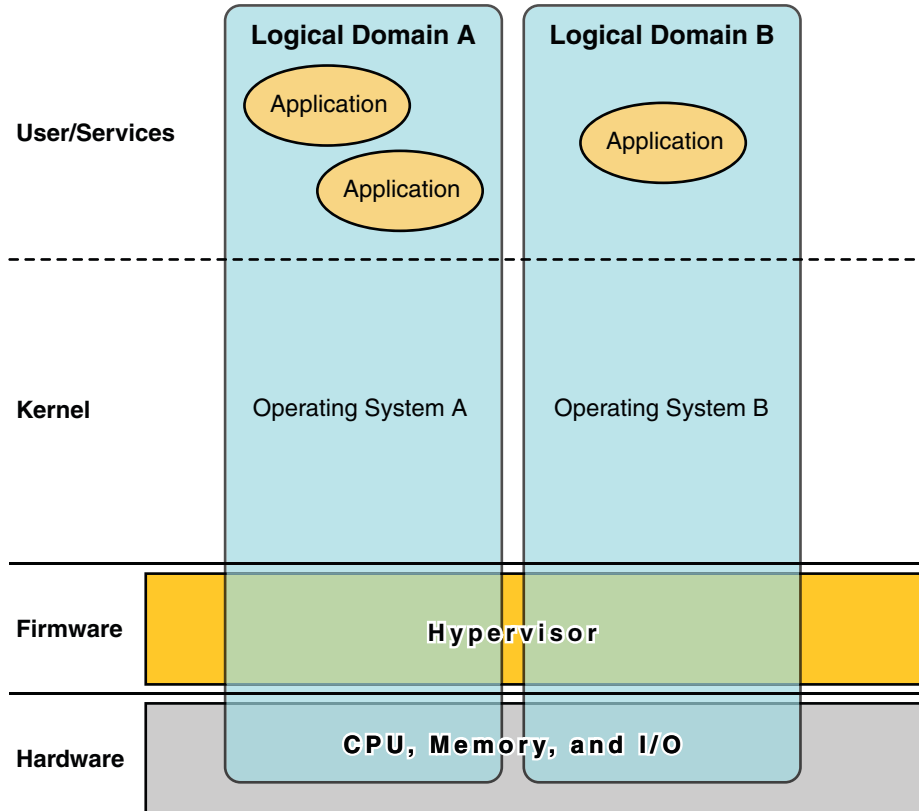
The number and capabilities of each logical domain that a specific SPARC hypervisor supports are server-dependent features. The hypervisor can allocate subsets of the overall CPU, memory,

and I/O resources of a server to a given logical domain. This enables support of multiple operating systems simultaneously, each within its own logical domain. Resources can be rearranged between separate logical domains with an arbitrary granularity. For example, CPUs are assignable to a logical domain with the granularity of a CPU thread.

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

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

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

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

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

Logical Domains Manager

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

Roles for Domains

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

- **Control domain.** The Logical Domains Manager runs in this domain, which enables you to create and manage other logical domains, and to allocate virtual resources to other domains. You can have only one control domain per server. The control domain is the first domain created when you install the Oracle VM Server for SPARC software. The control domain is named primary.

- **Service domain.** A service domain provides virtual device services to other domains, such as a virtual switch, a virtual console concentrator, and a virtual disk server. Any domain can be configured as a service domain.

■ **I/O domain.** An I/O domain has direct access to a physical I/O device, such as a network card in a PCI EXPRESS (PCIe) controller. An I/O domain can own a PCIe root complex, or it can own a PCIe slot or on-board PCIe device by using the direct I/O (DIO) feature. See "Assigning PCIe Endpoint Devices" on page 66.

An I/O domain can share physical I/O devices with other domains in the form of virtual devices when the I/O domain is also used as a service domain.

■ **Root domain.** A root domain has a PCIe root complex assigned to it. This domain owns the PCIe fabric and provides all fabric-related services, such as fabric error handling. A root domain is also an I/O domain, as it owns and has direct access to physical I/O devices.

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

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

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

Command-Line Interface

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

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

Virtual Input/Output

In a Logical Domains environment, you can provision up to 128 domains on an UltraSPARC T2 Plus processor system and a SPARC T3 processor system. These systems have a limited number of I/O buses and physical I/O slots. As a result, you cannot provide exclusive access to a physical disk and network devices to all domains on these systems. You can assign a PCIe bus or endpoint device to a domain to provide it with access to a physical device. Note that this

solution is insufficient to provide all domains with exclusive device access. See Chapter 6, "Setting Up I/O Domains." This limitation on the number of physical I/O devices that can be directly accessed is addressed by implementing a virtualized I/O model.

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

Virtual Network

Logical Domains uses the virtual network device and virtual network switch device to implement virtual networking. The virtual network (vnet) device emulates an Ethernet device and communicates with other vnet devices in the system by using a point-to-point channel. The virtual switch (vsw) device primarily functions as a multiplexor of all the virtual network's incoming and outgoing packets. The vsw device interfaces directly with a physical network adapter on a service domain, and sends and receives packets on behalf of a virtual network. The vsw device also functions as a simple layer-2 switch and switches packets between the vnet devices connected to it within the system.

Virtual Storage

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

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

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

Virtual Console

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

Resource Configuration

A system that runs the Oracle VM Server for SPARC software can configure resources, such as virtual CPUs, virtual I/O devices, cryptographic units, and memory. Some resources can be configured dynamically on a running domain, while others must be configured on a stopped domain. If a resource cannot be dynamically configured on the control domain, you must first initiate a delayed reconfiguration. The delayed reconfiguration postpones the configuration activities until after the control domain has been rebooted. For more information, see "Resource Reconfiguration" on page 149.

Persistent Configurations

You can use the ldm command to store the current configuration of a logical domain on the service processor. You can add a configuration, specify a configuration to be used, remove a configuration, and list the configurations. See the ldm(1M) man page. You can also specify a configuration to boot from the SP. See "Using Logical Domains With the Service Processor" on page 186.

For information about managing configurations, see "Managing Logical Domains Configurations" on page 178.

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

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

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

For information about the tool and about installing it, see Appendix A, "Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool." For information about the ldmp2v command, see the ldmp2v(1M) man page.

Oracle VM Server for SPARC Configuration Assistant

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

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

The Configuration Assistant is available as both a graphical user interface (GUI) and terminal-based tool.

For more information, see Appendix B, "Oracle VM Server for SPARC Configuration Assistant," and the ldmconfig(1M) man page.

2

Installing and Enabling Software

This chapter describes how to install or upgrade the different software components required to enable the Oracle VM Server for SPARC 2.0 software. Using the Oracle VM Server for SPARC software requires the following components:

- Supported platform, refer to "Supported Platforms" in *Oracle VM Server for SPARC 2.0 Release Notes* for a list of supported platforms.

- Control domain running an operating system at least equivalent to the Oracle Solaris 10 9/10 OS with any patches recommended in "Required Software and Patches" in *Oracle VM Server for SPARC 2.0 Release Notes*. See "Upgrading the Oracle Solaris OS" on page 32.

- At least system firmware version 7.3.0 for your Sun UltraSPARC T2 or T2 Plus platform and version 8.0.0 for your SPARC T3 platform. See "Upgrading the System Firmware" on page 26.

- Oracle VM Server for SPARC 2.0 software installed and enabled on the control domain. See "Installing the Logical Domains Manager" on page 29.

- (Optional) the Oracle VM Server for SPARC Management Information Base (MIB) software package. Refer to the *Logical Domains (LDoms) MIB 1.0.1 Administration Guide* for more information about using the Logical Domains MIB.

The Oracle Solaris OS and the system firmware must be installed or upgraded on your server before you install or upgrade the Logical Domains Manager. If your system is already using Oracle VM Server for SPARC software, see "Upgrading a System Already Using Oracle VM Server for SPARC" on page 32. Otherwise, see "Installing Oracle VM Server for SPARC Software on a New System" on page 26.

This chapter covers the following topics:

- "Installing Oracle VM Server for SPARC Software on a New System" on page 26
- "Upgrading a System Already Using Oracle VM Server for SPARC" on page 32
- "Factory Default Configuration and Disabling Logical Domains" on page 36

Note – The Solaris Security Toolkit (SST) software is no longer packaged with the Oracle VM Server for SPARC software. If you would like to use the most recent version of the SST software, see the *Oracle VM Server for SPARC 2.0 Release Notes*.

Installing Oracle VM Server for SPARC Software on a New System

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

Updating the Oracle Solaris OS

On a brand new system, you may want to reinstall the OS so that it conforms to your installation policy. In that case, refer to "Required and Recommended Oracle Solaris OS" in *Oracle VM Server for SPARC 2.0 Release Notes* to find the Oracle Solaris 10 OS that you should use for this version of the Oracle VM Server for SPARC software. Refer to your Oracle Solaris 10 OS installation guide for complete instructions for installing the Oracle Solaris OS. You can tailor your installation to the needs of your system.

If your system is already installed then it needs to be upgraded to the appropriate Oracle Solaris 10 OS that should be used for this version of the Oracle VM Server for SPARC software. Refer to "Required Software and Patches" in *Oracle VM Server for SPARC 2.0 Release Notes* to find the Oracle Solaris 10 OS that you should use for this version of the Oracle VM Server for SPARC software and the required and recommended patches. Refer to the Solaris 10 9/10 Release and Installation Collection (`http://docs.sun.com/app/docs/coll/1236.11`) for complete instructions for upgrading the OS.

Upgrading the System Firmware

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

For information about upgrading the system firmware by using the ILOM software, see "Update the Firmware" in *Sun SPARC Enterprise T5120 and T5220 Servers Topic Set* and "Updating ILOM Firmware" in the *Sun Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide* (`http://dlc.sun.com/pdf/820-6412-11/820-6412-11.pdf`).

▼ Upgrade System Firmware

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

For information about the required system firmware for the supported servers, see "Required System Firmware Patches" in *Oracle VM Server for SPARC 2.0 Release Notes*.

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

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

You can also use the ILOM web interface to upgrade system firmware, see "Updating ILOM Firmware" in the *Sun Integrated Lights Out Manager (ILOM) 3.0 Web Interface Procedures Guide* (http://dlc.sun.com/pdf/820-6411-11/820-6411-11.pdf).

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

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

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

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

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

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

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

This configuration is required to access the new flash image over the network. See "To Configure the Service Processor Network Management Port" in *Sun SPARC Enterprise T5120 and T5220 Servers Topic Set* and "Updating ILOM Firmware" in the *Sun Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide* (http://dlc.sun.com/pdf/820-6412-11/820-6412-11.pdf).

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

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

4   Verify that the host is powered off.

a. Type the following command:

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

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

```
-> stop /SYS
```

5   Verify that the `keyswitch_state` parameter is set to `normal`.

a. Type the following command:

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

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

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

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

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

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

- *IP-addr* is the IP address of a `tftp` server on the network that can access the flash image.

- *pathname* is the full path name to the flash image on the `tftp` server.

- *x_x_x* is the version number of the system firmware

- *nn* is the build number that applies to this release.

- *server-name* is the name of your server.

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

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

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

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

Downloading the Logical Domains Manager

▼ Download the Software

1   Download the zip file (`OVM_Server_SPARC-2_0.zip`).

You can find the software at http://www.oracle.com/virtualization.

2   Unzip the zip file.

```
$ unzip OVM_Server_SPARC-2_0.zip
```

See "Location of Oracle VM Server for SPARC 2.0 Software" in *Oracle VM Server for SPARC 2.0 Release Notes* for details about the structure of the file and what it includes.

Installing the Logical Domains Manager

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

- Using the installation script to install the packages and patches. This automatically installs the Logical Domains Manager software. See "Installing the Logical Domains Manager Software Automatically" on page 29.
- Using JumpStart to install the packages as part of an Oracle Solaris network installation. See "Using JumpStart to Install the Oracle VM Server for SPARC 2.0 Software" on page 30.
- Installing the package manually. See "Installing the Logical Domains Manager Software Manually" on page 31.

Note – Remember that you need to manually install the Logical Domains MIB software package after you install the Oracle VM Server for SPARC packages. It is not automatically installed with the other packages. Refer to the *Logical Domains (LDoms) MIB 1.0.1 Administration Guide* for more information about installing and using the Logical Domains MIB.

Installing the Logical Domains Manager Software Automatically

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

- **Using the** `install-ldm` **script with no options does the following automatically:**
    - Checks that the Oracle Solaris OS release is Oracle Solaris 10 9/10 OS at a minimum
    - Verifies that the package subdirectories `SUNWldm/` and `SUNWldmp2v/` are present
    - Verifies that the prerequisite Logical Domains driver packages, `SUNWldomr` and `SUNWldomu`, are present
    - Verifies that the `SUNWldm` and `SUNWldmp2v` packages have not been installed

- Installs the Oracle VM Server for SPARC 2.0 software
- Verifies that all packages are installed
- If the SST (`SUNWjass`) is already installed, you are prompted to harden the Oracle Solaris OS on the control domain.
- Determine whether to use the Oracle VM Server for SPARC Configuration Assistant (`ldmconfig`) to perform the installation.

- **Using the** `install-ldm` **script with the** `-c` **option automatically runs the Oracle VM Server for SPARC Configuration Assistant after the software is installed.**

- **Using the** `install-ldm` **script with the** `-s` **option skips the running of the Oracle VM Server for SPARC Configuration Assistant.**

- **Using the** `install-ldm` **script and the following options with the SST software enables you to do the following:**

  - `install-ldm -d`. Allows you to specify a SST driver other than a driver ending with `-secure.driver`. This option automatically performs all the functions listed in the preceding choice and hardens the Oracle Solaris OS on the control domain with the SST customized driver that you specify; for example, the `server-secure-myname.driver`.

  - `install-ldm -d none`. Specifies that you do *not* want to harden the Oracle Solaris OS running on your control domain by using the SST. This option automatically performs all the functions except hardening listed in the preceding choices. Bypassing the use of the SST is not suggested and should only be done when you intend to harden your control domain using an alternate process.

  - `install-ldm -p`. Specifies that you only want to perform the post-installation actions of enabling the Logical Domains Manager daemon (`ldmd`) and running the SST. For example, you would use this option if the `SUNWldm` and `SUNWjass` packages are preinstalled on your server.

Using JumpStart to Install the Oracle VM Server for SPARC 2.0 Software

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

▼ Set Up a JumpStart Server

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

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

   Perform the following steps:

   a. **See "Task Map: Preparing Custom JumpStart Installations" in** *Oracle Solaris 10 9/10 Installation Guide: Custom JumpStart and Advanced Installations***.**

   b. **Set up networked systems with the procedures in "Creating a Profile Server for Network Systems."**

   c. **Create a profile with the procedure in "Creating a Profile," and add a line to the profile to install the SUNWldm.v package by using the package profile keyword.**

      For example, add the following line to your profile to install the SUNWldm.v package from the extra directory from the HTTP server 192.168.254.255.

      ```
      package        SUNWldm.v http://192.168.254.255/extra timeout 5
      ```

   d. **Create the rules file with the procedure in "Creating the rules File."**

2 **Validate the rules file with the procedure in "Validating the rules File."**

Installing the Logical Domains Manager Software Manually

▼ Install the Oracle VM Server for SPARC 2.0 Software Manually

Before You Begin   Download the Oracle VM Server for SPARC 2.0 software (the SUNWldm and SUNWldmp2v packages). See "Download the Software" on page 29 for specific instructions.

1 **Use the pkgadd command to install the SUNWldm.v and SUNWldmp2v packages.**

   For more information about the pkgadd command, see the pkgadd(1M) man page.

   The -G option installs the package in the global zone only and the -d option specifies the path to the directory that contains the SUNWldm.v and SUNWldmp2v packages.

   ```
   # pkgadd -Gd . SUNWldm.v SUNWldmp2v
   ```

2 **Answer y for yes to all questions in the interactive prompts.**

3 **Use the pkginfo command to verify that the Oracle VM Server for SPARC 2.0 packages, SUNWldm and SUNWldmp2v, are installed.**

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

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

   ```
   # pkginfo -l SUNWldm | grep VERSION
   VERSION=2.0,REV=2010.08.03.10.20
   ```

Enabling the Logical Domains Manager Daemon

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

▼ Enable the Logical Domains Manager Daemon

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

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

For more information about the svcadm command, see the svcadm(1M) man page.

```
# svcadm enable ldmd
```

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

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

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

Upgrading a System Already Using Oracle VM Server for SPARC

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

If your system is already configured with the Oracle VM Server for SPARC software, then the control domain has to be upgraded. The other existing domains also have to be upgraded if you want to be able to use all features of the Oracle VM Server for SPARC 2.0 software.

Upgrading the Oracle Solaris OS

Refer to "Required Software and Patches" in *Oracle VM Server for SPARC 2.0 Release Notes* to find the Oracle Solaris 10 OS that you should use for this version of the Oracle VM Server for SPARC software, and the required and recommended patches for the different domains. Refer to the Oracle Solaris 10 installation guide for complete instructions for upgrading the Oracle Solaris OS.

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

Saving and Restoring Autosave Configuration Directories

Starting with the Logical Domains 1.2 release, you can save and restore autosave configuration directories prior to reinstalling the operating system on the control domain. Whenever you reinstall the operating system on the control domain, you must save and restore the Logical Domains autosave configuration data, which is found in the /var/opt/SUNWldm/autosave-*autosave-name* directories.

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

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

For more information about autosave configurations, see "Managing Logical Domains Configurations" on page 178.

▼ Save and Restore Autosave Directories

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

1 Save the autosave directories.

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

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

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

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

3 Restore the autosave directories.

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

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

Saving and Restoring the Logical Domains Constraints Database File

Whenever you upgrade the operating system on the control domain, you must save and restore the Logical Domains constraints database file that can be found in
`/var/opt/SUNWldm/ldom-db.xml`.

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

Preserving the Logical Domains Constraints Database File When Using Live Upgrade

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

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

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

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

If the control domain is upgraded from a Oracle Solaris 10 OS version older than Oracle Solaris 10 5/08 OS (or without patch 127127-11), and if volume manager volumes were exported as virtual disks, then the virtual disk back ends must be re-exported with `options=slice` after the Logical Domains Manager has been upgraded. See "Exporting Volumes and Backward Compatibility" on page 88 for more information.

Upgrading the Logical Domains Manager and the System Firmware

This section shows how to upgrade to Oracle VM Server for SPARC 2.0 software.

First download the Logical Domains Manager to the control domain. See "Downloading the Logical Domains Manager" on page 29.

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

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

1 **Bring down each domain to the `ok` prompt.**

2 **Stop all domains by using the `-a` option.**

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

3 **Issue the `unbind-domain` subcommand from the control domain for each domain.**

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

Upgrading to Oracle VM Server for SPARC 2.0 Software

This section shows how to upgrade to Oracle VM Server for SPARC 2.0 software.

Perform the procedure "Upgrade From Logical Domains 1.0 Software Only" in *Oracle VM Server for SPARC 2.0 Release Notes* if you want to use your existing Logical Domains 1.0 configurations with Oracle VM Server for SPARC 2.0 software. Existing Logical Domains 1.0 configurations do *not* work with Oracle VM Server for SPARC 2.0 software.

If you are upgrading from more recent versions of the Oracle VM Server for SPARC software, perform the procedure "Upgrade to the Oracle VM Server for SPARC 2.0 Software" on page 35. Such existing Logical Domains configurations *do* work with Oracle VM Server for SPARC 2.0 software.

▼ Upgrade to the Oracle VM Server for SPARC 2.0 Software

1 **Perform a flash upgrade of the system firmware.**

For the entire procedure, see "Upgrade System Firmware" on page 27.

2 **Disable the Logical Domains Manager daemon (`ldmd`).**

```
# svcadm disable ldmd
```

3 **Remove the old `SUNWldm` package.**

```
# pkgrm SUNWldm
```

4 **Add the new `SUNWldm` package.**

Specifying the `-d` option assumes that the package is in the current directory.

```
# pkgadd -Gd . SUNWldm
```

5   **Use the `ldm list` command to verify that the Logical Domains Manager is running.**

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

```
# ldm list
NAME            STATE    FLAGS   CONS    VCPU  MEMORY    UTIL  UPTIME
primary         active   ---c-   SP      32    3264M     0.3%  19d 9m
```

Factory Default Configuration and Disabling Logical Domains

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

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

▼ Remove All Guest Domains

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

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

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

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

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

---

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

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

▼ Remove All Logical Domains Configurations

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

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

2 Remove all configurations (_config-name_**) previously saved to the SP except for the**
   `factory-default` configuration.

   Use the following command for each such configuration:

   primary# **`ldm rm-config`** _config-name_

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

▼ Restore the Factory Default Configuration

1 Select the factory default configuration.

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

2 Stop the control domain.

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

3 Powercycle the system to load the factory default configuration.

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

▼ Disable the Logical Domains Manager

● **Disable the Logical Domains Manager from the control domain.**

   primary# **`svcadm disable ldmd`**

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

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

▼ Removing the Logical Domains Manager

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

● **Remove the Logical Domains Manager software.**

```
primary# pkgrm SUNWldm SUNWldmp2v
```

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

▼ Restore the Factory Default Configuration From the Service Processor

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

1 Restore the factory default configuration from the service processor.

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

2 Powercycle the system to load the factory default configuration.

```
-> reset /SYS
```

3

Security

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

This chapter covers the following topics:

- "Logical Domains Manager Authorization" on page 39
- "Creating Authorization and Profiles and Assigning Roles for User Accounts" on page 40
- "Configuring RBAC for Guest Console Access" on page 43
- "Enabling and Using BSM Auditing" on page 44

Logical Domains Manager Authorization

Authorization for the Logical Domains Manager has two levels:

- Read – allows you to view, but not modify the configuration.
- Read and write – allows you to view and change the configuration.

The changes are not made to the Oracle Solaris OS, but are added to the authorization file by the package script postinstall when the Logical Domains Manager is installed. Similarly, the authorization entries are removed by the package script preremove.

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

TABLE 3–1   The ldm Subcommands and User Authorizations

ldm Subcommand[1]	User Authorization
add-*	solaris.ldoms.write
bind-domain	solaris.ldoms.write

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

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

ldm Subcommand[1]	User Authorization
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

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

Creating Authorization and Profiles and Assigning Roles for User Accounts

You set up authorization and profiles and assign roles for user accounts using the Oracle Solaris OS Role-Based Access Control (RBAC) adapted for the Logical Domains Manager. Refer to the Solaris 10 System Administrator Collection (http://docs.sun.com/app/docs/coll/47.16) for more information about RBAC.

Authorization for the Logical Domains Manager has two levels:

- Read – allows you to view, but not modify the configuration.
- Read and write – allows you to view and change the configuration.

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

- solaris.ldoms.:::LDom administration::
- solaris.ldoms.grant:::Delegate LDom configuration::
- solaris.ldoms.read:::View LDom configuration::
- solaris.ldoms.write:::Manage LDom configuration::

Managing User Authorizations

▼ Add an Authorization for a User

Use the following steps as necessary to add authorizations in the /etc/security/auth_attr file for Logical Domains Manager users. Because the superuser already has solaris.* authorization, the superuser already has permission for solaris.ldoms.* authorizations.

1   **Create a local user account for each user who needs authorization to use the `ldm(1M)` subcommands.**

Note – To add Logical Domains Manager authorization for a user, a local (non-LDAP) account must be created for that user. Refer to the Oracle Solaris OS System Administrator Collection (`http://docs.sun.com/app/docs/coll/47.16`) for details.

---

2   **Do one of the following depending on which `ldm(1M)` subcommands you want the user to be able to access.**

See Table 3–1 for a list of `ldm`(1M) commands and their user authorizations.

- Add a read-only authorization for a user using the usermod(1M) command.

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

- Add a read and write authorization for a user using the usermod(1M) command.

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

▼ **Delete All Authorizations for a User**

● **Delete all authorizations for a local user account (the only possible option).**

    # **usermod -A ''** *username*

Managing User Profiles

The SUNWldm package adds two system-defined RBAC profiles in the /etc/security/prof_attr file for use in authorizing access to the Logical Domains Manager by non-superusers. The two Logical Domains-specific profiles are:

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

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

LDoms Management:suser:cmd:::/usr/sbin/ldm:privs=file_dac_read,file_dac_search

One of the preceding profiles can be assigned to a user account using the following procedure.

▼ **Add a Profile for 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 the Oracle Solaris 10 System Administrator Collection.

● **Add an administrative profile for a local user account; for example, LDoms Management.**

```
# usermod -P "LDoms Management" username
```

▼ **Delete All Profiles for a User**

● **Delete all profiles for a local user account (the only possible option).**

```
# usermod -P '' username
```

Assigning Roles to Users

The advantage of using this procedure is that only a user who has been assigned a specific role can assume the role. In assuming a role, a password is required if the role is given a password. This provides two layers of security. If a user has not been assigned a role, then the user cannot assume the role (by doing the su *role-name* command) even if the user has the correct password.

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

1 **Create a role.**

```
# roleadd -P "LDoms Review" ldm_read
```

2 **Assign a password to the role.**

```
# passwd ldm_read
```

3 **Assign the role to a user.**

For example, user_1.

```
# useradd -R ldm_read user_1
```

4 **Assign a password to the user (user_1).**

```
# passwd user_1
```

5 **Assign access only to the user_1 account to become the ldm_read account.**

```
# su user_1
```

6 **Type the user password when or if prompted.**

7 **Verify the user ID and access to the ldm_read role.**

```
$ id
uid=nn(user_1) gid=nn(group-name)
$ roles
ldm_read
```

8   Provide access to the user for `ldm` subcommands that have read authorization.

```
# su ldm_read
```

9   Type the user password when or if prompted.

10   Type the `id` command to show the user.

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

Configuring RBAC for Guest Console Access

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

By default, an authorization to access all guest consoles is added to the auth_attr database, when the vntsd service is enabled.

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

Superuser can use the usermod command to assign the required authorizations to other users or roles. This permits only the user or role who has the required authorizations to access a given domain console or console groups.

The following example gives user terry the authorization to access all domain consoles:

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

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

1. Add the new authorization entry to the auth_attr file for domain ldg1.

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

2. Assign this authorization to user sam:

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

For more information about authorizations and RBAC, see *System Administration Guide: Security Services*.

Enabling and Using BSM Auditing

The Logical Domains Manager uses the Oracle Solaris OS Basic Security module (BSM) auditing capability. BSM auditing provides the means to examine the history of actions and events on your control domain to determine what happened. The history is kept in a log of what was done, when it was done, by whom, and what was affected.

To enable and disable this auditing capability, use the Oracle Solaris OS bsmconv(1M) and bsmunconv(1M) commands. This section also includes tasks that show how to verify the auditing capability, print audit output, and rotate audit logs. You can find further information about BSM auditing in the Solaris 10 *System Administration Guide: Security Services*.

▼ Enable BSM Auditing

1 Add vs in the **flags:** line of the **/etc/security/audit_control** file.

2 Run the **bsmconv(1M)** command.

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

For more information about this command, see the bsmconv(1M) man page.

3 Reboot the Oracle Solaris OS for auditing to take effect.

▼ Verify That BSM Auditing Is Enabled

1 Type the following command.

```
# auditconfig -getcond
```

2 Check that **audit condition = auditing** appears in the output.

▼ Disable BSM Auditing

1 Run the **bsmunconv** command to disable BSM auditing.

```
# /etc/security/bsmunconv
```

For more information about this command, see the bsmunconv(1M) man page.

2 Reboot the Oracle Solaris OS for the disabling of auditing to take effect.

▼ Print Audit Output

● **Use one of the following to print BSM audit output:**

- Use the auditreduce(1M) and praudit(1M) commands to print audit output.

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

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

▼ Rotate Audit Logs

● **Use the audit -n command to rotate audit logs.**

4

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 Appendix B, "Oracle VM Server for SPARC Configuration Assistant."

This chapter covers the following topics:

- "Output Messages" on page 47
- "Creating Default Services" on page 48
- "Initial Configuration of the Control Domain" on page 49
- "Rebooting to Use Logical Domains" on page 50
- "Enabling Networking Between the Control/Service Domain and Other Domains" on page 51
- "Enabling the Virtual Network Terminal Server Daemon" on page 52

Output Messages

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

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

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

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

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

Creating Default Services

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

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

▼ Create Default Services

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

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

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

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

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

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

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

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

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

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

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

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

Dynamic Host Configuration Protocol (DHCP) server assigns the domain the same IP address. See "Enabling Networking Between the Control/Service Domain and Other Domains" on page 51.

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

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

Your output should look similar to the following.

```
primary# ldm list-services primary
VDS
    NAME                VOLUME          OPTIONS         DEVICE
    primary-vds0

VCC
    NAME                PORT-RANGE
    primary-vcc0        5000-5100

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

Initial Configuration of the Control Domain

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

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

▼ Set Up the Control Domain

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

---

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

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

2   Assign cryptographic resources to the control domain.

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

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

3   Assign virtual CPUs to the control domain.

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

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

4   Initiate a delayed reconfiguration on the control domain.

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

5   Assign memory to the control domain.

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

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

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

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

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

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

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

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

Rebooting to Use Logical Domains

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

▼ Reboot

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

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

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

Enabling Networking Between the Control/Service Domain and Other Domains

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

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

▼ Configure the Virtual Switch as the Primary Interface

1 Print out the addressing information for all interfaces.

```
primary# ifconfig -a
```

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

```
primary# ifconfig vsw0 plumb
```

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

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

4 Unplumb the physical network device assigned to the virtual switch (net-dev), which is nxge0 in this example.

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

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

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

  ```
  primary# ifconfig vsw0 IP_of_nxge0 netmask netmask_of_nxge0 broadcast + up
  ```
- If networking is configured using DHCP, enable DHCP for vsw0.

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

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

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

Note – If necessary, you can also configure the virtual switch as well as the physical network device. In this case, plumb the virtual switch as in Step 2, and do not unplumb the physical device (skip Step 4). You must then configure the virtual switch with either a static IP address or a dynamic IP address. You can obtain a dynamic IP address from a DHCP server. For additional information and an example of this case, see "Configuring Virtual Switch and Service Domain for NAT and Routing" on page 117.

Enabling the Virtual Network Terminal Server Daemon

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

▼ Enable the Virtual Network Terminal Server Daemon

Note – Be sure that you have created the default service vconscon (vcc) on the control domain before you enable vntsd. See "Creating Default Services" on page 48 for more information.

---

1   Use the svcadm(1M) command to enable the virtual network terminal server daemon, vntsd(1M).

```
primary# svcadm enable vntsd
```

2   Use the svcs(1) command to verify that the vntsd daemon is enabled.

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

5

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 Appendix B, "Oracle VM Server for SPARC Configuration Assistant."

This chapter covers the following topics:

- "Creating and Starting a Guest Domain" on page 53
- "Installing Oracle Solaris OS on a Guest Domain" on page 56

Creating and Starting a Guest Domain

The guest domain must run an operating system that understands both the sun4v platform and the virtual devices presented by the hypervisor. Currently, this means that you must run at least the Oracle Solaris 10 11/06 OS. Running the Oracle Solaris 10 9/10 OS provides you with all the Oracle VM Server for SPARC 2.0 features. See the *Oracle VM Server for SPARC 2.0 Release Notes* for any specific patches that might be necessary. Once you have created default services and reallocated resources from the control domain, you can create and start a guest domain.

▼ Create and Start a Guest Domain

1   Create a logical domain.

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

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

2   Add CPUs to the guest domain.

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

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

3   Add memory to the guest domain.

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

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

4   Add a virtual network device to the guest domain.

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

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

Where:

- vnet1 is a unique interface name to the logical domain, assigned to this virtual network device instance for reference on subsequent set-vnet or remove-vnet subcommands.

- primary-vsw0 is the name of an existing network service (virtual switch) to which to connect.

Note – Steps 5 and 6 are simplified instructions for adding a virtual disk server device (vdsdev) to the primary domain and a virtual disk (vdisk) to the guest domain. To learn how ZFS volumes and file systems can be used as virtual disks, see "Export a ZFS Volume as a Single-Slice Disk" on page 87 and "Using ZFS With Virtual Disks" on page 96.

---

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

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

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

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

  Where:

  - /dev/dsk/c2t1d0s2 is the path name of the actual physical device. When adding a device, the path name must be paired with the device name.

  - vol1 is a unique name you must specify for the device being added to the virtual disk server. The volume name must be unique to this virtual disk server instance, because this name is exported by this virtual disk server to the clients for adding. When adding a device, the volume name must be paired with the path name of the actual device.

  - primary-vds0 is the name of the virtual disk server to which to add this device.

- **File Example.** This second example is exporting a file as a block device.

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

Where:

- *backend* is the path name of the actual file exported as a block device. When adding a device, the back end must be paired with the device name.

- vol1 is a unique name you must specify for the device being added to the virtual disk server. The volume name must be unique to this virtual disk server instance, because this name is exported by this virtual disk server to the clients for adding. When adding a device, the volume name must be paired with the path name of the actual device.

- primary-vds0 is the name of the virtual disk server to which to add this device.

6   Add a virtual disk to the guest domain.

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

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

Where:

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

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

---

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

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

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

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

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

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

```
primary# ldm bind-domain ldg1
primary# ldm list-domain ldg1
NAME           STATE    FLAGS   CONS    VCPU MEMORY    UTIL  UPTIME
ldg1           bound    -----   5000    8    2G
```

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

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

10   Connect to the console of a guest domain from another terminal by logging into the control domain and connecting directly to the console port on the local host.

```
$ ssh hostname.domain-name
$ telnet localhost 5000
```

11   Start the guest domain ldg1.

```
primary# ldm start-domain ldg1
```

Installing Oracle Solaris OS on a Guest Domain

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

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

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

1   Insert the Oracle Solaris 10 OS DVD into the DVD drive.

2   Stop the volume management daemon, vold(1M), on the primary domain.

```
primary# svcadm disable volfs
```

3   Stop and unbind the guest domain (ldg1).

```
primary# ldm stop ldg1
primary# ldm unbind ldg1
```

4   Add the DVD with the DVD-ROM media as a secondary volume and virtual disk.

The following uses c0t0d0s2 as the DVD drive in which the Oracle Solaris media resides, dvd_vol@primary-vds0 as a secondary volume, and vdisk_cd_media as a virtual disk.

```
primary# ldm add-vdsdev /dev/dsk/c0t0d0s2 dvd_vol@primary-vds0
primary# ldm add-vdisk vdisk_cd_media dvd_vol@primary-vds0 ldg1
```

5   Check to see that the DVD is added as a secondary volume and virtual disk.

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

```
   dvd_vol                                    /dev/dsk/c0t0d0s2
....
-------------------------------------------------------------------------------
NAME              STATE    FLAGS    CONS    VCPU  MEMORY   UTIL  UPTIME
ldg1              inactive -----            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 027.0.0.1...
Connected to localhost.
Escape character is '^]'.

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

7 Show the device aliases in the client OpenBoot PROM.

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

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

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

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

9 Continue with the Oracle Solaris OS installation menu.

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

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

```
primary# ldm stop ldg1
primary# ldm unbind ldg1
```

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

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

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

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

```
primary# ldm list-bindings
NAME            STATE    FLAGS   CONS    VCPU MEMORY   UTIL UPTIME
primary         active   -n-cv   SP      4    4G       0.2% 22h 45m
...
VDS
   NAME            VOLUME       OPTIONS         DEVICE
   primary-vds0    vol1                         /dev/dsk/c2t1d0s2
   iso_vol                                      /export/solarisdvd.iso
....
--------------------------------------------------------------------------------
NAME            STATE    FLAGS   CONS    VCPU MEMORY   UTIL UPTIME
ldg1            inactive -----           60   6G
...
DISK
   NAME            VOLUME                       TOUT DEVICE  SERVER
   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 menu.**

▼ Perform a JumpStart Operation on a Guest Domain

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

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

Instead of the usual c*N*t*N*d*N*s*N* format, virtual disk device names use the c*N*d*N*s*N* format. c*N* is the virtual controller, d*N* is the virtual disk number, and s*N* is the slice number.

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

A virtual disk can appear either as a full disk or as a single-slice disk. The Oracle Solaris OS can be installed on a full disk by using a regular JumpStart profile that specifies multiple partitions. A single-slice disk only has a single partition, s0, that uses the entire disk. To install the Oracle Solaris OS on a single disk, you must use a profile that has a single partition (/) that uses the entire disk. You cannot define any other partitions, such as swap. For more information about full disks and single-slice disks, see "Virtual Disk Appearance" on page 81.

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

See *Oracle Solaris 10 9/10 Installation Guide: Custom JumpStart and Advanced Installations*.

Normal UFS Profile

```
filesys 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 9/10 Installation Guide: Custom JumpStart and Advanced Installations*.

Normal ZFS Profile

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

Actual ZFS Profile for Installing a Domain

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

6

Setting Up I/O Domains

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

This chapter covers the following topics:

- "I/O Domain Overview" on page 61
- "Assigning PCIe Buses" on page 62
- "Assigning PCIe Endpoint Devices" on page 66

I/O Domain Overview

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

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

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

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

- "Assigning PCIe Buses" on page 62
- "Assigning PCIe Endpoint Devices" on page 66

Note – You *cannot* migrate an I/O domain that is configured with PCIe endpoint devices. For information about other migration limitations, see Chapter 9, "Migrating Domains."

Assigning PCIe Buses

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

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

FIGURE 6–1 Assigning a PCIe Bus to an I/O Domain
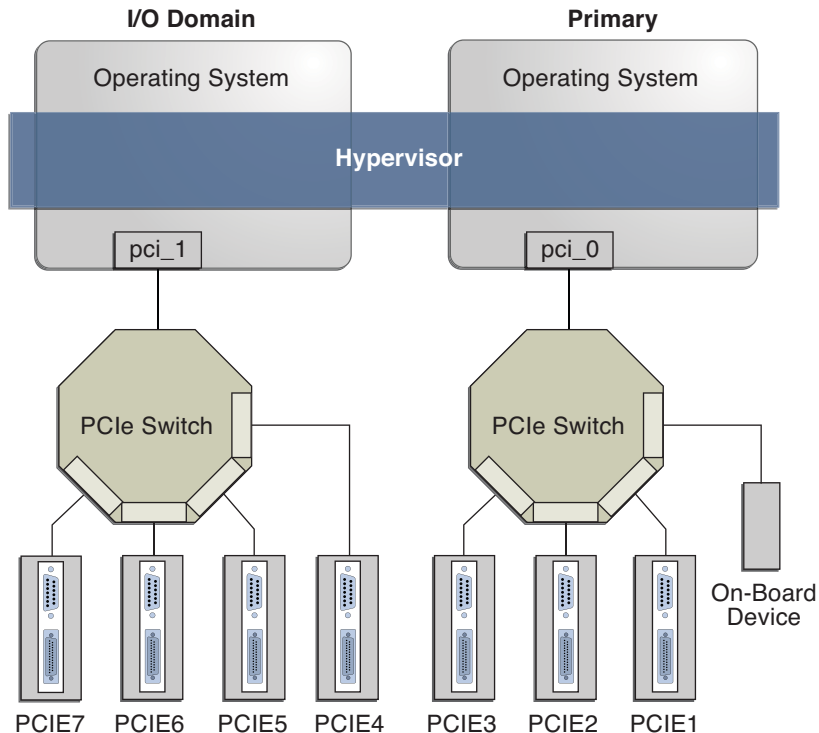
The maximum number of I/O domains that you can create with PCIe buses depends on the number of PCIe buses that are available on the server. For example, if you are using a Sun SPARC Enterprise T5440 server, you can have up to four I/O domains.

Note – Some Sun UltraSPARC servers have only one PCIe bus. In such cases, you can create an I/O domain by assigning a PCIe endpoint (or direct I/O-assignable) device to a domain. See "Assigning PCIe Endpoint Devices" on page 66. If the system has a Network Interface Unit (NIU), you can also assign an NIU to a domain to create an I/O domain.

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

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

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

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

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

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

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

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

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


PCIE                    PSEUDONYM  STATUS  DOMAIN
----                    ---------  ------  ------
pci@400/pci@0/pci@d     MB/PCIE0   EMP     -
pci@400/pci@0/pci@c     MB/PCIE1   OCC     primary
pci@400/pci@0/pci@1     MB/HBA     OCC     primary
pci@500/pci@0/pci@d     MB/PCIE4   EMP     -
pci@500/pci@0/pci@9     MB/PCIE5   EMP     -
pci@500/pci@0/pci@c     MB/NET0    OCC     primary
pci@600/pci@0/pci@c     MB/PCIE2   OCC     primary
pci@600/pci@0/pci@9     MB/PCIE3   OCC     primary
pci@700/pci@0/pci@c     MB/PCIE6   OCC     primary
pci@700/pci@0/pci@9     MB/PCIE7   EMP     -
```

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

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

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

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

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

        NAME        STATE     READ WRITE CKSUM
        rpool       ONLINE       0     0     0
          c0t1d0s0  ONLINE       0     0     0
```

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

The following example uses block device c1t0d0s0:

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

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

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

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

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

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

The following command uses the nxge0 network interface:

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

In this example, the physical device for the network interface used by the primary domain is under bus pci@500, which corresponds to the earlier listing of pci_1. So, the other two buses, pci_2 (pci@600) and pci_3 (pci@700), can safely be assigned to other domains because they are not used by the primary domain.

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

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

In this example, bus pci_2 and bus pci_3 are being removed from the primary domain. You might see a message from the ldm command that the primary domain is entering delayed reconfiguration mode.

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

7   Save this configuration to the service processor.

In this example, the configuration is io-domain.

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

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

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

---

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

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

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

The following example stops the ldg1 domain:

```
primary# ldm stop ldg1
```

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

The available bus is pci_2 and the domain is ldg1.

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

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

The following commands restart the ldg1 domain:

```
primary# ldm start ldg1
```

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

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

PCIE               PSEUDONYM  STATUS  DOMAIN
----               ---------  ------  ------
pci@400/pci@0/pci@d  MB/PCIE0   EMP     -
pci@400/pci@0/pci@c  MB/PCIE1   OCC     primary
pci@400/pci@0/pci@1  MB/HBA     OCC     primary
pci@500/pci@0/pci@d  MB/PCIE4   EMP     -
pci@500/pci@0/pci@9  MB/PCIE5   EMP     -
pci@500/pci@0/pci@c  MB/NET0    OCC     primary
pci@600/pci@0/pci@c  MB/PCIE2   UNK     -
pci@600/pci@0/pci@9  MB/PCIE3   UNK     -
pci@700/pci@0/pci@c  MB/PCIE6   UNK     -
pci@700/pci@0/pci@9  MB/PCIE7   UNK     -
```

This output confirms that the PCIe buses pci_0 and pci_1 and the devices below them are assigned to domain primary, and that pci_2 and its devices are assigned to ldg1.

Assigning PCIe Endpoint Devices

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

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

A PCIe endpoint device can be one of the following:

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

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

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

FIGURE 6–2   Assigning a PCIe Endpoint Device to an I/O Domain
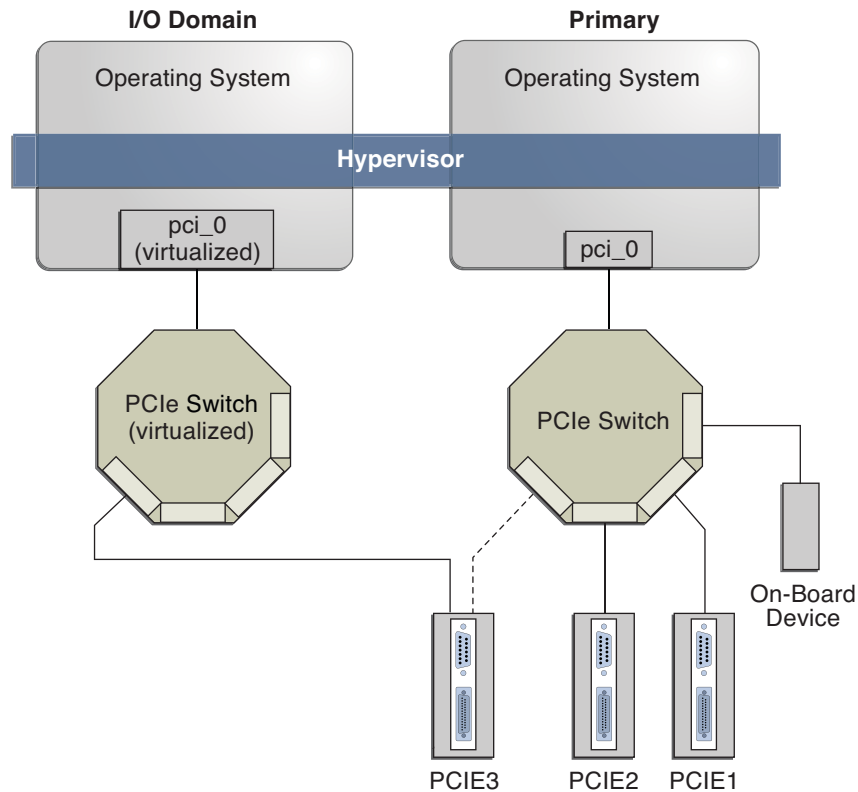
Use the ldm list-io command to list the PCIe endpoint devices.

Though the DIO feature permits any PCIe card in a slot to be assigned to an I/O domain, only certain PCIe cards are supported. See "Direct I/O Hardware and Software Requirements" in *Oracle VM Server for SPARC 2.0 Release Notes*.

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

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

- This feature is enabled only when all the software requirements are met. See "Direct I/O Hardware and Software Requirements" in *Oracle VM Server for SPARC 2.0 Release Notes*.
- Only PCIe endpoints that are connected to a PCIe bus assigned to the primary domain can be assigned to another domain with the DIO feature.
- I/O domains using DIO have access to the PCIe endpoint devices only when the primary domain is running.
- Rebooting the primary domain affects I/O domains that have PCIe endpoint devices. See "Rebooting the primary Domain" on page 70. The primary domain also has the following responsibilities:
    - Initializes the PCIe bus and manages the bus.
    - Handles all bus errors that are triggered by the PCIe endpoint devices that are assigned to I/O domains. Note that only the primary domain receives all PCIe bus-related errors.

Direct I/O Hardware and Software Requirements

To successfully use the DIO feature, you must be running the appropriate software and assign only the PCIe cards that are supported by the DIO feature to I/O domains. For the hardware and software requirements, see "Direct I/O Hardware and Software Requirements" in *Oracle VM Server for SPARC 2.0 Release Notes*.

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

Direct I/O Limitations

For information about how to work around the following limitations, see "Planning PCIe Endpoint Device Configuration" on page 69.

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

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

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

Planning PCIe Endpoint Device Configuration

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

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

For an example, see "Create an I/O Domain by Assigning a PCIe Endpoint Device" on page 72.

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

1. Understand and record your system hardware configuration.

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

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

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

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

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

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

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

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

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

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

   Use the `ldm add-config` command.

5. Reboot the `primary` domain to release the PCIe endpoint devices that you removed in Step 3.

6. Confirm that the PCIe endpoint devices you removed are no longer assigned to the `primary` domain.

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

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

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

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

   Use the `ldm add-io` command.

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

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

   See "Making PCIe Hardware Changes" on page 71.

Rebooting the `primary` Domain

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

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

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

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

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

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

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

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

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

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

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

Making PCIe Hardware Changes

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

- No action is required if you are installing a PCIe card into an empty slot. This PCIe card is automatically owned by the domain that owns the PCIe bus.

  To assign the new PCIe card to an I/O domain, use the ldm rm-io command to first remove the card from the primary domain. Then, use the ldm add-io command to assign the card to an I/O domain.

- No action is required if a PCIe card is removed from the system and assigned to the primary domain.

- To remove a PCIe card that is assigned to an I/O domain, first remove the device from the I/O domain. Then, add the device to the primary domain before you physically remove the device from the system.

- To replace a PCIe card that is assigned to an I/O domain, verify that the new card is supported by the DIO feature.

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

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

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

Plan all DIO deployments ahead of time to minimize downtime.

For an example of adding a PCIe endpoint device to create an I/O domain, see "Planning PCIe Endpoint Device Configuration" on page 69.

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

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

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

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

PCIE                    PSEUDONYM STATUS DOMAIN
----                    --------- ------ ------
pci@400/pci@0/pci@c  PCIE1     EMP    -
pci@400/pci@0/pci@9  PCIE2     OCC    primary
        network@0
        network@0,1
        network@0,2
        network@0,3
pci@400/pci@0/pci@d  PCIE3     OCC    primary
        SUNW,emlxs/fp/disk
        SUNW,emlxs@0,1/fp/disk
        SUNW,emlxs@0,1/fp@0,0
pci@400/pci@0/pci@8  MB/SASHBA OCC    primary
        scsi@0/tape
        scsi@0/disk
        scsi@0/sd@0,0
        scsi@0/sd@1,0
pci@500/pci@0/pci@9  PCIE0     EMP    -
pci@500/pci@0/pci@d  PCIE4     OCC    primary
        network@0
        network@0,1
```

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

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

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

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

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

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

        NAME       STATE     READ WRITE CKSUM
        rpool      ONLINE      0     0     0
          c0t1d0s0 ONLINE      0     0     0
```

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

The following example uses block device c0t1d0s:

```
primary# ls -l /dev/dsk/c0t1d0s
lrwxrwxrwx   1 root    root    49 Jul 20 22:17 /dev/dsk/c0t1d0s0 ->
../../devices/pci@400/pci@0/pci@8/scsi@0/sd@0,0:a
```

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

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

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
nxge0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 2
        inet 10.6.212.149 netmask fffffe00 broadcast 10.6.213.255
        ether 0:21:28:4:27:cc
```

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

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

The following command uses the nxge0 network interface:

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

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

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

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

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

a. Remove the PCIe endpoint devices.

> ⚠ **Caution** – Do *not* remove the devices that are used in the primary domain.
>
> If you mistakenly remove the wrong devices, use the ldm cancel-op reconf primary command to cancel the delayed reconfiguration on the primary domain.

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

```
# ldm rm-io PCIE2 primary
Initiating a delayed reconfiguration operation on the primary domain.
All configuration changes for other domains are disabled until the primary
domain reboots, at which time the new configuration for the primary domain
will also take effect.
# ldm rm-io PCIE3 primary
------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
------------------------------------------------------------------------------
```

```
# ldm rm-io PCIE4 primary
-------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
-------------------------------------------------------------------------------
# ldm rm-io PCIE5 primary
-------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
-------------------------------------------------------------------------------
```

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

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

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

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

```
# reboot -- -r
```

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

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

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

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

8   Assign a PCIe endpoint device to a domain.

a. Add the PCIE2 device to the ldg1 domain.

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

b. Bind and start the ldg1 domain.

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

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

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

```
# dladm show-dev
vnet0          link: up        speed: 0     Mbps        duplex: unknown
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
```

7

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

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

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

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

Logical devices can be any of the following:

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

FIGURE 7–1   Virtual Disks With Logical Domains
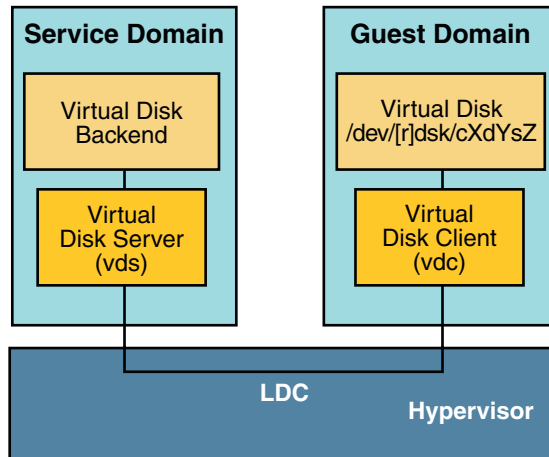
Managing Virtual Disks

This section describes adding a virtual disk to a guest domain, changing virtual disk and timeout options, and removing a virtual disk from a guest domain. See "Virtual Disk Back End Options" on page 82 for a description of virtual disk options. See "Virtual Disk Timeout" on page 95 for a description of the virtual disk timeout.

▼ Add a Virtual Disk

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

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

2 **Assign the back end to a guest domain.**

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

You can specify an ID of a new virtual disk device by setting the id property. By default, ID values are automatically generated, so set this property if you need to match an existing device name in the OS. See "Virtual Disk Identifier and Device Name" on page 80.

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

▼ Export a Virtual Disk Back End Multiple Times

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

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

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

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

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

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

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

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

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

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

▼ Change Virtual Disk Options

For more information about virtual disk options, see "Virtual Disk Back End Options" on page 82.

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

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

▼ Change the Timeout Option

For more information about virtual disk options, see "Virtual Disk Back End Options" on page 82.

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

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

▼ Remove a Virtual Disk

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

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

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

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

Virtual Disk Identifier and Device Name

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

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

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

When a domain with virtual disks is running the Oracle Solaris OS, each virtual disk appears in the domain as a c0d*n* disk device, where *n* is the device number of the virtual disk.

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

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

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

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

Virtual Disk Appearance

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

Full Disk

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

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

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

Single-Slice Disk

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

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

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

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

Virtual Disk Back End Options

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

Read-only (ro) Option

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

Exclusive (excl) Option

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

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

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

- When guest domains are running, if you want to be able to use commands such as format(1M) or luxadm(1M) to manage physical disks, then do not export these disks with the excl option.

- When you export an Solaris Volume Manager volume, such as a RAID or a mirrored volume, do not set the excl option. Otherwise, this can prevent Solaris Volume Manager from starting some recovery operation in case a component of the RAID or mirrored volume fails. See "Using Virtual Disks on Top of Solaris Volume Manager" on page 101 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 102 for more information.

- If you are exporting the same virtual disk back end multiple times from the same virtual disk service, see "Export a Virtual Disk Back End Multiple Times" on page 79 for more information.

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

Slice (`slice`) Option

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

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

For more information about this option, see "Virtual Disk Back End" on page 84.

Virtual Disk Back End

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

Physical Disk or Disk LUN

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

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

▼ Export a Physical Disk as a Virtual Disk

1 Export a physical disk as a virtual disk.

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

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

2 Assign the disk to a guest domain.

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

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

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

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

For example, the disk being checked is c0d1.

```
ldg1# ls -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 whether or not you specify the slice option. If the device is the slice 2 of the disk, you must set the slice option to export only slice 2 as a single-slice disk; otherwise, the entire disk is exported as full disk.

▼ Export a Physical Disk Slice as a Virtual Disk

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

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

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

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

2 Assign the disk to a guest domain.

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

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

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

```
ldg1# ls -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 8 devices, because the disk is a single-slice disk, only the first slice (s0) is usable.

▼ Export Slice 2

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

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

File and Volume

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

File or Volume Exported as a Full Disk

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

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

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

▼ Export a File as a Full Disk

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

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

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

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

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

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

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

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

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

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

A full disk is a regular disk with 8 slices.

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

```
ldg1# ls -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
```

File or Volume Exported as a Single-Slice Disk

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

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

▼ Export a ZFS Volume as a Single-Slice Disk

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

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

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

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

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

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

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

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

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

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

```
ldg1# ls -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

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

- Use the ldm set-vdsdev command in Oracle VM Server for SPARC 2.0 software, and set the slice option for all volumes you want to export as single-slice disks. Refer to the ldm(1M) man page for more information about this command.

- Add the following line to the /etc/system file on the service domain.

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

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

Summary of How Different Types of Back Ends Are Exported

Back End	No Slice Option	Slice Option Set
Disk (disk slice 2)	Full disk[1]	Single-slice disk[2]
Disk slice (not slice 2)	Single-slice disk[3]	Single-slice disk
File	Full disk	Single-slice disk

[1] Export the entire disk.

[2] Export only slice 2

[3] A slice is always exported as a single-slice disk.

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

Guidelines for Exporting Files and Disk Slices as Virtual Disks

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

Using the Loopback File (lofi) Driver

It is possible to use the loopback file (lofi) driver to export a file as a virtual disk. However, doing this adds an extra driver layer and impacts performance of the virtual disk. Instead, you can directly export a file as a full disk or as a single-slice disk. See "File and Volume" on page 86.

Directly or Indirectly Exporting a Disk Slice

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

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

Configuring Virtual Disk Multipathing

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

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

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

The following figure shows a virtual disk multipathing configuration, which is used as an example in the procedure . In this example, a multipathing group named foo is used to create a virtual disk, whose back end is accessible from two service domains: primary and alternate.

FIGURE 7–2 Configuring Virtual Disk Multipathing
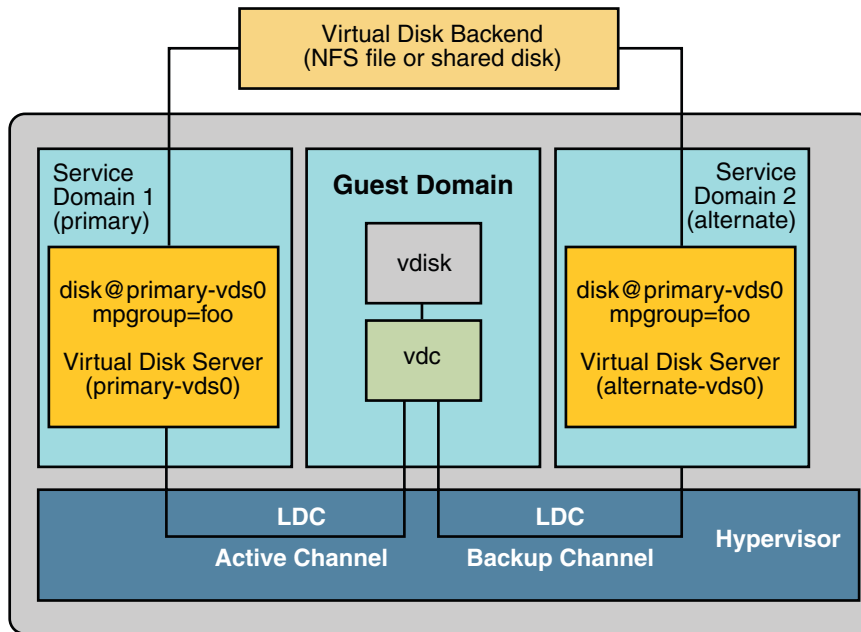
▼ Configure Virtual Disk Multipathing

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

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

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

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

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

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

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

3 Export the virtual disk to the guest domain.

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

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

More Information   Result of Virtual Disk Multipathing

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

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

CD, DVD and ISO Images

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

Note – You cannot export the CD or DVD drive itself; you only can export the CD or DVD that is inside the CD or DVD drive. Therefore, a CD or DVD must be present inside the drive before you can export it. Also, to be able to export a CD or DVD, that CD or DVD cannot be in use in the service domain. In particular, the Volume Management file system, volfs(7FS) service must not use the CD or DVD. See "Export a CD or DVD From the Service Domain to the Guest Domain" on page 92 for instructions on how to remove the device from use by volfs.

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

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

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

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

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

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

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

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

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

2. **Do one of the following.**

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

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

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

   See the vold.conf(4) man page.

   b. **Insert the CD or DVD in the CD or DVD drive.**

c. **From the service domain, restart the volume management file system service.**

```
service# svcadm refresh volfs
service# svcadm restart volfs
```

3 **From the service domain, find the disk path for the CD-ROM device.**

```
service# cdrw -l
Looking for CD devices...
   Node                 Connected Device                 Device type
---------------------+-------------------------------+-----------------
/dev/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 shows how to assign the exported CD or DVD to domain ldg1:

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

More Information   Exporting a CD or DVD Multiple Times

A CD or DVD can be exported multiple times and assigned to different guest domains. See "Export a Virtual Disk Back End Multiple Times" on page 79 for more information.

▼ Export an ISO Image From the `primary` Domain to Install a Guest Domain

This procedure shows how to export an ISO image from the `primary` domain and use it to install a guest domain. This procedure assumes that both the `primary` domain and the guest domain are configured.

For example, the following `ldm list` shows that both the `primary` and `ldom1` domains are configured:

```
# ldm list
NAME            STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary         active   -n-cv   SP      4     4G       0.3%  15m
ldom1           active   -t---   5000    4     1G        25%  8m
```

1 **Add a virtual disk server device to export the ISO image.**

In this example, the ISO image is /export/images/sol-10-u8-ga-sparc-dvd.iso.

```
# ldm add-vdsdev /export/images/sol-10-u8-ga-sparc-dvd.iso dvd-iso@primary-vds0
```

2 Stop the guest domain.

In this example, the logical domain is ldom1.

```
# ldm stop-domain ldom1
LDom ldom1 stopped
```

3 Add the virtual disk for the ISO image to the logical domain.

In this example, the logical domain is ldom1.

```
# ldm add-vdisk s10-dvd dvd-iso@primary-vds0 ldom1
```

4 Restart the guest domain.

In this example, the logical domain is ldom1.

```
# ldm start-domain ldom1
LDom ldom1 started
# ldm list
NAME            STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary         active     -n-cv   SP      4     4G       0.4%  25m
ldom1           active     -t---   5000    4     1G       0.0%  0s
```

In this example, the ldm list command shows that the ldom1 domain has just been started.

5 Connect to the guest domain.

```
# telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Connecting to console "ldom1" in group "ldom1" ....
Press ~? for control options ..
```

6 Verify the existence of the ISO image as a virtual disk.

```
{0} ok show-disks
a) /virtual-devices@100/channel-devices@200/disk@1
b) /virtual-devices@100/channel-devices@200/disk@0
q) NO SELECTION
Enter Selection, q to quit: q
```

In this example, the newly added device is
/virtual-devices@100/channel-devices@200/disk@1.

7 Boot the guest domain to install from the ISO image.

In this example, boot from the f slice of the
/virtual-devices@100/channel-devices@200/disk@1 disk.

```
{0} ok boot /virtual-devices@100/channel-devices@200/disk@1:f
```

Virtual Disk Timeout

By default, if the service domain providing access to a virtual disk back end is down, all I/O from the guest domain to the corresponding virtual disk is blocked. The I/O automatically is resumed when the service domain is operational and is servicing I/O requests to the virtual disk back end.

However, there are some cases when file systems or applications might not want the I/O operation to block, but for it to fail and report an error if the service domain is down for too long. It is now possible to set a connection timeout period for each virtual disk, which can then be used to establish a connection between the virtual disk client on a guest domain and the virtual disk server on the service domain. When that timeout period is reached, any pending I/O and any new I/O will fail as long as the service domain is down and the connection between the virtual disk client and server is not reestablished.

This timeout can be set by doing one of the following:

- Using the ldm add-vdisk command.

  ldm add-vdisk timeout=*seconds disk-name volume-name@service-name ldom*

- Using the ldm set-vdisk command.

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

Specify the timeout in seconds. If the timeout is set to 0, the timeout is disabled and I/O is blocked while the service domain is down (this is the default setting and behavior).

Alternatively, the timeout can be set by adding the following line to the /etc/system file on the guest domain.

```
set vdc:vdc_timeout=seconds
```

Note – If this tunable is set, it overwrites any timeout setting done using the ldm CLI. Also, the tunable sets the timeout for all virtual disks in the guest domain.

Virtual Disk and SCSI

If a physical SCSI disk or LUN is exported as a full disk, the corresponding virtual disk supports the user SCSI command interface, uscsi(7I) and multihost disk control operations mhd(7i). Other virtual disks, such as virtual disks having a file or a volume as a back end, do not support these interfaces.

As a consequence, applications or product features using SCSI commands (such as Solaris Volume Manager metaset, or Oracle Solaris Cluster shared devices) can be used in guest domains only with virtual disks having a physical SCSI disk as a back end.

> **Note –** SCSI operations are effectively executed by the service domain, which manages the physical SCSI disk or LUN used as a virtual disk back end. In particular, SCSI reservations are done by the service domain. Therefore, applications running in the service domain and in guest domains should not issue SCSI commands to the same physical SCSI disks; otherwise, this can lead to an unexpected disk state.

Virtual Disk and the `format(1M)` Command

The format(1M) command recognizes all virtual disks that are present in a domain. However, for virtual disks that are exported as single-slice disks, the format command cannot change the partition table of the virtual disk. Commands such as label will fail unless you try to write a disk label similar to the one that is already associated with the virtual disk.

Virtual disks whose back ends are SCSI disks support all format(1M) subcommands. Virtual disks whose back ends are not SCSI disks do not support some format(1M) subcommands, such as repair and defect. In that case, the behavior of format(1M) is similar to the behavior of Integrated Drive Electronics (IDE) disks.

Using ZFS With Virtual Disks

This section describes using the Zettabyte File System (ZFS) to store virtual disk back ends exported to guest domains. ZFS provides a convenient and powerful solution to create and manage virtual disk back ends. ZFS enables:

- Storing disk images in ZFS volumes or ZFS files
- Using snapshots to backup disk images
- Using clones to duplicate disk images and provision additional domains

Refer to the *Oracle Solaris ZFS Administration Guide* for more information about using the ZFS.

In the following descriptions and examples, the primary domain is also the service domain where disk images are stored.

Configuring a ZFS Pool in a Service Domain

To store the disk images, first create a ZFS storage pool in the service domain. For example, this command creates the ZFS storage pool ldmpool containing the disk c1t50d0 in the primary domain.

```
primary# zpool create ldmpool c1t50d0
```

Storing Disk Images With ZFS

The following command creates a disk image for guest domain `ldg1`. A ZFS file system for this guest domain is created, and all disk images of this guest domain will be stored on that file system.

```
primary# zfs create ldmpool/ldg1
```

Disk images can be stored on ZFS volumes or ZFS files. Creating a ZFS volume, whatever its size, is quick using the `zfs create -V` command. On the other hand, ZFS files have to be created using the `mkfile` command. The command can take some time to complete, especially if the file to create is quite large, which is often the case when creating a disk image.

Both ZFS volumes and ZFS files can take advantage of ZFS features such as the snapshot and clone features, but a ZFS volume is a pseudo device while a ZFS file is a regular file.

If the disk image is to be used as a virtual disk onto which an OS is installed, the disk image must be large enough to accommodate the OS installation requirements. This size depends on the version of the OS and on the type of installation performed. If you install the Oracle Solaris OS, you can use a disk size of 20 Gbytes to accommodate any type of installation of any version of the Oracle Solaris OS.

Examples of Storing Disk Images With ZFS

The following examples:

1. Create a 20-gigabyte image on a ZFS volume or file.
2. Export the ZFS volume or file as a virtual disk. The syntax to export a ZFS volume or file is the same, but the path to the back end is different.
3. Assign the exported ZFS volume or file to a guest domain.

When the guest domain is started, the ZFS volume or file appears as a virtual disk on which the Oracle Solaris OS can be installed.

▼ Create a Disk Image Using a ZFS Volume

● **For example, create a 20-gigabyte disk image on a ZFS volume.**
```
primary# zfs create -V 20gb ldmpool/ldg1/disk0
```

▼ Create a Disk Image Using a ZFS File

● **For example, create a 20-gigabyte disk image on a ZFS volume.**
```
primary# zfs create ldmpool/ldg1/disk0
primary# mkfile 20g /ldmpool/ldg1/disk0/file
```

▼ **Export the ZFS Volume**

● **Export the ZFS volume as a virtual disk.**

```
primary# ldm add-vdsdev /dev/zvol/dsk/ldmpool/ldg1/disk0 ldg1_disk0@primary-vds0
```

▼ **Export the ZFS File**

● **Export the ZFS file as a virtual disk.**

```
primary# ldm add-vdsdev /ldmpool/ldg1/disk0/file ldg1_disk0@primary-vds0
```

▼ **Assign the ZFS Volume or File to a Guest Domain**

● **Assign the ZFS volume or file to a guest domain; in this example, ldg1.**

```
primary# ldm add-vdisk disk0 ldg1_disk0@primary-vds0 ldg1
```

Creating a Snapshot of a Disk Image

When your disk image is stored on a ZFS volume or on a ZFS file, you can create snapshots of this disk image by using the ZFS snapshot command.

Before you create a snapshot of the disk image, ensure that the disk is not currently in use in the guest domain to ensure that data currently stored on the disk image are coherent. There are several ways to ensure that a disk is not in use in a guest domain. You can either:

- Stop and unbind the guest domain. This is the safest solution, and this is the only solution available if you want to create a snapshot of a disk image used as the boot disk of a guest domain.
- Alternatively, you can unmount any slices of the disk you want to snapshot used in the guest domain, and ensure that no slice is in use the guest domain.

In this example, because of the ZFS layout, the command to create a snapshot of the disk image is the same whether the disk image is stored on a ZFS volume or on a ZFS file.

▼ **Create a Snapshot of a Disk Image**

● **Create a snapshot of the disk image that was created for the ldg1 domain, for example.**

```
primary# zfs snapshot ldmpool/ldg1/disk0@version_1
```

Using Clone to Provision a New Domain

Once you have created a snapshot of a disk image, you can duplicate this disk image by using the ZFS clone command. Then the cloned image can be assigned to another domain. Cloning a boot disk image quickly creates a boot disk for a new guest domain without having to perform the entire Oracle Solaris OS installation process.

For example, if the disk0 created was the boot disk of domain ldg1, do the following to clone that disk to create a boot disk for domain ldg2.

```
primary# zfs create ldmpool/ldg2
primary# zfs clone ldmpool/ldg1/disk0@version_1 ldmpool/ldg2/disk0
```

Then ldompool/ldg2/disk0 can be exported as a virtual disk and assigned to the new ldg2 domain. The domain ldg2 can directly boot from that virtual disk without having to go through the OS installation process.

Cloning a Boot Disk Image

When a boot disk image is cloned, the new image is exactly the same as the original boot disk, and it contains any information that has been stored on the boot disk before the image was cloned, such as the host name, the IP address, the mounted file system table, or any system configuration or tuning.

Because the mounted file system table is the same on the original boot disk image and on the cloned disk image, the cloned disk image has to be assigned to the new domain in the same order as it was on the original domain. For example, if the boot disk image was assigned as the first disk of the original domain, then the cloned disk image has to be assigned as the first disk of the new domain. Otherwise, the new domain is unable to boot.

If the original domain was configured with a static IP address, then a new domain using the cloned image starts with the same IP address. In that case, you can change the network configuration of the new domain by using the sys-unconfig(1M) command. To avoid this problem you can also create a snapshot of a disk image of an unconfigured system.

If the original domain was configured with the Dynamic Host Configuration Protocol (DHCP), then a new domain using the cloned image also uses DHCP. In that case, you do not need to change the network configuration of the new domain because it automatically receives an IP address and its network configuration as it boots.

Note – The host ID of a domain is not stored on the boot disk, but it is assigned by the Logical Domains Manager when you create a domain. Therefore, when you clone a disk image, the new domain does not keep the host ID of the original domain.

▼ Create a Snapshot of a Disk Image of an Unconfigured System

1 **Bind and start the original domain.**

2 **Execute the `sys-config` command.**

3 **After the `sys-config` command completes, the domain halts.**

4 **Stop and unbind the domain; do** *not* **reboot it.**

5 **Take a snapshot of the domain boot disk image.**

For example:

```
primary# zfs snapshot ldmpool/ldg1/disk0@unconfigured
```

At this point you have the snapshot of the boot disk image of an unconfigured system.

6 **Clone this image to create a new domain which, when first booted, asks for the configuration of the system.**

Using Volume Managers in a Logical Domains Environment

This section describes using volume managers in a Logical Domains environment.

Using Virtual Disks on Top of Volume Managers

Any Zettabyte File System (ZFS), Solaris Volume Manager, or Veritas Volume Manager (VxVM) volume can be exported from a service domain to a guest domain as a virtual disk. A volume can be exported either as a single-slice disk (if the `slice` option is specified with the `ldm add-vdsdev` command) or as a full disk.

Note – The remainder of this section uses an Solaris Volume Manager volume as an example. However, the discussion also applies to ZFS and VxVM volumes.

The following examples show how to export a volume as a single-slice disk.

The virtual disk in the guest domain (for example, `/dev/dsk/c0d2s0`) is directly mapped to the associated volume (for example, `/dev/md/dsk/d0`), and data stored onto the virtual disk from the guest domain are directly stored onto the associated volume with no extra metadata. So data stored on the virtual disk from the guest domain can also be directly accessed from the service domain through the associated volume.

Examples

- If the Solaris Volume Manager volume d0 is exported from the primary domain to domain1, then the configuration of domain1 requires some extra steps.

```
primary# metainit d0 3 1 c2t70d0s6 1 c2t80d0s6 1 c2t90d0s6
primary# ldm add-vdsdev options=slice /dev/md/dsk/d0 vol3@primary-vds0
primary# ldm add-vdisk vdisk3 vol3@primary-vds0 domain1
```

- After domain1 has been bound and started, the exported volume appears as /dev/dsk/c0d2s0, for example, and you can use it.

```
domain1# newfs /dev/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 on Top of Solaris Volume Manager

When a RAID or mirror Solaris Volume Manager volume is used as a virtual disk by another domain, then it has to be exported without setting the exclusive (excl) option. Otherwise, if there is a failure on one of the components of the Solaris Volume Manager volume, then the recovery of the Solaris Volume Manager volume using the metareplace command or using a hot spare does not start. The metastat command sees the volume as resynchronizing, but the resynchronization does not progress.

For example, /dev/md/dsk/d0 is a RAID Solaris Volume Manager volume exported as a virtual disk with the excl option to another domain, and d0 is configured with some hot-spare devices. If a component of d0 fails, Solaris Volume Manager replaces the failing component with a hot spare and resynchronizes the Solaris Volume Manager volume. However, the resynchronization does not start. The volume is reported as resynchronizing, but the resynchronization does not progress.

```
# metastat d0
d0: RAID
    State: Resyncing
    Hot spare pool: hsp000
    Interlace: 32 blocks
    Size: 20097600 blocks (9.6 GB)
Original device:
    Size: 20100992 blocks (9.6 GB)
Device                                        Start Block  Dbase    State Reloc
c2t2d0s1                                              330  No        Okay Yes
c4t12d0s1                                             330  No        Okay Yes
/dev/dsk/c10t600C0FF00000000000015153295A4B100d0s1   330  No  Resyncing Yes
```

In such a situation, the domain using the Solaris Volume Manager volume as a virtual disk has to be stopped and unbound to complete the resynchronization. Then the Solaris Volume Manager volume can be resynchronized using the metasync command.

```
# metasync d0
```

Using Virtual Disks When VxVM Is Installed

When the Veritas Volume Manager (VxVM) is installed on your system, and if Veritas Dynamic Multipathing (DMP) is enabled on a physical disk or partition you want to export as virtual disk, then you have to export that disk or partition without setting the (non-default) excl option. Otherwise, you receive an error in /var/adm/messages while binding a domain that uses such a disk.

```
vd_setup_vd():  ldi_open_by_name(/dev/dsk/c4t12d0s2) = errno 16
vds_add_vd():   Failed to add vdisk ID 0
```

You can check if Veritas DMP is enabled by checking multipathing information in the output of the command vxdisk list; for example:

```
# vxdisk list Disk_3
Device:    Disk_3
devicetag: Disk_3
type:      auto
info:      format=none
flags:     online ready private autoconfig invalid
pubpaths:  block=/dev/vx/dmp/Disk_3s2 char=/dev/vx/rdmp/Disk_3s2
guid:      -
udid:      SEAGATE%5FST336753LSUN36G%5FDISKS%5F3032333948303144304E0000
site:      -
Multipathing information:
numpaths:  1
c4t12d0s2  state=enabled
```

Alternatively, if Veritas DMP is enabled on a disk or a slice that you want to export as a virtual disk with the excl option set, then you can disable DMP using the vxdmpadm command. For example:

```
# vxdmpadm -f disable path=/dev/dsk/c4t12d0s2
```

Using Volume Managers on Top of Virtual Disks

This section describes using volume managers on top of virtual disks.

Using ZFS on Top of Virtual Disks

Any virtual disk can be used with ZFS. A ZFS storage pool (zpool) can be imported in any domain that sees all the storage devices that are part of this zpool, regardless of whether the domain sees all these devices as virtual devices or real devices.

Using Solaris Volume Manager on Top of Virtual Disks

Any virtual disk can be used in the Solaris Volume Manager local disk set. For example, a virtual disk can be used for storing the Solaris Volume Manager metadevice state database, metadb(1M), of the local disk set or for creating Solaris Volume Manager volumes in the local disk set.

Any virtual disk whose back end is a SCSI disk can be used in a Solaris Volume Manager shared disk set, metaset(1M). Virtual disks whose back ends are not SCSI disks cannot be added into a Solaris Volume Manager share disk set. Trying to add a virtual disk whose back end is not a SCSI disk into a Solaris Volume Manager shared disk set fails with an error similar to the following.

```
# metaset -s test -a c2d2
metaset: domain1: test: failed to reserve any drives
```

Using VxVM on Top of Virtual Disks

For VxVM support in guest domains, refer to the VxVM documentation from Symantec.

8

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

A virtual network allows domains to communicate with each other without using any external physical networks. A virtual network also can allow domains to use the same physical network interface to access a physical network and communicate with remote systems. A virtual network is created by having a virtual switch to which you can connect virtual network devices.

Virtual Switch

A virtual switch (vsw) is a component running in a service domain and managed by the virtual switch driver. A virtual switch can be connected to some guest domains to enable network communications between those domains. In addition, if the virtual switch is associated also with a physical network interface, then this allows network communications between guest domains and the physical network over the physical network interface. A virtual switch also has a network interface, vsw*n*, which allows the service domain to communicate with the other domains connected to that virtual switch. It can be used like any regular network interface and configured with the ifconfig(1M) command.

Note – When a virtual switch is added to a service domain, its network interface is not plumbed. 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 plumbed and configured in the service domain. See "Enabling Networking Between the Control/Service Domain and Other Domains" on page 51 for instructions.

Virtual Network Device

A virtual network (vnet) device is a virtual device that is defined in a domain connected to a virtual switch. A virtual network device is managed by the virtual network driver, and it is connected to a virtual network through the hypervisor using logical domain channels (LDCs).

A virtual network device can be used as a network interface with the name vnet*n*, which can be used like any regular network interface and configured with the ifconfig(1M) command.

FIGURE 8–1 Setting Up a Virtual Network
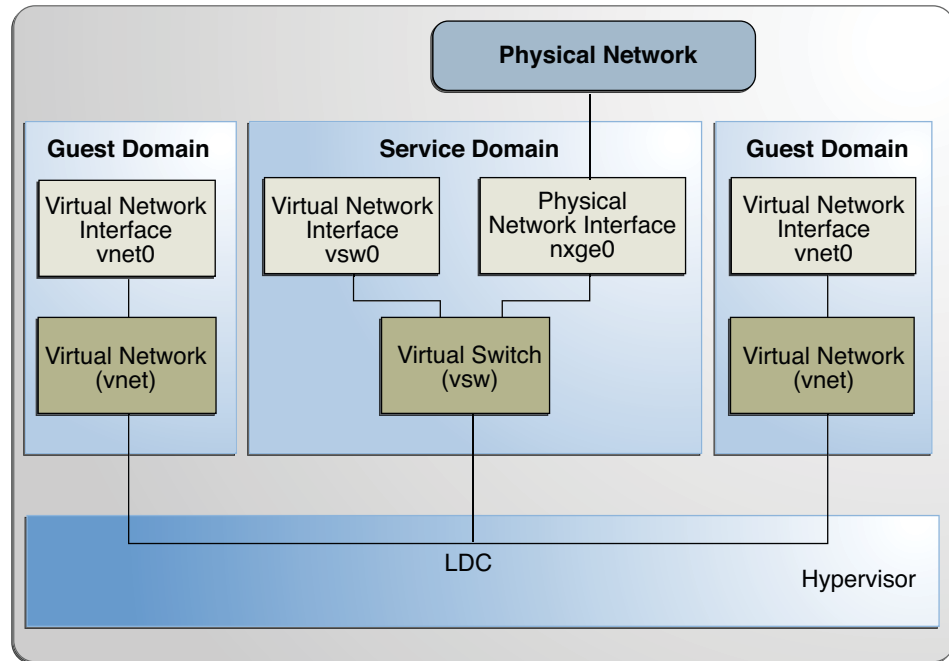
Following is a explanation for the example in Figure 8–1.

- The virtual switch in the service domain is connected to the guest domains. This allows guest domains to communicate with each other.

- The virtual switch is also connected to the physical network interface nxge0. This allows guest domains to communicate with the physical network.

- The virtual switch network interface vsw0 is plumbed in the service domain, so this allows the two guest domains to communicate with the service domain.

- The virtual switch network interface vsw0 in the service domain can be configured using the ifconfig(1M) command.

- The virtual network interfaces vnet0 in the guest domains can be configured using the ifconfig(1M) command.

Basically the virtual switch behaves like a regular physical network switch and switches network packets between the different systems, such as guest domains, service domain, and physical network, to which it is connected.

Managing a Virtual Switch

This section describes adding a virtual switch to a domain, setting options for a virtual switch, and removing a virtual switch.

▼ Add a Virtual Switch

● **Use the following command syntax to add a virtual switch.**

```
# ldm add-vsw [default-vlan-id=vlan-id] [pvid=[port-vlan-id]] [vid=vlan-id1,vlan-id2,...]
[linkprop=phys-state] [mac-addr=num] [net-dev=device] [mode=sc] [mtu=size]
[id=switch-id] vswitch-name ldom
```

Where:

- default-vlan-id=*vlan-id* specifies the default virtual local area network (VLAN) to which a virtual switch and its associated virtual network devices belong to implicitly, in untagged mode. It serves as the default port VLAN id (pvid) of the virtual switch and virtual network devices. Without this option, the default value of this property is 1. Normally, you would not need to use this option. It is provided only as a way to change the default value of 1. See "Using VLAN Tagging" on page 125 for more information.

- pvid=*port-vlan-id* specifies the VLAN to which the virtual switch needs to be a member, in untagged mode. See "Using VLAN Tagging" on page 125 for more information.

- vid=*vlan-id* specifies one or more VLANs to which a virtual switch needs to be a member, in tagged mode. See "Using VLAN Tagging" on page 125 for more information.

- linkprop=phys-state specifies whether the virtual device reports its link status based on the underlying physical network device. When linkprop=phys-state is specified on the command line, the virtual device link status reflects the physical link state. By default, the virtual device link status does not reflect the physical link state.

   Specify this option to use link-based IPMP. See "Using Link-Based IPMP in Logical Domains Virtual Networking" on page 121.

- mac-addr=*num* is the MAC address to be used by this switch. The number must be in standard octet notation; for example, 80:00:33:55:22:66. If you do not specify a MAC address, the switch is automatically assigned an address from the range of public MAC addresses allocated to the Logical Domains Manager. See "Assigning MAC Addresses Automatically or Manually" on page 114 for more information.

- net-dev=*device* is the path to the network device over which this switch operates.

- mode=sc enables virtual networking support for prioritized processing of Oracle Solaris Cluster heartbeat packets in a Logical Domains environment. Applications like Oracle Solaris Cluster need to ensure that high priority heartbeat packets are not dropped by congested virtual network and switch devices. This option prioritizes Oracle Solaris Cluster heartbeat frames and ensures that they are transferred in a reliable manner.

You must set this option when running Oracle Solaris Cluster in a Logical Domains environment and using guest domains as Oracle Solaris Cluster nodes. Do *not* set this option when you are not running Oracle Solaris Cluster software in guest domains, because you could impact virtual network performance.

- mtu=*size* specifies the maximum transmission unit (MTU) of a virtual switch device. Valid values are in the range of 1500-16000.

- id=*switch-id* is 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. See "Virtual Device Identifier and Network Interface Name" on page 111.

- *vswitch-name* is the unique name of the switch that is to be exported as a service. Clients (network) can attach to this service.

- *ldom* specifies the logical domain in which to add a virtual switch.

▼ Set Options for an Existing Virtual Switch

● **Use the following command syntax to set options for a virtual switch that already exists.**

```
# ldm set-vsw [pvid=[port-vlan-id]] [vid=[vlan-id1,vlan-id2,...]] [mac-addr=num]
  [linkprop=[phys-state]] [net-dev=[device]] [mode=[sc]] [mtu=[size]] vswitch-name
```

Where:

- mode= (left blank) stops special processing of Oracle Solaris Cluster heartbeat packets.

- Otherwise, the command arguments are the same as described in "Add a Virtual Switch" on page 108.

▼ Remove a Virtual Switch

● **Use the following command syntax to remove a virtual switch.**

```
# ldm rm-vsw [-f] vswitch-name
```

Where:

- -f attempts to force the removal of a virtual switch. The removal might fail.
- *vswitch-name* is the name of the switch that is to be removed as a service.

Managing a Virtual Network Device

This section describes adding a virtual network device to a domain, setting options for an existing virtual network device, and removing a virtual network device.

▼ Add a Virtual Network Device

● **Use the following command syntax to add a virtual network device.**

```
# ldm add-vnet [mac-addr=num] [mode=hybrid] [pvid=[port-vlan-id]]
[linkprop=phys-state] [vid=vlan-id1,vlan-id2,...] [mtu=size] [id=network-id]
if-name vswitch-name ldom
```

Where:

- mac-addr=*num* is the MAC address for this network device. The number must be in standard octet notation; for example, 80:00:33:55:22:66. See "Assigning MAC Addresses Automatically or Manually" on page 114 for more information.

- mode=hybrid to request the system to use NIU Hybrid I/O on this vnet if possible. If it is not possible, the system reverts to virtual I/O. This hybrid mode is considered a delayed reconfiguration if set on an active vnet. See "Using NIU Hybrid I/O" on page 128 for more information.

- pvid=*port-vlan-id* specifies the VLAN to which the virtual network device needs to be a member, in untagged mode. See "Using VLAN Tagging" on page 125 for more information.

- linkprop=phys-state specifies whether the virtual network device reports its link status based on the underlying physical network device. When linkprop=phys-state is specified on the command line, the virtual network device link status reflects the physical link state. By default, the virtual network device link status does not reflect the physical link state.

  Specify this option to use link-based IPMP. See "Using Link-Based IPMP in Logical Domains Virtual Networking" on page 121.

- vid=*vlan-id* specifies one or more VLANs to which a virtual network device needs to be a member, in tagged mode. See "Using VLAN Tagging" on page 125 for more information.

- mtu=*size* specifies the maximum transmission unit (MTU) of a virtual network device. Valid values are in the range of 1500-16000.

- id=*network-id* is the ID of a new virtual network device. By default, ID values are generated automatically, so set this property if you need to match an existing device name in the OS. See "Virtual Device Identifier and Network Interface Name" on page 111.

- *if-name*, interface name, is a unique name to the logical domain, assigned to this virtual network device instance for reference on subsequent ldm set-vnet or ldm rm-vnet commands.

- *vswitch-name* is the name of an existing network service (virtual switch) to which to connect.

- *ldom* specifies the logical domain to which to add the virtual network device.

▼ Set Options for an Existing Virtual Network Device

● **Use the following command syntax to set options for a virtual network device that already exists.**

```
# ldm set-vnet [mac-addr=num] [vswitch=vswitch-name] [mode=[hybrid]]
  [pvid=[port-vlan-id]] [linkprop=[phys-state]] [vid=[vlan-id1,vlan-id2,...]]
  [mtu=[size]] if-name ldom
```

Where:

- mode= (left blank) disables NIU Hybrid I/O.
- *if-name*, interface name, is the unique name assigned to the virtual network device you want to set.
- *ldom* specifies the logical domain from which to remove the virtual network device.
- Otherwise, the command arguments are the same as described in "Add a Virtual Network Device" on page 110.

▼ Remove a Virtual Network Device

● **Use the following command syntax to remove a virtual network device.**

```
# ldm rm-vnet [-f] if-name ldom
```

Where:

- -f attempts to force the removal of a virtual network device from a logical domain. The removal might fail.
- *if-name*, interface name, is the unique name assigned to the virtual network device you want to remove.
- *ldom* specifies the logical domain from which to remove the virtual network device.

Virtual Device Identifier and Network Interface Name

When you add a virtual switch or virtual network device to a domain, you can specify its device number by setting the id property.

```
# ldm add-vsw [id=switch-id] vswitch-name ldom
# ldm add-vnet [id=network-id] if-name vswitch-name ldom
```

Each virtual switch and virtual network device of a domain has a unique device number that is assigned when the domain is bound. If a virtual switch or virtual network device was added with

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

The following example shows that the `primary` domain has one virtual switch, `primary-vsw0`. This virtual switch has a device number of `0` (`switch@0`).

```
primary# ldm list-bindings primary
...
VSW
    NAME         MAC                  NET-DEV DEVICE   DEFAULT-VLAN-ID PVID VID MTU MODE
    primary-vsw0 00:14:4f:fb:54:f2 nxge0    switch@0 1                1    5,6 1500
...
```

The following example shows that the `ldg1` domain has two virtual network devices: `vnet` and `vnet1`. The `vnet` device has a device number of `0` (`network@0`) and the `vnet1` device has a device number of `1` (`network@1`).

```
primary# ldm list-bindings ldg1
...
NETWORK
    NAME  SERVICE             DEVICE    MAC               MODE   PVID VID MTU
    vnet  primary-vsw0@primary network@0 00:14:4f:fb:e0:4b hybrid 1        1500
    ...
    vnet1 primary-vsw0@primary network@1 00:14:4f:f8:e1:ea        1        1500
...
```

When a domain with a virtual switch is running the Oracle Solaris OS, the virtual switch has a network interface, vsw$N$. However, the network interface number of the virtual switch, N, is not necessarily the same as the device number of the virtual switch, n.

Similarly, when a domain with a virtual network device is running the Oracle Solaris OS, the virtual network device has a network interface, vnet$N$. However, the network interface number of the virtual network device, N, is not necessarily the same as the device number of the virtual network device, n.

⚠️ **Caution** – The Oracle Solaris OS preserves the mapping between the name of a network interface and a virtual switch or virtual network based on the device number. If a device number is not explicitly assigned to a virtual switch or virtual network device, its device number can change when the domain is unbound and is later bound again. In that case, the network interface name assigned by the OS running in the domain can also change and break the existing configuration of the system. This might happen, for example, when a virtual switch or a virtual network interface is removed from the configuration of the domain.

You cannot use the ldm list-* commands to directly determine the Oracle Solaris OS network interface name that corresponds to a virtual switch or virtual network device. However, you can obtain this information by using a combination of the output from ldm list -l command and from the entries under /devices on the Oracle Solaris OS.

▼ Find Oracle Solaris OS Network Interface Name

In this example procedure, guest domain ldg1 contains two virtual network devices, net-a and net-c. To find the Oracle Solaris OS network interface name in ldg1 that corresponds to net-c, do the following. This example also shows differences if you are looking for the network interface name of a virtual switch instead of a virtual network device.

1   Use the ldm command to find the virtual network device number for net-c.

```
# ldm list -l ldg1
...
NETWORK
NAME            SERVICE                  DEVICE        MAC
net-a           primary-vsw0@primary     network@0     00:14:4f:f8:91:4f
net-c           primary-vsw0@primary     network@2     00:14:4f:f8:dd:68
...
```

The virtual network device number for net-c is 2 (network@2).

To determine the network interface name of a virtual switch, find the virtual switch device number, *n* as switch@*n*.

2   To find the corresponding network interface on ldg1, log into ldg1 and find the entry for this device number under /devices.

```
# uname -n
ldg1
# find /devices/virtual-devices@100 -type c -name network@2\*
/devices/virtual-devices@100/channel-devices@200/network@2:vnet1
```

The network interface name is the part of the entry after the colon; that is, vnet1.

To determine the network interface name of a virtual switch, replace the argument to the -name option with virtual-network-switch@*n**. Then, find the network interface with the name vsw*N*.

3   Plumb vnet1 to see that it has the MAC address 00:14:4f:f8:dd:68 as shown in the ldm list -l output for net-c in Step 1.

```
# ifconfig vnet1
vnet1: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 0.0.0.0 netmask 0
        ether 0:14:4f:f8:dd:68
```

Assigning MAC Addresses Automatically or Manually

You must have enough media access control (MAC) addresses to assign to the number of logical domains, virtual switches, and virtual networks you are going to use. You can have the Logical Domains Manager automatically assign MAC addresses to a logical domain, a virtual network (vnet), and a virtual switch (vsw), or you can manually assign MAC addresses from your own pool of assigned MAC addresses. The ldm subcommands that set MAC addresses are add-domain, add-vsw, set-vsw, add-vnet, and set-vnet. If you do not specify a MAC address in these subcommands, the Logical Domains Manager assigns one automatically.

The advantage to having the Logical Domains Manager assign the MAC addresses is that it utilizes the block of MAC addresses dedicated for use with logical domains. Also, the Logical Domains Manager detects and prevents MAC address collisions with other Logical Domains Manager instances on the same subnet. This frees you from having to manually manage your pool of MAC addresses.

MAC address assignment happens as soon as a logical domain is created or a network device is configured into a domain. In addition, the assignment is persistent until the device, or the logical domain itself, is removed.

Range of MAC Addresses Assigned to Logical Domains

Logical Domains have been assigned the following block of 512K MAC addresses:

```
00:14:4F:F8:00:00 ~ 00:14:4F:FF:FF:FF
```

The lower 256K addresses are used by the Logical Domains Manager for *automatic MAC address allocation*, and you *cannot* manually request an address in this range:

```
00:14:4F:F8:00:00 - 00:14:4F:FB:FF:FF
```

You can use the upper half of this range for *manual MAC address allocation*:

```
00:14:4F:FC:00:00 - 00:14:4F:FF:FF:FF
```

Automatic Assignment Algorithm

When you do not specify a MAC address in creating logical domain or a network device, the Logical Domains Manager automatically allocates and assigns a MAC address to that logical domain or network device. To obtain this MAC address, the Logical Domains Manager iteratively attempts to select an address and then checks for potential collisions.

Before selecting a potential address, the Logical Domains Manager first looks to see if it has a recently freed, automatically assigned address saved in a database for this purpose (see "Freed MAC Addresses" on page 116). If so, the Logical Domains Manager selects its candidate address from the database.

If no recently freed addresses are available, the MAC address is randomly selected from the 256K range of addresses set aside for this purpose. The MAC address is selected randomly to lessen the chance of a duplicate MAC address being selected as a candidate.

The address selected is then checked against other Logical Domains Managers on other systems to prevent duplicate MAC addresses from actually being assigned. The algorithm employed is described in "Duplicate MAC Address Detection" on page 115. If the address is already assigned, the Logical Domains Manager iterates, choosing another address, and again checking for collisions. This continues until a MAC address is found that is not already allocated, or a time limit of 30 seconds has elapsed. If the time limit is reached, then the creation of the device fails, and an error message similar to the following is shown.

```
Automatic MAC allocation failed.  Please set the vnet MAC address manually.
```

Duplicate MAC Address Detection

To prevent the same MAC address from being allocated to different devices, one Logical Domains Manager checks with other Logical Domains Managers on other systems by sending a multicast message over the control domain's default network interface, including the address that the Logical Domains Manager wants to assign to the device. The Logical Domains Manager attempting to assign the MAC address waits for one second for a response back. If a different device on another Logical Domains-enabled system has already been assigned that MAC address, the Logical Domains Manager on that system sends back a response containing the MAC address in question. If the requesting Logical Domains Manager receives a response, it knows the chosen MAC address has already been allocated, chooses another, and iterates.

By default, these multicast messages are sent only to other managers on the same subnet; the default time-to-live (TTL) is 1. The TTL can be configured using the Service Management Facilities (SMF) property ldmd/hops.

Each Logical Domains Manager is responsible for:

- Listening for multicast messages
- Keeping track of MAC addresses assigned to its domains
- Looking for duplicates
- Responding so that duplicates do not occur

If the Logical Domains Manager on a system is shut down for any reason, duplicate MAC addresses could occur while the Logical Domains Manager is down.

Automatic MAC allocation occurs at the time the logical domain or network device is created and persists until the device or the logical domain is removed.

Note – A detection check for duplicate MAC addresses is performed when the logical domain or network device is created, and the logical domain is started.

Freed MAC Addresses

When a logical domain or a device associated with an automatic MAC address is removed, that MAC address is saved in a database of recently freed MAC addresses for possible later use on that system. These MAC addresses are saved to prevent the exhaustion of Internet Protocol (IP) addresses from a Dynamic Host Configuration Protocol (DHCP) server. When DHCP servers allocate IP addresses, they do so for a period of time (the lease time). The lease duration is often configured to be quite long, generally hours or days. If network devices are created and removed at a high rate without the Logical Domains Manager reusing automatically allocated MAC addresses, the number of MAC addresses allocated could soon overwhelm a typically configured DHCP server.

When the Logical Domains Manager is requested to automatically obtain a MAC address for a logical domain or network device, it first looks to the freed MAC address database to see if there is a previously assigned MAC address it can reuse. If there is a MAC address available from this database, the duplicate MAC address detection algorithm is run. If the MAC address had not been assigned to someone else since it was previously freed, it will be reused and removed from the database. If a collision is detected, the address is simply removed from the database. The Logical Domains Manager then either tries the next address in the database or if none is available, randomly picks a new MAC address.

Using Network Adapters With Logical Domains

In a logical domains environment, the virtual switch service running in a service domain can directly interact with GLDv3-compliant network adapters. Though non-GLDv3 compliant network adapters can be used in these systems, the virtual switch cannot interface with them directly. See "Configuring Virtual Switch and Service Domain for NAT and Routing" on page 117 for information about how to use non-GLDv3 compliant network adapters.

For information about using link aggregations, see "Using Link Aggregation With a Virtual Switch" on page 132.

▼ Determine If a Network Adapter Is GLDv3-Compliant

1   Use the Oracle Solaris OS `dladm`(1M) command, where, for example, `bge0` is the network device name.

```
# dladm show-link bge0
bge0            type: non-vlan   mtu: 1500      device: bge0
```

2   Look at `type:` in the output:

- GLDv3-compliant drivers will have a type of `non-vlan` or `vlan`.
- Non-GLDv3-compliant drivers will have a type of `legacy`.

Configuring Virtual Switch and Service Domain for NAT and Routing

The virtual switch (`vsw`) is a layer-2 switch, that also can be used as a network device in the service domain. The virtual switch can be configured to act only as a switch between the virtual network (`vnet`) devices in the various logical domains but with no connectivity to a network outside the box through a physical device. In this mode, plumbing the `vsw` as a network device and enabling IP routing in the service domain enables virtual networks to communicate outside the box using the service domain as a router. This mode of operation is very essential to provide external connectivity to the domains when the physical network adapter is not GLDv3-compliant.

The advantages of this configuration are:

- The virtual switch does not need to use a physical device directly and can provide external connectivity even when the underlying device is not GLDv3-compliant.
- The configuration can take advantage of the IP routing and filtering capabilities of the Oracle Solaris OS.

FIGURE 8–2   Virtual Network Routing
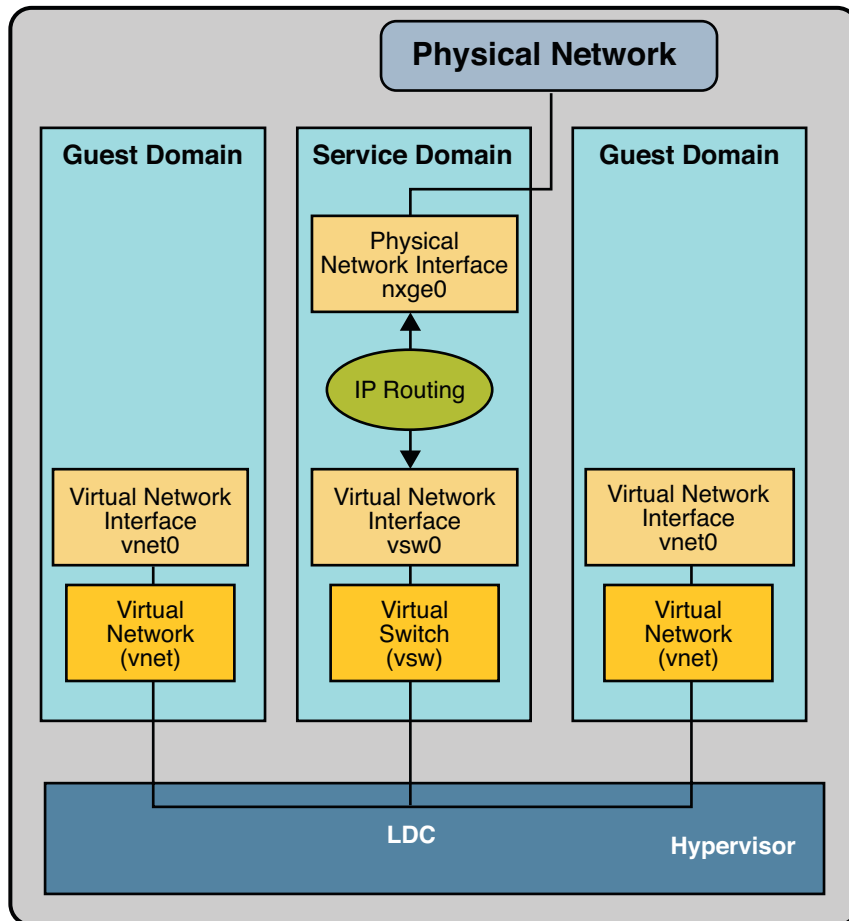
▼ Set Up the Virtual Switch to Provide External Connectivity to Domains

1   Create a virtual switch with no associated physical device.

If assigning an address, ensure that the virtual switch has an unique MAC address.

```
primary# ldm add-vsw [mac-addr=xx:xx:xx:xx:xx:xx] primary-vsw0 primary
```

2   Plumb the virtual switch as a network device in addition to the physical network device being used by the domain.

See "Configure the Virtual Switch as the Primary Interface" on page 51 for more information about plumbing the virtual switch.

3   **Configure the virtual switch device for DHCP, if needed.**

See "Configure the Virtual Switch as the Primary Interface" on page 51 for more information about configuring the virtual switch device for DHCP.

4   **Create the `/etc/dhcp.vsw` file, if needed.**

5   **Configure IP routing in the service domain, and set up required routing tables in all the domains.**

For information about how to do this, refer to "Packet Forwarding and Routing on IPv4 Networks" in *System Administration Guide: IP Services*.

Configuring IPMP in a Logical Domains Environment

The Logical Domains 1.3 release introduced support for link-based IPMP with virtual network devices. When configuring an IPMP group with virtual network devices, configure the group to use link-based detection. If using older versions of the Oracle VM Server for SPARC (Logical Domains) software, you can only configure probe-based detection with virtual network devices.

Configuring Virtual Network Devices Into an IPMP Group in a Domain

The following diagram shows two virtual networks (`vnet1` and `vnet2`) connected to separate virtual switch instances (`vsw0` and `vsw1`) in the service domain, which, in turn, use two different physical interfaces (`nxge0` and `nxge1`). In the event of a physical link failure in the service domain, the virtual switch device that is bound to that physical device detects the link failure. Then, the virtual switch device propagates the failure to the corresponding virtual network device that is bound to this virtual switch. The virtual network device sends notification of this link event to the IP layer in the guest `LDom_A`, which results in failover to the other virtual network device in the IPMP group.

FIGURE 8–3   Two Virtual Networks Connected to Separate Virtual Switch Instances
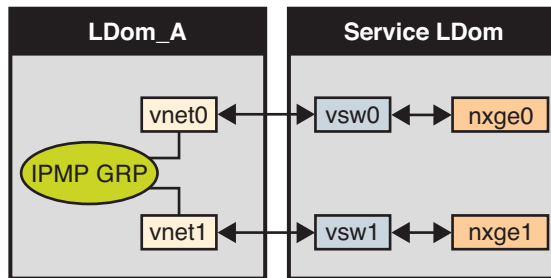
Further reliability can be achieved in the logical domain by connecting each virtual network device (vnet0 and vnet1) to virtual switch instances in different service domains (as shown in the following diagram). In this case, in addition to physical network failure, LDom_A can detect virtual network failure and trigger a failover following a service domain crash or shutdown.

FIGURE 8–4   Each Virtual Network Device Connected to Different Service Domains
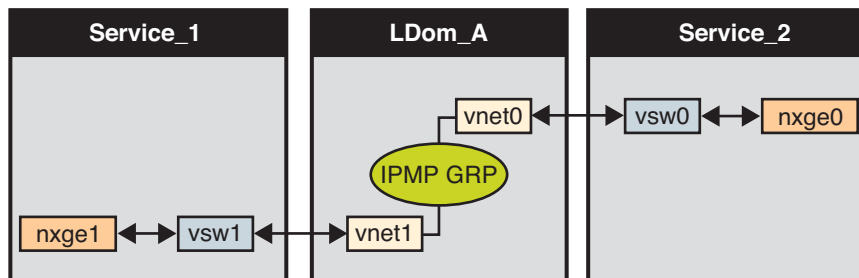
Refer to the Oracle Solaris 10 *System Administration Guide: IP Services* for more information about how to configure and use IPMP groups.

Configuring and Using IPMP in the Service Domain

IPMP can be configured in the service domain by configuring virtual switch interfaces into a group. The following diagram shows two virtual switch instances (vsw0 and vsw1) that are bound to two different physical devices. The two virtual switch interfaces can then be plumbed and configured into an IPMP group. In the event of a physical link failure, the virtual switch device that is bound to that physical device detects the link failure. Then, the virtual switch device sends notification of this link event to the IP layer in the service domain, which results in failover to the other virtual switch device in the IPMP group.

FIGURE 8–5   Two Virtual Switch Interfaces Configured as Part of IPMP Group
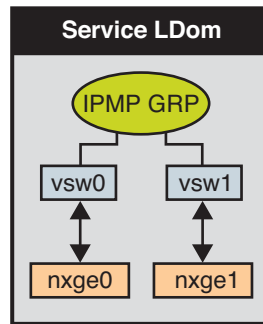
Using Link-Based IPMP in Logical Domains Virtual Networking

Starting with Logical Domains 1.3, the virtual network and virtual switch devices support link status updates to the network stack. By default, a virtual network device reports the status of its virtual link (its LDC to the virtual switch). This setup is enabled by default and does not require you to perform additional configuration steps.

Sometimes it might be necessary to detect physical network link state changes. For instance, if a physical device has been assigned to a virtual switch, even if the link from a virtual network device to its virtual switch device is up, the physical network link from the service domain to the external network might be down. In such a case, it might be necessary to obtain and report the physical link status to the virtual network device and its stack.

The linkprop=phys-state option can be used to configure physical link state tracking for virtual network devices as well as for virtual switch devices. When this option is enabled, the virtual device (virtual network or virtual switch) reports its link state based on the physical link state while it is plumbed as an interface in the domain. You can use standard Oracle Solaris network administration commands such as dladm and ifconfig to check the link status. See the dladm(1M) and ifconfig(1M) man pages. In addition, the link status is also logged in the /var/adm/messages file.

Note – You can run both link-state-unaware and link-state-aware vnet and vsw drivers concurrently on a Logical Domains system. However, if you intend to configure link-based IPMP, you must install the link-state-aware driver. If you intend to enable physical link state updates, upgrade both the vnet and vsw drivers to the Oracle Solaris 10 9/10 OS, and run at least Version 1.3 of the Logical Domains Manager.

▼ **Configure Physical Link Status Updates**

This procedure shows how to enable physical link status updates for virtual network devices.

You can also enable physical link status updates for a virtual switch device by following similar steps and specifying the `linkprop=phys-state` option to the `ldm add-vsw` and `ldm set-vsw` commands.

Note – You need to use the `linkprop=phys-state` option only if the virtual switch device itself is plumbed as an interface. If `linkprop=phys-state` is specified and the physical link is down, the virtual network device reports its link status as down, even if the connection to the virtual switch is up. This situation occurs because the Oracle Solaris OS does not currently provide interfaces to report two distinct link states, such as virtual-link-state and physical-link-state.

---

1 Become superuser or assume an equivalent role.

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

2 Enable physical link status updates for the virtual device.

You can enable physical link status updates for a virtual network device in the following ways:

- Create a virtual network device by specifying `linkprop=phys-state` when running the `ldm add-vnet` command.

  Specifying the `linkprop=phys-state` option configures the virtual network device to obtain physical link state updates and report them to the stack.

  Note – If `linkprop=phys-state` is specified and the physical link is down (even if the connection to the virtual switch is up), the virtual network device reports its link status as down. This situation occurs because the Oracle Solaris OS does not currently provide interfaces to report two distinct link states, such as virtual-link-state and physical-link-state.

  ```
  # ldm add-vnet linkprop=phys-state if-name vswitch-name ldom
  ```

  The following example enables physical link status updates for vnet0 connected to primary-vsw0 on the logical domain ldom1:

  ```
  # ldm add-vnet linkprop=phys-state vnet0 primary-vsw0 ldom1
  ```

- Modify an existing virtual network device by specifying `linkprop=phys-state` when running the `ldm set-vnet` command.

  ```
  # ldm set-vnet linkprop=phys-state if-name ldom
  ```

  The following example enables physical link status updates for vnet0 on the logical domain ldom1:

  ```
  # ldm set-vnet linkprop=phys-state vnet0 ldom1
  ```

To disable physical link state updates, specify `linkprop=` by running the `ldm set-vnet` command.

The following example disables physical link status updates for vnet0 on the logical domain ldom1:

```
# ldm set-vnet linkprop= vnet0 ldom1
```

Example 8–1   Configuring Link-Based IPMP

The following examples show how to configure link-based IPMP both with and without enabling physical link status updates:

- The following example configures two virtual network devices on a domain. Each virtual network device is connected to a separate virtual switch device on the service domain to use link-based IPMP.

  Note – Test addresses are not configured on these virtual network devices. Also, you do not need to perform additional configuration when you use the ldm add-vnet command to create these virtual network devices.

  The following commands add the virtual network devices to the domain. Note that because linkprop=phys-state is not specified, only the link to the virtual switch is monitored for state changes.

  ```
  # ldm add-vnet vnet0 primary-vsw0 ldom1
  # ldm add-vnet vnet1 primary-vsw1 ldom1
  ```

  The following commands configure the virtual network devices on the guest domain and assign them to an IPMP group. Note that test addresses are not configured on these virtual network devices because link-based failure detection is being used.

  ```
  # ifconfig vnet0 plumb
  # ifconfig vnet1 plumb
  # ifconfig vnet0 192.168.1.1/24 up
  # ifconfig vnet1 192.168.1.2/24 up
  # ifconfig vnet0 group ipmp0
  # ifconfig vnet1 group ipmp0
  ```

- The following example configures two virtual network devices on a domain. Each domain is connected to a separate virtual switch device on the service domain to use link-based IPMP. The virtual network devices are also configured to obtain physical link state updates.

  ```
  # ldm add-vnet linkprop=phys-state vnet0 primary-vsw0 ldom1
  # ldm add-vnet linkprop=phys-state vnet1 primary-vsw1 ldom1
  ```

  Note – The virtual switch must have a physical network device assigned for the domain to successfully bind. If the domain is already bound and the virtual switch does not have a physical network device assigned, the ldm add-vnet commands will fail.

The following commands plumb the virtual network devices and assign them to an IPMP group:

```
# ifconfig vnet0 plumb
# ifconfig vnet1 plumb
# ifconfig vnet0 192.168.1.1/24 up
# ifconfig vnet1 192.168.1.2/24 up
# ifconfig vnet0 group ipmp0
# ifconfig vnet1 group ipmp0
```

Configuring and Using IPMP in Releases Prior to Logical Domains 1.3

In Logical Domains releases prior to 1.3, the virtual switch and the virtual network devices are not capable of performing link failure detection. In those releases, network failure detection and recovery can be set up by using probe-based IPMP.

Configuring IPMP in the Guest Domain

The virtual network devices in a guest domain can be configured into an IPMP group as shown in Figure 8–3 and Figure 8–4. The only difference is that probe-based failure detection is used by configuring test addresses on the virtual network devices. See *System Administration Guide: IP Services* for more information about configuring probe-based IPMP.

Configuring IPMP in the Service Domain

In Logical Domains releases prior to 1.3, the virtual switch device is not capable of physical link failure detection. In such cases, network failure detection and recovery can be set up by configuring the physical interfaces in the service domain into an IPMP group. To do this, configure the virtual switch in the service domain without assigning a physical network device to it. Namely, do not specify a value for the net-dev (net-dev=) property while you use the ldm add-vswitch command to create the virtual switch. Plumb the virtual switch interface in the service domain and configure the service domain itself to act as an IP router. Refer to the Oracle Solaris 10 *System Administration Guide: IP Services* for information about setting up IP routing.

Once configured, the virtual switch sends all packets originating from virtual networks (and destined for an external machine) to its IP layer, instead of sending the packets directly by means of the physical device. In the event of a physical interface failure, the IP layer detects failure and automatically re-routes packets through the secondary interface.

Since the physical interfaces are directly being configured into an IPMP group, the group can be set up for either link-based or probe-based detection. The following diagram shows two network interfaces (nxge0 and nxge1) configured as part of an IPMP group. The virtual switch instance (vsw0) has been plumbed as a network device to send packets to its IP layer.

FIGURE 8–6 Two Network Interfaces Configured as Part of IPMP Group
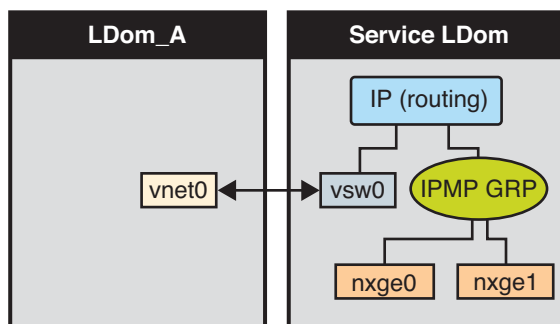
▼ Configure a Host Route for Probe-Based IPMP

Note – This procedure only applies to guest domains and to releases prior to 1.3, where only probe-based IPMP is supported.

If no explicit route is configured for a router in the network corresponding to the IPMP interfaces, then one or more explicit host routes to target systems need to be configured for the IPMP probe-based detection to work as expected. Otherwise, probe detection can fail to detect the network failures.

● **Configure a host route.**

   # **route add -host** *destination-IP gateway-IP* **-static**

   For example:

   # **route add -host 192.168.102.1 192.168.102.1 -static**

   Refer to "Configuring Target Systems" in *System Administration Guide: IP Services* for more information.

Using VLAN Tagging

As of the release of Oracle Solaris 10 10/08 OS and Logical Domains 1.1 software, 802.1Q VLAN-Tagging support is available in the Logical Domains network infrastructure.

Note – Tagged VLANs are not supported in any of the previous releases for Logical Domains networking components.

The virtual switch (vsw) and virtual network (vnet) devices support switching of Ethernet packets based on the virtual local area network (VLAN) identifier (ID) and handle the necessary tagging or untagging of Ethernet frames.

You can create multiple VLAN interfaces over a vnet device in a guest domain. You can use the Oracle Solaris OS ifconfig(1M) command to create a VLAN interface over a virtual network device, the same way it is used to configure a VLAN interface over any other physical network device. The additional requirement in the Logical Domains environment is that you must assign the vnet to the corresponding VLANs using the Logical Domains Manager CLI commands. Refer to the ldm(1M) for complete information about the Logical Domains Manager CLI commands.

Similarly, you can configure VLAN interfaces over a virtual switch device in the service domain. VLAN IDs 2 through 4094 are valid; VLAN ID 1 is reserved as the default-vlan-id.

When you create a vnet device on a guest domain, you must assign it to the required VLANs by specifying a port VLAN ID and zero or more VLAN IDs for this vnet, using the pvid= and vid= arguments to the ldm add-vnet command. This configures the virtual switch to support multiple VLANs in the Logical Domains network and switch packets using both MAC address and VLAN IDs in the network.

Similarly, any VLANs to which the vsw device itself should belong, when plumbed as a network interface, must be configured in the vsw device using the pvid= and vid= arguments to the ldm add-vsw command.

You can change the VLANs to which a device belongs using ldm set-vnet or ldm set-vsw command.

Port VLAN ID (PVID)

The PVID indicates a VLAN to which the virtual network device needs to be a member, in untagged mode. In this case, the vsw device provides the necessary tagging or untagging of frames for the vnet device over the VLAN specified by its PVID. Any outbound frames from the virtual network that are untagged are tagged with its PVID by the virtual switch. Inbound frames tagged with this PVID are untagged by the virtual switch, before sending it to the vnet device. Thus, assigning a PVID to a vnet implicitly means that the corresponding virtual network port on the virtual switch is marked untagged for the VLAN specified by the PVID. You can have only one PVID for a vnet device.

The corresponding virtual network interface, when configured using the ifconfig(1M) command 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 plumb vnet instance 0, using the following command, and if the pvid= argument for the vnet has been specified as 10, the vnet0 interface would be implicitly assigned to belong to the VLAN 10.

```
# ifconfig vnet0 plumb
```

VLAN ID (VID)

The VID indicates the VLAN to which a virtual network device or virtual switch needs to be a member, in tagged mode. The virtual network device sends and receives tagged frames over the VLANs specified by its VIDs. The virtual switch passes any frames that are tagged with the specified VID between the virtual network device and the external network.

▼ Assign VLANs to a Virtual Switch and Virtual Network Device

1 **Assign the virtual switch (`vsw`) to two VLANs.**

For example, configure VLAN 21 as untagged and VLAN 20 as tagged. Assign the virtual network (vnet) to three VLANs. Configure VLAN 20 as untagged and VLAN 21 and 22 as tagged.

```
# ldm add-vsw net-dev=nxge0 pvid=21 vid=20 primary-vsw0 primary
# ldm add-vnet pvid=20 vid=21,22 vnet01 primary-vsw0 ldom1
```

2 **Plumb the VLAN interfaces.**

This example assumes that the instance number of these devices is 0 in the domains and the VLANs are mapped to these subnets:

VLAN	Subnet
20	192.168.1.0 (netmask: 255.255.255.0)
21	192.168.2.0 (netmask: 255.255.255.0)
22	192.168.3.0 (netmask: 255.255.255.0)

a. **Plumb the VLAN interface in the service (`primary`) domain.**

```
primary# ifconfig vsw0 plumb
primary# ifconfig vsw0 192.168.2.100 netmask 0xffffff00 broadcast + up
primary# ifconfig vsw20000 plumb
primary# ifconfig vsw20000 192.168.1.100 netmask 0xffffff00 broadcast + up
```

b. **Plumb the VLAN interface in the guest (`ldom1`) domain.**

```
ldom1# ifconfig vnet0 plumb
ldom1# ifconfig vnet0 192.168.1.101 netmask 0xffffff00 broadcast + up
ldom1# ifconfig vnet21000 plumb
ldom1# ifconfig vnet21000 192.168.2.101 netmask 0xffffff00 broadcast + up
ldom1# ifconfig vnet22000 plumb
ldom1# ifconfig vnet22000 192.168.3.101 netmask 0xffffff00 broadcast + up
```

For more information about how to configure VLAN interfaces in the Oracle Solaris OS, refer to "Administering Virtual Local Area Networks" in *System Administration Guide: IP Services*.

▼ Install a Guest Domain When the Install Server Is in a VLAN

Be careful when installing a guest domain over the network (JumpStart) and the installation server is in a VLAN. Specify the VLAN ID that is associated with the installation server as the PVID of the virtual network device, and do not configure any tagged VLANs (vid) for that virtual network device. You must do this because OBP is not aware of VLANs and cannot handle VLAN-tagged network packets. The virtual switch handles the untagging and tagging of packets to and from the guest domain during network installation. After the network installation completes and the Oracle Solaris OS boots, you can configure the virtual network device to be tagged in that VLAN. You can then add the virtual network device to additional VLANs in tagged mode.

For information about using JumpStart to install a guest domain, see "Perform a JumpStart Operation on a Guest Domain" on page 59.

1   Initially configure the network device in untagged mode.

For example, if the install server is in VLAN 21, configure the virtual network initially as follows:

```
primary# ldm add-vnet pvid=21 vnet01 primary-vsw0 ldom1
```

2   After the installation is complete and the Oracle Solaris OS boots, configure the virtual network in tagged mode.

```
primary# ldm set-vnet pvid= vid=21, 22, 23 vnet01 primary-vsw0 ldom1
```

Using NIU Hybrid I/O

The virtual I/O framework implements a *hybrid* I/O model for improved functionality and performance. The hybrid I/O model combines direct and virtualized I/O to allow flexible deployment of I/O resources to virtual machines. It is particularly useful when direct I/O does not provide full capability for the virtual machine, or direct I/O is not persistently or consistently available to the virtual machine. This could be because of resource availability or virtual machine migration. The hybrid I/O architecture is well-suited for the Network Interface Unit (NIU) on Sun UltraSPARC T2 and SPARC T3 platforms. An NIU is a network I/O interface that is integrated on chip. This architecture enables the dynamic assignment of Direct Memory Access (DMA) resources to virtual networking devices and, thereby, provides consistent performance to applications in the domain.

NIU hybrid I/O is available for Sun UltraSPARC T2 and SPARC T3 platforms. This feature is enabled by an optional hybrid mode that provides for a virtual network (vnet) device where the DMA hardware resources are loaned to a vnet device in a guest domain for improved performance. In the hybrid mode, a vnet device in a guest domain can send and receive unicast traffic from an external network directly into the guest domain using the DMA hardware resources. The broadcast or multicast traffic and unicast traffic to the other guest domains in the same system continue to be sent using the virtual I/O communication mechanism.

Note – NIU hybrid I/O is not available on UltraSPARC T2 Plus platforms.

FIGURE 8–7   Hybrid Virtual Networking
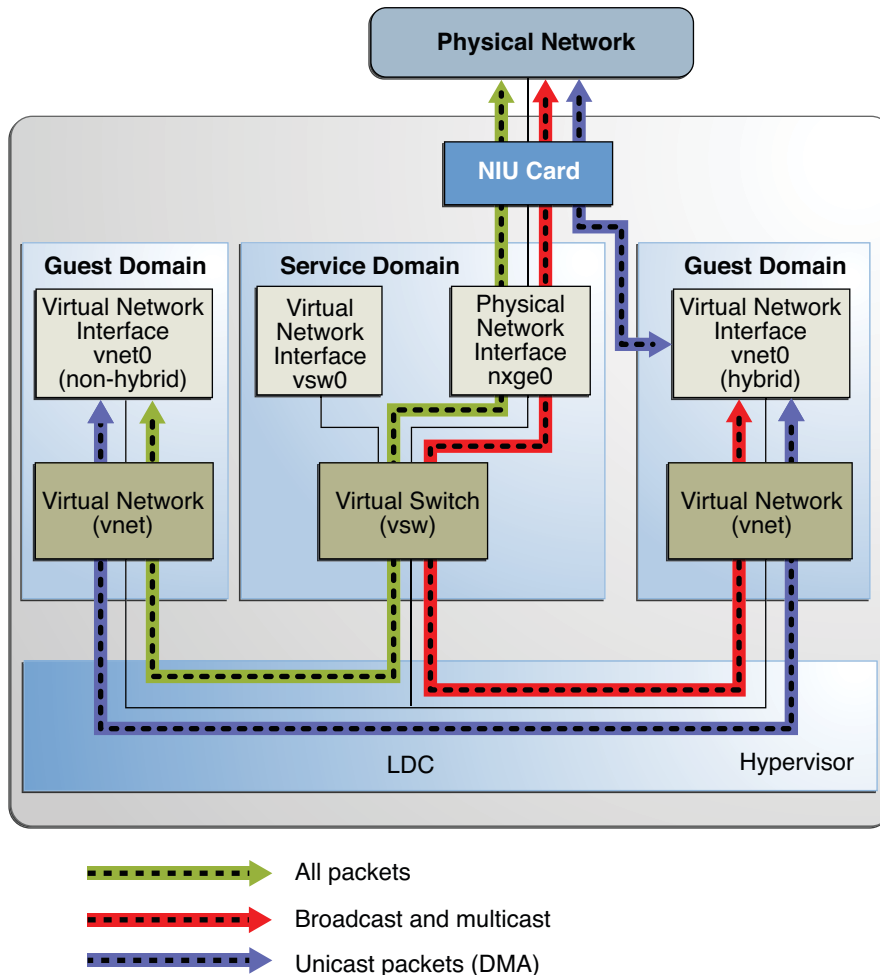
The hybrid mode applies only for the vnet devices that are associated with a virtual switch (vsw) configured to use an NIU network device. As the shareable DMA hardware resources are limited, up to only three vnet devices per vsw can have DMA hardware resources assigned at a given time. If more than three vnet devices have the hybrid mode enabled, the assignment is done on a first-come, first-served basis. As 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.

Following are points you need to be aware of when using this feature:

- Hybrid mode option for a vnet device is treated as a suggestion only. That means the DMA resources are assigned only when they are available and the device is capable of using them.
- Logical Domains Manager CLI commands do not validate the hybrid mode option; that is, it is possible to set the hybrid mode on any vnet or any number of vnet devices.
- Guest domains and the service domain need to run Oracle Solaris 10 10/08 OS at a minimum.
- Up to a maximum of only three vnet devices per vsw can have DMA hardware resources loaned at a given time. As there are two NIU network devices, there can be a total of six vnet devices with DMA hardware resources loaned.

Note – Set the hybrid mode only for three vnet devices per vsw so that they are guaranteed to have DMA hardware resources assigned.

- Hybrid mode is disabled by default for a vnet device. It needs to be explicitly enabled with Logical Domains Manager CLI commands. See "Enable Hybrid Mode" on page 132.

  (Refer to the ldm(1M) man page for more details.)
- The hybrid mode option cannot be changed dynamically while the guest domain is active.
- The DMA hardware resources are assigned only when a vnet device is active that is plumbed in the guest domain.
- The NIU 10-gigabit Ethernet driver (nxge) is used for the NIU card. The same driver is also used for other 10-gigabit network cards. However, the NIU hybrid I/O feature is available for NIU network devices only.

▼ Configure a Virtual Switch With an NIU Network Device

1 Determine an NIU network device.

The following example shows the output on an UltraSPARC T2 server:

```
# grep nxge /etc/path_to_inst
"/niu@80/network@0" 0 "nxge"
"/niu@80/network@1" 1 "nxge"
```

The following example shows the output on a SPARC T3-1 server:

```
# grep nxge /etc/path_to_inst
"/niu@480/network@0" 0 "nxge"
"/niu@480/network@1" 1 "nxge"
```

2 Configure a virtual switch.

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

▼ Enable Hybrid Mode

● **For example, enable a hybrid mode for a `vnet` device while it is being created.**

```
# ldm add-vnet mode=hybrid vnet01 primary-vsw0 ldom01
```

▼ Disable Hybrid Mode

● **For example, disable hybrid mode for a `vnet` device.**

```
# ldm set-vnet mode= vnet01 ldom01
```

Using Link Aggregation With a Virtual Switch

As of the release of the Oracle Solaris 10 10/08 OS and the Logical Domains 1.1 software, the virtual switch can be configured to use a link aggregation. A link aggregation is used as the virtual switch's network device to connect to the physical network. This configuration enables the virtual switch to leverage the features provided by the IEEE 802.3ad Link Aggregation Standard. Such features include increased bandwidth, load balancing, and failover. For information about how to configure link aggregation, see the *System Administration Guide: IP Services*.

After you create a link aggregation, you can assign it to the virtual switch. Making this assignment is similar to assigning a physical network device to a virtual switch. Use the ldm add-vswitch or ldm set-vswitch command to set the net-dev property.

When the link aggregation is assigned to the virtual switch, traffic to and from the physical network flows through the aggregation. Any necessary load balancing or failover is handled transparently by the underlying aggregation framework. Link aggregation is completely transparent to the virtual network (`vnet`) devices that are on the guest domains and that are bound to a virtual switch that uses an aggregation.

Note – You cannot group the virtual network devices (`vnet` and `vsw`) into a link aggregation.

You can plumb and use the virtual switch that is configured to use a link aggregation in the service domain. See "Configure the Virtual Switch as the Primary Interface" on page 51.

The following figure illustrates a virtual switch configured to use an aggregation, aggr1, over physical interfaces nxge0 and nxge1.

FIGURE 8–8   Configuring a Virtual Switch to Use a Link Aggregation
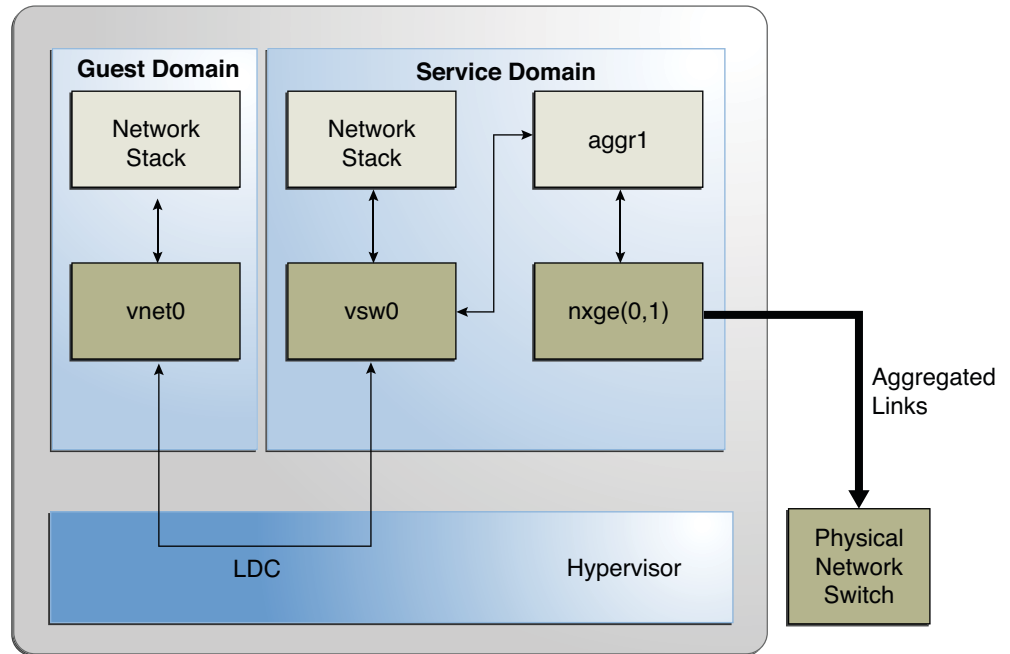
Configuring Jumbo Frames

The Logical Domains virtual switch (vsw) and virtual network (vnet) devices can now support Ethernet frames with payload sizes larger than 1500 bytes. This change results in these drivers being able to increase network throughput.

▼ Configure Virtual Network and Virtual Switch Devices to Use Jumbo Frames

You enable jumbo frames by specifying the maximum transmission unit (MTU) for the virtual switch device. In such cases, the virtual switch device and all virtual network devices that are bound to the virtual switch device use the specified MTU value.

In certain circumstances, you can specify an MTU value directly on a virtual network device. You might do this if the required MTU value for the virtual network device should be less than that supported by the virtual switch.

> **Note –** On the Oracle Solaris 10 5/09 OS, the MTU of a physical device must be configured to match the MTU of the virtual switch. For more information about configuring particular drivers, see the man page that corresponds to that driver in Section 7D of the Oracle Solaris reference manual. For example, to obtain information about the nxge driver, see the nxge(7D) man page.

1 Log in to the control domain.

2 Become superuser or assume an equivalent role.

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

3 Determine the value of MTU that you want to use for the virtual network.

You can specify an MTU value from 1500 to 16000 bytes. The specified MTU must match the MTU of the physical network device that is assigned to the virtual switch.

4 Specify the MTU value of a virtual switch device or virtual network device.

Do one of the following:

- Enable jumbo frames on a new virtual switch device in the service domain by specifying its MTU as a value of the mtu property.

  # **ldm add-vsw mtu=**_value vswitch-name ldom_

  In addition to configuring the virtual switch, this command updates the MTU value of each virtual network device that will be bound to this virtual switch.

- Enable jumbo frames on an existing virtual switch device in the service domain by specifying its MTU as a value of the mtu property.

  # **ldm set-vsw mtu=**_value vswitch-name_

  In addition to configuring the virtual switch, this command updates the MTU value of each virtual network device that will be bound to this virtual switch.

In rare circumstances, you might need to use the ldm add-vnet or ldm set-vnet command to specify an MTU value for a virtual network device that differs from the MTU value of the virtual switch. For example, you might change the virtual network device's MTU value if you configure VLANs over a virtual network device and the largest VLAN MTU is less than the MTU value on the virtual switch. A vnet driver that supports jumbo frames might not be required for domains where only the default MTU value is used. However, if the domains have virtual network devices bound to a virtual switch that uses jumbo frames, ensure that the vnet driver supports jumbo frames.

If you use the ldm set-vnet command to specify an mtu value on a virtual network device, future updates to the MTU value of the virtual switch device are not propagated to that virtual

network device. To reenable the virtual network device to obtain the MTU value from the virtual switch device, run the following command:

```
# ldm set-vnet mtu= vnet-name ldom
```

Note that enabling jumbo frames for a virtual network device automatically enables jumbo frames for any HybridIO resource that is assigned to that virtual network device.

On the control domain, the Logical Domains Manager updates the MTU values that are initiated by the ldm set-vsw and ldm set-vnet commands as delayed reconfiguration operations. To make MTU updates to domains other than the control domain, you must stop a domain prior to running the ldm set-vsw or ldm set-vnet command to modify the MTU value.

Example 8–2   Configuring Jumbo Frames on Virtual Switch and Virtual Network Devices

- The following example shows how to add a new virtual switch device that uses an MTU value of 9000. This MTU value is propagated from the virtual switch device to all of the client virtual network devices.

   First, the ldm add-vsw command creates the virtual switch device, primary-vsw0, with an MTU value of 9000. Note that instance 0 of the network device nxge0 is specified as a value of the net-dev property.

   ```
   # ldm add-vsw net-dev=nxge0 mtu=9000 primary-vsw0 primary
   ```

   Next, the ldm add-vnet command adds a client virtual network device to this virtual switch, primary-vsw0. Note that the MTU of the virtual network device is implicitly assigned from the virtual switch to which it is bound. As a result, the ldm add-vnet command does not require that you specify a value for the mtu property.

   ```
   # ldm add-vnet vnet01 primary-vsw0 ldom1
   ```

   The ifconfig command plumbs the virtual switch interface in the service domain, primary. The ifconfig vsw0 command output shows that the value of the mtu property is 9000.

   ```
   # ifconfig vsw0 plumb
   # ifconfig vsw0 192.168.1.100/24 up
   # ifconfig vsw0
   vsw0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 9000 index 5
           inet 192.168.1.100 netmask ffffff00 broadcast 192.168.1.255
           ether 0:14:4f:fa:0:99
   ```

   The ifconfig command plumbs the virtual network interface in the guest domain, ldom1. The ifconfig vnet0 command output shows that the value of the mtu property is 9000.

   ```
   # ifconfig vnet0 plumb
   # ifconfig vnet0 192.168.1.101/24 up
   # ifconfig vnet0
   vnet0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 9000 index 4
           inet 192.168.1.101 netmask ffffff00 broadcast 192.168.1.255
           ether 0:14:4f:f9:c4:13
   ```

■ The following example shows how to use the `ifconfig` command to change the MTU of the interface to 4000.

Note that the MTU of an interface can only be changed to a value that is less than the MTU of the device that is assigned by the Logical Domains Manager. This method is useful when VLANs are configured and each VLAN interface needs a different MTU.

```
# ifconfig vnet0 mtu 4000
# ifconfig vnet0
vnet0: flags=1201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS,FIXEDMTU>
mtu 4000 index 4
        inet 192.168.1.101 netmask ffffff00 broadcast 192.168.1.255
        ether 0:14:4f:f9:c4:13
```

Compatibility With Older (Jumbo-Unaware) Versions of the `vnet` and `vsw` Drivers

Drivers that support jumbo frames can interoperate with drivers that do not support jumbo frames on the same system. This interoperability is possible as long as jumbo frame support is not enabled when you create the virtual switch.

Note – Do not set the `mtu` property if any guest or service domains that are associated with the virtual switch do not use Logical Domains drivers that support jumbo frames.

Jumbo frames can be enabled by changing the `mtu` property of a virtual switch from the default value of 1500. In this instance, older driver versions ignore the `mtu` setting and continue to use the default value. Note that the `ldm list` output will show the MTU value you specified and not the default value. Any frames larger than the default MTU are not sent to those devices and are dropped by the new drivers. This situation might result in inconsistent network behavior with those guests that still use the older drivers. This applies to both client guest domains and the service domain.

So, while jumbo frames are enabled, ensure that all virtual devices in the Logical Domains network are upgraded to use the new drivers that support jumbo frames. Also ensure that you upgrade to at least Logical Domains 1.2 so that you can configure jumbo frames.

9

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

Domain migration provides the ability to migrate a logical domain from one host machine to another. The host where the migration is initiated is referred to as the source machine, and the host where the domain is migrated to is referred to as the target machine. Similarly, once a migration is started, the domain to be migrated is referred to as the source domain and the shell of a domain created on the target machine is referred to as the target domain while the migration is in progress.

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 running on the target machine. Once this connection has been established, the migration occurs. The migration itself can be broken down into different phases.

Phase 1: After connecting with the Logical Domains Manager running in the target host, information about the source machine and domain are transferred to the target host. This information is used to perform a series of checks to determine whether a migration is possible. The checks differ depending on the state of the source domain. For example, if the source domain is active, a different set of checks are performed than if the domain is bound or inactive.

Phase 2: When all checks in Phase 1 have passed, the source and target machines prepare for the migration, and the source domain is suspended. On the target machine, a domain is created to receive the source domain.

Phase 3: For an active domain, the next phase is to transfer all the runtime state information for the domain to the target. This information is retrieved from the hypervisor. On the target, the state information is installed in the hypervisor.

Phase 4: Handoff. After all state information is transferred, the handoff occurs when the target domain resumes execution (if the source was active) and the source domain is destroyed. From this point on, the target 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 hypervisor on both the source and target machines must have a compatible version of firmware installed. See "Required Software to Enable Oracle VM Server for SPARC 2.0 Features" in *Oracle VM Server for SPARC 2.0 Release Notes*.

- A compatible version of the Logical Domains Manager must be running on both machines.

Note – The migration feature was first released with the Logical Domains 1.1 software and corresponding firmware. For information about the latest firmware for your platform, see the *Oracle VM Server for SPARC 2.0 Release Notes*.

Authentication for Migration Operations

Because the migration operation executes on two machines, a user must be authenticated on both the source and target host. In particular, a user other than superuser must have the `solaris.ldoms.read` and `solaris.ldoms.write` authorizations.

The `ldm` command-line interface for migration allows the user to specify an optional alternate user name for authentication on the target host. If this is not specified, the user name of the user executing the migration command is used. In both cases, the user is prompted for a password for the target system, unless the `-p` option is used to initiate a non-interactive migration.

Migrating a Domain

You can use the `ldm migrate-domain` command to initiate a migration of a domain from one system to another system.

For information about the migration options and operands, see the `ldm(1M)` man page.

Performing a Dry Run

When you provide the `-n` option to the `migrate-domain` subcommand, migration checks are performed, but the source domain is not migrated. Any requirement that is not satisfied is reported as an error. This allows you to correct any configuration errors before attempting a real migration.

Note – Because of the dynamic nature of logical domains, it is possible for a dry run to succeed and a migration to fail and vice-versa.

Performing Non-Interactive Migrations

Until the release of the Logical Domains 1.3 software, migrations were interactive operations. When you initiated the migration, you were prompted for the password to use for the target machine. Starting with Logical Domains 1.3, you can use the `ldm migrate-domain -p` *filename* command to initiate a non-interactive migration operation.

The file name you specify as an argument to the `-p` option must have the following properties:

- 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 on 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

For the migration of an active domain to occur with Oracle VM Server for SPARC 2.0 software, there is a certain set of requirements and restrictions imposed on the source logical domain, the source machine, and the target machine. The sections following describe these requirements and restrictions for each of the resource types.

Note – The migration operation speeds up when the `primary` domain on the source and target systems have cryptographic units assigned. Starting with Logical Domains 1.3, you can speed up migration by adding more virtual CPUs to the `primary` domains of both the source and target systems.

Migrating CPUs in an Active Domain

Following are the requirements and restrictions on CPUs when you are performing a migration:

- The source and target machines must have the same processor type running at the same frequency.
- The target machine must have sufficient free strands to accommodate the number of strands in use by the domain.

Additional requirements and restrictions apply under any of the following conditions:

- The target system is not running at least version 2.0 of the Logical Domains Manager. In this case, you might see the following message during the migration:

  ```
  The target machine is running an older version of the domain
  manager that does not support the latest migration functionality.
  ```

- The source system is not running at least version 2.0 of the Logical Domains Manager. Because the legacy Logical Domains Manager in the source domain is unable to detect the software mismatch, the migration proceeds without issuing a message.

- The source domain is running an Oracle Solaris OS version older than the Oracle Solaris 10 9/10 OS. In this case, you might see the following message during the migration:

  ```
  Domain ldom is not running an operating system that is
  compatible with the latest migration functionality.
  ```

If any of these conditions apply, the following requirements and restrictions on CPUs apply:

- Full cores must be allocated for the migrated domain. If the number of strands in the source domain is less than a full core, the extra strands are unavailable to any domain until after the migrated domain is rebooted.

- After a migration, CPU dynamic reconfiguration (DR) is disabled for the target domain until it has been rebooted. After a reboot has occurred, CPU DR becomes available for that domain.

- The target system must have enough full cores that are entirely free to provide the number of strands required for the migrated domain. After the migration, if a full core is only partially used by the migrated domain, any extra strands are unavailable to any domain until after the migrated domain is rebooted.

Migrating Memory in an Active Domain

There must be sufficient free memory on the target machine to accommodate the migration of the source domain. In addition, following are a few properties that must be maintained across the migration:

- It must be possible to create the same number of identically-sized memory blocks.

- The physical addresses of the memory blocks do not need to match, but the same real addresses must be maintained across the migration.

The target machine must have sufficient free memory to accommodate the migration of the source domain. In addition, the layout of the available memory on the target machine must be compatible with the memory layout of the source domain or the migration will fail.

In particular, if the memory on the target machine is fragmented into multiple small address ranges, but the source domain requires a single large address range, the migration will fail. The following example illustrates this scenario. The target domain has two Gbytes of free memory in two memory blocks:

```
# ldm list-devices memory
MEMORY
    PA                SIZE
    0x108000000       1G
    0x188000000       1G
```

The source domain, ldg-src, also has two Gbytes of free memory, but it is laid out in a single memory block:

```
# ldm list -o memory ldg-src
NAME
ldg-src

MEMORY
    RA            PA            SIZE
    0x8000000     0x208000000   2G
```

Given this memory layout situation, the migration fails:

```
# ldm migrate-domain ldg-src dt212-239
Target Password:
Unable to bind 2G memory region at real address 0x8000000
Domain Migration of LDom ldg-src failed
```

Note – After a migration, memory dynamic reconfiguration (DR) is disabled for the target domain until it has been rebooted. After the reboot completes, memory DR is re-enabled for the domain.

Migrating Physical I/O Devices in an Active Domain

Virtual devices that are associated with physical devices can be migrated. However, domains that have direct access to physical devices cannot be migrated. For instance, you cannot migrate I/O domains.

Migrating Virtual I/O Devices in an Active Domain

All virtual I/O (VIO) services used by the source domain must be available on the target machine. In other words, the following conditions must exist:

■ Each logical volume used in the source logical domain must also be available on the target host and must refer to the same storage.

Caution – If the logical volume used by the source as a boot device exists on the target but does not refer to the same storage, the migration appears to succeed, but the machine is not usable as it is unable to access its boot device. The domain has to be stopped, the configuration issue corrected, and then the domain restarted. Otherwise, the domain could be left in an inconsistent state.

■ For each virtual network device in the source domain, a virtual network switch must exist on the target host, with the same name as the virtual network switch the device is attached to on the source host.

For example, if vnet0 in the source domain is attached to a virtual switch service name switch-y, then there must be a logical domain on the target host providing a virtual switch service named switch-y.

Note – The switches do not have to be connected to the same network for the migration to occur, though the migrated domain can experience networking problems if the switches are not connected to the same network.

MAC addresses used by the source domain that are in the automatically allocated range must be available for use on the target host.

- A virtual console concentrator (`vcc`) service must exist on the target host and have at least one free port. Explicit console constraints are ignored during the migration. The console for the target domain is created using the target domain name as the console group and using any available port on the first `vcc` device in the control domain. If there is a conflict with the default group name, the migration fails.

Migrating NIU Hybrid Input/Output in an Active Domain

A domain using NIU Hybrid I/O resources can be migrated. A constraint specifying NIU Hybrid I/O resources is not a hard requirement of a logical domain. If such a domain is migrated to a machine that does not have available NIU resources, the constraint is preserved, but not fulfilled.

Migrating Cryptographic Units in an Active Domain

Starting with Logical Domains 1.3, you can migrate a guest domain that has bound cryptographic units if it runs an operating system that supports cryptographic unit dynamic reconfiguration (DR).

The following Oracle Solaris OS versions support cryptographic unit DR:

- At least the Solaris 10 10/09 OS
- At least the Oracle Solaris 10 5/08 OS plus patch ID 142245-01

At the start of the migration, the Logical Domains Manager determines whether the source domain 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, a migration operation might still complete successfully. In such a case, the domain might end up with 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 hosts prevent a migration from starting. Delayed reconfiguration operations are blocked while a migration is in progress.

Migrating While an Active Domain Is in Elastic Mode

Domain migrations are not supported for a source or target machine in elastic mode. If a migration is underway while the domain is in performance mode and the power management (PM) policy is set to elastic mode, the policy switch is deferred until the migration completes. The migration command returns an error if either the source or target machine is in elastic mode and a domain migration is attempted.

Operations on Other Domains

While a migration is in progress on a machine, any operation that could 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 Bound or Inactive Domains

Because a bound or inactive domain is not executing at the time of the migration, there are fewer restrictions than when you migrate an active domain.

The migration of a bound domain requires that the target is able to satisfy the CPU, memory, and I/O constraints of the source domain. Otherwise, the migration will fail. The migration of an inactive domain does not have such requirements. However, the target must satisfy the domain's constraints when a bind is later attempted. Otherwise, the domain binding will fail.

Migrating CPUs in a Bound or Inactive Domain

You can migrate a bound or inactive domain between machines running different processor types and machines that are running at different frequencies.

The Oracle Solaris OS image in the guest must support the processor type on the target machine.

Migrating Virtual Input/Output in a Bound or Inactive Domain

For an inactive domain, there are no checks performed against the virtual input/output (VIO) constraints. So, the VIO servers do not need to exist for the migration to succeed. As with any inactive domain, the VIO servers need to exist and be available at the time the domain is bound.

Migrating PCIe Endpoint Devices in a Bound or Inactive Domain

You cannot perform a domain migration on an I/O domain that is configured with PCIe endpoint devices.

For information about the direct I/O (DIO) feature, see "Assigning PCIe Endpoint Devices" on page 66.

Monitoring a Migration in Progress

When a migration is in progress, the source and target domains are shown differently in the status output. The output of the ldm list command indicates the state of the migrating domain.

The sixth column in the FLAGS field shows one of the following values:

- The source domain shows an s to indicate that it is the source of the migration.
- The target domain shows a t to indicate that it is the target of a migration.
- If an error occurs that requires user intervention, an e is shown.

The following shows that ldg-src is the source domain 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 shows that ldg-tgt is the target domain of the migration:

```
# ldm list ldg-tgt
NAME       STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg-tgt    bound      -----t  5000    1     1G
```

In the long form of the status output, additional information is shown about the migration. On the source, the percentage of the operation complete is displayed along with the target host and domain name. Similarly, on the target, the percentage of the operation complete is displayed along with the source host and domain name.

EXAMPLE 9–1   Monitoring a Migration in Progress

```
# ldm list -o status ldg-src
NAME
ldg-src

STATUS
    OPERATION    PROGRESS    TARGET
    migration    17%         t5440-sys-2
```

Canceling a Migration in Progress

Once a migration starts, if the ldm command is interrupted with a KILL signal, the migration is terminated. The target domain is destroyed, and the source domain 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 terminates the migration in progress, and the source domain resumes as the active domain. The ldm cancel-operation command should be initiated from the source system. On a given system, any migration-related command impacts the migration operation that was started from that system. A system cannot control a migration operation when it is the target system.

Note – Once a migration has been initiated, suspending the ldm(1M) process does not pause the operation, because it is the Logical Domains Manager daemon (ldmd) on the source and target machines that are effecting the migration. The ldm process waits for a signal from the ldmd that the migration has been completed before returning.

Recovering From a Failed Migration

If the network connection is lost after the source has completed sending all the runtime state information to the target, but before the target can acknowledge that the domain has been resumed, the migration operation terminates, and the source is placed in an error state. This indicates that user interaction is required to determine whether or not the migration was completed successfully. In such a situation, take the following steps.

- Determine whether the target domain has resumed successfully. The target domain will be in one of two states:
  - If the migration completed successfully, the target domain is in the normal state.
  - If the migration failed, the target cleans up and destroys the target domain.
- If the target is resumed, it is safe to destroy the source domain in the error state. If the target is not present, the source domain is still the master version of the domain, and it must be recovered. To do this, execute the cancel command on the source machine. This clears the error state and restores the source domain back to its original condition.

Migration Examples

Example 9–2 shows how a domain, called ldg1, can be migrated to a machine called t5440-sys-2.

EXAMPLE 9–2   Migrating a Guest Domain

```
# ldm migrate-domain ldg1 t5440-sys-2
Target Password:
```

To perform this migration without being prompted for the target password, use the following command:

```
# ldm migrate-domain -p pfile ldg1 t5440-sys-2
```

The -p option takes a file name as an argument. The specified file contains the superuser password for the target. In this example, pfile contains the password for the target, t5440-sys-2.

Example 9–3 shows that a domain can be renamed as part of the migration. In this example, ldg-src is the source domain, and it is renamed to ldg-tgt on the target machine (t5440-sys-2) as part of the migration. In addition, the user name (root) on the target machine is explicitly specified.

EXAMPLE 9–3   Migrating and Renaming a Guest Domain

```
# ldm migrate ldg-src root@t5440-sys-2:ldg-tgt
Target Password:
```

Example 9–4 shows a sample failure message if the target domain does not have migration support, that is, if you are running an Logical Domains version prior to Version 1.1.

EXAMPLE 9–4 Migration Failure Message

```
# ldm migrate ldg1 t5440-sys-2
Target Password:
Failed to establish connection with ldmd(1m) on target: t5440-sys-2
Check that the 'ldmd' service is enabled on the target machine and
that the version supports Domain Migration. Check that the 'xmpp_enabled'
and 'incoming_migration_enabled' properties of the 'ldmd' service on
the target machine are set to 'true' using svccfg(1M).
```

Example 9–5 shows how to obtain status on a target domain while the migration is in progress. In this example, the source machine is t5440-sys-1.

EXAMPLE 9–5 Obtaining Target Domain Status

```
# ldm list -o status ldg-tgt
NAME
ldg-tgt

STATUS
    OPERATION    PROGRESS    SOURCE
    migration    55%         t5440-sys-1
```

Example 9–6 shows how to obtain parseable status on the source domain while the migration is in progress. In this example, the target machine is t5440-sys-2.

EXAMPLE 9–6 Obtaining Source Domain Parseable Status

```
# ldm list -o status -p ldg-src
VERSION 1.3
DOMAIN|name=ldg-src|
STATUS
|op=migration|progress=42|error=no|target=t5440-sys-2
```

10

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

A system that runs the Oracle VM Server for SPARC software is able to configure resources, such as virtual CPUs, virtual I/O devices, cryptographic units, and memory. Some resources can be configured dynamically on a running domain, while others must be configured on a stopped domain. If a resource cannot be dynamically configured on the control domain, you must first initiate a delayed reconfiguration. The delayed reconfiguration postpones the configuration activities until after the control domain has been rebooted.

Dynamic Reconfiguration

Dynamic reconfiguration (DR) enables resources to be added or removed while the operating system (OS) is running. The capability to perform DR of a particular resource type is dependent on having support in the OS running in the logical domain.

Dynamic reconfiguration is supported for the following resources:

- **Virtual CPUs** – Supported in all versions of the Oracle Solaris 10 OS
- **Virtual I/O devices** – Supported in at least the Solaris 10 10/08 OS
- **Cryptographic units** – Supported in at least the Oracle Solaris 10 9/10 OS
- **Memory** – Supported starting with the Oracle VM Server for SPARC 2.0 release (see "Using Memory Dynamic Reconfiguration" on page 155)
- **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

Starting with the Logical Domains 1.2 software, 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.

Starting with the Oracle VM Server for SPARC 2.0 software, you must first initiate a delayed reconfiguration on the control domain prior to performing resource configuration operations. You can initiate the delayed reconfiguration by using the ldm start-reconf primary command.

When a delayed reconfiguration is in progress on the control domain, other reconfiguration requests for the control domain are deferred until it is rebooted, or stopped and started. Also, when a delayed reconfiguration is pending for the control domain, reconfiguration requests for other logical domains are severely restricted and will fail with an appropriate error message.

The ldm cancel-operation reconf command cancels delayed reconfiguration operations on the control domain. You can list delayed reconfiguration operations by using the ldm list-domain command. For more information about how to use the delayed reconfiguration feature, see the ldm(1M) man page.

Note – You cannot use the ldm cancel-operation reconf command if any other ldm remove-* commands have already performed a delayed reconfiguration operation on virtual I/O devices. The ldm cancel-operation reconf command fails in these circumstances.

Resource Allocation

Starting with the Oracle VM Server for SPARC 2.0 release, the resource allocation mechanism uses resource allocation constraints and hints to assign resources to a domain at bind time.

A *resource allocation constraint* is a hard requirement that the system *must* meet when you assign a resource to a domain. If the constraint cannot be met, both the resource allocation and the binding of the domain fail.

A *resource allocation hint* is a soft requirement that the system *attempts* to meet when you assign a resource to a domain. The allocation of a resource can still succeed, and the domain can be bound even if the hint cannot be fully met. This situation can happen if the system can assign the resource in a way that does not necessarily meet the requirement.

CPU Allocation

The CPU allocation mechanism uses the following constraints and hint for CPU resources:

- **Whole-core constraint.** This constraint specifies that virtual CPUs are allocated to a domain based on a specified number of CPU cores. The system must be able to allocate the specified number of cores and must also assign all the virtual CPUs of those allocated cores to the domain. If the system cannot allocate the specified number of cores, the domain fails to bind.

- **Maximum number of cores constraint.** This constraint specifies the maximum number of cores that can be assigned to a bound or active domain. This constraint is automatically enabled when the whole-core constraint is set on a domain. In that case, the maximum number of cores is automatically set to the number of cores configured when the domain is inactive. Currently, this constraint cannot be enabled independently of the whole-core constraint, and the maximum number of cores cannot be set manually.

- **Core affinity hint.** This hint requests that virtual CPUs allocated to a domain come from the same CPU cores or from the fewest number of CPU cores. The system makes its best effort to honor this request. The domain fails to bind *only* if insufficient free virtual CPUs are available on the system.

  The core affinity hint is enabled by default and cannot be disabled.

Note – The whole-core constraint and the core affinity hint only address the location of a virtual CPU on cores. They do not address the location of a core on chips or of a chip on sockets.

Enabling the Whole-Core Constraint

The whole-core constraint is automatically enabled when you specify the number of cores to assign to a domain. By default, you specify the virtual CPUs to assign to a domain. You can only

enable the whole-core constraint on an inactive domain, not on a domain that is bound or active. Before you enable the whole-core constraint on the control domain, you must first initiate a delayed reconfiguration.

Use the `ldm add-vcpu -c` *number*, `ldm set-vcpu -c` *number*, or `ldm remove-vcpu -c` *number* command to assign or remove CPU cores to and from a domain. *number* specifies the number of CPU cores and enables the whole-core constraint. For more information, see the `ldm(1M)` man page.

You can use the `ldm add-vcpu -c` *number* or `ldm remove-vcpu -c` *number* command on a domain that was previously configured with virtual CPUs. In that case, the existing number of virtual CPUs is automatically converted to the corresponding number of cores. This conversion is possible only if the existing number of virtual CPUs is a multiple of the number of virtual CPUs per core. If not, the conversion cannot be performed, and the command fails.

Note – If you use these commands to enable the whole-core constraint on an inactive domain or on the control domain in delayed reconfiguration mode, the maximum number of cores is also set. The maximum number of cores is not affected when you use these commands on a bound or active domain.

For example, a core is comprised of eight virtual CPUs. If a domain has seven virtual CPUs assigned, an `ldm add-vcpu -c` or `ldm remove-vcpu -c` command could not meet the whole-core constraint. Instead, you could use the `set-vcpu -c` command to specify the number of cores and to enable the whole-core constraint.

The following example enables the whole-core constraint on the inactive `ldg1` domain. The `ldm list` command verifies that the whole-core constraint is enabled.

```
primary# ldm add-vcpu -c 1 ldg1
primary# ldm list -o resmgmt ldg1
NAME
ldg1
CONSTRAINT
   whole-core
   max-cores=1
```

Note – When the whole-core constraint is enabled on a domain, the cryptographic units that are associated with those cores are unaffected by core additions. So, the system does not automatically add or remove the associated cryptographic units to or from the domain. Also, you cannot remove cores if the corresponding cryptographic units are assigned to the domain.

Disabling the Whole-Core Constraint

When a domain is assigned virtual CPUs instead of cores, the whole-core constraint is disabled. You can only disable the whole-core constraint on an inactive domain, not on a domain that is bound or active. Before you disable the whole-core constraint on the control domain, you must first initiate a delayed reconfiguration.

Use the ldm add-vcpu *number*, ldm set-vcpu *number*, or ldm remove-vcpu *number* command to assign or remove virtual CPUs to and from a domain. *number* specifies the number of virtual CPUs and disables the whole-core constraint. For more information, see the ldm(1M) man page.

You can use the ldm add-vcpu *number* or ldm rm-vcpu *number* command on a domain that was previously configured with CPU cores. In that case, the existing number of CPU cores is automatically converted to the corresponding number of virtual CPUs.

Note – When you disable the whole-core constraint, the maximum core constraint is also automatically disabled.

The following example disables the whole-core constraint on the inactive ldg1 domain:

```
primary# ldm set-vcpu 1 ldg1
```

Allocating CPUs to the Control Domain

To enable the whole-core constraint on the control domain, the control domain must be in delayed reconfiguration mode. Enabling the whole-core constraint on the control domain only succeeds if sufficient CPU cores are available to meet the requested constraint. That is, unused cores, cores that are already used by the control domain, or cores that are partially used by the control domain must be available. Otherwise, the CPU allocation on the control domain remains unchanged.

Note – When the control domain is in delayed reconfiguration mode, the whole-core constraint and the setting of the number of cores also specifies the maximum number of cores.

The following example enables the whole-core constraint on the control domain (primary). First, initiate a delayed reconfiguration on the control domain. Next, assign one whole core to the control domain, and then reboot the domain to make the changes take effect.

```
primary# ldm start-reconf primary
Initiating a delayed reconfiguration operation on the primary domain.
All configuration changes for other domains are disabled until the
```

```
primary domain reboots, at which time the new configuration for the
primary domain also takes effect.

primary# ldm add-vcpu -c 1 primary
primary# reboot
```

Interactions Between the Whole-Core Constraint and Other Domain Features

This section describes the interactions between the whole-core constraint and the following features:

- "CPU Dynamic Reconfiguration" on page 154
- "Dynamic Resource Management" on page 154
- "Domain Migration" on page 155
- "Power Management" on page 155

CPU Dynamic Reconfiguration

The whole-core constraint is fully compatible with CPU dynamic reconfiguration (DR). When a domain is defined with the whole-core constraint, you can use the ldm add-vcpu -c, ldm set-vcpu -c, or remove-vcpu -c command to change the number of cores on an active domain.

However, if a bound or active domain is not in delayed reconfiguration mode, its number of cores cannot exceed the maximum number of cores. This maximum is set with the maximum core constraint, which is automatically enabled when the whole-core constraint is enabled. Any CPU DR operation that does not satisfy the maximum core constraint fails.

Dynamic Resource Management

The whole-core constraint is not compatible with dynamic resource management (DRM). When a DRM policy is enabled on a domain that uses the whole-core constraint, the policy is automatically disabled. The whole-core constraint remains enabled.

Even though a DRM policy cannot be enabled when the whole-core constraint is in effect, you can still define a DRM policy for the domain. Note that when a policy is automatically disabled, it still remains active. The policy is automatically re-enabled if the domain is restarted without the whole-core constraint.

The following are the expected interactions between the whole-core constraint and DRM:

- If the whole-core constraint is set on a domain, a warning message is issued when you attempt to enable a DRM policy on that domain.
- If a DRM policy is in effect on an inactive domain, you are permitted to enable the whole-core constraint on the domain. When the domain becomes active and the policy is enabled, the system automatically disables the DRM policy for the domain.
- If a DRM policy is enabled on an active or bound domain, you are not permitted to enable the whole-core constraint.

Domain Migration

CPU whole-core configuration is incompatible with domain migration. However, you can still migrate a domain that is configured with CPU whole cores. To restore the whole-core constraint after such a migration, stop the domain and reconfigure it for whole-core allocation.

Power Management

The whole-core constraint is fully compatible with the power management (PM) performance and elastic modes. When elastic mode is enabled, the PM subsystem can add or remove CPU cores to or from domains that are configured with the whole-core constraint. In that case, the whole-core constraint continues to be honored, and domains that use that constraint remain configured only with whole cores.

Using Memory Dynamic Reconfiguration

The Oracle VM Server for SPARC 2.0 release introduces memory dynamic reconfiguration (DR). This feature is capacity-based and enables you to add or remove an arbitrary amount of memory to or from an active logical domain.

The following are the requirements and restrictions for using the memory DR feature:

- You can perform memory DR operations on any domain. However, only a single memory DR operation can be in progress on a domain at a given time.
- The memory DR feature enforces 256-Mbyte alignment on the address and size of the memory involved in a given operation. See "Memory Alignment" on page 157.
- 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 158.

If the memory of a domain cannot be reconfigured by using a memory DR operation, the domain must be stopped before the memory can be reconfigured. If the domain is the control domain, you must first initiate a delayed reconfiguration.

Adding Memory

If a domain is active, you can use the `ldm add-memory` command to dynamically add memory to the domain. The `ldm set-memory` command can also dynamically add memory if the specified memory size is greater than the current memory size of the domain.

Removing Memory

If a domain is active, you can use the `ldm remove-memory` command to dynamically remove memory from the domain. The `ldm set-memory` command can also dynamically remove memory if the specified memory size is smaller than the current memory size of the domain.

Memory removal can be a long-running operation. You can track the progress of an operation or cancel an ongoing memory DR request.

Tracking the Progress of a Memory DR Request

You can track the progress of an `ldm remove-memory` command by running the `ldm list -l` command for the specified domain.

Canceling a Memory DR Request

You can cancel a removal request that is in progress by interrupting the `ldm remove-memory` command (by pressing Control-C) or by issuing the `ldm cancel-operation memdr` command. If you cancel a memory removal request, only the outstanding portion of the removal request is affected, namely, the amount of memory still to be removed from the domain.

Partial Memory DR Requests

A memory addition request is rejected if there is insufficient free memory to fulfill the entire request. However, a memory addition request can be partially fulfilled if the target domain fails to add any of the memory requested by the Logical Domains Manager.

A memory removal request is rejected if there is insufficient memory in the domain to fulfill the entire request. However, a memory removal request can be partially fulfilled if the target domain fails to remove any of the memory requested by the Logical Domains Manager.

Note – Memory is cleared after it is removed from a domain and before it is added to another domain.

Memory Reconfiguration of the Control Domain

The memory DR feature can be used to reconfigure the memory of the control domain. If a memory DR request cannot be performed on the control domain, you must first initiate a delayed reconfiguration.

Using memory DR might not be appropriate for removing large amounts of memory from an active domain because memory DR operations might be long running. In particular, during the intial configuration of the system, you should use delayed reconfiguration to decrease the memory in the control domain.

Decrease the Control Domain's Memory

Use a delayed reconfiguration instead of a memory DR to decrease the control domain's memory from an initial factory default configuration. In such a case, the control domain owns all of the host system's memory. The memory DR feature is not well suited for this purpose because an active domain is not guaranteed to add or more typically give up all of the requested memory. Rather, the OS running in that domain makes a best effort to fulfill the request. In addition, memory removal can be a long-running operation. These issues are amplified when large memory operations are involved, as is the case for the initial decrease of the control domain's memory.

For these reasons, use a delayed reconfiguration by following these steps:

1. Use the `ldm start-reconf primary` command to put the control domain in delayed reconfiguration mode.
2. Partition the host system's resources that are owned by the control domain, as necessary.
3. Use the `ldm cancel-reconf` command to undo the operations in Step 2, if necessary, and start over.
4. Reboot the control domain to make the reconfiguration changes take effect.

Dynamic and Delayed Reconfiguration

If a delayed reconfiguration is pending in the control domain, a memory reconfiguration request is rejected for any other domain. If a delayed reconfiguration is not pending in the control domain, a memory reconfiguration request is rejected for any domain that does not support memory DR. A memory reconfiguration request on a control domain that does not support memory DR is converted to a delayed reconfiguration request.

Memory Alignment

Memory reconfiguration requests have different alignment requirments that depend on the state of the domain to which the request is applied.

Memory Alignment for Active Domains

- **Dynamic addition and removal.** The address and size of a memory block are 256-Mbyte-aligned for dynamic addition and dynamic removal. The minimum operation size is 256 Mbytes.

  A nonaligned request or a removal request that is larger than the bound size is rejected.

  Use the following commands to adjust memory allocations:

  - `ldm add-memory`. If you specify the `--auto-adj` option with this command, the amount of memory to be added is 256-Mbyte-aligned, which might increase the amount of memory actually added to the domain.
  - `ldm remove-memory`. If you specify the `--auto-adj` option with this command, the amount of memory to be removed is 256-Mbyte-aligned, which might decrease the amount of memory actually removed from the domain.
  - `ldm set-memory`. This command is treated as an addition or a removal operation. If you specify the `--auto-adj` option, the amount of memory to be added or removed is 256-Mbyte-aligned as previously described. Note that this alignment might increase the resulting memory size of the domain.

- **Delayed reconfiguration.** The address and size of a memory block are 4-Mbyte-aligned. If you make a nonaligned request, the request is rounded up to be 4-Mbyte-aligned.

Memory Alignment for Bound Domains

The address and size of a memory block are 4-Mbyte-aligned for bound domains. If you make a nonaligned request, the request is rounded up to be 4-Mbyte-aligned. This means that the resulting memory size of the domain might be more than you specified.

For the `ldm add-memory`, `ldm set-memory`, and `ldm remove-memory` commands, the `--auto-adj` option rounds up the size of the resulting memory to be 256-Mbyte-aligned. This means that the resulting memory might be more than you specified.

Memory Alignment for Inactive Domains

For the `ldm add-memory`, `ldm set-memory`, and `ldm remove-memory` commands, the `--auto-adj` option rounds up the size of the resulting memory to be 256-Mbyte-aligned. There is no alignment requirement for an inactive domain. The restrictions described in "Memory Alignment for Bound Domains" on page 158 take effect after such a domain is bound.

Adding Unaligned Memory

The memory DR feature enforces 256-Mbyte memory alignment on the address and size of the memory that is dynamically added to or removed from an active domain. This means that any unaligned memory in an active domain cannot be removed by using memory DR.

This also means than any unaligned memory in the free memory pool cannot be added to an active domain by using memory DR.

After all the aligned memory has been allocated, you can use the `ldm add-memory` command to add the remaining unaligned memory to a bound or inactive domain. You can also use this command to add the remaining unaligned memory to the control domain by means of a delayed reconfiguration operation.

The following example shows how to add the two remaining 128-Mbyte memory blocks to the `primary` and `ldom1` domains. The `ldom1` domain is in the bound state. The following commands add the two remaining memory blocks. The first command initiates a delayed reconfiguration operation on the control domain. The second command adds one of the 128-Mbyte memory blocks to the control domain. The fifth command adds the other 128-Mbyte memory block to the `ldom1` domain.

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

# ldm add-memory 128M primary
------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
------------------------------------------------------------------------------

# ldm list
NAME            STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary         active     -ndcv-  SP      8     2688M    0.1%  23d 8h 8m

# ldm list
NAME            STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary         active     -n-cv-  SP      8     2560M    0.5%  23d 8h 9m
ldom1           bound      ------  5000    1     524M

# ldm add-mem 128M ldom1
# ldm list
NAME            STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary         active     -n-cv-  SP      8     2560M    0.1%  23d 8h 9m
ldom1           bound      ------  5000    1     652M
```

Memory DR Examples

The following examples show how to perform memory DR operations. For information about the related CLI commands, see the ldm(1M) man page.

EXAMPLE 10–1 Memory DR Operations on Active Domains

This example shows how to dynamically add memory to and remove it from an active domain, `ldom1`.

EXAMPLE 10–1   Memory DR Operations on Active Domains   *(Continued)*

The ldm list output shows the memory for each domain in the Memory field. The first ldm add-mem command exits with an error because you must specify memory in multiples of 256 Mbytes. The next ldm add-mem command uses the --auto-adj option so that even though you specify 200M as the amount of memory to add, the amount is rounded up to 256 Mbytes.

The ldm rm-mem command exits with an error because you must specify memory in multiples of 256 Mbytes. When you add the --auto-adj option to the same command, the memory removal succeeds, as the amount of memory is rounded down to the next 256-Mbyte boundary.

```
# ldm list
NAME             STATE      FLAGS    CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-   SP      4     27392M   0.4%  1d 22h 53m
ldom1            active     -n----   5000    2     2G       0.4%  1d 1h 23m
ldom2            bound      ------   5001    2     200M

# ldm add-mem 200M ldom1
The size of memory must be a multiple of 256MB.

# ldm add-mem --auto-adj 200M ldom1
Adjusting request size to 256M.
The ldom1 domain has been allocated 56M more memory
than requested because of memory alignment constraints.

# ldm list
NAME             STATE      FLAGS    CONS    VCPU    MEMORY    UTIL    UPTIME
primary          active     -n-cv-   SP      4       27392M    5.0%    8m
ldom1            active     -n----   5000    2       2304M     0.5%    1m
ldom2            bound      ------   5001    2       200M

# ldm rm-mem --auto-adj 300M ldom1
Adjusting requested size to 256M.
The ldom1 domain has been allocated 44M more memory
than requested because of memory alignment constraints.

# ldm list
NAME             STATE      FLAGS    CONS    VCPU    MEMORY    UTIL    UPTIME
primary          active     -n-cv-   SP      4       27392M    0.3%    8m
ldom1            active     -n----   5000    2       2G        0.2%    2m
ldom2            bound      ------   5001    2       200M
```

EXAMPLE 10–2   Memory DR Operations on Bound Domains

This example shows how to add memory to and remove it from a bound domain, ldom2.

The ldm list output shows the memory for each domain in the Memory field. The first ldm add-mem command adds 100 Mbytes of memory to the ldom2 domain. The next ldm add-mem command specifies the --auto-adj option, which causes an additional 112 Mbytes of memory to be dynamically added to ldom2.

The ldm rm-mem command dynamically removes 100 Mbytes from the ldom2 domain. If you specify the --auto-adj option to the same command to remove 300 Mbytes of memory, the amount of memory is rounded down to the next 256-Mbyte boundary.

EXAMPLE 10–2 Memory DR Operations on Bound Domains   *(Continued)*

```
# ldm list
NAME            STATE      FLAGS    CONS    VCPU    MEMORY    UTIL    UPTIME
primary         active     -n-cv-   SP      4       27392M    0.4%    1d 22h 53m
ldom1           active     -n----   5000    2       2G        0.4%    1d 1h 23m
ldom2           bound      ------   5001    2       200M

# ldm add-mem 100M ldom2

# ldm list
NAME            STATE      FLAGS    CONS    VCPU    MEMORY    UTIL    UPTIME
primary         active     -n-cv-   SP      4       27392M    0.5%    1d 22h 54m
ldom1           active     -n----   5000    2       2G        0.2%    1d 1h 25m
ldom2           bound      ------   5001    2       300M

# ldm add-mem --auto-adj 100M ldom2
Adjusting request size to 256M.
The ldom2 domain has been allocated 112M more memory
than requested because of memory alignment constraints.

# ldm list
NAME            STATE      FLAGS    CONS    VCPU    MEMORY    UTIL    UPTIME
primary         active     -n-cv-   SP      4       27392M    0.4%    1d 22h 55m
ldom1           active     -n----   5000    2       2G        0.5%    1d 1h 25m
ldom2           bound      ------   5001    2       512M

# ldm rm-mem 100M ldom2
# ldm list
NAME            STATE      FLAGS    CONS    VCPU    MEMORY    UTIL    UPTIME
primary         active     -n-cv-   SP      4       27392M    3.3%    1d 22h 55m
ldom1           active     -n----   5000    2       2G        0.2%    1d 1h 25m
ldom2           bound      ------   5001    2       412M

# ldm rm-mem --auto-adj 300M ldom2
Adjusting request size to 256M.
The ldom2 domain has been allocated 144M more memory
than requested because of memory alignment constraints.

# ldm list
NAME            STATE      FLAGS    CONS    VCPU    MEMORY    UTIL    UPTIME
primary         active     -n-cv-   SP      4       27392M    0.5%    1d 22h 55m
ldom1           active     -n----   5000    2       2G        0.2%    1d 1h 26m
ldom2           bound      ------   5001    2       256M
```

EXAMPLE 10–3 Setting Domain Memory Sizes

This example shows how to use the `ldm set-memory` command to add memory to and remove it from a domain.

The `ldm list` output shows the memory for each domain in the Memory field. The first `ldm set-mem` command attempts to set the `primary` domain's size to 3400 Mbytes. The resulting error states that the specified value is not on a 256-Mbyte boundary. Adding the `--auto-adj` option to the same command enables you to successfully remove some memory and stay on the 256-Mbyte boundary. This command also issues a warning to state that not all of the requested memory could be removed as the domain is using that memory.

EXAMPLE 10–3   Setting Domain Memory Sizes   *(Continued)*

The next ldm set-mem command sets the memory size of the ldom2 domain, which is in the bound state, to 690 Mbytes. If you add the --auto-adj option to the same command, an additional 78 Mbytes of memory is dynamically added to ldom2 to stay on a 256-Mbyte boundary.

```
# ldm list
NAME            STATE       FLAGS    CONS    VCPU  MEMORY    UTIL  UPTIME
primary         active      -n-cv-   SP      4     27392M    0.5%  1d 22h 55m
ldom1           active      -n----   5000    2     2G        0.2%  1d 1h 26m
ldom2           bound       ------   5001    2     256M

# ldm set-mem 3400M primary
An ldm set-mem 3400M command would remove 23992MB, which is not a multiple
of 256MB. Instead, run ldm rm-mem 23808MB to ensure a 256MB alignment.

# ldm set-mem --auto-adj 3400M primary
Adjusting request size to 3.4G.
The primary domain has been allocated 184M more memory
than requested because of memory alignment constraints.
Only 9472M of memory could be removed from the primary domain
because the rest of the memory is in use.

# ldm set-mem 690M ldom2
# ldm list
NAME            STATE       FLAGS    CONS    VCPU  MEMORY    UTIL  UPTIME
primary         active      -n-cv-   SP      4     17920M    0.5%  1d 22h 56m
ldom1           active      -n----   5000    2     2G        0.6%  1d 1h 27m
ldom2           bound       ------   5001    2     690M

# ldm set-mem --auto-adj 690M ldom2
Adjusting request size to 256M.
The ldom2 domain has been allocated 78M more memory
than requested because of memory alignment constraints.

# ldm list
NAME            STATE       FLAGS    CONS    VCPU  MEMORY    UTIL  UPTIME
primary         active      -n-cv-   SP      4     17920M    2.1%  1d 22h 57m
ldom1           active      -n----   5000    2     2G        0.2%  1d 1h 27m
ldom2           bound       ------   5001    2     768M
```

Using Power Management

To use Power Management (PM), you first need to set the PM mode in the Oracle Integrated Lights Out Manager (ILOM) 3.0 firmware. This section summarizes the information that you need to be able to use PM with the Oracle VM Server for SPARC software.

For more information about ILOM, see the following:

- "Monitoring Power Consumption" in the *Sun Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide* (`http://dlc.sun.com/pdf/820-6412-11/820-6412-11.pdf`)

- *Oracle Integrated Lights Out Manager (ILOM) 3.0 Feature Updates and Release Notes* (`http://dlc.sun.com/pdf/820-7329-16/820-7329-16.pdf`)

The power mode is the setting that governs system power usage at any point in time. Starting with the Logical Domains 1.3 release, the following power modes are supported, assuming that the underlying platform has implemented PM features:

- **Performance mode.** The system is allowed to use all the power that is available.

- **Elastic mode.** The system power usage is adapted to the current utilization level. For example, the power state of resources is reduced as utilization decreases.

Following are the PM features:

- **CPU core auto-disabling.** PM automatically disables a CPU core when all the strands on that core have been disabled.

- **CPU clock cycle skip.** Starting with the Oracle VM Server for SPARC 2.0 release, PM can automatically adjust the CPU clock cycle skip on SPARC T3 platforms. The adjustment can increase or decrease the number of clock cycles that are skipped to keep all domains within the power utilization thresholds. PM determines whether to make such adjustments based on CPU utilization. When the system goes in to performance mode, the number of clock cycles that are skipped is automatically adjusted to being none.

- **Memory operations in deep idle mode.** Starting with the Oracle VM Server for SPARC 2.0 release, when SPARC T3 platforms are in elastic mode, they automatically configure under-utilized memory to operate in a deeper idle mode to save power.

- **Power limit.** Starting with the Oracle VM Server for SPARC 2.0 release, you can set a *power limit* on SPARC T3 platforms to restrict the power draw of a system. If the power draw is greater than the power limit, PM techniques are used to reduce power. You can use the ILOM service processor (SP) to set the power limit.

  See the following documents:

  - *Sun Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide* (`http://dlc.sun.com/pdf/820-6412-11/820-6412-11.pdf`)

  - *Oracle Integrated Lights Out Manager (ILOM) 3.0 Feature Updates and Release Notes* (`http://dlc.sun.com/pdf/820-7329-16/820-7329-16.pdf`)

  You can use the ILOM interface to set a power limit, grace period, and violation action. If the power limit is exceeded for more than the grace period, the violation action is performed.

  If the current power draw exceeds the power limit, an attempt is made to reduce the power state of resources that can be power-managed. If the power draw drops below the power

limit, the power state of those resources is permitted to increase. If the system is in elastic mode, an increase in the power state of resources is driven by the utilization level.

When a system is in elastic mode, some domain configuration modifications are first validated to confirm that the power limit is not exceeded. If the power limit is exceeded, only some of the resources might be modified or added as requested. If the power limit is later increased, you can then add any resources that were unsuccessfully modified.

If a domain's load causes resources to consume more power, only the resources that maintain the power draw being under the power limit are successfully powered up.

For instructions on configuring the power mode by using the ILOM 3.0 firmware CLI, see "Monitoring Power Consumption" in the *Sun Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide* (http://dlc.sun.com/pdf/820-6412-11/820-6412-11.pdf).

Listing CPU Power-Managed Strands

This section shows how to list power-managed strands and virtual CPUs.

▼ List CPU Power-Managed Strands

● **List power-managed strands by using one of the following commands:**

a. **Use the `list -l` subcommand.**

In the output, a dash (---) in the UTIL column for the CPU means that the strand is power-managed.

```
# ldm list -l primary
NAME          STATE    FLAGS  CONS  VCPU MEMORY  UTIL  UPTIME
primary       active   -n-cv  SP    8    4G      4.3%  7d 19h 43m

SOFTSTATE
Solaris running

MAC
    00:14:4f:fa:ed:88

HOSTID
    0x84faed88

CONTROL
    failure-policy=ignore

DEPENDENCY
    master=

VCPU
    VID     PID     UTIL STRAND
    0       0       0.0%  100%
```

```
1        1        ---      100%
2        2        ---      100%
3        3        ---      100%
4        4        ---      100%
5        5        ---      100%
6        6        ---      100%
7        7        ---      100%
....
```

b. Use the parseable option (-p) to the list -l subcommand.

In the output, a blank after util= means that the strand is power-managed.

```
# ldm list -l -p

VCPU
|vid=0|pid=0|util=0.7%|strand=100
|vid=1|pid=1|util=|strand=100
|vid=2|pid=2|util=|strand=100
|vid=3|pid=3|util=|strand=100
|vid=4|pid=4|util=0.7%|strand=100
|vid=5|pid=5|util=|strand=100
|vid=6|pid=6|util=|strand=100
|vid=7|pid=7|util=|strand=100
```

▼ List Power-Managed CPUs

● **List power-managed CPUs by using one of the following commands:**

a. Use the list-devices -a cpu command.

In the PM column of the output, a yes means that the CPU is power-managed, and a no means that the CPU is powered on. It is assumed that 100 percent free CPUs are power-managed by default, hence the dash (---) under the PM column for them.

```
# ldm list-devices -a cpu
VCPU
    PID      %FREE    PM
    0        0        no
    1        0        yes
    2        0        yes
    3        0        yes
    4        100      ---
    5        100      ---
    6        100      ---
    7        100      ---
```

b. Use the parseable option (-p) to the list-devices -a cpu subcommand.

In the pm= field in the output, a yes means that the CPU is power-managed, and a no means that the CPU is powered on. It is assumed that 100 percent free CPUs are power-managed by default, hence the blank in that field for them.

```
# ldm list-devices -a -p cpu
VERSION 1.4
VCPU
```

```
|pid=0|free=0|pm=no
|pid=1|free=0|pm=yes
|pid=2|free=0|pm=yes
|pid=3|free=0|pm=yes
|pid=4|free=0|pm=no
|pid=5|free=0|pm=yes
|pid=6|free=0|pm=yes
|pid=7|free=0|pm=yes
|pid=8|free=100|pm=
|pid=9|free=100|pm=
|pid=10|free=100|pm=
```

Using Dynamic Resource Management

Starting with the Logical Domains 1.3 software, you can use policies to determine how to automatically perform DR activities. At this time, you can *only* create policies to govern the dynamic resource management of virtual CPUs.

> ⚠ **Caution** – The following issues affect CPU dynamic resource management (DRM):
>
> - When PM is in elastic mode, DRM cannot be enabled.
> - Any change from performance mode to elastic mode is delayed while DRM is enabled.
> - Ensure that you disable CPU DRM prior to performing a domain migration operation.
> - DRM policies do not apply to domains that are configured with the whole-core constraint.

A *resource management policy* specifies under what conditions virtual CPUs can be automatically added to and removed from a logical domain. A policy is managed by using the ldm add-policy, ldm set-policy, and ldm remove-policy commands:

```
ldm add-policy [enable=yes|no] [priority=value] [attack=value] [decay=value]
  [elastic-margin=value] [sample-rate=value] [tod-begin=hh:mm[:ss]]
  [tod-end=hh:mm[:ss]] [util-lower=percent] [util-upper=percent] [vcpu-min=value]
  [vcpu-max=value] name=policy-name ldom...
ldm set-policy [enable=[yes|no]] [priority=[value]] [attack=[value]] [decay=[value]]
  [elastic-margin=[value]] [sample-rate=[value]] [tod-begin=[hh:mm:ss]]
  [tod-end=[hh:mm:ss]] [util-lower=[percent]] [util-upper=[percent]] [vcpu-min=[value]]
  [vcpu-max=[value]] name=policy-name ldom...
ldm remove-policy [name=]policy-name... ldom
```

For information about these commands and about creating resource management policies, see the ldm(1M) man page.

A policy is in effect during the times specified by the tod-begin and tod-end properties. The policy uses the value of the priority property to determine which policy to use if more than one policy is in effect simultaneously.

The policy uses the util-high and util-low property values to specify the high and low thresholds for CPU utilization. If the utilization exceeds the value of util-high, virtual CPUs

are added to the domain until the number is between the vcpu-min and vcpu-max values. If the utilization drops below the util-low value, virtual CPUs are removed from the domain until the number is between the vcpu-min and vcpu-max values. If vcpu-min is reached, no more virtual CPUs can be dynamically removed. If the vcpu-max is reached, no more virtual CPUs can be dynamically added.

EXAMPLE 10–4 Adding Resource Management Policies

For example, after observing the typical utilization of your systems over several weeks, you might set up policies to optimize resource usage. The highest usage is daily from 9:00 a.m. to 6:00 p.m. Pacific, and the low usage is daily from 6:00 p.m. to 9:00 a.m. Pacific.

Based on this system utilization observation, you decide to create the following high and low policies based on overall system utilization:

- **High:** Daily from 9:00 a.m. to 6:00 p.m. Pacific
- **Low:** Daily from 6:00 p.m. to 9:00 a.m. Pacific

The following ldm add-policy command creates the high-usage policy to be used during the high utilization period on the ldom1 domain.

The following high-usage policy does the following:

- Specifies that the beginning and ending times are 9:00 a.m. and 6:00 p.m. by setting the tod-begin and tod-end properties, respectively.
- Specifies that the lower and upper limits at which to perform policy analysis are 25 percent and 75 percent by setting the util-lower and util-upper properties, respectively.
- Specifies that the minimum and maximum number of virtual CPUs is 2 and 16 by setting the vcpu-min and vcpu-max properties, respectively.
- Specifies that the maximum number of virtual CPUs to be added during any one resource control cycle is 1 by setting the attack property.
- Specifies that the maximum number of virtual CPUs to be removed during any one resource control cycle is 1 by setting the decay property.
- Specifies that the priority of this policy is 1 by setting the priority property. A priority of 1 means that this policy will be enforced even if another policy can take effect.
- Specifies that the name of the policy file is high-usage by setting the name property.
- Uses the default values for those properties that are not specified, such as enable and sample-rate. See the ldm(1M) man page.

```
# ldm add-policy tod-begin=09:00 tod-end=18:00 util-lower=25 util-upper=75 \
vcpu-min=2 vcpu-max=16 attack=1 decay=1 priority=1 name=high-usage ldom1
```

The following ldm add-policy command creates the med-usage policy to be used during the low utilization period on the ldom1 domain.

EXAMPLE 10–4 Adding Resource Management Policies   *(Continued)*

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.

```
# ldm add-policy tod-begin=18:00 tod-end=09:00 util-lower=10 util-upper=50 \
 vcpu-min=2 vcpu-max=16 attack=1 decay=1 priority=1 name=med-usage ldom1
```

Listing Domain Resources

This section shows the syntax usage for the ldm subcommands, defines some output terms, such as flags and utilization statistics, and provides examples that are similar to what you actually see as output.

Machine-Readable Output

If you are creating scripts that use ldm list command output, *always* use the -p option to produce the machine-readable form of the output. See "Generate a Parseable, Machine-Readable List (-p)" on page 170 for more information.

▼ Show Syntax Usage for ldm Subcommands

● **Look at syntax usage for all ldm subcommands.**

primary# **ldm --help**

For more information about the ldm subcommands, see the ldm(1M) man page.

Flag Definitions

The following flags can be shown in the output for a domain (`ldm list`). If you use the long, parseable options (`-l -p`) for the command, the flags are spelled out; for example, `flags=normal,control,vio-service`. If not, you see the letter abbreviation; for example `-n-cv-`. The list flag values are position dependent. Following are the values that can appear in each of the six columns from left to right.

Column 1

- `s` starting or stopping
- `-` placeholder

Column 2

- `n` normal
- `t` transition

Column 3

- `d` delayed reconfiguration
- `r` memory dynamic reconfiguration (DR)
- `-` placeholder

Column 4

- `c` control domain
- `-` placeholder

Column 5

- `v` virtual I/O service domain
- `-` placeholder

Column 6

- `s` source domain in a migration
- `t` target domain in a migration
- `e` error occurred during a migration
- `-` placeholder

Utilization Statistic Definition

The per virtual CPU utilization statistic (`UTIL`) is shown on the long (`-l`) option of the `ldm list` command. The statistic is the percentage of time that the virtual CPU spent executing on behalf of the guest operating system. A virtual CPU is considered to be executing on behalf of the guest operating system except when it has been yielded to the hypervisor. If the guest operating system does not yield virtual CPUs to the hypervisor, the utilization of CPUs in the guest operating system will always show as `100%`.

The utilization statistic reported for a logical domain is the average of the virtual CPU utilizations for the virtual CPUs in the domain. A dash (- - -) in the UTIL column means that the strand is power-managed.

Viewing Various Lists

▼ Show Software Versions (-V)

● **View the current software versions installed.**

```
primary# ldm -V
```

▼ Generate a Short List

● **Generate a short list for all domains.**

```
primary# ldm list
```

▼ Generate a Long List (-l)

● **Generate a long list for all domains.**

```
primary# ldm list -l
```

▼ Generate an Extended List (-e)

● **Generate an extended list of all domains.**

```
primary# ldm list -e
```

▼ Generate a Parseable, Machine-Readable List (-p)

● **Generate a parseable, machine-readable list of all domains.**

```
primary# ldm list -p
```

▼ Generate a Subset of a Long List (-o *format*)

● **Generate output as a subset of resources by entering one or more of the following *format* options. If you specify more than one format, delimit the items by a comma with no spaces.**

```
primary# ldm list -o resource[,resource...] ldom
```

- console – Output contains virtual console (vcons) and virtual console concentrator (vcc) service
- core – Output contains information about domains that have whole cores allocated

- `cpu` – Output contains information about the virtual CPU (`vcpu`), physical CPU (`pcpu`), and core ID
- `crypto` – Cryptographic unit output contains Modular Arithmetic Unit (`mau`) and any other LDoms-supported cryptographic unit, such as the Control Word Queue (CWQ)
- `disk` – Output contains virtual disk (`vdisk`) and virtual disk server (`vds`)
- `domain` – Output contains variables (`var`), host ID (`hostid`), domain state, flags, UUID, and software state
- `memory` – Output contains `memory`
- `network` – Output contains media access control (`mac`) address , virtual network switch (`vsw`), and virtual network (`vnet`) device
- `physio` – Physical input/output contains peripheral component interconnect (`pci`) and network interface unit (`niu`)
- `resmgmt` – Output contains dynamic resource management (DRM) policy information, indicates which policy is currently running, and lists constraints related to whole-core configuration
- `serial` – Output contains virtual logical domain channel (`vldc`) service, virtual logical domain channel client (`vldcc`), virtual data plane channel client (`vdpcc`), virtual data plane channel service (`vdpcs`)
- `stats` – Output contains statistics that are related to resource management policies
- `status` – Output contains status about a domain migration in progress

The following examples show various subsets of output that you can specify:

- List CPU information for the control domain

  ```
  # ldm list -o cpu primary
  ```
- List domain information for a guest domain

  ```
  # ldm list -o domain ldm2
  ```
- List memory and network information for a guest domain

  ```
  # ldm list -o network,memory ldm1
  ```
- List DRM policy information for a guest domain

  ```
  # ldm list -o resmgmt,stats ldm1
  ```

▼ List a Variable

● **Show a variable and its value for a domain.**

```
primary# ldm list-variable variable-name ldom
```

For example, the following command shows the value for the boot-device variable on the ldg1 domain:

```
primary# ldm list-variable boot-device ldg1
boot-device=/virtual-devices@100/channel-devices@200/disk@0:a
```

▼ List Bindings

● **List the resources that are bound to a domain.**

```
primary# ldm list-bindings ldom
```

▼ List Configurations

● **List logical domain configurations that have been stored on the SP.**

Example 10–5 Configurations List

The ldm list-config command lists the logical domain configurations that are stored on the service processor. When used with the -r option, this command lists those configurations for which autosave files exist on the control domain.

For more information about configurations, see "Managing Logical Domains Configurations" on page 178. For more examples, see the ldm(1M) man page.

```
primary# ldm list-config
factory-default
3guests
foo [next poweron]
primary
reconfig-primary
```

More Information Meaning of Labels

The labels to the right of the configuration name mean the following:

- [current] – Last booted configuration, only as long as it matches the currently running configuration; that is, until you initiate a reconfiguration. After the reconfiguration, the annotation changes to [next poweron].

- [next poweron] – Configuration to be used at the next powercycle.

▼ List Devices

● **List all server resources, bound and unbound.**

```
primary# ldm list-devices -a
```

▼ List Available Memory

● **List the amount of memory available to be allocated.**

```
primary# ldm list-devices mem
MEMORY
    PA                SIZE
    0x14e000000       2848M
```

▼ List Services

● **List the services that are available.**

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

Listing Constraints

To the Logical Domains Manager, constraints are one or more resources you want to have assigned to a particular domain. You either receive all the resources you ask to be added to a domain or you get none of them, depending upon the available resources. The list-constraints subcommand lists those resources you requested assigned to the domain.

▼ List Constraints for One Domain

● **List constraints for one domain.**

```
primary# ldm list-constraints ldom
```

▼ List Constraints in XML Format

● **List constraints in XML format for a particular domain.**

```
primary# ldm list-constraints -x ldom
```

▼ List Constraints in a Machine-Readable Format

● **List constraints for all domains in a parseable format.**

```
primary# ldm list-constraints -p
```

11

Managing Configurations

This chapter contains information about managing domain configurations.

This chapter covers the following topics:

- "Saving Domain Configurations for Future Rebuilding" on page 175
- "Managing Logical Domains Configurations" on page 178

Saving Domain Configurations for Future Rebuilding

The basic process is to save the resource constraints information for each domain into an XML file, which can then be re-issued to the Logical Domains Manager, for example, after a hardware failure to rebuild a desired configuration.

"Restore a Domain Configuration From an XML File (`ldm add-domain`)" on page 176 works for guest domains, but not for the control (`primary`) domain. You can save the `primary` domain's constraints to an XML file, but you cannot feed the file back into the `ldm add-domain -i` command. However, you can use the `ldm init-system` command and the resource constraints from the XML file to reconfigure your `primary` domain. You can also use the `ldm init-system` command to reconfigure other domains that are described in the XML file, but those domains are left inactive when the configuration is complete.

The method that follows does not preserve actual bindings, only the constraints used to create those bindings. This means that, after this procedure, the domains will have the same virtual resources, but will not necessarily be bound to the same physical resources.

▼ Save Domain Configurations

This procedure shows how to save a domain configuration for a single domain or for all the domains on a system.

● **Save a domain configuration for one or more domains.**

   ■ **To save the configuration for a single domain, create an XML file containing the domain's constraints.**

   # **ldm list-constraints -x** *ldom* **>***ldom***.xml**

   The following example shows how to create an XML file, ldg1.xml, which contains the ldg1 domain's constraints:

   # **ldm list-constraints -x ldg1 >ldg1.xml**

   ■ **To save the configurations for all the domains on a system, create an XML file containing the constraints for all domains.**

   # **ldm list-constraints -x >***file***.xml**

   The following example shows how to create an XML file, config.xml, which contains the constraints for all the domains on a system:

   # **ldm list-constraints -x >config.xml**

▼ Restore a Domain Configuration From an XML File (`ldm add-domain`)

Instead of this procedure, you can use the ldm init-system command to restore domain configurations from an XML file. See "Restore a Domain Configuration From an XML File (ldm init-system)" on page 177.

1 Create the domain by using the XML file that you created as input.

   # **ldm add-domain -i** *ldom***.xml**

2 Bind the domain.

   # **ldm bind-domain** *ldom*

3 Start the domain.

   # **ldm start-domain** *ldom*

Example 11–1 Restoring a Single Domain From an XML File

The following example shows how to restore a single domain. First, you restore the ldg1 domain from the XML file. Then, you bind and restart the ldg1 domain that you restored.

```
# ldm add-domain -i ldg1.xml
# ldm bind ldg1
# ldm start ldg1
```

▼ Restore a Domain Configuration From an XML File (`ldm init-system`)

This procedure explains how to use the `ldm init-system` command with an XML file to re-create a previously saved configuration. The XML file describes one or more domain configurations. The XML file can be created by running the `ldm ls-constraints -x` command. The `ldm init-system` command is expected to be run in the `factory-default` configuration, but it can restore any configuration from an XML file. The `primary` domain is reconfigured as specified in the file, and any non-`primary` domains that have configurations in the XML file are reconfigured but left inactive.

Instead of this procedure, you can use the `ldm add-domain` command to restore a single domain configuration from an XML file. See .

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 .

3 Become superuser or assume an equivalent role.

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

4 Restore the domain configuration or configurations from the XML file.

```
# ldm init-system [-rs] -i filename.xml
```

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.

Example 11–2 Restoring Domains From XML Configuration Files

The following examples show how to use the `ldm init-system` command to restore the `primary` domain and all the domains on a system from the `factory-default` configuration.

- **Restore the `primary` domain.** The `-r` option is used to reboot the `primary` domain after the configuration completes. The `primary.xml` file contains the XML domain configuration that you saved at an earlier time.

  ```
  primary# ldm init-system -r -i primary.xml
  ```

- **Restore all the domains on a system.** Restore the domains on the system to the configurations in the config.xml XML file. The config.xml file contains the XML domain configurations that you saved at an earlier time. The primary domain is restarted automatically by the ldm init-system command. Any other domains are restored, but not bound and restarted.

```
# ldm init-system -r -i config.xml
```

After the system reboots, the following commands bind and restart the ldg1 and ldg2 domains:

```
# ldm bind ldg1
# ldm start ldg1
# ldm bind ldg2
# ldm start ldg2
```

Managing Logical Domains Configurations

A Logical Domains *configuration* is a complete description of all the domains and their resource allocations within a single system. You can save and store configurations on the service processor (SP) for later use.

When you power up a system, the SP boots the selected configuration. By booting a configuration, the system runs the same set of domains, and uses the same virtualization and partitioning resource allocations that are specified in the configuration. The default configuration is the one that is most recently saved.

Starting with the Logical Domains 1.2 release, a copy of the current configuration is automatically saved on the control domain whenever the Logical Domains configuration is changed.

The autosave operation occurs immediately, even in the following situations:

- When the new configuration is not explicitly saved on the SP
- When the actual configuration change is not made until after the affected domain reboots

This autosave operation enables you to recover a configuration when the configurations that are saved on the SP are lost. This operation also enables you to recover a configuration when the current configuration was not explicitly saved to the SP when the system powercycled. In these circumstances, the Logical Domains Manager can restore that configuration on restart if it is newer than the configuration marked for the next boot.

Note – Power management, FMA, ASR, and PRI update events do not cause an update to the autosave files.

You can automatically or manually restore autosave files to new or existing configurations. By default, when an autosave configuration is newer than the corresponding running configuration, a message is written to the Logical Domains log. Thus, you must use the ldm add-spconfig -r command to manually update an existing configuration or create a new one based on the autosave data.

Note – When a delayed reconfiguration is pending, the configuration changes are immediately autosaved. As a result, if you run the ldm list-config -r command, the autosave configuration is shown as being newer than the current configuration.

For information about how to use the ldm *-spconfig commands to manage configurations and to manually recover autosave files, see the ldm(1M) man page.

For information about how to select a configuration to boot, see "Using Logical Domains With the Service Processor" on page 186.

▼ Modify the Autorecovery Policy

The autorecovery policy specifies how to handle the recovery of a configuration when one configuration that is automatically saved on the control domain is newer than the corresponding running configuration. The autorecovery policy is specified by setting the autorecovery_policy property of the ldmd SMF service. The autorecovery_policy property can have the following values:

- autorecovery_policy=1 – Logs warning messages when an autosave configuration is newer than the corresponding running configuration. These messages are logged in the ldmd SMF log file. The user must manually perform any configuration recovery. This is the default policy.

- autorecovery_policy=2 – Displays a notification message if an autosave configuration is newer than the corresponding running configuration. This notification message is printed in the output of any ldm command the first time an ldm command is issued after each restart of the Logical Domains Manager. The user must manually perform any configuration recovery.

- autorecovery_policy=3 – Automatically updates the configuration if an autosave configuration is newer than the corresponding running configuration. This action overwrites the SP configuration that will be used during the next powercycle. This configuration is updated with the newer configuration that is saved on the control domain. This action does not impact the currently running configuration. It only impacts the configuration that will be used during the next powercycle. A message is also logged, which states that a newer configuration has been saved on the SP and that it will be booted the next time the system is powercycled. These messages are logged in the ldmd SMF log file.

1 **Log in to the control domain.**

2 Become superuser or assume an equivalent role.

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

3 View the `autorecovery_policy` property value.

```
# svccfg -s ldmd listprop ldmd/autorecovery_policy
```

4 Stop the `ldmd` service.

```
# svcadm disable ldmd
```

5 Change the `autorecovery_policy` property value.

```
# svccfg -s ldmd setprop ldmd/autorecovery_policy=value
```

For example, to set the policy to perform autorecovery, set the property value to 3:

```
# svccfg -s ldmd setprop ldmd/autorecovery_policy=3
```

6 Refresh and restart the `ldmd` service.

```
# svcadm refresh ldmd
# svcadm enable ldmd
```

Example 11–3   Modifying the Autorecovery Policy From Log to Autorecovery

The following example shows how to view the current value of the `autorecovery_policy` property and change it to a new value. The original value of this property is 1, which means that autosave changes are logged. The svcadm command is used to stop and restart the ldmd service, and the svccfg command is used to view and set the property value.

```
# svccfg -s ldmd listprop ldmd/autorecovery_policy
ldmd/autorecovery_policy integer 1
# svcadm disable ldmd
# svccfg -s ldmd setprop ldmd/autorecovery_policy=3
# svcadm refresh ldmd
# svcadm enable ldmd
```

12

Performing Other Administration Tasks

This chapter contains information and tasks about using the Oracle VM Server for SPARC software that are not described in the preceding chapters.

This chapter covers the following topics:

Entering Names in the CLI

The following sections describe the restrictions on entering names in the Logical Domains Manager CLI.

File Names (*file*) and Variable Names (*var-name*)

- First character must be a letter, a number, or a forward slash (/).
- Subsequent letters must be letters, numbers, or punctuation.

Virtual Disk Server *backend* and Virtual Switch Device Names

The names must contain letters, numbers, or punctuation.

Configuration Name (*config-name*)

The logical domain configuration name (*config-name*) that you assign to a configuration stored on the service processor (SP) must have no more than 64 characters.

All Other Names

The remainder of the names, such as the logical domain name (*ldom*), service names (*vswitch-name*, *service-name*, *vdpcs-service-name*, and *vcc-name*), virtual network name (*if-name*), and virtual disk name (*disk-name*), must be in the following format:

- First character must be a letter or number.
- Subsequent characters must be letters, numbers, or any of the following characters
  `-_+#.:;~()`.

Connecting to a Guest Console Over a Network

You can connect to a guest console over a network if the listen_addr property is set to the IP address of the control domain in the vntsd(1M) SMF manifest. For example:

```
$ telnet host-name 5001
```

Note – Enabling network access to a console has security implications. Any user can connect to a console and for this reason it is disabled by default.

A Service Management Facility manifest is an XML file that describes a service. For more information about creating an SMF manifest, refer to the Solaris 10 System Administrator Collection (http://docs.sun.com/app/docs/coll/47.16).

Note – To access a non-English OS in a guest domain through the console, the terminal for the console must be in the locale required by the OS.

Using Console Groups

The virtual network terminal server daemon, vntsd(1M), enables you to provide access for multiple domain consoles using a single TCP port. At the time of domain creation, the Logical Domains Manager assigns a unique TCP port to each console by creating a new default group for that domain's console. The TCP port is then assigned to the console group as opposed to the console itself. The console can be bound to an existing group using the set-vcons subcommand.

▼ Combine Multiple Consoles Into One Group

1 Bind the consoles for the domains into one group.

The following example shows binding the console for three different domains (ldg1, ldg2, and ldg3) to the same console group (group1).

```
primary# ldm set-vcons group=group1 service=primary-vcc0 ldg1
primary# ldm set-vcons group=group1 service=primary-vcc0 ldg2
primary# ldm set-vcons group=group1 service=primary-vcc0 ldg3
```

2 Connect to the associated TCP port (localhost at port 5000 in this example).

```
# telnet localhost 5000
primary-vnts-group1: h, l, c{id}, n{name}, q:
```

You are prompted to select one of the domain consoles.

3 List the domains within the group by selecting l (list).

```
primary-vnts-group1: h, l, c{id}, n{name}, q: l
DOMAIN ID           DOMAIN NAME                 DOMAIN STATE
0                   ldg1                        online
1                   ldg2                        online
2                   ldg3                        online
```

Note – To re-assign the console to a different group or vcc instance, the domain must be unbound; that is, it has to be in the inactive state. Refer to the Oracle Solaris 10 OS vntsd(1M) man page for more information about configuring and using SMF to manage vntsd and using console groups.

Stopping a Heavily Loaded Domain Can Time Out

An `ldm stop-domain` command can time out before the domain completes shutting down. When this happens, an error similar to the following is returned by the Logical Domains Manager.

```
LDom ldg8 stop notification failed
```

However, the domain could still be processing the shutdown request. Use the `ldm list-domain` command to verify the status of the domain. For example:

```
# ldm list-domain ldg8
NAME         STATE    FLAGS  CONS   VCPU MEMORY  UTIL UPTIME
ldg8         active   s----  5000   22   3328M   0.3% 1d 14h 31m
```

The preceding list shows the domain as active, but the `s` flag indicates that the domain is in the process of stopping. This should be a transitory state.

The following example shows the domain has now stopped.

```
# ldm list-domain ldg8
NAME         STATE    FLAGS  CONS   VCPU MEMORY  UTIL UPTIME
ldg8         bound    -----  5000   22   3328M
```

Operating the Oracle Solaris OS With Oracle VM Server for SPARC

This section describes the changes in behavior in using the Oracle Solaris OS that occur once a configuration created by the Logical Domains Manager is instantiated.

OpenBoot Firmware Not Available After Oracle Solaris OS Has Started

The OpenBoot firmware is not available after the Oracle Solaris OS has started because it is removed from memory.

To reach the `ok` prompt from the Oracle Solaris OS, you must halt the domain. You can use the Oracle Solaris OS `halt` command to halt the domain.

Powercycling a Server

Whenever performing any maintenance on a system running Oracle VM Server for SPARC software that requires powercycling the server, you must save your current logical domain configurations to the SP first.

▼ **Save Your Current Domain Configurations to the SP**

● **Use the following command.**

   # **ldm add-config** *config-name*

Do Not Use the `psradm(1M)` Command on Active CPUs in a Power-Managed Domain

Do not attempt to change an active CPU's operational status in a power-managed domain by using the psradm(1M) command.

Result of Oracle Solaris OS Breaks

The behavior described in this section is seen when you do the following:

1. Press the L1-A key sequence when the input device is set to keyboard.
2. Enter the send break command when the virtual console is at the telnet prompt.

After these types of breaks, you receive the following prompt:

```
c)ontinue, s)ync, r)eset, h)alt?
```

Type the letter that represents what you want the system to do after these types of breaks.

Results From Halting or Rebooting the Control Domain

The following table shows the expected behavior of halting or rebooting the control (primary) domain.

TABLE 12–1   Expected Behavior of Halting or Rebooting the Control (primary) Domain

Command	Other Domain Configured?	Behavior
halt	Not Configured	Host powered off and stays off until powered on at the SP.
	Configured	Soft resets and boots up if the variable auto-boot?=true. Soft resets and halts at ok prompt if the variable auto-boot?=false.
reboot	Not Configured	Reboots the host, no power off.
	Configured	Reboots the host, no power off.

TABLE 12–1 Expected Behavior of Halting or Rebooting the Control (primary) Domain   *(Continued)*

Command	Other Domain Configured?	Behavior
shutdown -i 5	Not Configured	Host powered off, stays off until powered on at the SP.
	Configured	Soft resets and reboots.

For information about the consequences of rebooting a control domain that has the root domain role, see "Rebooting the primary Domain" on page 70.

Using Logical Domains With the Service Processor

The section describes information to be aware of in using the Integrated Lights Out Manager (ILOM) service processor (SP) with the Logical Domains Manager. For more information about using the ILOM software, see the documents for your specific platform, such as *Sun SPARC Enterprise T5120 and T5220 Servers Topic Set* for the Sun SPARC Enterprise T5120 and T5220 servers.

An additional option is available to the existing ILOM command.

-> **set /HOST/bootmode config=***config-name*

The config=*config-name* option enables you to set the configuration on the next power on to another configuration, including the factory-default shipping configuration.

You can invoke the command whether the host is powered on or off. It takes effect on the next host reset or power on.

▼ Reset the Domain Configuration to the Default or Another Configuration

● **Reset the logical domain configuration on the next power on to the default shipping configuration by executing this command:**

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

You also can select other configurations that have been created with the Logical Domains Manager using the ldm add-config command and stored on the service processor (SP). The name you specify in the Logical Domains Manager ldm add-config command can be used to select that configuration with the ILOM bootmode command. For example, assume you stored the configuration with the name ldm-config1.

-> **set /HOST/bootmode config=ldm-config1**

Now, you must powercycle the system to load the new configuration.

See the ldm(1M) man page for more information about the ldm add-config command.

Configuring Domain Dependencies

You can use the Logical Domains Manager to establish dependency relationships between domains. A domain that has one or more domains that depend on it is called a *master domain*. A domain that depends on another domain is called a *slave domain*.

Each slave domain can specify up to four master domains by setting the master property. For example, the pine slave domain specifies its four master domains in the following comma-separated list:

```
# ldm add-domain master=apple,lemon,orange,peach pine
```

Each master domain can specify what happens to its slave domains in the event that the master domain fails. For instance, if a master domain fails, it might require its slave domains to panic. If a slave domain has more than one master domain, the first master domain to fail triggers its defined failure policy on all of its slave domains.

Note – If more than one master domain fails simultaneously, only one of the specified failure policies will be enforced on all the affected slave domains. For example, if the failed master domains have failure policies of stop and panic, all slave domains will be either stopped or panicked.

The master domain's failure policy is controlled by setting one of the following values to the failure-policy property:

- ignore ignores any slave domains when the master domain fails.
- panic panics any slave domains when the master domain fails.
- reset resets any slave domains when the master domain fails.
- stop stops any slave domains when the master domain fails.

In this example, the master domains specify their failure policy as follows:

```
# ldm set-domain failure-policy=ignore apple
# ldm set-domain failure-policy=panic lemon
# ldm set-domain failure-policy=reset orange
# ldm set-domain failure-policy=stop peach
```

You can use this mechanism to create explicit dependencies between domains. For example, a guest domain implicitly depends on the service domain to provide its virtual devices. A guest domain's I/O is blocked when the service domain on which it depends is not up and running. By defining a guest domain as a slave of its service domain, you can specify the behavior of the

guest domain when its service domain goes down. When no such dependency is established, a guest domain just waits for its service domain to return to service.

Note – The Logical Domains Manager does not permit you to create domain relationships that create a dependency cycle. For more information, see "Dependency Cycles" on page 189.

For domain dependency XML examples, see Example D–6.

Domain Dependency Examples

The following examples show how to configure domain dependencies.

- The first command creates a master domain called `twizzle`. This command uses `failure-policy=reset` to specify that slave domains reset if the `twizzle` domain fails. The second command modifies a master domain called `primary`. This command uses `failure-policy=panic` to specify that slave domains panic if the `primary` domain fails. The third command creates a slave domain called `chocktaw` that depends on two master domains, `twizzle` and `primary`. The slave domain uses `master=twizzle,primary` to specify its master domains. In the event either the `twizzle` or `primary` domain fails, the `chocktaw` domain will reset or panic. The first master domain to fail is the one responsible for determining the behavior of the slave domains.

  ```
  # ldm add-domain failure-policy=reset twizzle
  # ldm set-domain failure-policy=panic primary
  # ldm add-domain master=twizzle,primary chocktaw
  ```

- This example shows how to use the `ldm set-domain` command to modify the `orange` domain to assign `primary` as the master domain. The second command uses the `ldm set-domain` command to assign `orange` and `primary` as master domains for the `tangerine` domain. The third command lists information about all of these domains.

  ```
  # ldm set-domain master=primary orange
  # ldm set-domain master=orange,primary tangerine
  # ldm list -o domain
  NAME            STATE      FLAGS    UTIL
  primary         active     -n-cv-   0.2%

  SOFTSTATE
  Solaris running

  HOSTID
      0x83d8b31c

  CONTROL
      failure-policy=ignore

  DEPENDENCY
      master=
  ```

```
------------------------------------------------------------------------------
NAME              STATE      FLAGS   UTIL
orange            bound      ------

HOSTID
    0x84fb28ef

CONTROL
    failure-policy=stop

DEPENDENCY
    master=primary

VARIABLES
    test_var=Aloha

------------------------------------------------------------------------------
NAME              STATE      FLAGS   UTIL
tangerine         bound      ------

HOSTID
    0x84f948e9

CONTROL
    failure-policy=ignore

DEPENDENCY
    master=orange,primary

VARIABLES
    test_var=A hui hou
```
- The following shows an example listing with parseable output:

    # **ldm list -o domain -p**

Dependency Cycles

The Logical Domains Manager does not permit you to create domain relationships that create a dependency cycle. A *dependency cycle* is a relationship between two or more domains that lead to a situation where a slave domain depends on itself, or a master domain depends on one of its slave domains.

The Logical Domains Manager determines whether a dependency cycle exists before adding a dependency. The Logical Domains Manager starts at the slave domain and searches along all paths that are specified by the master array until the end of the path is reached. Any dependency cycles found along the way are reported as errors.

The following example shows how a dependency cycle might be created. The first command creates a slave domain called mohawk that specifies its master domain as primary. So, mohawk depends on primary in the following dependency chain:

FIGURE 12–1   Single Domain Dependency

The second command creates a slave domain called primary that specifies its master domain as counter. So, mohawk depends on primary, which depends on counter in the following dependency chain:

FIGURE 12–2   Multiple Domain Dependency
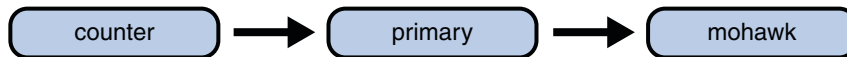
The third command attempts to create a dependency between the counter and mohawk domains, which would produce the following dependency cycle:

FIGURE 12–3   Domain Dependency Cycle
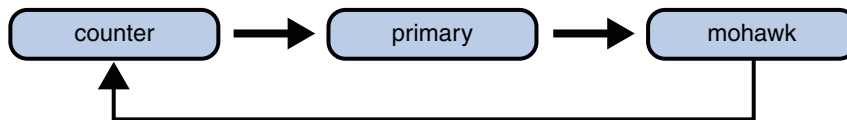
The ldm set-domain command will fail with the following error message:

```
# ldm add-domain master=primary mohawk
# ldm set-domain master=counter primary
# ldm set-domain master=mohawk counter
Dependency cycle detected: LDom "counter" indicates "primary" as its master
```

Determining Where Errors Occur by Mapping CPU and Memory Addresses

This section describes how you can correlate the information that is reported by the Oracle Solaris Fault Management Architecture (FMA) with the logical domain resources that are marked as being faulty.

FMA reports CPU errors in terms of physical CPU numbers and memory errors in terms of physical memory addresses.

If you want to determine within which logical domain an error occurred and the corresponding virtual CPU number or real memory address within the domain, then you must perform a mapping.

CPU Mapping

The domain and the virtual CPU number within the domain, which correspond to a given physical CPU number, can be determined with the following procedures.

▼ Determine the CPU Number

1 Generate a long parseable list for all domains.

```
primary# ldm list -l -p
```

2 Look for the entry in the list's VCPU sections that has a pid field equal to the physical CPU number.

- **If you find such an entry, the CPU is in the domain the entry is listed under, and the virtual CPU number within the domain is given by the entry's vid field.**

- **If you do not find such an entry, the CPU is not in any domain.**

Memory Mapping

The domain and the real memory address within the domain, which correspond to a given physical memory address (PA), can be determined as follows.

▼ Determine the Real Memory Address

1 Generate a long parseable list for all domains.

```
primary# ldm list -l -p
```

2 Look for the line in the list's MEMORY sections where the PA falls within the inclusive range pa to $(pa + size$ - **1); that is, $pa <= $ PA $ < (pa + size$ - **1**).**

Here pa and $size$ refer to the values in the corresponding fields of the line.

- **If you find such an entry, the PA is in the domain the entry is listed under and the corresponding real address within the domain is given by $ra + $ (PA - pa).**

- **If you do not find such an entry, the PA is not in any domain.**

Examples of CPU and Memory Mapping

Suppose you have a logical domain configuration as shown in Example 12–1, and you want to determine the domain and the virtual CPU corresponding to physical CPU number 5, and the domain and the real address corresponding to physical address 0x7e816000.

Looking through the VCPU entries in the list for the one with the pid field equal to 5, you can find the following entry under logical domain ldg1.

```
|vid=1|pid=5|util=29|strand=100
```

Hence, the physical CPU number 5 is in domain ldg1 and within the domain it has virtual CPU number 1.

Looking through the MEMORY entries in the list, you can find the following entry under domain ldg2.

```
ra=0x8000000|pa=0x78000000|size=1073741824
```

Where 0x78000000 <= 0x7e816000 <= (0x78000000 + 1073741824 - 1); that is, $pa <= PA <= (pa + size - 1)$. Hence, the PA is in domain ldg2 and the corresponding real address is 0x8000000 + (0x7e816000 - 0x78000000) = 0xe816000.

EXAMPLE 12–1 Long Parseable List of Logical Domains Configurations

```
primary# ldm list -l -p
VERSION 1.0
DOMAIN|name=primary|state=active|flags=normal,control,vio-service|cons=SP|ncpu=4|mem=1073741824|util=0.6|
uptime=64801|softstate=Solaris running
VCPU
|vid=0|pid=0|util=0.9|strand=100
|vid=1|pid=1|util=0.5|strand=100
|vid=2|pid=2|util=0.6|strand=100
|vid=3|pid=3|util=0.6|strand=100
MEMORY
|ra=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
...
```

Using Universally Unique Identifiers

Starting with the Oracle VM Server for SPARC 2.0 release, each domain is assigned a universally unique identifier (UUID). The UUID is assigned when a domain is created. For legacy domains, the UUID is assigned when the ldmd daemon initializes.

Note – The UUID is lost if you use the ldm migrate-domain -f command to migrate a domain to a target machine that runs an older version of the Logical Domains Manager. When you migrate a domain from a source machine that runs an older version of the Logical Domains Manager, the domain is assigned a new UUID as part of the migration. Otherwise, the UUID is migrated.

You can obtain the UUID for a domain by running the ldm list -l, ldm list-bindings, or ldm list -o domain command. The following examples show the UUID for the ldg1 domain:

```
primary# ldm create ldg1
primary# ldm ls -l ldg1
NAME            STATE     FLAGS   CONS   VCPU  MEMORY   UTIL  UPTIME
ldg1            inactive  ------

UUID
    6c908858-12ef-e520-9eb3-f1cd3dbc3a59

primary# ldm ls -l -p ldg1
VERSION 1.4
DOMAIN|name=ldg1|state=inactive|flags=|cons=|ncpu=|mem=|util=|uptime=
UUID|uuid=6c908858-12ef-e520-9eb3-f1cd3dbc3a59
```

Virtual Domain Information Command and API

The virtinfo command enables you to gather information about a running virtual domain. You can also use the Virtual Domain Information API to create programs to gather information related to virtual domains.

The following list shows some of the information that you can gather about a virtual domain by using the command or API:

- Domain type (implementation, control, guest, I/O, service, root)
- Domain name determined by the Virtual Domain Manager
- Universally unique identifier (UUID) of the domain
- Network node name of the domain's control domain
- Chassis serial number on which the domain is running

For information about the virtinfo command, see the virtinfo(1M) man page. For information about the API, see the libv12n(3LIB) and v12n(3EXT) man pages.

A

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

This appendix covers the following topics:

Oracle VM Server for SPARC P2V Tool Overview

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

- Any `sun4u` SPARC based system that runs at least the Solaris 8 OS
- Any `sun4v` system that runs the Oracle Solaris 10 OS, but does not run in a logical domain

The conversion from a physical system to a virtual system is performed in the following phases:

- **Collection phase.** Runs on the physical source system. In the `collect` phase, a file system image of the source system is created based on the configuration information that it collects about the source system.

- **Preparation phase.** Runs on the control domain of the target system. In the `prepare` phase, a logical domain is created on the target system based on the configuration information collected in the `collect` phase. The file system image is restored to one or more virtual disks. You can use the P2V tool to create virtual disks on plain files or ZFS volumes. You can also create virtual disks on physical disks or LUNs, or on volume manager volumes that you created. The image is modified to enable it to run as a logical domain.

- **Conversion phase.** Runs on the control domain of the target system. In the `convert` phase, the created logical domain is converted into a logical domain that runs the Oracle Solaris 10 OS by using the standard Solaris upgrade process.

For information about the P2V tool, see the ldmp2v(1M) man page.

The following sections describe how the conversion from a physical system to a virtual system is performed in phases.

Collection Phase

The Collection phase runs on the system to be converted. To create a consistent file system image, ensure that the system is as quiet as possible and that all applications are stopped. The ldmp2v command creates a backup of all mounted UFS file systems, so ensure that any file systems to be moved to a logical domain are mounted. You can exclude mounted file systems that you do not want to move, such as file systems on SAN storage or file systems that will be moved by other means. Use the -x option to exclude such file systems. File systems that are excluded by the -x option are not re-created on the guest domain. You can use the -O option to exclude files and directories.

No changes are required on the source system. The only requirement is the ldmp2v script that was installed on the control domain. Ensure that the flarcreate utility is present on the source system.

Preparation Phase

The preparation phase uses the data collected during the collection phase to create a logical domain that is comparable to the source system.

You can use the ldmp2v prepare command in one of the following ways:

- **Automatic mode.** This mode automatically creates virtual disks and restores file system data.

    - Creates the logical domain and the required virtual disks of the same size as on the source system.

    - Partitions the disks and restores the file systems.

        If the combined size of the /, /usr, and /var file systems is less than 10 Gbytes, the sizes of these file systems are automatically adjusted to allow for the larger disk space requirements of the Oracle Solaris 10 OS. Automatic resize can be disabled by using the -x no-auto-adjust-fs option or by using the -m option to manually resize a file system.

    - Modifies the OS image of the logical domain to replace all references to physical hardware with versions that are appropriate for a logical domain. This enables you to upgrade the system to the Oracle Solaris 10 OS by using the normal Solaris upgrade process. Modifications include updating the /etc/vfstab file to account for new disk names. Any Solaris Volume Manager or Veritas Volume Manager (VxVM) encapsulated boot disks are automatically unencapsulated during this process. When a

disk is unencapsulated, it is converted into plain disk slices. If VxVM is installed on the source system, the P2V process disables VxVM on the created guest domain.

- **Non-automatic mode.** You must create the virtual disks and restore the file system data manually. This mode enables you to change the size and number of disks, the partitioning, and the file system layout. The preparation phase in this mode only runs the logical domain creation and the OS image modification steps on the file system.

- **Cleanup mode.** Removes a logical domain and all of the underlying back-end devices that are created by ldmp2v.

Conversion Phase

In the conversion phase, the logical domain uses the Solaris upgrade process to upgrade to the Oracle Solaris 10 OS. The upgrade operation removes all existing packages and installs the Oracle Solaris 10 sun4v packages, which automatically performs a sun4u-to-sun4v conversion. The convert phase can use an Oracle Solaris DVD iso image or a network install image. You can also use Custom JumpStart to perform a fully automated hands-off upgrade operation.

Back-End Devices

You can create virtual disks for a guest domain on a number of back-end types: files (file), ZFS volumes (zvol), physical disks or LUNs (disk), or volume manager volumes (disk). The ldmp2v command automatically creates files or ZFS volumes of the appropriate size if you specify file or zvol as the back-end type in one of the following ways:

- By using the -b option
- By specifying the value of the BACKEND_TYPE parameter in the /etc/ldmp2v.conf file

The disk back-end type enables you to use a physical disk, LUN, or volume manager volume (Solaris Volume Manager and Veritas Volume Manager (VxVM)) as a back-end device for virtual disks. You must create the disk or volume with an appropriate size prior to beginning the prepare phase. For a physical disk or LUN, specify the back-end device as slice 2 of the block or character device of the disk, such as /dev/dsk/c0t3d0s2. For a volume manager volume, specify the block or character device for the volume, such as /dev/md/dsk/d100 for Solaris Volume Manager or /dev/vx/dsk/ldomdg/vol1 for VxVM.

Unless you specify the volume and virtual disk names with the -B *backend*:*volume*:*vdisk* option, the volumes and virtual disks that you create for the guest are given default names.

- *backend* specifies the name of the back end to use. You must specify *backend* for the disk back-end type. *backend* is optional for the file and zvol back-end types, and can be used to set a non-default name for the file or ZFS volume that ldmp2v creates. The default name is $BACKEND_PREFIX/*guest-name*/disk*N*.

- *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*-vol*N*.

- *vdisk* is optional for all back-end types and specifies the name of the volume in the guest domain. If not specified, *vdisk* is disk*N*.

Note – During the conversion process, the virtual disk is temporarily named *guest-name*-disk*N* 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 installed and configured *only* on the control domain of the target system. You do not need to install the package on the source system. Instead, you can simply copy the /usr/sbin/ldmp2v script from the target system to the source system.

Prerequisites

Before you can run the Oracle VM Server for SPARC P2V Tool, ensure that the following conditions are met:

- The following Flash utility patches are installed on the source system:
    - **For the Solaris 8 OS:** At least patch ID 109318-34
    - **For the Solaris 9 OS:** At least patch ID 113343-06
- Target system runs at least Logical Domains 1.1 on the following:
    - 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
- 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

Version 2.0 of the Oracle VM Server for SPARC P2V Tool has the following limitations:

- Only UFS file systems are supported.
- Only plain disks (/dev/dsk/c0t0d0s0), Solaris Volume Manager metadevices (/dev/md/dsk/d*NNN*), and VxVM encapsulated boot disks are supported on the source system.
- During the P2V process, each guest domain can have only a single virtual switch and virtual disk server. You can add more virtual switches and virtual disk servers to the domain after the P2V conversion.
- Support for VxVM volumes is limited to the following volumes on an encapsulated boot disk: rootvol, swapvol, usr, var, opt, and home. The original slices for these volumes must still be present on the boot disk. The P2V tool supports Veritas Volume Manager 5.x on the Solaris 10 OS. However, you can also use the P2V tool to convert Solaris 8 and Solaris 9 operating systems that use VxVM.
- You cannot convert Solaris 10 systems that are configured with zones.

▼ Install the Oracle VM Server for SPARC P2V Tool

1 **Go to the Oracle VM Server for SPARC download page at `http://www.sun.com/servers/coolthreads/ldoms/get.jsp`.**

2 **Download the P2V software package, `SUNWldmp2v`.**

Starting with the Logical Domains 1.2 release, the SUNWldmp2v package is included in the Oracle VM Server for SPARC zip file.

3 **Become superuser or assume an equivalent role.**

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

4 **Use the `pkgadd` command to install the `SUNWldmp2v` package.**

```
# pkgadd -d . SUNWldmp2v
```

5   **Create the `/etc/ldmp2v.conf` file and configure the following default properties:**

- VDS – Name of the virtual disk service, such as VDS="primary-vds0"

- VSW – Name of the virtual switch, such as VSW="primary-vsw0"

- VCC – Name of the virtual console concentrator, such as VCC="primary-vcc0"

- `BACKEND_TYPE` – Back-end type of `zvol`, `file`, or `disk`

- `BACKEND_SPARSE` – Whether to create back-end devices as sparse volumes or files BACKEND_SPARSE="yes", or non-sparse volumes or files BACKEND_SPARSE="no"

- `BACKEND_PREFIX` – Location to create virtual disk back-end devices

  When `BACKEND_TYPE="zvol"`, specify the `BACKEND_PREFIX` value as a ZFS dataset name. When `BACKEND_TYPE="files"`, the `BACKEND_PREFIX` value is interpreted as a path name of a directory that is relative to /.

  For example, BACKEND_PREFIX="tank/ldoms" would result in having ZVOLs created in the tank/ldoms/*domain-name* dataset, and files created in the /tank/ldoms/*domain-name* subdirectory.

  The `BACKEND_PREFIX` property is not applicable to the `disk` back end.

- `BOOT_TIMEOUT` – Timeout for Oracle Solaris OS boot in seconds

For more information, see the ldmp2v.conf.sample configuration file that is part of the downloadable bundle.

Using the `ldmp2v` Command

This section includes examples for the three phases.

EXAMPLE A–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
```

EXAMPLE A–1 Collection Phase Examples *(Continued)*

The following shows how to run the collection tool when the source and target systems share an NFS-mounted file system:

```
volumia# ldmp2v collect -d home/dana/volumia
Collecting system configuration ...
Archiving file systems ...
Determining which filesystems will be included in the archive...
Creating the archive...
895080 blocks
Archive creation complete.
```

- **Not sharing an NFS-mounted file system.** When the source and target systems do not share an NFS-mounted file system, the file system image can be written to local storage and later copied to the control domain. The Flash utility automatically excludes the archive that it creates.

```
volumia# ldmp2v collect -d /var/tmp/volumia
Collecting system configuration ...
Archiving file systems ...
Determining which filesystems will be included in the archive...
Creating the archive...
895080 blocks
Archive creation complete.
```

Copy the Flash archive and the `manifest` file from the /var/tmp/volumia directory to the target system.

- **Skip file-system backup step.** If backups of the system are already available using a third-party backup tool such as NetBackup, you can skip the file system backup step by using the none archiving method. When you use this option, only the system configuration manifest is created.

```
volumia# ldmp2v collect -d /home/dana/p2v/volumia -a none
Collecting system configuration ...
The following file system(s) must be archived manually: / /u01 /var
```

Note that if the directory specified by -d is not shared by the source and target systems, you must copy the contents of that directory to the control domain. The directory contents must be copied to the control domain prior to the preparation phase.

EXAMPLE A–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 ...
```

EXAMPLE A–2 Preparation Phase Examples   *(Continued)*

```
Unmounting guest file systems ...
Creating domain volumia ...
Attaching vdisks to domain volumia ...
```

- The following command shows information about the volumia logical domain:

```
# ldm list -l volumia
NAME              STATE      FLAGS  CONS   VCPU  MEMORY   UTIL  UPTIME
volumia           inactive   ------         2    4G

NETWORK
    NAME     SERVICE                 DEVICE    MAC                     MODE   PVID VID
    vnet0    primary-vsw0                      00:03:ba:1d:7a:5a              1

DISK
    NAME    DEVICE   TOUT MPGROUP       VOLUME                              SERVER
    disk0                                       volumia-vol0@primary-vds0
    disk1                                       volumia-vol1@primary-vds0
```

- The following shows that you can completely remove a domain and its back-end devices by using the -C option:

```
# ldmp2v prepare -C volumia
Cleaning up domain volumia ...
Removing vdisk disk0 ...
Removing vdisk disk1 ...
Removing domain volumia ...
Removing volume volumia-vol0@primary-vds0 ...
Removing ZFS volume tank/ldoms/volumia/disk0 ...
Removing volume volumia-vol1@primary-vds0 ...
Removing ZFS volume tank/ldoms/volumia/disk1 ...
```

- The following shows that you can resize one or more file systems during P2V by specifying the mount point and the new size with the -m option:

```
# ldmp2v prepare -d /home/dana/p2v/normaal -m /:8g normaal
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 normaal ...
Attaching vdisks to domain normaal ...
```

EXAMPLE A–3 Conversion Phase Examples

The following examples show how you might use the ldmp2v convert command.

- **Using a network installation server.** The ldmp2v convert command boots the domain over the network by using the specified virtual network interface. You must run the setup_install_server and add_install_client scripts on the installation server.

  You can use the Custom JumpStart feature to perform a completely hands-off conversion. This feature requires that you create and configure the appropriate sysidcfg and profile files for the client on the JumpStart server. The profile should consist of the following lines:

  ```
  install_type    upgrade
  root_device     c0d0s0
  ```

  The sysidcfg file is only used for the upgrade operation, so a configuration such as the following should be sufficient:

  ```
  name_service=NONE
  root_password=uQkoXlMLCsZhI
  system_locale=C
  timeserver=localhost
  timezone=Europe/Amsterdam
  terminal=vt100
  security_policy=NONE
  nfs4_domain=dynamic
  network_interface=PRIMARY {netmask=255.255.255.192
          default_route=none protocol_ipv6=no}
  ```

  For more information about using Custom JumpStart, see *Oracle Solaris 10 9/10 Installation Guide: Custom JumpStart and Advanced Installations*.

  ```
  # 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...
  ```

EXAMPLE A–3   Conversion Phase Examples   *(Continued)*

```
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.

Note – The answers to the sysid questions are *only* used for the duration of the upgrade process. This data is not applied to the existing OS image on disk. The fastest and simplest way to run the conversion is to select Non-networked. The root password that you specify does not need to match the root password of the source system. The system's original identity is preserved by the upgrade and takes effect after the post-upgrade reboot. The time required to perform the upgrade depends on the Oracle Solaris Cluster that is installed on the original system.

```
# ldmp2v convert -i /tank/iso/s10s_u5.iso -d /home/dana/p2v/volumia volumia
Testing original system status ...
```

EXAMPLE A–3   Conversion Phase Examples   *(Continued)*

```
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.

  -----------------------------------------------------------------------------
```

EXAMPLE A–3 Conversion Phase Examples   *(Continued)*

```
F2_Upgrade    F3_Go Back    F4_Initial    F5_Exit    F6_Help
```

B

Oracle VM Server for SPARC Configuration Assistant

The Oracle VM Server for SPARC Configuration Assistant leads you through the configuration of a logical domain by setting basic properties. It runs on chip multithreading (CMT)-based systems.

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

The Configuration Assistant is available as both a graphical user interface (GUI) and terminal-based tool, ldmconfig.

For information about the terminal-based tool, see "Using the Configuration Assistant (ldmconfig)" on page 208 and the ldmconfig(1M) man page.

For information about starting the GUI tool, see "Using the Configuration Assistant (GUI)" on page 207.

Using the Configuration Assistant (GUI)

The Configuration Assistant GUI is delivered as part of the Oracle VM Server for SPARC zip bundle.

Ensure that the target system is running at least the Logical Domains 1.2 software and that your system is running at least Version 1.6 of the Java SE Runtime Environment.

To run the Configuration Assistant GUI from the command line, type the following:

```
$ java -jar "Configurator.jar"
```

This GUI tool includes on-screen documentation to help you create the configuration for your system.

Using the Configuration Assistant (`ldmconfig`)

The terminal-based Configuration Assistant, `ldmconfig`, works through a series of operations that correspond to user interface screens. The end result is the creation of a configuration that you can deploy to a logical domain.

The following sections describe how to install the `ldmconfig` command and some features of the Configuration Assistant tool.

Installing the Configuration Assistant

The Configuration Assistant is delivered as part of the `SUNWldm` package.

After you install the `SUNWldm` package, you can find the `ldmconfig` command in the `/usr/sbin` directory. The command is also installed in the `/opt/SUNWldm/bin` directory for legacy purposes.

Prerequisites

Before you install and run the Configuration Assistant, ensure that the following conditions are met:

- The target system must be running at least the Logical Domains 1.2 software.
- Your terminal window must be at least 80 characters wide by 24 lines long.

Limitations and Known Issues

The Configuration Assistant has the following limitations:

- Resizing the terminal while using `ldmconfig` might cause garbled output
- Support for UFS disk files as virtual disks only
- Only works with systems where no existing logical domains configurations are present
- Virtual console concentrator ports are from `5000` to `5100`
- Default names that are used for guest domains, services, and devices cannot be changed

`ldmconfig` Features

The terminal-based Configuration Assistant, `ldmconfig`, works through a series of operations that correspond to user interface screens. You can navigate backward (previous) and forward (next) through these screens until you reach the final step. The final step produces the configuration. At any time you can quit the Configuration Assistant or reset the configuration to use the defaults. From the final screen, you can deploy the configuration to a logical domain.

First, the Configuration Assistant automatically inspects the system to determine the most suitable default property values based on best practices, and then shows those properties that are required to control a deployment. Note that this is not an exhaustive list. You can set other properties to further customize the configuration.

For information about the using the ldmconfig tool, see the ldmconfig(1M) man page.

You can adjust the following properties:

- **Number of guest domains.** Specify the number of guest domains for the application to create. The minimum is one guest domain. The maximum value is determined by the availability of VCPU resources. For example, you could create up to 60 guest domains with a single thread each on a 64-thread CMT system, and four threads reserved for the control domain. If best practices are selected, the minimum number of VCPU resources per guest domain is a single core. So, on an 8-core, 8-thread-per-core system with best practices selected, you could create up to seven guest domains with one core each. Also, one core is assigned to the control domain.

  The Configuration Assistant shows the maximum number of domains that can be configured for this system.

  The Configuration Assistant performs the following tasks to create domains:

  - **For all domains.**
    - Creates a virtual terminal service on ports from 5000 to 5100
    - Creates a virtual disk service
    - Creates a virtual network switch on the network adapter nominated
    - Enables the virtual terminal server daemon

  - **For each domain.**
    - Creates the logical domain
    - Configures VCPUs assigned to the domain
    - Configures memory assigned to the domain
    - Creates a UFS disk file to use as a virtual disk
    - Creates a virtual disk server device (`vdsdev`) for the disk file
    - Assigns the disk file as virtual disk `vdisk0` for the domain
    - Adds a virtual network adapter attached to the virtual switch on the network adapter nominated
    - Sets the OBP property `auto-boot?=true`
    - Sets the OBP property `boot-device=vdisk0`
    - Binds the domain
    - Starts the domain

- **Default network.** Specify the network adapter that the new domains will use for virtual networking. The adapter must be present in the system. The Configuration Assistant highlights those adapters that are currently in use by the system as default adapters, and those that have active link status (cabled adapters).

- **Virtual disk size.** Create virtual disks for each of the new domains. These virtual disks are created based on the disk files that are located in the local file systems. This property controls the size of each virtual disk in Gbytes. The minimum size, 8 Gbytes, is based on the approximate size required to contain a Oracle Solaris 10 OS, and the maximum size is 100 Gbytes.

  If the Configuration Assistant cannot find file systems that have adequate space to contain the disk files for all domains, an error screen is shown. In this case, you might need to do the following before rerunning the application:

  - Reduce the size of the virtual disks
  - Reduce the number of domains
  - Add more higher-capacity file systems

- **Virtual disk directory.** Specify a file system that has sufficient capacity on which to store the files to be created as virtual disks for the new domains. The directory is based on the number of domains that are selected and the size of the virtual disks. The value must be recalculated and destination directories selected any time that these property values are changed. The Configuration Assistant presents you with a list of file systems that have sufficient space. After you specify the file system name, the Configuration Assistant creates a directory in this file system called `/ldoms/disks` in which to create the disk images.

- **Best practice.** Specify whether to use best practice for property values.

  - When the value is `yes`, the Configuration Assistant uses best practice for several configuration property values. It forces the minimum of one core per domain, including the system domains. As a result, this limits the maximum number of guest domains to the total number of cores present in the system minus one core for the system domains. For example, in the case of a two-socket SPARC Enterprise T5140 with eight cores each, the maximum number of guest domains is 15 plus the system domain.

  - When the value is `no`, the Configuration Assistant permits the creation of domains that have a minimum of one thread, but maintain at least four threads for the system domain.

Next, the Configuration Assistant summarizes the deployment configuration to be created, which includes the following information:

- Number of domains
- CPU assigned to each guest domain
- Memory assigned to each guest domain
- Size and location of the virtual disks
- Network adapter to be used for virtual network services for guest domains
- Amount of CPU and memory to be used by the system for services

■ If a valid Oracle Solaris OS DVD was identified, it will be used to create a shared virtual CD-ROM device to permit guest domains to install the Oracle Solaris OS

Finally, the Configuration Assistant configures the system to create the specified Logical Domains deployment. It also describes the actions to be taken and shows the commands to be run to configure the system. This information can assist you in learning how to use the ldm commands that are needed to configure the system.

> **Caution –** Do *not* interact with this configuration step and do *not* interrupt this process as it might result in a partially configured system.

After the commands have been completed successfully, reboot the system for the changes to take effect.

C

Logical Domains Manager Discovery

Logical Domains Managers can be discovered on a subnet by using multicast messages. The ldmd daemon is able to listen on a network for a specific multicast packet. If that multicast message is of a certain type, ldmd replies to the caller. This enables ldmd to be discovered on systems that are running Oracle VM Server for SPARC.

This appendix provides information about discovering the Logical Domains Manager running on systems on a subnet.

Discovering Systems Running the Logical Domains Manager

Multicast Communication

This discovery mechanism uses the same multicast network that is used by the ldmd daemon to detect collisions when automatically assigning MAC addresses. To configure the multicast socket, you must supply the following information:

```
#define    MAC_MULTI_PORT        64535
#define    MAC_MULTI_GROUP        "239.129.9.27"
```

By default, *only* multicast packets can be sent on the subnet to which the machine is attached. You can change the behavior by setting the ldmd/hops SMF property for the ldmd daemon.

Message Format

The discovery messages must be clearly marked so as not to be confused with other messages. The following multicast message format ensures that discovery messages can be distinguished by the discovery listening process:

```
#include <netdb.h> /* Used for MAXHOSTNAMELEN definition */
#define    MAC_MULTI_MAGIC_NO    92792004
#define    MAC_MULTI_VERSION    1

enum {
    SEND_MSG = 0,
    RESPONSE_MSG,
    LDMD_DISC_SEND,
    LDMD_DISC_RESP,
};

typedef struct {
    uint32_t    version_no;
    uint32_t    magic_no;
    uint32_t    msg_type;
    uint32_t    resv;
    union {
        mac_lookup_t        Mac_lookup;
        ldmd_discovery_t    Ldmd_discovery;
    } payload;
#define    lookup        payload.Mac_lookup
#define    discovery    payload.Ldmd_discovery
} multicast_msg_t;

#define    LDMD_VERSION_LEN    32

typedef struct {
    uint64_t mac_addr;
    char        source_ip[INET_ADDRSTRLEN];
} mac_lookup_t;

typedef struct {
    char        ldmd_version[LDMD_VERSION_LEN];
    char        hostname[MAXHOSTNAMELEN];
    struct in_addr    ip_address;
    int        port_no;
} ldmd_discovery_t;
```

▼ Discover Logical Domains Managers Running on Your Subnet

1   Open a multicast socket.

Ensure that you use the port and group information specified in "Multicast Communication" on page 213.

2   Send a `multicast_msg_t` message over the socket.

The message should include the following:

- Valid value for version_no, which is 1 as defined by MAC_MULTI_VERSION
- Valid value for magic_no, which is 92792004 as defined by MAC_MULTI_MAGIC_NO
- msg_type of LDMD_DISC_SEND

3   Listen on the multicast socket for responses from Logical Domains Managers.

The responses must be a `multicast_msg_t` message with the following:

- Valid value for `version_no`
- Valid value for `magic_no`
- `msg_type` set to `LDMD_DISC_RESP`
- Payload consisting of a `ldmd_discovery_t` structure, which contains the following information:
  - `ldmd_version` – Version of the Logical Domains Manager running on the system
  - `hostname` – Host name of the system
  - `ip_address` – IP address of the system
  - `port_no` – Port number being used by the Logical Domains Manager for communications, which should be XMPP port 6482

When listening for a response from Logical Domains Managers, ensure that any auto-allocation MAC collision-detection packets are discarded.

D

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 217
- "XML Protocol" on page 218
- "Event Messages" on page 223
- "Logical Domains Manager Actions" on page 227
- "Logical Domains Manager Resources and Properties" on page 228

For various schemas to use with the Logical Domains Manager, see Appendix E, "Logical Domains Manager XML Schemas."

XML Transport

External programs can use the Extensible Messaging and Presence Protocol (XMPP – RFC 3920) to communicate with the Logical Domains Manager. XMPP is supported for both local and remote connections and is on by default. To shut off a remote connection, set the ldmd/xmpp_enabled SMF property to false and restart the Logical Domains Manager.

```
# svccfg -s ldom/ldmd setprop ldmd/xmpp_enabled=false
# svcadm refresh ldmd
# svcadm restart ldmd
```

Note – Disabling the XMPP server also prevents domain migration and the dynamic reconfiguration of memory.

XMPP Server

The Logical Domains Manager implements an XMPP server which can communicate with numerous available XMPP client applications and libraries. The Logical Domains Manager uses the following security mechanisms:

- Transport Layer Security (TLS) to secure the communication channel between the client and itself.

- Simple Authentication and Security Layer (SASL) for authentication. PLAIN is the only SASL mechanism supported. You must send in a user name and password to the server, so it can authorize you before allowing monitoring or management operations.

Local Connections

The Logical Domains Manager detects whether user clients are running on the same domain as itself and, if so, does a minimal XMPP handshake with that client. Specifically, the SASL authentication step after the setup of a secure channel through TLS is skipped. Authentication and authorization are done based on the credentials of the process implementing the client interface.

Clients can choose to implement a full XMPP client or to simply run a streaming XML parser, such as the libxml2 Simple API for XML (SAX) parser. Either way the client has to handle an XMPP handshake to the point of TLS negotiation. Refer to the XMPP specification for the sequence needed.

XML Protocol

After communication initialization is complete, Logical Domains-defined XML messages are sent next. There are two general types of XML messages:

- Request and response messages use the <LDM_interface> tag. This type of XML message is used for communicating commands and getting results back from the Logical Domains Manager, analogous to executing commands using the command-line interface (CLI). This tag is also used for event registration and unregistration.

- Event messages use the <LDM_event> tag. This type of XML message is used to asynchronously report events posted by the Logical Domains Manager.

Request and Response Messages

The XML interface into Logical Domains has two different formats:

- One format for sending commands into the Logical Domains Manager
- Another format for Logical Domains Manager to respond on the status of the incoming message and the actions requested within that message.

The two formats share many common XML structures, but are separated in this discussion for a better understanding of the differences between them. This document also contains an XML Schema which details the combined incoming and outgoing XML (See "LDM_Event XML Schema" on page 243).

Request Messages

An incoming XML request to the Logical Domains Manager at its most basic level includes a description of a single command, operating on a single object. More complicated requests can handle multiple commands and multiple objects per command. Following is the structure of a basic XML command.

EXAMPLE D–1 Format of a Single Command Operating on a Single Object

```
<LDM_interface version="1.0">
  <cmd>
    <action>Place command here</action>
    <option>Place options for certain commands here</option>
    <data version="3.0">
      <Envelope>
        <References/>
        <!-- Note a <Section> section can be here instead of <Content> -->
        <Content xsi:type="ovf:VirtualSystem_Type" id="Domain name">
          <Section xsi:type="ovf:ResourceAllocationSection_type">
            <Item>
              <rasd:OtherResourceType>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>
```

The <LDM_interface> Tag

All commands sent to the Logical Domains Manager must start with the <LDM_interface> tag. Any document sent into the Logical Domains Manager must have only one <LDM_interface> tag contained within it. The <LDM_interface> tag must include a version attribute as shown in Example D–1.

The <cmd> Tag

Within the <LDM_interface> tag, the document must include at least one <cmd> tag. Each <cmd> section must have only one <action> tag. Use the <action> tag to describe the command to run. Each <cmd> tag must include at least one <data> tag to describe the objects on which the command is to operate.

The <cmd> tag can also have an <option> tag, which is used for options and flags that are associated with some commands. The following commands use options:

- The remove-domain command can use the -a option.
- The stop-domain command can use the -f option.
- The cancel-operation command can use the migration or reconf option.
- The add-spconfig command can use the -r *autosave-name* option.
- The remove-spconfig command can use the -r option.
- The list-spconfig command can use the -r [*autosave-name*] option.

The <data> Tag

Each <data> section contains a description of an object pertinent to the command specified. The format of the data section is based on the XML schema portion of the Open Virtualization Format (OVF) draft specification. That schema defines an <Envelope> section which contains a <References> tag (unused by Logical Domains) and <Content> and <Section> sections.

For Logical Domains, the <Content> section is used to identify and describe a particular domain. The domain name in the id= attribute of the <Content> node identifies the domain. Within the <Content> section are one or more <Section> sections describing resources of the domain as needed by the particular command.

If you only need to identify a domain name, then you do not need to use any <Section> tags. Conversely, if no domain identifier is needed for the command, then you do need to provide a <Section> section, describing the resources needed for the command, outside of a <Content> section, but still within the <Envelope> section.

A <data> section does not need to contain an <Envelope> tag in cases where the object information can be inferred. This situation mainly applies to requests for monitoring all objects applicable to an action, and event registration and unregistration requests.

To allow use of the OVF specification's schema to properly define all types of objects, two additional OVF types have been defined:

- <gprop:GenericProperty> tag (See "The GenericProperty XML Schema" on page 260.)
- <Binding> tag (See "Binding_Type XML Schema" on page 260.)

The <gprop:GenericProperty> tag was defined to handle any object's property for which the OVF specification does not have a definition. The property name is defined in the key= attribute of the node and the value of the property is the contents of the node. The <binding> tag is used in the list-bindings subcommand output to define resources that are bound to other resources.

Response Messages

An outgoing XML response closely matches the structure of the incoming request in terms of the commands and objects included, with the addition of a <Response> section for each object and command specified, as well as an overall <Response> section for the request. The <Response> sections provide status and message information as described in Example D–2. Following is the structure of a response to a basic XML request.

EXAMPLE D–2   Format of a Response to a Single Command Operating on a Single Object

```
<LDM_interface version="1.0">
  <cmd>
    <action>Place command here</action>
    <data version="3.0">
      <Envelope>
        <References/>
        <!-- Note a <Section> section can be here instead of <Content> -->
        <Content xsi:type="ovf:VirtualSystem_Type" id="Domain name">
          <Section xsi:type="ovf:ResourceAllocationSection_type">
            <Item>
              <rasd:OtherResourceType>
                LDom Resource Type
              </rasd:OtherResourceType>
              <gprop:GenericProperty
              key="Property name">
                Property Value
              </gprop:GenericProperty>
            </Item>
          </Section>
          <!-- Note: More <Section> sections can be placed here -->
        </Content>
      </Envelope>
      <response>
        <status>success or failure</status>
        <resp_msg>Reason for failure</resp_msg>
      </response>
    </data>
    <!-- Note: More Data sections can be placed here -->
    <response>
      <status>success or failure</status>
      <resp_msg>Reason for failure</resp_msg>
    </response>
```

```
  </cmd>
  <!-- Note: More Command sections can be placed here -->
  <response>
    <status>success or failure</status>
    <resp_msg>Reason for failure</resp_msg>
  </response>
</LDM_interface>
```

Overall Response

This `<response>` section, which is the direct child of the `<LDM_interface>` section, indicates overall success or failure of the entire request. Unless the incoming XML document is malformed, the `<response>` section includes only a `<status>` tag. If this response status indicates success, all commands on all objects have succeeded. If this response status is a failure and there is no `<resp_msg>` tag, then one of the commands included in the original request failed. The <resp_msg> tag is used only to describe some problem with the XML document itself.

Command Response

The `<response>` section under the `<cmd>` section alerts the user to success or failure of that particular command. The `<status>` tag shows if that command succeeds or fails. As with the overall response, if the command fails, the `<response>` section includes only a `<resp_msg>` tag if the contents of the `<cmd>` section of the request is malformed. Otherwise, the failed status means one of the objects the command ran against caused a failure.

Object Response

Finally, each `<data>` section in a `<cmd>` section also has a `<response>` section. This shows if the command being run on this particular object passes or fails. If the status of the response is SUCCESS, there is no `<resp_msg>` tag in the `<response>` section. If the status is FAILURE, there are one or more `<resp_msg>` tags in the `<response>` field, depending on the errors encountered when running the command against that object. Object errors can result from problems found when running the command, or a malformed or unknown object.

In addition to the `<response>` section, the `<data>` section can contain other information. This information is in the same format as an incoming `<data>` field, describing the object that caused a failure. See "The <data> Tag" on page 220. 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 224 for complete details.

Registration and Unregistration

Use an `<LDM_interface>` message to register for events. See "The `<LDM_interface>` Tag" on page 220. The action tag details the type of event for which to register or unregister and the `<data>` section is left empty.

EXAMPLE D–3   Example Event Registration Request Message

```
<LDM_interface version="1.0">
  <cmd>
    <action>reg-domain-events</action>
    <data version="3.0"/>
  </cmd>
</LDM_interface>
```

The Logical Domains Manager responds with an `<LDM_interface>` response message stating whether the registration or unregistration was successful.

EXAMPLE D–4   Example Event Registration Response Message

```
<LDM_interface version="1.0">
  <cmd>
    <action>reg-domain-events</action>
    <data version="3.0"/>
      <response>
        <status>success</status>
      </response>
    </data>
    <response>
      <status>success</status>
    </response>
  </cmd>
  <response>
    <status>success</status>
  </response>
</LDM_interface>
```

The action string for each type of event is listed in the events subsection.

The <LDM_event> Messages

Event messages have the same format as an incoming <LDM_interface> message with the exception that the start tag for the message is <LDM_event>. The action tag of the message is the action that was performed to trigger the event. The data section of the message describes the object associated with the event; the details depend on the type of event that occurred.

EXAMPLE D–5 Example <LDM_event> Notification

```
<LDM_event version='1.0'>
  <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

Following are the event types to which you can subscribe:

- Domain events
- Hardware events
- Progress events
- Resource events

All the events correspond to ldm subcommands.

Domain Events

Domain events describe what actions can be performed directly to a domain. The following table shows the domain events which can be listed in the <action> tag in the <LDM_event> message.

Domain Events	Domain Events	Domain Events
add-domain	remove-domain	bind-domain
unbind-domain	start-domain	stop-domain

Domain Events	Domain Events	Domain Events
domain-reset	panic-domain	migrate-domain

These events always contain *only* a <Content> tag in the OVF data section that describes to which domain the event happened. To register for the domain events, send an <LDM_interface> message with the <action> tag set to **reg-domain-events**. Unregistering for these events requires an <LDM_interface> message with the action tag set to **unreg-domain-events**.

Hardware Events

Hardware events pertain to changing the physical system hardware. In the case of Oracle VM Server for SPARC software, the only hardware changes that can be made are those to the service processor (SP) when a user adds, removes, or sets an SP configuration. Currently, the only three events of this type are:

- add-spconfig
- set-spconfig
- remove-spconfig

The hardware events always contain *only* a <Section> tag in the OVF data section which describes which SP configuration to which the event is happening. To register for these events, send an <LDM_interface> message with the <action> tag set to **reg-hardware-events**. Unregistering for these events requires an <LDM_interface> message with the <action> tag set to **unreg-hardware-events**.

Progress Events

Progress events are issued for long-running commands, such as a domain migration. These events report the amount of progress that has been made during the life of the command. At this time, only the migration-process event is reported.

Progress events always contain *only* a <Section> tag in the OVF data section that describes the SP configuration affected by the event. To register for these events, send an <LDM_interface> message with the <action> tag set to reg-hardware-events. Unregistering for these events requires an <LDM_interface> message with the <action> tag set to unreg-hardware-events.

The <data> section of a progress event consists of a <content> section that describes the affected domain. This <content> section uses an ldom_info <Section> tag to update progress. The following generic properties are shown in the ldom_info section:

- --progress – Percentage of the progress made by the command
- --status – Command status, which can be one of ongoing, failed, or done
- --source – Machine that is reporting the progress

Resource Events

Resource events occur when resources are added, removed, or changed in any domain. The data section for some of these events contains the <Content> tag with a <Section> tag giving a service name in the OVF data section. The following table shows events which can be listed in the <action> tag in the <LDM_event> message.

Resource Events	Resource Events
add-vdiskserverdevice	remove-vdiskserverdevice
set-vdiskserverdevice	remove-vdiskserver
set-vconscon	remove-vconscon
set-vswitch	remove-vswitch
remove-vdpcs	

The remaining resource events always contain *only* the <Content> tag in the OVF data section that describes to which domain the event happened.

Resource Events	Resource Events	Resource Events
add-vcpu	add-crypto	add-memory
add-io	add-variable	add-vconscon
add-vdisk	add-vdiskserver	add-vnet
add-vswitch	add-vdpcs	add-vdpcc
set-vcpu	set-crypto	set-memory
set-variable	set-vnet	set-vconsole
set-vdisk	remove-vcpu	remove-crypto
remove-memory	remove-io	remove-variable
remove-vdisk	remove-vnet	remove-vdpcc

To register for the resource events, send an <LDM_interface> message with the <action> tag set to **reg-resource-events**. Unregistering for these events requires an <LDM_interface> message with the <action> tag set to **unreg-resource-events**.

All Events

You can also register to listen for all three type of events without having to register for each one individually. To register for all three types of events simultaneously, send an <LDM_interface> message with the <action> tag set to **reg-all-events**. Unregistering for these events require an <LDM_interface> message with the <action> tag set to **unreg-all-events**.

Logical Domains Manager Actions

The commands specified in the <action> tag, with the exception of *-*-events commands, correspond to those of the ldm command-line interface. For details about ldm subcommands, see the ldm(1M) man page.

Note – The XML interface does *not* support the verb or command *aliases* supported by the Logical Domains Manager CLI.

The supported strings in the <action> tag are as follows:

Logical Domains Manager Actions	Logical Domains Manager Actions	Logical Domains Manager Actions
list-bindings	list-services	list-constraints
list-devices	add-domain	remove-domain
list-domain	start-domain	stop-domain
bind-domain	unbind-domain	add-io
remove-io	add-mau	set-mau
remove-mau	add-memory	set-memory
remove-memory	remove-reconf	add-spconfig
set-spconfig	remove-spconfig	list-spconfig
add-variable	set-variable	remove-variable
list-variable	add-vconscon	set-vconscon
remove-vconscon	set-vconsole	add-vcpu
set-vcpu	remove-vcpu	add-vdisk
remove-vdisk	add-vdiskserver	remove-vdiskserver
add-vdpcc	remove-vdpcc	add-vdpcs
remove-vdpcs	add-vdiskserverdevice	remove-vdiskserverdevice

Logical Domains Manager Actions	Logical Domains Manager Actions	Logical Domains Manager Actions
add-vnet	set-vnet	remove-vnet
add-vswitch	set-vswitch	remove-vswitch
reg-domain-events	unreg-domain-events	reg-resource-events
unreg-resource-events	reg-hardware-events	unreg-hardware-events
reg-all-events	unreg-all-events	migrate-domain
cancel-operation	set-domain	

Logical Domains Manager Resources and Properties

Following are the Logical Domains Manager resources and the properties that can be defined for each of those resources. The resources and properties are shown in **bold** type in the XML examples. These examples show resources, not binding output. The constraint output can be used to create input for the Logical Domains Manager actions. The exception to this is domain migration output. See "Domain Migration" on page 238. Each resource is defined in a <Section> OVF section and is specified by a <rasd:OtherResourceType> tag.

Domain Information (`ldom_info`) Resource

EXAMPLE D–6   Example ldom_info XML Output

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="primary">
    <Section xsi:type="ovf:ResourceAllocationSection_type">
      <Item>
        <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
        <rasd:Address>00:03:ba:d8:ba:f6</rasd:Address>
        <gprop:GenericPropertykey="hostid">83d8baf6</gprop:GenericProperty>
        <gprop:GenericProperty key="master">plum</gprop:GenericProperty>
        <gprop:GenericProperty key="failure-policy">reset</gprop:GenericProperty>
        <gprop:GenericProperty key="progress">45%</gprop:GenericProperty>
        <gprop:GenericProperty key="status">ongoing</gprop:GenericProperty>
        <gprop:GenericProperty key="source">dt90-319</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

The `ldom_info` resource is always contained within a `<Content>` section. The following properties within the `ldom_info` resource are optional properties:

- `<rasd:Address>` tag, which specifies the MAC address to be assigned to a domain.

- `<gprop:GenericPropertykey="failure-policy">` tag, which specifies how slave domains should behave should the master domain fail. The default value is `ignore`. Following are the valid property values:

  - `ignore` ignores failures of the master domain (slave domains are unaffected).
  - `panic` panics any slave domains when the master domain fails.
  - `reset` resets any slave domains when the master domain fails.
  - `stop` stops any slave domains when the master domain fails.

- `<gprop:GenericPropertykey="hostid">` tag, which specifies the host ID to be assigned to the domain.

- `<gprop:GenericPropertykey="master">` tag, which specifies up to four comma-separated master domain names.

- `<gprop:GenericPropertykey="progress">` tag, which specifies the percentage of progress made by the command.

- `<gprop:GenericPropertykey="source">` tag, which specifies the machine reporting on the progress of the command.

- `<gprop:GenericPropertykey="status">` tag, which specifies the status of the command (`done`, `failed`, or `ongoing`).

CPU (`cpu`) Resource

The equivalent of the `add-vcpu`, `set-vcpu`, and `remove-vcpu` XML request actions is to set the value of the `<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.

EXAMPLE D–7   Example cpu XML

The following example shows the XML request equivalent for the `ldm add-vcpu -c 1 ldg1` command:

```
<?xml version="1.0"?>
<LDM_interface version="1.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="./schemas/combined-v3.xsd"
xmlns:ovf="./schemas/envelope"
xmlns:rasd="./schemas/CIM_ResourceAllocationSettingData"
```

EXAMPLE D–7 Example cpu XML   *(Continued)*

```
xmlns:vssd="./schemas/CIM_VirtualSystemSettingData"
xmlns:gprop="./schemas/GenericProperty"
xmlns:bind="./schemas/Binding">
  <cmd>
    <action>add-vcpu</action>
    <data version="3.0">
      <Envelope>
        <References/>
        <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="ldg1">
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>cpu</rasd:OtherResourceType>
              <rasd:AllocationUnits>8</rasd:AllocationUnits>
              <gprop:GenericProperty key="wcore">1</gprop:GenericProperty>
            </Item>
          </Section>
        </Content>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

A cpu resource is always contained within a <Content> section.

MAU (mau) Resource

Note – The mau resource is any supported cryptographic unit on a supported server. Currently, the two cryptographic units supported are the Modular Arithmetic Unit (MAU) and the Control Word Queue (CWQ).

EXAMPLE D–8 Example mau XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>mau</rasd:OtherResourceType>
        <rasd:AllocationUnits>1</rasd:AllocationUnits>
      </Item>
    </Section>
  </Content>
</Envelope>
```

A mau resource is always contained within a <Content> section. The only property is the <rasd:AllocationUnits> tag, which signifies the number of MAUs or other cryptographic units.

Memory (`memory`) Resource

EXAMPLE D–9 Example memory XML

```
<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>
```

A memory resource is always contained within a <Content> section. The only property is the <rasd:AllocationUnits> tag, which signifies the amount of memory.

Virtual Disk Server (`vds`) Resource

EXAMPLE D–10 Example vds XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vds</rasd:OtherResourceType>
        <gprop:GenericProperty
          key="service_name">vdstmp</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

A virtual disk server (vds) resource can be in a <Content> section as part of a domain description, or it can appear on its own in an <Envelope> section. The only property is the <gprop:GenericProperty> tag with a key of service_name and which contains the name of the vds resource being described.

Virtual Disk Server Volume (`vds_volume`) Resource

EXAMPLE D–11 Example vds_volume XML

```
<Envelope>
  <References/>
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
```

EXAMPLE D–11 Example vds_volume XML   *(Continued)*

```
      <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>
```

A vds_volume resource can be in a <Content> section as part of a domain description, or it can appear on its own in an <Envelope> section. It must have <gprop:GenericProperty> tags with the following keys:

- vol_name – Name of the volume
- service_name – Name of the virtual disk server to which this volume is to be bound
- block_dev – File or device name to be associated with this volume

Optionally, a vds_volume resource can also have the following properties:

- vol_opts – One or more of the following, comma-separated, within one string: {ro,slice,excl}
- mpgroup – Name of the multipath (failover) group

Disk (`disk`) Resource

EXAMPLE D–12 Example disk XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>disk</rasd:OtherResourceType>
        <gprop:GenericProperty key="vdisk_name">vdisk0</gprop:GenericProperty>
        <gprop:GenericProperty
          key="service_name">primary-vds0</gprop:GenericProperty>
        <gprop:GenericProperty key="vol_name">vdsdev0</gprop:GenericProperty>
        <gprop:GenericProperty key="timeout">60</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

A `disk` resource is always contained within a `<Content>` section. It must have
`<gprop:GenericProperty>` tags with the following keys:

- `vdisk_name` – Name of the virtual disk

- `service_name` – Name of the virtual disk server to which this virtual disk is to be bound

- `vol_name` – Virtual disk service device with which this virtual disk is to be associated

Optionally, the `disk` resource can also have the `timeout` property, which is the timeout value in
seconds for establishing a connection between a virtual disk client (vdc) and a virtual disk
server (vds). If there are multiple virtual disk (vdisk) paths, then the vdc can try to connect to a
different vds, and the timeout ensures that a connection to any vds is established within the
specified amount of time.

Virtual Switch (vsw) Resource

EXAMPLE D–13   Example vsw XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vsw</rasd:OtherResourceType>
        <gprop:GenericProperty key="service_name">vsw1-ldg1</gprop:GenericProperty>
        <gprop:GenericProperty key="dev_path">bge0</gprop:GenericProperty>
        <gprop:GenericProperty key="linkprop">phys-state</gprop:GenericProperty>
        <rasd:Address>00:14:4f:fc:00:01</rasd:Address>
        <gprop:GenericProperty key="mode">sc</gprop:GenericProperty>
        <gprop:GenericProperty key="pvid">12345678</gprop:GenericProperty>
        <gprop:GenericProperty key="vid">87654321</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

A vsw resource can be either in a `<Content>` section as part of a domain description, or it can
appear on its own in an `<Envelope>` section. It must have `<gprop:GenericProperty>` tags with
the following keys:

- `service_name` – Name to be assigned to the virtual switch.

- `linkprop` – Specifies whether the virtual device should get physical link state updates. When
  the value is `phys-state`, the virtual device gets physical link state updates. When the value is
  blank, the virtual device does not get physical link state updates. By default, the virtual
  device does not get physical link state updates.

- `dev_path` – Path of the network device to be associated with this virtual switch

Optionally, the vsw 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 – sc for Oracle Solaris Cluster heartbeat support.

Network (`network`) Resource

EXAMPLE D–14   Example network XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>network</rasd:OtherResourceType>
        <gprop:GenericProperty key="linkprop">phys-state</gprop:GenericProperty>
        <gprop:GenericProperty key="vnet_name">ldg1-vnet0</gprop:GenericProperty>
        <gprop:GenericProperty
          key="service_name">primary-vsw0</gprop:GenericProperty>
        <rasd:Address>00:14:4f:fc:00:01</rasd:Address>
      </Item>
    </Section>
  </Content>
</Envelope>
```

A network resource is always contained within a `<Content>` section. It must have `<gprop:GenericProperty>` tags with the following keys:

- linkprop – Specifies whether the virtual device should get physical link state updates. When the value is phys-state, the virtual device gets physical link state updates. When the value is blank, the virtual device does not get physical link state updates. By default, the virtual device does not get physical link state updates.

- vnet_name – Name of the virtual network (vnet)

- service_name – Name of the virtual switch (vswitch) to which this virtual network is to be bound

Optionally, the network resource can also have the following properties:

- `<rasd:Address>` – Assigns a MAC address to the virtual switch

- pvid – Port virtual local area network (VLAN) identifier (ID) indicates the VLAN of which the virtual network needs to be a member, in untagged mode.

- vid – Virtual local area network (VLAN) identifier (ID) indicates the VLAN of which a virtual network and virtual switch need to be a member, in tagged mode.

- mode – hybrid to enable hybrid I/O for that virtual network.

Virtual Console Concentrator (`vcc`) Resource

EXAMPLE D–15 Example vcc XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vcc</rasd:OtherResourceType>
        <gprop:GenericProperty key="service_name">vcc1</gprop:GenericProperty>
        <gprop:GenericProperty key="min_port">6000</gprop:GenericProperty>
        <gprop:GenericProperty key="max_port">6100</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

A vcc resource can be either in a <Content> section as part of a domain description, or it can appear on its own in an <Envelope> section. It can have <gprop:GenericProperty> tags with the following keys:

- service_name – Name to be assigned to the virtual console concentrator service
- min_port – Minimum port number to be associated with this vcc
- max_port – Maximum port number to be associated with this vcc

Variable (`var`) Resource

EXAMPLE D–16 Example var XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>var</rasd:OtherResourceType>
        <gprop:GenericProperty key="name">test_var</gprop:GenericProperty>
        <gprop:GenericProperty key="value">test1</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

A var resource is always contained within a <Content> section. It can have
<gprop:GenericProperty> tags with the following keys:

- name – Name of the variable
- value – Value of the variable

Physical I/O Device (`physio_device`) Resource

EXAMPLE D–17 Example physio_device XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
        <gprop:GenericProperty key="name">pci@780</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

A physio_device resource is always contained within a <Content> section. The only property
is the <gprop:GenericProperty> tag with the name key property value, which is the name of the
I/O device being described.

SP Configuration (`spconfig`) Resource

EXAMPLE D–18 Example spconfig XML

```
<Envelope>
    <Section xsi:type="ovf:ResourceAllocationSection_type">
      <Item>
        <rasd:OtherResourceType>spconfig</rasd:OtherResourceType>
        <gprop:GenericProperty
          key="spconfig_name">primary</gprop:GenericProperty>
        <gprop:GenericProperty
          key="spconfig_status">current</gprop:GenericProperty>
      </Item>
    </Section>
</Envelope>
```

A service processor (SP) configuration (spconfig) resource always appears on its own in an <Envelope> section. It can have <gprop:GenericProperty> tags with the following keys

- spconfig_name – Name of a configuration to be stored on the SP
- spconfig_status – The current status of a particular SP configuration. This property is used in the output of an ldm list-spconfig command.

Virtual Data Plane Channel Service (vdpcs) Resource

EXAMPLE D–19 Example vdpcs XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vdpcs</rasd:OtherResourceType>
        <gprop:GenericProperty key="service_name">dg1-vdpcs</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

This resource is only of interest in a Netra DPS environment. A vdpcs resource can be either in a <Content> section as part of a domain description, or it can appear on its own in an <Envelope> section. The only property is the <gprop:GenericProperty> tag with the service_name key property value, which is the name of the virtual data plane channel service (vdpcs) resource being described.

Virtual Data Plane Channel Client (vdpcc) Resource

EXAMPLE D–20 Example vdpcc XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vdpcc</rasd:OtherResourceType>
        <gprop:GenericProperty key="vdpcc_name">vdpcc</gprop:GenericProperty>
        <gprop:GenericProperty
          key="service_name">ldg1-vdpcs</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

This resource is only of interest in a Netra DPS environment. A virtual data plane channel client resource is always contained within a <Content> section. It can have <gprop:GenericProperty> tags with the following keys:

- vdpcc_name – Name of the virtual data plane channel client (vdpcc)
- service_name – Name of the virtual data plane channel service vdpcs to which this vdpcc is to be bound

Console (`console`) Resource

EXAMPLE D–21 Example console XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>console</rasd:OtherResourceType>
        <gprop:GenericProperty key="port">6000</gprop:GenericProperty>
        <gprop:GenericProperty key="service_name">vcc2</gprop:GenericProperty>
        <gprop:GenericProperty key="group">group-name</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

A console resource is always contained within a <Content> section. It can have <gprop:GenericProperty> tags with the following keys:

- port – Port to which to change this virtual console (console)
- service_name – Virtual console concentrator (vcc) service to which to bind this console
- group – Name of the group to which to bind this console

Domain Migration

This example shows what is contained in the <data> section for a migrate-domain subcommand.

EXAMPLE D–22 Example migrate-domain <data> Section

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="ldg1"/>
  <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="ldg1"/>
    <Section xsi:type="ovf:ResourceAllocationSection_Type">
      <Item>
        <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
```

EXAMPLE D–22   Example `migrate-domain <data>` Section   *(Continued)*

```
        <gprop:GenericProperty key="target">target-host</gprop:GenericProperty>
        <gprop:GenericProperty key="username">user-name</gprop:GenericProperty>
        <gprop:GenericProperty key="password">password</gprop:GenericProperty>
      <Item>
    </Section>
  </Content>
</Envelope>
```

Where:

- First `<Content>` node (without an `<ldom_info>` section) is the source domain to migrate.

- Second `<Content>` node (with an `<ldom_info>` section) is the target domain to which to migrate. The source and target domain names can be the same.

- The `<ldom_info>` section for the target domain describes the machine to which to migrate and the details needed to migrate to that machine:

  - `target-host` is the target machine to which to migrate.

  - `user-name` is the login user name for the target machine. Must be SASL 64-bit encoded.

  - `password` is the password to use for logging into the target machine. Must be SASL 64-bit encoded.

Note – The Logical Domains Manager uses `sasl_decode64()` to decode the target user name and password and uses `sasl_encode64()` to encode these values. SASL 64 encoding is equivalent to `base64` encoding.

E

Logical Domains Manager XML Schemas

This appendix provides various XML schemas for your use with the Logical Domains Manager.

This chapter covers the following topics:

LDM_interface XML Schema

This schema is a snapshot of the Open Virtualization Format (OVF) Draft Specification version 0.98

EXAMPLE E–1 LDM_interface XML Schema

```
<?xml version="1.0"?>
xs:schema
     xmlns:ovf="/var/opt/SUNWldom/envelope"
     xmlns:xs="http://www.w3.org/2001/XMLSchema">
     <xs:import namespace="/var/opt/SUNWldom/envelope" schemaLocation="ovf-envelope.xsd"/>

     <xs:annotation>
         <xs:documentation>
             Copyright (c) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
         </xs:documentation>
```

EXAMPLE E–1 LDM_interface XML Schema   *(Continued)*

```
      </xs:annotation>
<!--
=================
Type Definitions
=================
-->
  <xs:simpleType name="statusStringType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="success"/>
      <xs:enumeration value="failure"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="responseType">
    <xs:sequence>
      <xs:element name="status" type="statusStringType"/>
      <xs:element name="resp_msg" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- LDM interface document -->
  <xs:element name="LDM_interface">
    <xs:complexType>
      <xs:sequence>

        <!-- START cmd -->
        <xs:element name="cmd" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="action" type="xs:string" minOccurs="0"/>

              <!-- START data -->
              <xs:element name="data" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:choice minOccurs="1" maxOccurs="unbounded">

                    <!--OVF Evelope Version 0.9 -->
                    <xs:element name="Envelope" type="ovf:Envelope_Type"/>
                    <!-- DATA response -->
                    <xs:element name="response" type="responseType" minOccurs="0" maxOccurs="1"/>
                  </xs:choice>
                  <xs:attribute name="version" type="xs:string" use="required"/>
                </xs:complexType>
              </xs:element> <!-- END data -->

              <!-- CMD response -->
              <xs:element name="response" type="responseType" minOccurs="0" maxOccurs="1"/>

            </xs:sequence>
          </xs:complexType>
        </xs:element> <!-- END cmd -->

        <!-- DOCUMENT response -->
        <xs:element name="response" type="responseType" minOccurs="0" maxOccurs="1"/>
```

```
      </xs:sequence>
      <xs:attribute name="version" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element> <!-- LDM interface document -->

</xs:schema>
```

LDM_Event XML Schema

EXAMPLE E–2   LDM_Event XML Schema

```
<?xml version="1.0"?>
<xs:schema
    xmlns:ovf="/var/opt/SUNWldom/envelope"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:import namespace="/var/opt/SUNWldom/envelope" schemaLocation="ovf-envelope.xsd"/>

    <xs:annotation>
      <xs:documentation>
        Copyright (c) 2007, 2010, Oracle and/or its affiliates.  All rights reserved.
      </xs:documentation>
    </xs:annotation>

  <!-- LDM interface document -->
  <xs:element name="LDM_event">
    <xs:complexType>
      <xs:sequence>

        <!-- START cmd -->
        <xs:element name="cmd" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="action" type="xs:string" minOccurs="0"/>

              <!-- START data -->
              <xs:element name="data" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:choice minOccurs="1" maxOccurs="unbounded">

                  <!--OVF Evelope Version 0.9 -->
                  <xs:element name="Envelope" type="ovf:Envelope_Type"/>

                  </xs:choice>
                  <xs:attribute name="version" type="xs:string" use="required"/>
                </xs:complexType>
              </xs:element> <!-- END data -->

            </xs:sequence>
          </xs:complexType>
        </xs:element> <!-- END cmd -->

      </xs:sequence>
```

```
        <xs:attribute name="version" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element> <!-- LDM interface document -->

</xs:schema>
```

The `ovf-envelope.xsd` Schema

EXAMPLE E–3   The ovf-envelope.xsd Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/envelope"
  xmlns:ovf="/var/opt/SUNWldom/envelope"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- Include virtual hardware schema -->
  <xs:include schemaLocation="./ovf-section.xsd"/>
  <xs:include schemaLocation="./cim-virtualhardware.xsd"/>
  <xs:include schemaLocation="./ovf-core.xsd"/>

  <!-- Root element of a OVF package-->
  <xs:element name="Envelope" type="ovf:Envelope_Type"/>

  <xs:complexType name="Envelope_Type">
    <xs:sequence>
      <!--- References to all external files -->
      <xs:element name="References" type="ovf:References_Type"/>

      <!-- Package level meta-data -->
      <xs:element name="Section" type="ovf:Section_Type" minOccurs="0" maxOccurs="unbounded"/>

      <!-- Content. A virtual machine or a vService -->
      <xs:element name="Content" type="ovf:Entity_Type" minOccurs="0" maxOccurs="unbounded"/>

      <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="signed" type="xs:boolean" use="optional"/>
    <xs:attribute name="manifest" type="xs:boolean" use="optional"/>
    <xs:anyAttribute namespace="##any"/>
  </xs:complexType>

  <xs:complexType name="References_Type">
    <xs:sequence>
      <xs:element name="File" type="ovf:File_Type" minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
<xs:anyAttribute namespace="##any"/>
  </xs:complexType>

  <!--Type for an external reference to a resource -->
```

```
  <xs:complexType name="File_Type">
    <xs:sequence>
      <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>

    <!-- Reference key used in other parts of the package -->
    <xs:attribute name="id" type="xs:string" use="required"/>
    <!-- Same as using a single part element -->
    <xs:attribute name="href" type="xs:string" use="required"/>
    <!-- Size in bytes of the files (if known) -->
    <xs:attribute name="size" type="xs:integer" use="optional"/>
    <!-- Estimated size in bytes of the files (if a good guess is known) -->
    <xs:attribute name="estSize" type="xs:integer" use="optional"/>
    <!-- Compression type (gzip or bzip2) -->
    <xs:attribute name="compression" type="xs:string" use="optional"/>
    <!-- Chunk size (except of last chunk) -->
    <xs:attribute name="chunkSize" type="xs:long" use="optional"/>

    <xs:anyAttribute namespace="##any"/>
  </xs:complexType>

  <!-- Base class for an entity -->
  <xs:complexType name="Entity_Type" abstract="true">
    <xs:sequence>
      <xs:element name="Info" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Section" type="ovf:Section_Type" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- A Virtual Machine Entity -->
  <xs:complexType name="VirtualSystem_Type">
<xs:complexContent>
      <xs:extension base="ovf:Entity_Type"> </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- A Composite Service -->
  <xs:complexType name="VirtualSystemCollection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Entity_Type">
       <xs:sequence>
        <xs:element name="Content" type="ovf:Entity_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
       </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

The `ovf-section.xsd` Schema

EXAMPLE E–4   The ovf-section.xsd Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/envelope"
  xmlns:ovf="/var/opt/SUNWldom/envelope"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <!-- The base class for a section. Subclassing this is the most common form of extensibility -->
  <xs:complexType name="Section_Type" abstract="true">
    <xs:sequence>
      <!-- The info element specifies the meaning of the section. This is typically shown
        if the section is not understood by the importer -->
      <xs:element name="Info" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  <!-- Whether the import should fail or not, if the section is not understood -->
    <xs:attribute name="required" type="xs:boolean" use="optional"/>
    <xs:anyAttribute namespace="##any"/>
    <!-- Subtypes defines more specific elements -->
  </xs:complexType>

  <!-- A basic type for a localizable string -->
  <xs:complexType name="Info_Type">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute ref="xml:lang"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
```

The `ovf-core.xsd` Schema

EXAMPLE E–5   The ovf-core.xsd Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/envelope"
  xmlns:ovf="/var/opt/SUNWldom/envelope"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:include schemaLocation="ovf-section.xsd"/>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <!-- A user defined annotation on an entity -->
  <xs:complexType name="AnnotationSection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Section_Type">
        <xs:sequence>
```

```
        <!-- Several localized annotations can be included -->
        <xs:element name="Annotation" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Product information about a virtual appliance -->
<xs:complexType name="ProductSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Product" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Vendor" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Version" type="xs:string" minOccurs="0"/>
        <xs:element name="Full-version" type="xs:string" minOccurs="0"/>
        <xs:element name="ProductUrl" type="xs:string" minOccurs="0"/>
        <xs:element name="VendorUrl" type="xs:string" minOccurs="0"/>
        <xs:element name="AppUrl" type="xs:string" minOccurs="0"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Configuration parameters that can be passed to the virtual machine for
  application-level configuration -->
<xs:complexType name="PropertySection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Property" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Description" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
              <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
                maxOccurs="unbounded"/>
              <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="key" type="xs:string"/>
            <xs:attribute name="type" type="xs:string"/>
            <xs:attribute name="configurableByUser" type="xs:boolean" use="optional"/>
            <xs:attribute name="configurableAtRuntime" type="xs:boolean" use="optional"/>
            <xs:attribute name="defaultValue" type="xs:string" use="optional"/>
            <xs:anyAttribute namespace="##any"/>
          </xs:complexType>
        </xs:element>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
```

```
      <!-- A comma-separated list of transports that are supported by the virtual machine to
        access the OVF environment. -->
      <xs:attribute name="transport" type="xs:string" use="optional"/>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Provides descriptions for the logical networks used within the package. These descriptions are
  typically used as an aid when the package is deployed. -->
<xs:complexType name="NetworkSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Network" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Description" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
              <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
                maxOccurs="unbounded"/>
              <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:anyAttribute namespace="##any"/>
          </xs:complexType>
        </xs:element>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Provides meta-information description of the virtual disks in the package -->
<xs:complexType name="DiskSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Disk" type="ovf:VirtualDiskDesc_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- Disk -->
<xs:complexType name="VirtualDiskDesc_Type">
  <!-- A logical ID for the virtual disk within this package -->
  <xs:attribute name="diskId" type="xs:string" use="required"/>
  <!-- A file reference to the virtual disk file. If this is not specified a blank virtual disk is
    created of the given size -->
  <xs:attribute name="fileRef" type="xs:string" use="optional"/>
  <!-- Capacity in bytes. The capacity can be specified as either a size or as a reference to a property
    using $(property_name) -->
```

```
  <xs:attribute name="capacity" type="xs:string" use="required"/>
  <!-- Format of the disk. The format is an URL that identifies the disk type,
    e.g., http://www.vmware.com/format/vmdk.html#sparse -->
  <xs:attribute name="format" type="xs:string" use="required"/>
  <!-- Populated size of disk. This is an estimation of how much storage the disk needs if backed by
    a non pre-allocated (aka. sparse) disk. This size does not take the meta-data into
    account used by a sparse disk. -->
  <xs:attribute name="populatedSize" type="xs:long" use="optional"/>
  <!-- Reference to a potential parent disk -->
  <xs:attribute name="parentRef" type="xs:string" use="optional"/>
</xs:complexType>

<!-- CPU Architecture requirements for the guest software. -->
<xs:complexType name="CpuCompatibilitySection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Level" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="level" type="xs:int" use="optional"/>
            <xs:attribute name="eax" type="xs:string" use="optional"/>
            <xs:attribute name="ebx" type="xs:string" use="optional"/>
            <xs:attribute name="ecx" type="xs:string" use="optional"/>
            <xs:attribute name="edx" type="xs:string" use="optional"/>
          </xs:complexType>
        </xs:element>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="Vendor" type="xs:string"/>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Specification of the operating system installed in the guest -->
<xs:complexType name="OperatingSystemSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Description" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <!-- The IDs are the enumeration used in CIM_OperatingSystem_Type -->
      <xs:attribute name="id" type="xs:string"/>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- End-User License Agreement -->
<xs:complexType name="EulaSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
```

```
        <!-- Contains the license agreement in plain text. Several different locales can be
          specified -->
        <xs:element name="License" type="ovf:Info_Type" minOccurs="1" maxOccurs="unbounded"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- For a VirtualSystemCollection, this section is used to specify the order in which the
  contained entities are to be powered on. -->
<xs:complexType name="StartupSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <!-- Id of entity in collection -->
            <xs:attribute name="id" type="xs:string"/>
            <!-- Startup order. Entities are started up starting with lower-numbers first. Items with
              same order identifier may be started up concurrently or in any order.
              The order is reversed for shutdown. -->
            <xs:attribute name="order" type="xs:int"/>
            <!-- Delay in seconds to wait for the power on to complete -->
            <xs:attribute name="startDelay" type="xs:int"/>
            <!-- Whether to resume power-on sequence, once the guest reports ok. -->
            <xs:attribute name="waitingForGuest" type="xs:boolean"/>
            <!-- Delay in seconds to wait for the power on to complete -->
            <xs:attribute name="stopDelay" type="xs:int"/>
            <!-- Stop action to use. Valid values are: 'powerOn' (default), 'none'. -->
            <xs:attribute name="startAction" type="xs:string"/>
            <!-- Stop action to use. Valid values are: 'powerOff' (default), 'guestShutdown',
              'suspend'. -->
            <xs:attribute name="stopAction" type="xs:string"/>
            <xs:anyAttribute namespace="##any"/>
          </xs:complexType>
        </xs:element>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <!-- A comma-separated list of transports that the virtual machine supports to provide
        feedback. -->
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- If this section is present, it indicates that the virtual machine needs to be initially
  booted to install and configure the software. -->
<xs:complexType name="InstallSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
```

```
              maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
          <!-- A comma-separated list of transports that the virtual machine supports to provide
            feedback. -->
          <xs:attribute name="transport" type="xs:string"/>
          <xs:anyAttribute namespace="##any"/>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

The `ovf-virtualhardware.xsc` Schema

EXAMPLE E–6   The ovf-virtualhardware.xsc Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/envelope"
  xmlns:ovf="/var/opt/SUNWldom/envelope"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:vssd="/var/opt/SUNWldom/CIM_VirtualSystemSettingData
  xmlns:rasd="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData">

  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <xs:include schemaLocation="ovf-section.xsd"/>

  <xs:import namespace="/var/opt/SUNWldom/CIM_VirtualSystemSettingData" schemaLocation="cim-vssd.xsd"/>
  <xs:import namespace="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
    schemaLocation="cim-rasd.xsd"/>

  <!-- Specifies the virtual hardware for a virtual machine -->
  <xs:complexType name="VirtualHardwareSection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Section_Type">
        <xs:sequence>
          <xs:element name="System" type="vssd:CIM_VirtualSystemSettingData_Type" minOccurs="0"/>
          <xs:element name="Item" type="rasd:CIM_ResourceAllocationSettingData_Type"
            minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Specifies a section for resource constraints on a VirtualSystemCollection -->
  <xs:complexType name="ResourceAllocationSection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Section_Type">
        <xs:sequence>
          <xs:element name="Item" type="rasd:CIM_ResourceAllocationSettingData_Type"
            minOccurs="0" maxOccurs="unbounded"/>
```

EXAMPLE E–6   The ovf-virtualhardware.xsc Schema   *(Continued)*

```
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

The `cim-rasd.xsd` Schema

EXAMPLE E–7   The cim-rasd.xsd Schema

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
  xmlns:class="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
  xmlns:cim="/var/opt/SUNWldom/common"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:import namespace="/var/opt/SUNWldom/common" schemaLocation="cim-common.xsd"/>

  <xs:element name="Caption" nillable="true" type="cim:cimString"/>

  <xs:element name="Description" nillable="true" type="cim:cimString"/>

  <xs:element name="InstanceId" nillable="true" type="cim:cimString"/>

  <xs:element name="ResourceType" nillable="true">
    <xs:complexType>
      <xs:simpleContent>
        <xs:restriction base="xs:anyType">
          <xs:simpleType>
            <xs:union>
              <xs:simpleType>
                <xs:restriction base="xs:unsignedShort">
                  <xs:enumeration value="1"/> <!-- Other -->
                  <xs:enumeration value="2"/> <!-- Computer System -->
                  <xs:enumeration value="3"/> <!-- Processor-->
                  <xs:enumeration value="4"/> <!-- Memory-->
                  <xs:enumeration value="5"/> <!-- IDE Controller -->
                  <xs:enumeration value="6"/> <!-- Parallel SCSI HBA -->
                  <xs:enumeration value="7"/> <!-- FC HBA -->
                  <xs:enumeration value="8"/> <!-- iSCSI HBA -->
                  <xs:enumeration value="9"/> <!-- IB HCA -->
                  <xs:enumeration value="10"/> <!-- Ethernet Adapter -->
                  <xs:enumeration value="11"/> <!-- Other Network Adapter -->
                  <xs:enumeration value="12"/> <!-- I/O Slot -->
                  <xs:enumeration value="13"/> <!-- I/O Device -->
                  <xs:enumeration value="14"/> <!-- Floppy Drive -->
                  <xs:enumeration value="15"/> <!-- CD Drive -->
                  <xs:enumeration value="16"/> <!-- DVD drive -->
                  <xs:enumeration value="17"/> <!-- Disk Drive -->
                  <xs:enumeration value="18"/> <!-- Tape Drive -->
                  <xs:enumeration value="19"/> <!-- Storage Extent -->
                  <xs:enumeration value="20"/> <!-- Other storage device -->
```

```
                <xs:enumeration value="21"/> <!-- Serial port -->
                <xs:enumeration value="22"/> <!-- Parallel port -->
                <xs:enumeration value="23"/> <!-- USB Controller -->
                <xs:enumeration value="24"/> <!-- Graphics controller -->
                <xs:enumeration value="25"/> <!-- IEEE 1394 Controller -->
                <xs:enumeration value="26"/> <!-- Partitionable Unit -->
                <xs:enumeration value="27"/> <!-- Base Partitionable Unit -->
                <xs:enumeration value="28"/> <!-- Power Supply -->
                <xs:enumeration value="29"/> <!-- Cooling Device -->
                <xs:enumeration value="29"/> <!-- Cooling Device -->
                <xs:enumeration value="31"/> <!-- PS2 Controller -->
                <xs:enumeration value="32"/> <!-- SIO Controller -->
                <xs:enumeration value="33"/> <!-- Keyboard -->
                <xs:enumeration value="34"/> <!-- Pointing Device -->
              </xs:restriction>
            </xs:simpleType>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:minInclusive value="30"/>
                <xs:maxInclusive value="32769"/>
              </xs:restriction>
            </xs:simpleType>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:minInclusive value="32768"/>
                <xs:maxInclusive value="65535"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:union>
        </xs:simpleType>
        <xs:anyAttribute namespace="##any"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="OtherResourceType" nillable="true" type="cim:cimString"/>

<xs:element name="ResourceSubType" nillable="true" type="cim:cimString"/>

<xs:element name="PoolID" nillable="true" type="cim:cimString"/>

<xs:element name="ConsumerVisibility" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="xs:anyType">
        <xs:simpleType>
          <xs:union>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:enumeration value="0"/>
                <xs:enumeration value="2"/>
                <xs:enumeration value="3"/>
                <xs:enumeration value="4"/>
              </xs:restriction>
            </xs:simpleType>
```

EXAMPLE E–7 The cim-rasd.xsd Schema   *(Continued)*

```
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:minInclusive value="5"/>
                <xs:maxInclusive value="32768"/>
              </xs:restriction>
            </xs:simpleType>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:minInclusive value="32767"/>
                <xs:maxInclusive value="65535"/
              </xs:restriction>
            </xs:simpleType>
          </xs:union>
        </xs:simpleType>
        <xs:anyAttribute namespace="##any"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="HostResource" nillable="true" type="xs:anyType"/>
<xs:element name="AllocationUnits" nillable="true" type="cim:cimString"/>
<xs:element name="VirtualQuantity" nillable="true" type="cim:cimUnsignedLong"/>
<xs:element name="Reservation" nillable="true" type="cim:cimUnsignedLong"/>
<xs:element name="Limit" nillable="true" type="cim:cimUnsignedLong"/>
<xs:element name="Weight" nillable="true" type="cim:cimUnsignedInt"/>
<xs:element name="AutomaticAllocation" nillable="true" type="cim:cimBoolean"/>
<xs:element name="AutomaticDeallocation" nillable="true" type="cim:cimBoolean"/>
<xs:element name="Parent" nillable="true" type="cim:cimString"/>
<xs:element name="Connection" nillable="true" type="cim:cimString"/>
<xs:element name="Address" nillable="true" type="cim:cimString"/
<xs:element name="MappingBehavior" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="xs:anyType">
        <xs:simpleType>
          <xs:union>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:enumeration value="0"/>
                <xs:enumeration value="1"/>
                <xs:enumeration value="2"/>
                <xs:enumeration value="3"/>
                <xs:enumeration value="4"/>
              </xs:restriction>
            </xs:simpleType>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:minInclusive value="5"/>
                <xs:maxInclusive value="32768"/>
              </xs:restriction>
            </xs:simpleType>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort"
                <xs:minInclusive value="32767"/>
                <xs:maxInclusive value="65535"/>
```

```
                </xs:restriction>
              </xs:simpleType>
            </xs:union>
          </xs:simpleType>
          <xs:anyAttribute namespace="##any"/>
        </xs:restriction>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="AddressOnParent" nillable="true" type="cim:cimString"/>

  <xs:element name="BusNumber" nillable="true" type="cim:cimUnsignedShort"/>

  <xs:complexType name="CIM_ResourceAllocationSettingData_Type">
    <xs:sequence>
      <xs:element ref="class:Caption" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="class:Description" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="class:InstanceId" minOccurs="0"/>
      <xs:element ref="class:ResourceType" minOccurs="0"/>
      <xs:element ref="class:OtherResourceType" minOccurs="0"/>
      <xs:element ref="class:ResourceSubType" minOccurs="0"/>
      <xs:element ref="class:PoolID" minOccurs="0"/>
      <xs:element ref="class:ConsumerVisibility" minOccurs="0"/>
      <xs:element ref="class:HostResource" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="class:AllocationUnits" minOccurs="0"/>
      <xs:element ref="class:VirtualQuantity" minOccurs="0"/>
      <xs:element ref="class:Reservation" minOccurs="0"/>
      <xs:element ref="class:Limit" minOccurs="0"/>
      <xs:element ref="class:Weight" minOccurs="0"/>
      <xs:element ref="class:AutomaticAllocation" minOccurs="0"/>
      <xs:element ref="class:AutomaticDeallocation" minOccurs="0"/>
      <xs:element ref="class:Parent" minOccurs="0"/>
      <xs:element ref="class:Connection" maxOccurs="unbounded" minOccurs="0"/>
      <xs:element ref="class:Address" minOccurs="0"/>
      <xs:element ref="class:MappingBehavior" minOccurs="0"/>
      <xs:element ref="class:AddressOnParent" minOccurs="0"/>
      <xs:element ref="class:BusNumber" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any"/>
  </xs:complexType>

  <xs:element name="CIM_ResourceAllocationSettingData"
    type="class:CIM_ResourceAllocationSettingData_Type"/>
</xs:schema>
```

The `cim-vssd.xsd` Schema

EXAMPLE E–8   The cim-vssd.xsd Schema

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/CIM_VirtualSystemSettingData"
  xmlns:class="/var/opt/SUNWldom/CIM_VirtualSystemSettingData"
  xmlns:cim="/var/opt/SUNWldom/common"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:import namespace="/var/opt/SUNWldom/common"
    schemaLocation="cim-common.xsd"/>

  <xs:element name="Caption" nillable="true" type="cim:cimString"/>

  <xs:element name="Description" nillable="true" type="cim:cimString"/>

  <xs:element name="InstanceId" nillable="true" type="cim:cimString"/>

  <xs:element name="VirtualSystemIdentifier" nillable="true" type="cim:cimString"/>

  <xs:element name="VirtualSystemType" nillable="true" type="cim:cimString"/>

  <xs:complexType name="CIM_VirtualSystemSettingData_Type">
    <xs:sequence>
      <xs:element ref="class:Caption" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="class:Description" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="class:InstanceId" minOccurs="0"/>
      <xs:element ref="class:VirtualSystemIdentifier" minOccurs="0"/>
      <xs:element ref="class:VirtualSystemType" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any"/>
  </xs:complexType>

  <xs:element name="CIM_VirtualSystemSettingData" type="class:CIM_VirtualSystemSettingData_Type"/>

</xs:schema>
```

The `cim-common.xsd` Schema

EXAMPLE E–9   The cim-common.xsd Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/common"
  xmlns:cim="/var/opt/SUNWldom/common"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

  <!-- The following are runtime attribute definitions -->
  <xs:attribute name="Key" type="xs:boolean"/>

  <xs:attribute name="Version" type="xs:string"/>
```

```
<!-- The following section defines the extended WS-CIM datatypes -->
<xs:complexType name="cimDateTime">
  <xs:choice>
    <xs:element name="CIM_DateTime" type="xs:string" nillable="true"/>
    <xs:element name="Interval" type="xs:duration"/>
    <xs:element name="Date" type="xs:date"/>
    <xs:element name="Time" type="xs:time"/>
    <xs:element name="Datetime" type="xs:dateTime"/>
  </xs:choice>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<xs:complexType name="cimUnsignedByte">
  <xs:simpleContent>
    <xs:extension base="xs:unsignedByte">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimByte">
  <xs:simpleContent>
    <xs:extension base="xs:byte">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimUnsignedShort">
  <xs:simpleContent>
    <xs:extension base="xs:unsignedShort">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimShort">
  <xs:simpleContent>
    <xs:extension base="xs:short">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimUnsignedInt">
  <xs:simpleContent>
    <xs:extension base="xs:unsignedInt">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimInt">
  <xs:simpleContent>
    <xs:extension base="xs:int">
```

EXAMPLE E–9 The cim-common.xsd Schema *(Continued)*

```
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimUnsignedLong">
  <xs:simpleContent>
    <xs:extension base="xs:unsignedLong">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimLong">
  <xs:simpleContent>
    <xs:extension base="xs:long">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimString">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimBoolean">
  <xs:simpleContent>
    <xs:extension base="xs:boolean">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimFloat">
  <xs:simpleContent>
    <xs:extension base="xs:float">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimDouble">
  <xs:simpleContent>
    <xs:extension base="xs:double">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimChar16">
  <xs:simpleContent>
      <xs:restriction base="cim:cimString">
```

EXAMPLE E–9 The `cim-common.xsd` Schema *(Continued)*

```
      <xs:maxLength value="1"/>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimBase64Binary">
  <xs:simpleContent>
    <xs:extension base="xs:base64Binary">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimHexBinary">
  <xs:simpleContent>
    <xs:extension base="xs:hexBinary">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimReference">
  <xs:sequence>
    <xs:any namespace="##other" maxOccurs="unbounded"/>
  </xs:sequence>
  xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<!-- The following datatypes are used exclusively to define metadata fragments -->
<xs:attribute name="qualifier" type="xs:boolean"/>

<xs:complexType name="qualifierString">
  <xs:simpleContent>
    <xs:extension base="cim:cimString">
      <xs:attribute ref="cim:qualifier" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="qualifierBoolean">
  <xs:simpleContent>
    <xs:extension base="cim:cimBoolean">
      <xs:attribute ref="cim:qualifier" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="qualifierUInt32">
  <xs:simpleContent>
    <xs:extension base="cim:cimUnsignedInt">
      <xs:attribute ref="cim:qualifier" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

EXAMPLE E–9 The cim-common.xsd Schema   *(Continued)*

```
  <xs:complexType name="qualifierSInt64">
    <xs:simpleContent>
      <xs:extension base="cim:cimLong">
        <xs:attribute ref="cim:qualifier" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
<!--
  <xs:complexType name="qualifierSArray">
    <xs:complexContent>
      <xs:extension base="cim:qualifierString"/>
    </xs:complexContent>
  </xs:complexType>
-->
  <!-- The following element is to be used only for defining metadata -->
  <xs:element name=" DefaultValue" type="xs:anySimpleType"/>
</xs:schema>
```

The `GenericProperty` XML Schema

This schema is an extension to the Open Virtualization Format (OVF) schema.

EXAMPLE E–10 The GenericProperty XML Schema

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/GenericProperty"
  xmlns:class="/var/opt/SUNWldom/GenericProperty"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:complexType name="GenericProperty_Type" type="xs:string">
  <xs:attribute name="key" type="xs:string" use="required"/>

  </xs:complexType>
  <xs:element name="GenericProperty" type="class:GenericProperty_Type"/>

</xs:schema>
```

`Binding_Type` XML Schema

This schema is an extension to the Open Virtualization Format (OVF) schema.

EXAMPLE E–11 Binding_Type XML Schema

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/Binding"
  xmlns:class="/var/opt/SUNWldom/Binding"
  xmlns:rasd="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
```

EXAMPLE E–11   Binding_Type XML Schema   *(Continued)*

```
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:import namespace="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
      schemaLocation="cim-rasd.xsd"/>

  <xs:complexType name="Binding_Type">
    <xs:sequence>
      <xs:element name="Item"
        type="rasd:CIM_ResourceAllocationSettingData_Type"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Glossary

This list defines terminology, abbreviations, and acronyms in the Oracle VM Server for SPARC documentation.

A

API	Application programming interface
auditreduce	Merge and select audit records from audit trail files (see the auditreduce(1M) man page).
auditing	Using Oracle Solaris OS auditing to identify the source of security changes
authorization	Setting up authorization using the Oracle Solaris OS RBAC

B

bge	Broadcom Gigabit Ethernet driver on Broadcom BCM57xx devices
BSM	Basic Security module
bsmconv	Enable the BSM (see the bsmconv(1M) man page).
bsmunconv	Disable the BSM (see the bsmunconv(1M) man page).

C

CD	Compact disc
CLI	Command-line interface
compliance	Determining if a system's configuration is in compliance with a predefined security profile
configuration	Name of logical domain configuration that is saved on the service processor

CMT	Chip multithreading
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.
control domain	Domain that creates and manages other logical domains and services
CPU	Central processing unit
CWQ	Control Word Queue; cryptographic unit for Oracle's Sun UltraSPARC T2 based platforms

D

DHCP	Dynamic Host Configuration Protocol
DMA	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.
DMP	Dynamic Multipathing (Veritas)
Logical Domains Manager	A CLI to create and manage logical domains and allocate resources to domains
DPS	Data plane software
DR	Dynamic reconfiguration
drd	Oracle Solaris 10 OS dynamic reconfiguration daemon for Logical Domains Manager (see the drd(1M) man page).
DS	Domain Services module (Oracle Solaris 10 OS)
DVD	Digital versatile disc

E

EFI	Extensible firmware interface
ETM	Encoding Table Management module (Oracle Solaris 10 OS)

F

FC_AL	Fiber Channel Arbitrated Loop

FMA	Fault Management Architecture
`fmd`	Oracle Solaris 10 OS fault manager daemon (see the `fmd(1M)` man page).
`format`	Disk partitioning and maintenance utility (see the `format(1M)` man page).
`fmthard`	Populate label on hard disks (see the `fmthard(1M)` man page).
FTP	File Transfer Protocol

G

Gb	Gigabit
guest domain	Uses services from the I/O and service domains and is managed by the control domain.
GLDv3	Generic LAN Driver version 3.

H

hardening	Modifying Oracle Solaris OS configuration to improve security
HDD	Hard disk drive
hypervisor	Firmware layer interposed between the operating system and the hardware layer

I

I/O domain	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
IB	Infiniband
IDE	Integrated Drive Electronics
IDR	Interim Diagnostics Release
ILOM	Integrated Lights Out Manager
`io`	I/O devices, such as internal disks and PCIe controllers and their attached adapters and devices
`ioctl`	Input/output control call
IP	Internet Protocol

IPMP	Internet Protocol Network Multipathing
ISO	International Organization for Standardization

K

`kaio`	Kernel asynchronous input/output
KB	Kilobyte
KU	Kernel update

L

LAN	Local-area network
LDAP	Lightweight Directory Access Protocol
LDC	Logical domain channel
`ldm`	Logical Domains Manager utility (see the `ldm`(1M) man page).
`ldmd`	Logical Domains Manager daemon
`lofi`	Loopback file
logical domain	A virtual machine comprised of a discrete logical grouping of resources, which has its own operating system and identity within a single computer system
LUN	Logical unit number

M

MAC	Media access control address, which Logical Domains can automatically assign or you can assign manually
MAU	Modular Arithmetic Unit
MB	Megabyte
MD	Machine description in the server database
`mem,memory`	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.

metadb	Create and delete replicas of the Solaris Volume Manager metadevice state database (see the metadb(1M) man page).
metaset	Configure disk sets (see the metaset(1M) man page).
mhd	Multihost disk control operations (see the mhd(7i) man page).
MIB	Management Information Base
minimizing	Installing the minimum number of core Oracle Solaris OS package necessary
MMF	Multimode fiber
MMU	Memory management unit
mpgroup	Multipathing group name for virtual disk failover
mtu	Maximum transmission unit

N

NAT	Network Address Translation
ndpsldcc	Netra DPS Logical Domain Channel Client. *See also* vdpcc.
ndpsldcs	Netra DPS Logical Domain Channel Service. *See also* vdpcs.
NFS	Network file system
NIS	Network Information Services
NIU	Network Interface Unit (Oracle's Sun SPARC Enterprise T5120 and T5220 servers)
NTS	Network terminal server
NVRAM	Non-volatile random-access memory
nxge	Driver for an NIU 10Gb Ethernet adapter

O

OS	Operating system
OVF	Open Virtualization Format

P

P2V	Logical Domains Physical-to-Virtual Conversion Tool
PA	Physical address
PCI	Peripheral component interconnect bus
PCIe	PCI EXPRESS bus
PCI-X	PCI Extended bus
pcpu	Physical CPU
physio	Physical input/output
PICL	Platform Information and Control Library
picld	PICL daemon (see the picld(1M) man page).
PM	Power management of virtual CPUs and memory
praudit	Print contents of an audit trail file (see the praudit(1M) man page).
PRI	Priority

R

RA	Real address
RAID	Redundant Array of Inexpensive Disks
RBAC	Role-Based Access Control
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.
system controller (SC)	Also see service processor
SCSI	Small Computer System Interface

service domain	Logical domain that provides devices, such as virtual switches, virtual console connectors, and virtual disk servers, to other logical domains
SMA	System Management Agent
SMF	Service Management Facility
SNMP	Simple Network Management Protocol
service processor (SP)	The SP, also known as the system controller (SC), monitors and runs the physical machine.
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).

T

TCP	Transmission Control Protocol
TLS	Transport Layer Security

U

UDP	User Diagram Protocol
UFS	UNIX File System
unicast	Network communication that takes place between a single sender and a single receiver
USB	Universal Serial Bus
uscsi	User SCSI command interface (see the uscsi(7I) man page).
UTP	Unshielded twisted pair

V

var	Variable
VBSC	Virtual blade system controller
vcc, vconscon	Virtual console concentrator service with a specific port range to assign to guest domains
vcons, vconsole	Virtual console for accessing system-level messages. A connection is achieved by connecting to the vconscon service in the control domain at a specific port.
vcpu	Virtual central processing unit. Each core in a server is represented as a virtual CPU. For example, an 8-core Sun Fire T2000 Server from Oracle has 32 virtual CPUs that can be allocated among the logical domains.
vdc	Virtual disk client
vdisk	A virtual disk is a generic block device associated with different types of physical devices, volumes, or files.
vdpcc	Virtual data plane channel client in a Netra DPS environment
vdpcs	Virtual data plane channel service in a Netra DPS environment
vds, vdiskserver	Virtual disk server enables you to import virtual disks into a logical domain
vdsdev, vdiskserverdevice	Virtual disk server device is exported by the virtual disk server. The device can be an entire disk, a slice on a disk, a file, or a disk volume.
VLAN	Virtual local area network
vldc	Virtual logical domain channel service
vldcc	Virtual logical domain channel client
vnet	Virtual network device implements a virtual Ethernet device and communicates with other vnet devices in the system by using the virtual network switch (vswitch)
vntsd	Oracle Solaris 10 OS virtual network terminal server daemon for Logical Domains consoles (see the vntsd(1M) man page).
volfs	Volume Management file system (see the volfs(7FS) man page).
vsw, vswitch	Virtual network switch that connects the virtual network devices to the external network and also switches packets between them
VTOC	Volume table of contents
VxDMP	Veritas Dynamic Multipathing
VxVM	Veritas Volume Manager

W

WAN   Wide-area network

X

XFP   eXtreme Fast Path

XML   Extensible Markup Language

XMPP   Extensible Messaging and Presence Protocol

Z

ZFS   Zettabyte File System (Oracle Solaris 10 OS)

zpool   ZFS storage pool (see the zpool(1M) man page).

ZVOL   ZFS Volume Emulation Driver

Index

U

UltraSPARC T2 Plus server, 20
using link-based IPMP, 121–124

V

virtual devices
   I/O, 21
   virtual console concentrator (`vcc`), 21
   virtual disk client (`vdc`), 21
   virtual disk service (`vds`), 21
   virtual network (`vnet`), 21
   virtual switch (`vsw`), 21
virtual disk multipathing, 89
virtual machine, 19
virtual network terminal server daemon (`vntsd`), 21

X

XML schema
   Logical Domains Manager used with, 217, 241