Oracle® VM Server for SPARC 3.3 Administration Guide

Part No: E62358
October 2015

ORACLE

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

Accès au support électronique

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

Contents

Preface

- **Overview** – Provides detailed information and procedures that describe the overview, security considerations, installation, configuration, modification, and execution of common tasks for the Oracle VM Server for SPARC 3.3 software on supported servers, blades, and server modules. See "Supported Platforms" in *Oracle VM Server for SPARC 3.3 Installation Guide*.

  Note – The features that are described in this book can be used with all of the supported system software and hardware platforms that are listed in *Oracle VM Server for SPARC 3.3 Installation Guide*. However, some features are only available on a subset of the supported system software and hardware platforms. For information about these exceptions, see "What's New in This Release" in *Oracle VM Server for SPARC 3.3 Release Notes* and What's New in Oracle VM Server for SPARC Software (`http://www.oracle.com/technetwork/server-storage/vm/documentation/sparc-whatsnew-330281.html`).

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

Product Documentation Library

Documentation and resources for this product and related products are available at `http://www.oracle.com/technetwork/documentation/vm-sparc-194287.html`.

Feedback

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

PART I

Oracle VM Server for SPARC 3.3 Software

This part introduces the Oracle VM Server for SPARC 3.3 software, which provides highly efficient, enterprise-class virtualization capabilities for SPARC T-Series, SPARC M-Series, and Fujitsu M10 platforms.

◆ ◆ ◆ CHAPTER 1

Overview of the Oracle VM Server for SPARC Software

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

Oracle VM Server for SPARC provides highly efficient, enterprise-class virtualization capabilities for Oracle SPARC T-Series and SPARC M-Series platforms, and Fujitsu M10 platforms. Using the Oracle VM Server for SPARC software, you can create up to 128 virtual servers, called logical domains, on a single system. This kind of configuration enables you to take advantage of the massive thread scale offered by SPARC T-Series and SPARC M-Series platforms, Fujitsu M10 platforms and the Oracle Solaris OS.

This chapter covers the following topics:

About Oracle VM Server for SPARC and Oracle Solaris OS Versions

The Oracle VM Server for SPARC software depends on particular Oracle Solaris OS versions, required software patches, and particular versions of system firmware. For more information, see "Fully Qualified Oracle Solaris OS Versions" in *Oracle VM Server for SPARC 3.3 Installation Guide*.

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

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

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

Hypervisor and Logical Domains

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

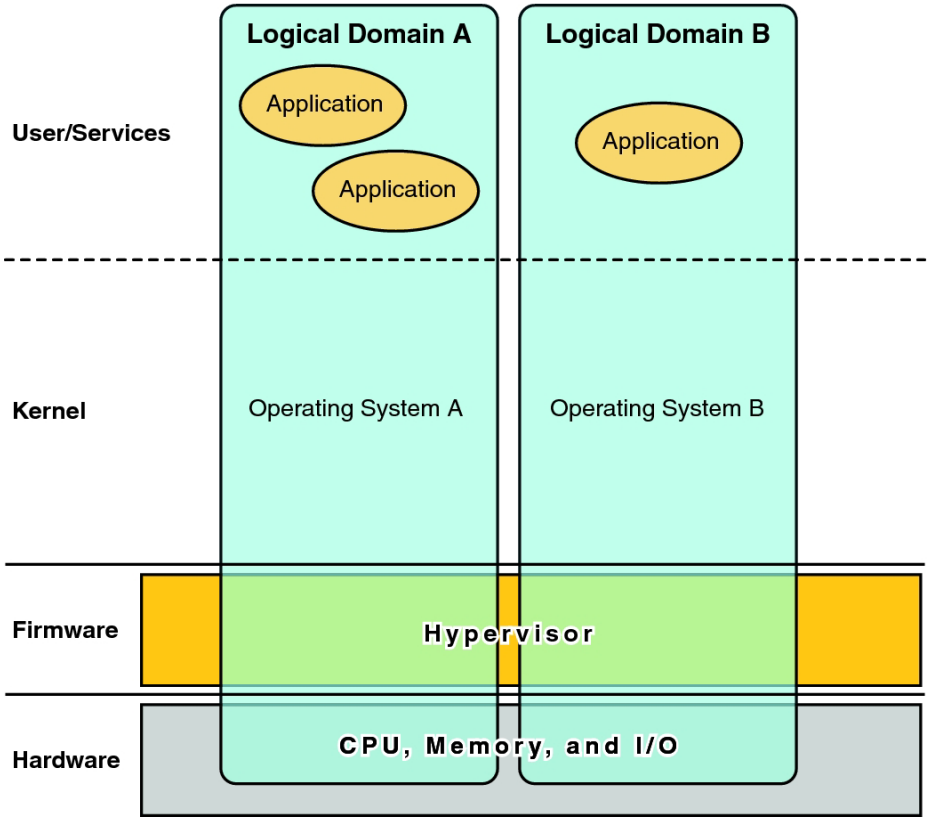
The SPARC *hypervisor* is a small firmware layer that provides a stable virtualized machine architecture to which an operating system can be written. SPARC servers that use the hypervisor provide hardware features to support the hypervisor's control over a logical operating system's activities.

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

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

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

FIGURE 1–1 Hypervisor Supporting Two Domains



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

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

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

Each logical domain can be stopped, started, and rebooted independently of each other without requiring you to perform a power cycle of the server.

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

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

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

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

Logical Domains Manager

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

Roles for Domains

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

- **Control domain.** The Logical Domains Manager runs in this domain, which enables you to create and manage other logical domains, and to allocate virtual resources to other domains. You can have only one control domain per server. The control domain is the first domain created when you install the Oracle VM Server for SPARC software. The control domain is named `primary`.
- **Service domain.** A service domain provides virtual device services to other domains, such as a virtual switch, a virtual console concentrator, and a virtual disk server. You can have more than one service domain, and any domain can be configured as a service domain.
- **I/O domain.** An I/O domain has direct access to a physical I/O device, such as a network card in a PCI EXPRESS (PCIe) controller. An I/O domain can own the following:
  - A PCIe root complex.
  - A PCIe slot or on-board PCIe device by using the direct I/O (DIO) feature. See "Creating an I/O Domain by Assigning PCIe Endpoint Devices" on page 143.

- A PCIe SR-IOV virtual function. See Chapter 7, "Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions."

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

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

  The number of root domains that you can have depends on your platform architecture. For example, if you are using an eight-socket Oracle SPARC T5-8 server, you can have up to 16 root domains.

  The default root domain is the `primary` domain. You can use non-`primary` domains to act as root domains.
- **Guest domain.** A guest domain is a non-I/O domain that consumes virtual device services that are provided by one or more service domains. A guest domain does not have any physical I/O devices but only has virtual I/O devices, such as virtual disks and virtual network interfaces.

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

Command-Line Interface

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

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

Virtual Input/Output

In an Oracle VM Server for SPARC environment, you can provision up to 128 domains on a system (up to 256 on a Fujitsu M10 server). Some servers, particularly single-processor and some dual-processor systems, have a limited number of I/O buses and physical I/O slots. As a result, you might be unable to provide exclusive access to a physical disk and network devices to all domains on these systems. You can assign a PCIe bus or endpoint device to a domain to provide it with access to a physical device. Note that this solution is insufficient to provide all

domains with exclusive device access. This limitation on the number of physical I/O devices that can be directly accessed is addressed by implementing a virtualized I/O model. See Chapter 5, "Configuring I/O Domains."

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

Virtual Network

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

Virtual Storage

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

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

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

Virtual Console

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

Resource Configuration

A system that runs the Oracle VM Server for SPARC software can configure resources such as virtual CPUs, virtual I/O devices, cryptographic units, and memory. Some resources can be configured dynamically on a running domain, while others must be configured on a stopped domain. If a resource cannot be dynamically configured on the control domain, you must first initiate a delayed reconfiguration. The delayed reconfiguration postpones the configuration activities until after the control domain has been rebooted. For more information, see "Resource Reconfiguration" on page 305.

Persistent Configurations

You can use the `ldm` command to store the current configuration of a logical domain on the service processor. You can add a configuration, specify a configuration to be used, remove a configuration, and list the configurations. For details, see the `ldm(1M)` man page. You can also specify a configuration to boot from the SP, as described in "Using Oracle VM Server for SPARC With the Service Processor" on page 366.

For information about managing configurations, see "Managing Domain Configurations" on page 343.

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

The Oracle VM Server for SPARC Physical-to-Virtual (P2V) Conversion tool automatically converts an existing physical system to a virtual system that runs the Oracle Solaris 10 OS in a logical domain on a chip multithreading (CMT) system. You can run the `ldmp2v` command from a control domain that runs the Oracle Solaris 11 OS to convert one of the following source systems to a logical domain:

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

---

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

For information about the tool and about installing it, see Chapter 19, "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 Management Information Base

The Oracle VM Server for SPARC Management Information Base (MIB) enables third-party system management applications to perform remote monitoring of domains, and to start and stop logical domains (domains) by using the Simple Network Management Protocol (SNMP). For more information, see Chapter 21, "Using the Oracle VM Server for SPARC Management Information Base Software."

Oracle VM Server for SPARC Troubleshooting

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

- "Known Issues" in *Oracle VM Server for SPARC 3.3 Release Notes*
- Information Center: Overview of Oracle VM Server for SPARC (LDoms) (Doc ID 1589473.2) (`https://mosemp.us.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=227880986952919&id=1589473.2&_afrWindowMode=0&_adf.ctrl-state=`

◆ ◆ ◆ CHAPTER 2

Oracle VM Server for SPARC Security

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

This chapter covers the following topics:

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

Delegating the Management of Logical Domains by Using Rights

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

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

These rights profiles can be assigned directly to users or to a role that is then assigned to users. When one of these profiles is assigned directly to a user, you must use the `pfexec` command or a profile shell, such as `pfbash` or `pfksh`, to successfully use the `ldm` command to manage your domains. Determine whether to use roles or rights profiles based on your rights configuration. See *System Administration Guide: Security Services* or *Securing Users and Processes in Oracle Solaris 11.3* .

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

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

Installing the Logical Domains Manager adds the necessary rights profiles to the local files. To configure profiles and roles in a naming service, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* . For an overview of the authorizations and execution attributes delivered by the Logical Domains Manager package, see "Logical Domains Manager Profile Contents" on page 36. All of the examples in this chapter assume that the rights configuration uses local files.

Using Rights Profiles and Roles

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

- For the Oracle Solaris 11 OS, add values by using the plus sign (+) for each authorization you add.

  For example, the `usermod -A` +*auth username* command grants the *auth* authorization to the *username* user; similarly for the `rolemod` command.
- For the Oracle Solaris 10 OS, the `usermod` or `rolemod` command replaces any existing values.

  To add values instead of replacing them, specify a comma-separated list of existing values and the new values.

Managing User Rights Profiles

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

▼ How to Assign a Rights Profile to a User

Users who have been directly assigned the `LDoms Management` profile *must* invoke a profile shell to run the `ldm` command with security attributes. For more information, see *System Administration Guide: Security Services* or *Securing Users and Processes in Oracle Solaris 11.3* .

1 Become an administrator.

For Oracle Solaris 11.3, see Chapter 1, "About Using Rights to Control Users and Processes," in *Securing Users and Processes in Oracle Solaris 11.3* .

2. **Assign an administrative profile to a local user account.**

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

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

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

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

Assigning Roles to Users

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

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

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

1. **Become an administrator.**

   For Oracle Solaris 11.3, see Chapter 1, "About Using Rights to Control Users and Processes," in *Securing Users and Processes in Oracle Solaris 11.3* .

2. **Create a role.**

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

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

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

   ```
   # passwd role-name
   ```

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

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

5. **Assign a password to the user.**

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

   ```
   # passwd username
   ```

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

   ```
   # su username
   ```

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

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

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

```
$ su role-name
```

9 Verify that the user has assumed the role.

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

Example 2–1 Creating a Role and Assigning the Role to a User

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

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

Logical Domains Manager Profile Contents

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

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

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

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

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

TABLE 2–1 `ldm` Subcommands and User Authorizations

ldm Subcommand[1]	User Authorization
`add-*`	`solaris.ldoms.write`
`bind-domain`	`solaris.ldoms.write`
`list`	`solaris.ldoms.read`
`list-*`	`solaris.ldoms.read`
`panic-domain`	`solaris.ldoms.write`
`remove-*`	`solaris.ldoms.write`
`set-*`	`solaris.ldoms.write`
`start-domain`	`solaris.ldoms.write`
`stop-domain`	`solaris.ldoms.write`
`unbind-domain`	`solaris.ldoms.write`

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

Controlling Access to a Domain Console by Using Rights

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

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

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

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

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

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

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

- To control the access to all domain consoles, see "How to Control Access to All Domain Consoles by Using Roles" on page 38 and "How to Control Access to All Domain Consoles by Using Rights Profiles" on page 40.
- To control access to a single domain console, see "How to Control Access to a Single Console by Using Roles" on page 41 and "How to Control Access to a Single Console by Using Rights Profiles" on page 42.

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

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

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

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

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

3 Assign the new role to a user.

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

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

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

```
primary# svccfg -s vntsd setprop vntsd/authorization = true
primary# svcadm refresh vntsd
primary# svcadm restart vntsd
primary# ldm ls
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
```

```
primary          active     -n-cv-  UART    8     16G      0.2%  47m
ldg1             active     -n--v-  5000    2     1G       0.1%  17h 50m
ldg2             active     -t----  5001    4     2G        25%  11s
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Use the `profiles` command to create a new profile.

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

3 Assign the rights profile to a user.

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

4 Connect to the domain console as the user.

```
$ telnet localhost 5000
```

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

The following example shows how to use rights profiles to control access to all domain consoles. Use the `profiles` command to create a rights profile with the `solaris.vntsd.consoles` authorization in the rights profile description database.

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

Assign the rights profile to a user.

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

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

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

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

How to Control Access to a Single Console by Using Roles

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

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

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

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

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

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

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

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

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

Example 2–4 Accessing a Single Domain Console

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Use the `profiles` command to create a new profile.

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

4 Assign the rights profile.

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

Enabling and Using Logical Domains Manager Auditing

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

Note – In Oracle VM Server for SPARC, audit records are not generated for Logical Domains Manager actions by default.

You can enable and disable the Oracle Solaris OS auditing feature. Use the `audit` command. See the `audit`(1M) man page and *Managing Auditing in Oracle Solaris 11.3* .

▼ How to Enable Logical Domains Manager Auditing

Although Oracle Solaris 11 auditing is enabled by default, you must configure Logical Domains Manager auditing.

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

---

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

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

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

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

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

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

2 Preselect the `vs` audit class.

a. Determine which auditing classes are already selected.

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

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

b. Add the `vs` auditing class.

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

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

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

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

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

If you do not want to log out, see "How to Update the Preselection Mask of Logged In Users" in *Managing Auditing in Oracle Solaris 11.3* .

3 Verify that Oracle Solaris OS auditing is enabled.

```
# auditconfig -getcond
```

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

4 Configure Logical Domains Manager to generate audit records.

a. Set the `ldmd/audit` SMF property value to `true`.

```
# svccfg -s ldmd setprop ldmd/audit = boolean: true
```

b. Refresh the `ldmd` service.

```
# svcadm refresh ldmd
```

c. Restart the `ldmd` service.

```
# svcadm restart ldmd
```

▼ How to Disable Logical Domains Manager Auditing

1 Set the `ldmd/audit` SMF property value to `false`.

```
# svccfg -s ldmd setprop ldmd/audit = boolean: false
```

2 Refresh the `ldmd` service.

```
# svcadm refresh ldmd
```

3 Restart the `ldmd` service.

```
# svcadm restart ldmd
```

▼ How to Review Audit Records

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

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

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

Using Domain Console Logging

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

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

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

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

▼ How to Enable or Disable Console Logging

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

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

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

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

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

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

3 **Enable or disable console logging.**

- **To enable console logging.**

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

- **To disable console logging.**

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

Service Domain Requirements for Domain Console Logging

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

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

◆ ◆ ◆ CHAPTER 3

Setting Up Services and the Control Domain

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

This chapter covers the following topics:

Output Messages

If a resource cannot be dynamically configured on the control domain, the recommended practice is to first initiate a delayed reconfiguration. The delayed reconfiguration postpones the configuration activities until after the control domain has been rebooted.

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

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

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

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

Creating Default Services

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

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

▼ How to Create Default Services

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

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

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

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

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

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

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

Assign a GLDv3-compliant network adapter to the virtual switch if each logical domain must communicate outside the box through the virtual switch.

Add a virtual switch service (`primary-vsw0`) on a network device that you want to use for guest domain networking.

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

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

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

You can use the `ldm list-netdev -b` command to determine the backend network devices that are available for the virtual switch. See "Virtual Switch" on page 227.

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

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 net0      switch@0   prog,promisc
```

Initial Configuration of the Control Domain

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

Configuring the Control Domain

▼ How to Configure the Control Domain

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

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

1 **Assign virtual CPUs to the control domain.**

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

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

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

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

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

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

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

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

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

3. **Initiate a delayed reconfiguration on the control domain.**

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

4. **Assign memory to the control domain.**

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

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

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

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

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

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

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

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

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

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

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

When in the `factory-default` configuration, the control domain owns all of the host system's memory. The memory DR feature is not well suited for this purpose because an active domain is not guaranteed to add or, more typically, give up, all of the requested memory. Rather, the OS

running in that domain makes a best effort to fulfill the request. In addition, memory removal can be a long-running operation. These issues are amplified when large memory operations are involved, as is the case for the initial decrease of the control domain's memory.

Note – When the Oracle Solaris OS is installed on a ZFS file system, it automatically sizes and creates swap and dump areas as ZFS volumes in the ZFS root pool based on the amount of physical memory that is present. If you change the domain's memory allocation, it might alter the recommended size of these volumes. The allocations might be larger than needed after reducing control domain memory. For swap space recommendations, see "Planning for Swap Space" in *Managing File Systems in Oracle Solaris 11.3*. Before you free disk space, you can optionally change the swap and dump space. See "Managing ZFS Swap and Dump Devices" in *Managing ZFS File Systems in Oracle Solaris 11.3*.

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

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

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

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

2. **Configure the control domain.**

   See "How to Configure the Control Domain" on page 49.

Rebooting to Use Domains

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

▼ How to Reboot

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

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

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

Enabling Networking Between the Oracle Solaris 10 Service Domain and Other Domains

By default, networking between an Oracle Solaris 10 service domain and other domains in the system is disabled. Because networking is not enabled by default in the Oracle Solaris 10 OS, you must enable networking by configuring the virtual switch device as a network device. The virtual switch can either replace the underlying physical device (`nxge0` in this example) as the primary interface or be configured as an additional network interface in the domain.

Guest domains can automatically communicate with the Oracle Solaris 10 service domain as long as the corresponding network back-end device is configured in the same virtual LAN or virtual network.

▼ How to Configure the Virtual Switch as the Primary Interface

Note – Perform the following procedure from the Oracle Solaris 10 service domain's console, as the procedure could temporarily disrupt network connectivity to the domain.

If necessary, you can configure the virtual switch as well as the physical network device. In this case, create the virtual switch as in Step 2, and do not delete the physical device (skip Step 3). You must then configure the virtual switch with either a static IP address or a dynamic IP address. You can obtain a dynamic IP address from a DHCP server. For additional information and an example of this case, see "Configuring a Virtual Switch and the Service Domain for NAT and Routing" on page 244.

1. **Print the addressing information for all interfaces.**

   ```
   # ifconfig -a
   ```

2. **Configure the virtual switch network interface.**

   ```
   # ifconfig vsw0 plumb
   ```

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

   ```
   # ifconfig nxge0 down unplumb
   ```

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

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

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

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

  ```
  # ifconfig vsw0 dhcp start
  ```

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

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

Enabling the Virtual Network Terminal Server Daemon

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

▼ How to Enable the Virtual Network Terminal Server Daemon

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

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

```
primary# svcadm enable vntsd
```

2 **Verify that the `vntsd` daemon is enabled.**

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

Verifying That the ILOM Interconnect Is Enabled

The ILOM interconnect is required for communication between the `ldmd` daemon and the service processor (SP) on SPARC T7 series servers and SPARC M7 series servers and should not be disabled. For more information, see the `ilomconfig`(1M) man page.

Note – Avoid disabling the ILOM interconnect on other SPARC T-Series systems, SPARC M5, and SPARC M6 systems. However, if you do so, the `ldmd` daemon can still communicate with the SP.

If an attempt to use the `ldm` command to manage domain configurations on SPARC T7 series servers and SPARC M7 series servers fails because of an error communicating with the SP, check the ILOM interconnect state and re-enable the `ilomconfig-interconnect` service if necessary. See "How to Verify the ILOM Interconnect Configuration" on page 54 and "How to Re-Enable the ILOM Interconnect Service" on page 55.

▼ How to Verify the ILOM Interconnect Configuration

1 **Verify that the `ilomconfig-interconnect` service is enabled.**

```
primary# svcs ilomconfig-interconnect
STATE          STIME    FMRI
online          9:53:28 svc:/network/ilomconfig-interconnect:default
```

2 **Verify that the ILOM interconnect is configured properly.**

A proper configuration shows the State value as `enabled` and the Host Interconnect IP Address value as an IP address and not `none`.

```
primary# ilomconfig list interconnect
Interconnect
============
State: enabled
Type: USB Ethernet
SP Interconnect IP Address: 169.254.182.76
Host Interconnect IP Address: 169.254.182.77
Interconnect Netmask: 255.255.255.0
SP Interconnect MAC Address: 02:21:28:57:47:16
Host Interconnect MAC Address: 02:21:28:57:47:17
```

3 **Verify that the `ldmd` daemon can communicate with the SP.**

```
primary# ldm list-spconfig
```

How to Re-Enable the ILOM Interconnect Service

The `ilomconfig-interconnect` service is enabled by default. Use this procedure if you need to re-enable this service manually.

1 Enable the ILOM interconnect service.

```
primary# svcadm enable ilomconfig-interconnect
```

2 Verify that the `ilomconfig-interconnect` service is enabled.

```
primary# svcs ilomconfig-interconnect
STATE          STIME    FMRI
online          9:53:28 svc:/network/ilomconfig-interconnect:default
```

3 Verify that the ILOM interconnect is configured properly.

A proper configuration shows the State value as `enabled` and the Host Interconnect IP Address value as an IP address and not `none`.

```
primary# ilomconfig list interconnect
Interconnect
============
State: enabled
Type: USB Ethernet
SP Interconnect IP Address: 169.254.182.76
Host Interconnect IP Address: 169.254.182.77
Interconnect Netmask: 255.255.255.0
SP Interconnect MAC Address: 02:21:28:57:47:16
Host Interconnect MAC Address: 02:21:28:57:47:17
```

4 Verify that the `ldmd` daemon can communicate with the SP.

```
primary# ldm list-spconfig
```

◆ ◆ ◆ CHAPTER 4

Setting Up Guest Domains

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

This chapter covers the following topics:

Creating and Starting a Guest Domain

The guest domain must run an operating system that is compatible with both the `sun4v` platform and the virtual devices presented by the hypervisor. Currently, this requirement means that you must run at least the Oracle Solaris 10 11/06 OS. Running the Oracle Solaris 10 1/13 OS provides you with all the Oracle VM Server for SPARC 3.3 features. See *Oracle VM Server for SPARC 3.3 Installation Guide* for any specific patches that might be necessary. Once you have created default services and reallocated resources from the control domain, you can create and start a guest domain.

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

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

How to Create and Start a Guest Domain

1. **Create a logical domain.**

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

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

2. **Add CPUs to the guest domain.**

   Do one of the following:

   - **Add virtual CPUs.**

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

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

   - **Add whole cores.**

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

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

3. **Add memory to the guest domain.**

   The following command would add 2 gigabytes of memory to guest domain `ldg1`.

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

4. **Add a virtual network device to the guest domain.**

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

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

   Where:

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

   Note – Steps 5 and 6 are simplified instructions for adding a virtual disk server device (`vdsdev`) to the primary domain and a virtual disk (`vdisk`) to the guest domain. To learn how ZFS volumes and file systems can be used as virtual disks, see "How to Export a ZFS Volume as a Single-Slice Disk" on page 181 and "Using ZFS With Virtual Disks" on page 192.

   ---

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

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

- **Physical Disk Example.** This example adds a physical disk with these specifics:

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

  Where:

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

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

  Where:

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

6 Add a virtual disk to the guest domain.

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

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

Where:

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

Note – The virtual disks are generic block devices that are associated with different types of physical devices, volumes, or files. A virtual disk is not synonymous with a SCSI disk and, therefore, excludes the target ID in the disk label. Virtual disks in a logical domain have the following format: 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 the `auto-boot?` and `boot-device` variables for the guest domain.

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

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

The following example command sets `boot-device` to `vdisk1` for guest domain `ldg1`.

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

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

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

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

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

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

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

11 Start the guest domain `ldg1`.

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

Installing Oracle Solaris OS on a Guest Domain

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

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

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

During the Oracle Solaris 11 installation, select the Manual networking configuration. After the Oracle Solaris 11 installation, ensure that the `DefaultFixed` NCP is enabled by using the `netadm list` command. See Chapters 2, 3, 5, and 6 of *Configuring and Managing Network Components in Oracle Solaris 11.3* .

Memory Size Requirements

The Oracle VM Server for SPARC software does not impose a memory size limitation when you create a domain. The memory size requirement is a characteristic of the guest operating system. Some Oracle VM Server for SPARC functionality might not work if the amount of memory present is smaller than the recommended size. For recommended and minimum memory requirements for the Oracle Solaris 10 OS, see "System Requirements and Recommendations" in *Oracle Solaris 10 1/13 Installation Guide: Planning for Installation and Upgrade* . For recommended and minimum memory requirements for the Oracle Solaris 11 OS, see *Oracle Solaris 11 Release Notes* , *Oracle Solaris 11.1 Release Notes* , *Oracle Solaris 11.2 Release Notes* , and *Oracle Solaris 11.3 Release Notes* .

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

The memory dynamic reconfiguration (DR) feature enforces 256-Mbyte alignment on the address and size of the memory involved in a given operation. See "Memory Alignment" on page 324.

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

1 Insert the Oracle Solaris 10 OS or Oracle Solaris 11 OS DVD into the DVD drive.

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

```
primary# svcadm disable volfs
```

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

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

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

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

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

5 Verify that the DVD is added as a secondary volume and virtual disk.

```
primary# ldm list-bindings
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active   -n-cv   SP      8     8G       0.2%  22h 45m
...
VDS
   NAME             VOLUME         OPTIONS          DEVICE
   primary-vds0     vol1                            /dev/dsk/c2t1d0s2
   dvd_vol                                          /dev/dsk/c0t0d0s2
....
--------------------------------------------------------------------------------
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg1             inactive -----           60    6G
...
DISK
   NAME             VOLUME                      TOUT DEVICE  SERVER
   vdisk1           vol1@primary-vds0
   vdisk_cd_media   dvd_vol@primary-vds0
....
```

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

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

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

7 Show the device aliases in the client OpenBoot PROM.

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

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

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

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

9 Continue with the Oracle Solaris OS installation.

▼ How to Install the Oracle Solaris OS on a Guest Domain From an Oracle Solaris ISO File

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

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

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

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

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

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

```
primary# ldm list-bindings
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active   -n-cv   SP      8     8G       0.2%  22h 45m
...
VDS
   NAME             VOLUME         OPTIONS          DEVICE
   primary-vds0     vol1                            /dev/dsk/c2t1d0s2
   iso_vol                                          /export/solarisdvd.iso
....
------------------------------------------------------------------------------
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg1             inactive -----           60    6G
...
DISK
   NAME       VOLUME               TOUT ID DEVICE  SERVER  MPGROUP
   vdisk1     vol1@primary-vds0
   vdisk_iso  iso_vol@primary-vds0
....
```

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

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

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

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

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

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

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

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

7 **Continue with the Oracle Solaris OS installation.**

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

---

Note – The Oracle Solaris JumpStart feature is available only for the Oracle Solaris 10 OS. See *Oracle Solaris 10 1/13 Installation Guide: JumpStart Installations* .

To perform an automated installation of the Oracle Solaris 11 OS, you can use the Automated Installer (AI) feature. See *Transitioning From Oracle Solaris 10 to Oracle Solaris 11.3* .

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

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

  Note – A virtual disk can appear either as a full disk or as a single-slice disk. The Oracle Solaris OS can be installed on a full disk by using a regular JumpStart profile that specifies multiple partitions. A single-slice disk only has a single partition, `s0`, that uses the entire disk. To install the Oracle Solaris OS on a single disk, you must use a profile that has a single partition (`/`) that uses the entire disk. You cannot define any other partitions, such as swap. For more information about full disks and single-slice disks, see "Virtual Disk Appearance" on page 173.

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

    See *Oracle Solaris 10 1/13 Installation Guide: JumpStart Installations* .

Normal UFS Profile

```
filesys c1t1d0s0 free /
filesys c1t1d0s1 2048 swap
filesys c1t1d0s5 120 /spare1
filesys c1t1d0s6 120 /spare2
```

Actual UFS Profile for Installing a Domain on a Full Disk

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

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

```
filesys c0d0s0 free /
```

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

  See Chapter 9, "Installing a ZFS Root Pool With JumpStart," in *Oracle Solaris 10 1/13 Installation Guide: JumpStart Installations* .

  Normal ZFS Profile

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

  Actual ZFS Profile for Installing a Domain

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

◆ ◆ ◆ CHAPTER 5

Configuring I/O Domains

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

This chapter covers the following topics:

I/O Domain Overview

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

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

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

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

- Chapter 6, "Creating a Root Domain by Assigning PCIe Buses"
- Chapter 8, "Creating an I/O Domain by Using Direct I/O"
- Chapter 7, "Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions"
- Chapter 9, "Using Non-`primary` Root Domains"

Note – You cannot migrate a domain that has PCIe buses, PCIe endpoint devices, or SR-IOV virtual functions. For information about other migration limitations, see Chapter 13, "Migrating Domains."

General Guidelines for Creating an I/O Domain

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

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

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

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

- Assign CPU resources at the granularity of CPU cores. Assign one or more CPU cores based on the type of I/O device and the number of I/O devices in the I/O domain.

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

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

SPARC systems, up to and including the SPARC T5 and SPARC M6 platforms, provide a finite number of interrupts, so Oracle Solaris limits the number of interrupts that each device can use. The default limit should match the needs of a typical system configuration but you might need to adjust this value for certain system configurations. For more information, see "Adjusting the Interrupt Limit" on page 380.

◆ ◆ ◆ CHAPTER 6

Creating a Root Domain by Assigning PCIe Buses

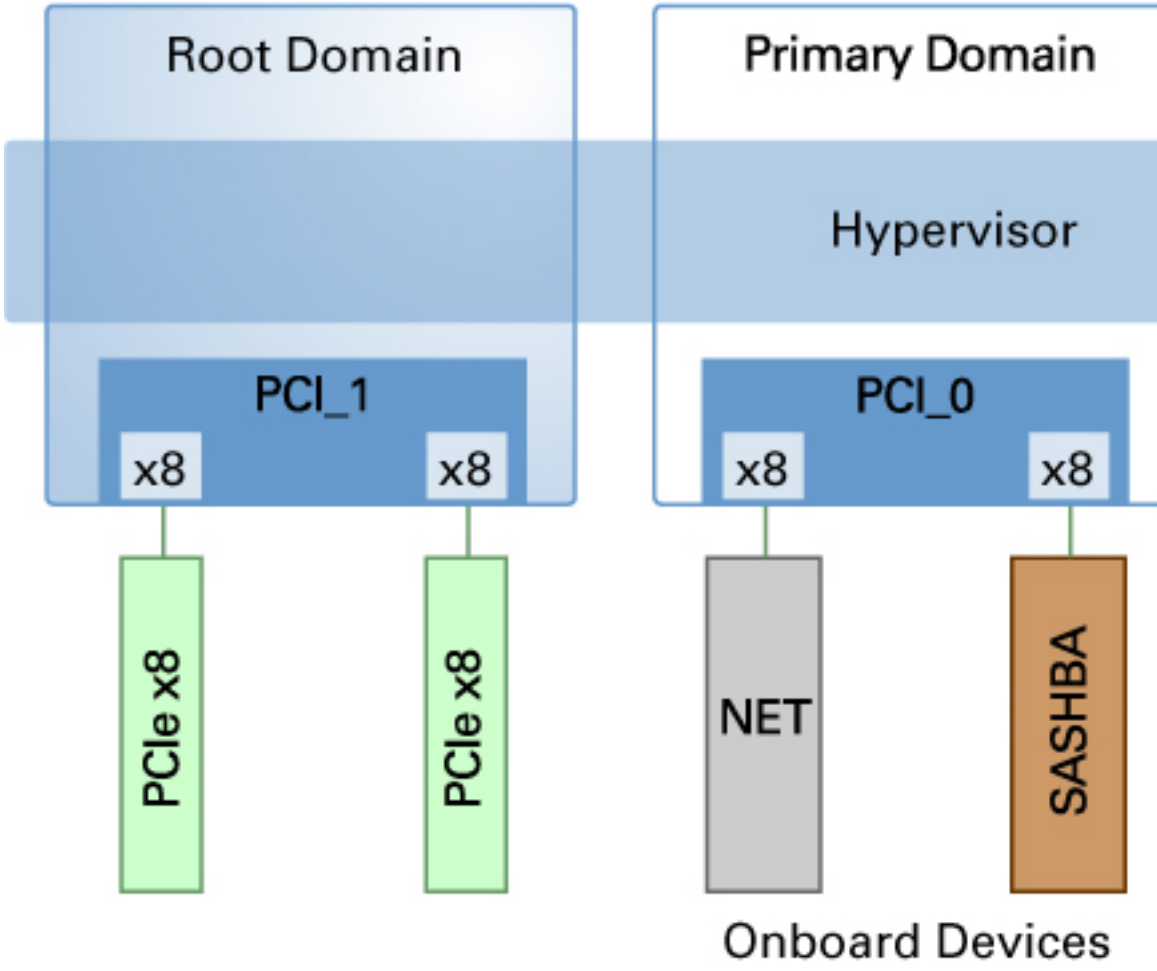
This chapter describes how to create a root domain by assigning PCIe buses.

Creating a Root Domain by Assigning PCIe Buses

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

The following diagram shows a system that has three root complexes, `pci_0`, `pci_1`, and `pci_2`.

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



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

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

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

Static PCIe Bus Assignment

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

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

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

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

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

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

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

Dynamic PCIe Bus Assignment

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

The dynamic PCIe bus assignment feature is enabled when your system runs the required firmware and software. See "Dynamic PCIe Bus Assignment Requirements" on page 71. If your system does not run the required firmware and software, the `ldm add-io` and `ldm remove-io` commands fail gracefully.

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

Dynamic PCIe Bus Assignment Requirements

The dynamic PCIe bus assignment feature is supported on SPARC M5 servers, SPARC M6 servers, SPARC M7 series servers, SPARC T7 series servers, and Fujitsu M10 servers that run the Oracle Solaris 11 OS in the root domain. SPARC M5 servers and SPARC M6 servers must run at

least the 9.4.2 version of the system firmware, SPARC T7 series servers and SPARC M7 series servers must run at least 9.4.3, and the Fujitsu M10 server must run at least XCP2240.

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

This example procedure shows how to create a new root domain from an initial configuration where several buses are owned by the `primary` domain. By default the `primary` domain owns all buses present on the system. This example is for a SPARC T4-2 server. This procedure can also be used on other servers. The instructions for different servers might vary slightly from these, but you can obtain the basic principles from this example.

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

Caution – All internal disks on the supported servers might be connected to a single PCIe bus. If a domain is booted from an internal disk, do not remove that bus from the domain.

Ensure that you do not remove a bus that has devices that are used by a domain, such as network ports or `usbecm` devices. If you remove the wrong bus, a domain might not be able to access the required devices and could become unusable. To remove a bus that has devices that are used by a domain, reconfigure that domain to use devices from other buses. For example, you might have to reconfigure the domain to use a different on-board network port or a PCIe card from a different PCIe slot.

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

In this example, the `primary` domain uses only a ZFS pool (`rpool`) and network interface (`igb0`). If the `primary` domain uses more devices, repeat Steps 2-4 for each device to ensure that none are located on the bus that will be removed.

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

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

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

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

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

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

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

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

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

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

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

3 Obtain information about the system's boot disk.

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

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

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

  ```
  primary# mpathadm show lu /dev/rdsk/c0t5000CCA03C138904d0s0
  Logical Unit:  /dev/rdsk/c0t5000CCA03C138904d0s2
          mpath-support:  libmpscsi_vhci.so
          Vendor:  HITACHI
          Product:  H106030SDSUN300G
  ```

```
Revision:  A2B0
Name Type:  unknown type
Name:  5000cca03c138904
Asymmetric:  no
Current Load Balance:  round-robin
Logical Unit Group ID:  NA
Auto Failback:  on
Auto Probing:  NA

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

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

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

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

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

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

  The following example uses block device `c1t0d0s0`:

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

  In this example, the physical device for the `primary` domain's boot disk is connected to the `pci@400` bus.

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

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

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

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

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

The following command uses the `igb0` network interface:

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

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

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

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

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

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

- **Dynamic method:**

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

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

- **Static method:**

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

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

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

7 Add a bus to a domain.

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

- **Dynamic method:**

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

- **Static method:**

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

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

8 Save this configuration to the service processor.

In this example, the configuration is `io-domain`.

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

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

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

```
primary# ldm list-io
NAME                                      TYPE   BUS      DOMAIN   STATUS
----                                      ----   ---      ------   ------
pci_0                                     BUS    pci_0    primary
pci_1                                     BUS    pci_1    primary
pci_2                                     BUS    pci_2    ldg1
pci_3                                     BUS    pci_3    primary
/SYS/MB/PCIE1                             PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA0                           PCIE   pci_0    primary  OCC
/SYS/MB/NET0                              PCIE   pci_0    primary  OCC
/SYS/MB/PCIE5                             PCIE   pci_1    primary  EMP
/SYS/MB/PCIE6                             PCIE   pci_1    primary  EMP
/SYS/MB/PCIE7                             PCIE   pci_1    primary  EMP
/SYS/MB/PCIE2                             PCIE   pci_2    ldg1     EMP
/SYS/MB/PCIE3                             PCIE   pci_2    ldg1     EMP
/SYS/MB/PCIE4                             PCIE   pci_2    ldg1     EMP
/SYS/MB/PCIE8                             PCIE   pci_3    primary  EMP
/SYS/MB/SASHBA1                           PCIE   pci_3    primary  OCC
/SYS/MB/NET2                              PCIE   pci_3    primary  OCC
/SYS/MB/NET0/IOVNET.PF0                   PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF1                   PF     pci_0    primary
/SYS/MB/NET2/IOVNET.PF0                   PF     pci_3    primary
/SYS/MB/NET2/IOVNET.PF1                   PF     pci_3    primary
```

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

◆ ◆ ◆ CHAPTER 7

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

This chapter covers the following PCIe SR-IOV topics:

SR-IOV Overview

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

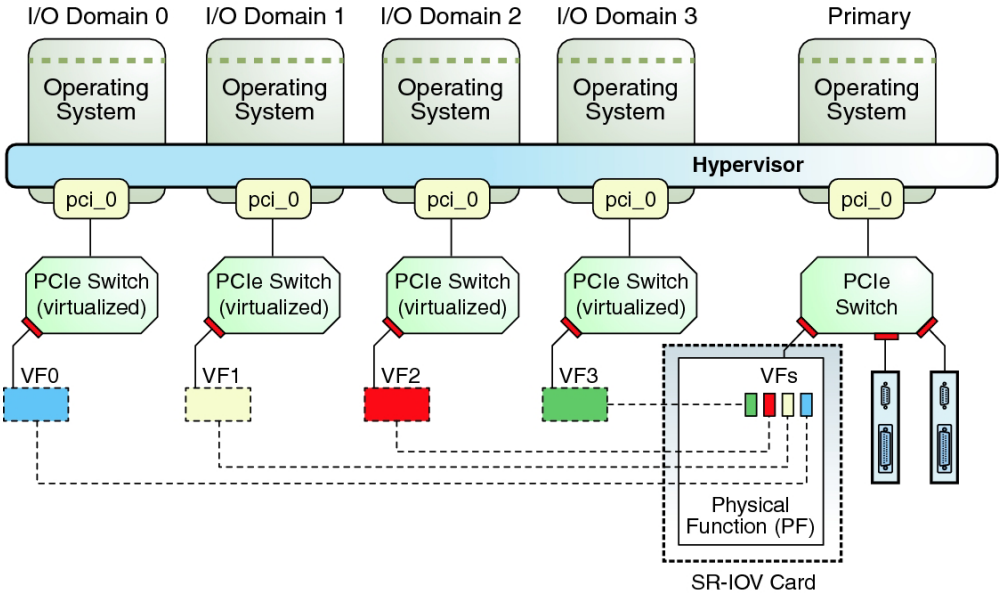
The Peripheral Component Interconnect Express (PCIe) single root I/O virtualization (SR-IOV) implementation is based on version 1.1 of the standard as defined by the PCI-SIG. The SR-IOV standard enables the efficient sharing of PCIe devices among virtual machines and is implemented in the hardware to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

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

For more information about SR-IOV, see the PCI-SIG web site (`http://www.pcisig.com/`).

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

FIGURE 7–1 Using Virtual Functions and a Physical Function in an I/O Domain



SR-IOV has the following function types:

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

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

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

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

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

The Oracle VM Server for SPARC SR-IOV implementation includes both static and dynamic configuration methods. For more information, see "Static SR-IOV" on page 84 and "Dynamic SR-IOV" on page 85.

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

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

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

Note – On SPARC M7 series servers, SPARC T7 series servers, and Fujitsu M10 servers, PCIe buses are enabled for I/O virtualization by default.

Assigning a SR-IOV virtual function to a domain creates an implicit dependency on the domain providing the SR-IOV physical function service. You can view these dependencies or view domains that depend on this SR-IOV physical function by using the `ldm list-dependencies` command. See "Listing Domain I/O Dependencies" on page 382.

SR-IOV Hardware and Software Requirements

The dynamic and static PCIe SR-IOV features are supported on the SPARC T4 server, SPARC T5 server, SPARC T7 series server, SPARC M5 server, SPARC M6 server, and SPARC M7 series server. The dynamic feature is supported on Fujitsu M10 platforms for Ethernet devices only while the other device types require that you use the static method. The SPARC T3 platform supports only the static PCIe SR-IOV feature.

- **Hardware Requirements.**

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

  - **Ethernet SR-IOV.** To use the SR-IOV feature, you can use on-board PCIe SR-IOV devices as well as PCIe SR-IOV plug-in cards. All on-board SR-IOV devices in a given platform are supported unless otherwise explicitly stated in the platform documentation.
  - **InfiniBand SR-IOV.** InfiniBand devices are supported on the SPARC T4 server, SPARC T5 server, SPARC T7 series server, SPARC M5 server, SPARC M6 server, SPARC M7 series server, and Fujitsu M10 server.
  - **Fibre Channel SR-IOV.** Fibre Channel devices are supported on the SPARC T4 server, SPARC T5 server, SPARC T7 series server, SPARC M5 server, SPARC M6 server, SPARC M7 series server, and Fujitsu M10 server.

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

- **Firmware Requirements.**
  - **Ethernet SR-IOV.** To use the dynamic SR-IOV feature, SPARC T4 server must run at least version 8.4.0.a of the system firmware. SPARC T5 servers, SPARC M5 servers, and SPARC M6 servers must run at least version 9.1.0.a of the system firmware. SPARC T7 series servers and SPARC M7 series servers must run at least version 9.4.3 of the system firmware. Fujitsu M10 servers must run at least version XCP2210 of the system firmware. The SPARC T3 server supports only the static SR-IOV feature.

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

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

       Perform these commands from the `ok` prompt:

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

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

       `LP` Sun Dual 10GbE SFP+ PCIe 2.0 LP FCode 3.01 4/2/2012

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

       `FEM` Sun Dual 10GbE SFP+ PCIe 2.0 FEM FCode 3.01 4/2/2012

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

    3. Install the patch.

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

  - **InfiniBand SR-IOV.** To use this feature, your system must run at least the following version of the system firmware:
    - **SPARC T4 Servers** – 8.4
    - **SPARC T5 Servers** – 9.1.0.*x*
    - **SPARC T7 Series Servers** – 9.4.3
    - **SPARC M5 and SPARC M6 Servers** – 9.1.0.*x*
    - **SPARC M7 Series Servers** – 9.4.3
    - **Fujitsu M10 Server** – XCP2210

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

    - **Low Profile (X4242A)** – Patch ID 16340059

- **Express Module (X4243A)** – Patch ID 16340042

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

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

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

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

- **Fibre Channel SR-IOV.** To use this feature, your system must run at least the following version of the system firmware:
  - **SPARC T4 Server** – 8.4.2.c
  - **SPARC T5 Server** – 9.1.2.d
  - **SPARC T7 Series Server** – 9.4.3
  - **SPARC M5 Server** – 9.1.2.d
  - **SPARC M6 Server** – 9.1.2.d
  - **SPARC M7 Series Server** – 9.4.3
  - **Fujitsu M10 Server** – XCP2210

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

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

- **Software Requirements.**
  - **Ethernet SR-IOV.** To use the SR-IOV feature, all domains must be running at least the Oracle Solaris 11.1 SRU 10 OS.
  - **InfiniBand SR-IOV.** The following domains must run the supported Oracle Solaris OS:
    - The `primary` domain or a non-`primary` root domain must run at least the Oracle Solaris 11.1 SRU 10 OS.
    - The I/O domains must run at least the Oracle Solaris 11.1 SRU 10 OS.

- Update the `/etc/system` file on any root domain that has an InfiniBand SR-IOV physical function from which you plan to configure virtual functions.

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

  For information about correctly creating or updating `/etc/system` property values, see "Updating Property Values in the /etc/system File" on page 362.

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

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

- **Fibre Channel SR-IOV.** To use the SR-IOV feature, all domains must be running at least the Oracle Solaris 11.1 SRU 17 OS.

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

- "Static SR-IOV Software Requirements" on page 85
- "Dynamic SR-IOV Software Requirements" on page 86

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

- "Ethernet SR-IOV Hardware Requirements" on page 89
- "InfiniBand SR-IOV Hardware Requirements" on page 108
- "Fibre Channel SR-IOV Hardware Requirements" on page 121

Current SR-IOV Feature Limitations

The SR-IOV feature has the following limitations:

- An I/O domain cannot start if any associated root domain is not running.
- Migration is disabled for any domain that has one or more virtual functions assigned to it.
- You can destroy only the last virtual function that was created for a physical function. So, if you create three virtual functions, the first virtual function that you can destroy must be the third one.
- If an SR-IOV card is assigned to a domain by using the Direct I/O (DIO) feature, the SR-IOV feature is not enabled for that card.
- The PCIe endpoint devices and SR-IOV virtual functions from a particular PCIe bus can be assigned up to a maximum of 15 domains on supported SPARC T-Series and SPARC M-Series systems. On SPARC T7 series servers and SPARC M7 series servers, you can assign PCIe endpoint devices and SR-IOV virtual functions from a particular PCIe bus to a maximum of 31 domains. On a Fujitsu M10 server you can assign PCIe endpoint devices and SR-IOV virtual functions from a particular PCIe bus to a maximum of 24 domains. The

PCIe resources, such as interrupt vectors for each PCIe bus, are divided among the root domain and I/O domains. As a result, the number of devices that you can assign to a particular I/O domain is also limited. Make sure that you do not assign a large number virtual functions to the same I/O domain. There is no interrupt limitation for the SPARC T7 series servers and SPARC M7 series servers. For a description of the problems related to SR-IOV, see *Oracle VM Server for SPARC 3.3 Release Notes*.

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

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

  If the I/O domain is resilient, it can continue to operate even if the root domain that is the owner of the PCIe bus becomes unavailable. See "I/O Domain Resiliency" on page 134.
- SPARC systems, up to and including the SPARC T5 and SPARC M6 platforms, provide a finite number of interrupts, so Oracle Solaris limits the number of interrupts that each device can use. The default limit should match the needs of a typical system configuration but you might need to adjust this value for certain system configurations. For more information, see "Adjusting the Interrupt Limit" on page 380.

Static SR-IOV

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

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

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

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

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

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

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

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

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

Static SR-IOV Software Requirements

The static SR-IOV features are supported by at least the Oracle VM Server for SPARC 3.0 software and firmware. See "PCIe SR-IOV Hardware and Software Requirements" in *Oracle VM Server for SPARC 3.0 Release Notes* .

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

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

Dynamic SR-IOV

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

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

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

Dynamic SR-IOV Software Requirements

For information about the required PCIe SR-IOV software and firmware versions, see "SR-IOV Hardware and Software Requirements" on page 80.

Note – If your system does not meet the dynamic SR-IOV software and firmware requirements, you must use the static SR-IOV method to perform SR-IOV-related tasks. See "Static SR-IOV" on page 84.

Dynamic SR-IOV Configuration Requirements

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

- I/O virtualization has been enabled for a PCIe bus before you begin to configure virtual functions.
- The OS that runs on the root domain and on I/O domains must be at least the Oracle Solaris 11.1 SRU 10 OS.
- The physical function device is not configured in the OS or is in a multipathing configuration. For example, you can unplumb an Ethernet SR-IOV device or have it in an IPMP or an aggregation to successfully create or destroy a virtual function.

  An operation to create or destroy a virtual function requires that the physical function device driver toggle between the offline and online states. A multipathing configuration permits the device driver to toggle between these states.
- The virtual function is either not in use or in a multipathing configuration before you remove a virtual function from an I/O domain. For example, you can either unplumb an Ethernet SR-IOV virtual function or not use it in an IPMP configuration.

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

Enabling I/O Virtualization

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

Note – On SPARC M7 series servers, SPARC T7 series servers, and Fujitsu M10 servers, PCIe buses are enabled for I/O virtualization by default.

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

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

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

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

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

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

   Run one of the following commands:

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

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

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

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

3. **Reboot the root domain.**

   Run one of the following commands:

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

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

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

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

Planning for the Use of PCIe SR-IOV Virtual Functions

Plan ahead to determine how you want to use virtual functions in your configuration. Determine which virtual functions from the SR-IOV devices will satisfy your current and future configuration needs.

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

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

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

For information about I/O domains, see "General Guidelines for Creating an I/O Domain" on page 68.

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

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

   Use the following commands to identify the required information:

`ldm list-io`	Identifies the available SR-IOV physical function devices.
`prtdiag -v`	Identifies which PCIe SR-IOV cards and on-board devices are available.
`ldm list-io -l` *pf-name*	Identifies additional information about a specified physical function, such as the maximum number of virtual functions that are supported by the device.
`ldm list-io -d` *pf-name*	Identifies the device-specific properties that are supported by the device. See "Advanced SR-IOV Topics: Ethernet SR-IOV" on page 101.

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

   See "How to Enable I/O Virtualization for a PCIe Bus" on page 87.

3. Create the required number of virtual functions on the specified SR-IOV physical function.

   Use the following command to create the virtual functions for the physical function:

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

For more information, see "How to Create an Ethernet SR-IOV Virtual Function" on page 90, "How to Create an InfiniBand Virtual Function" on page 108, and "How to Create a Fibre Channel SR-IOV Virtual Function" on page 124.

4. Use the `ldm add-config` command to save the configuration to the SP.

   For more information, see "How to Add an Ethernet SR-IOV Virtual Function to an I/O Domain" on page 99, "How to Add an InfiniBand Virtual Function to an I/O Domain" on page 113, and "How to Add a Fibre Channel SR-IOV Virtual Function to an I/O Domain" on page 131.

Using Ethernet SR-IOV Virtual Functions

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

Ethernet SR-IOV Hardware Requirements

For information about the required PCIe Ethernet SR-IOV hardware, see "SR-IOV Hardware and Software Requirements" on page 80.

Ethernet SR-IOV Limitations

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

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

Planning for the Use of Ethernet SR-IOV Virtual Functions

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

If you cannot use multipathing or must plumb the physical function, use the static method to create the virtual functions. See "Static SR-IOV" on page 84.

Ethernet Device-Specific and Network-Specific Properties

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

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

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

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

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

Creating Ethernet Virtual Functions

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

▼ How to Create an Ethernet SR-IOV Virtual Function

If you cannot use this dynamic method, use the static method instead. See "Static SR-IOV" on page 84.

1 Identify the physical function device.

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

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

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

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

See "How to Enable I/O Virtualization for a PCIe Bus" on page 87.

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

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

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

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

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

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

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

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

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

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

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

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

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

- **Static method:**

  a. **Initiate a delayed reconfiguration.**

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

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

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

  c. **Reboot the root domain.**

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

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

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

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

Example 7–1 Displaying Information About the Ethernet Physical Function

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

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

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

```
/SYS/MB/NET2/IOVNET.PF0                    PF     pci_1    primary
/SYS/MB/NET2/IOVNET.PF1                    PF     pci_1    primary
```

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

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

Example 7–2 Dynamically Creating an Ethernet Virtual Function Without Setting Optional Properties

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

Ensure that I/O virtualization is enabled on the `pci_0` PCIe bus. See "How to Enable I/O Virtualization for a PCIe Bus" on page 87.

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

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

Example 7–3 Dynamically Creating an Ethernet Virtual Function and Setting Properties

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

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

Example 7–4 Dynamically Creating an Ethernet Virtual Function With Two Alternate MAC Addresses

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

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

Example 7–5 Statically Creating a Virtual Function Without Setting Optional Properties

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

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

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

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

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

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

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

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

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

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

Example 7–6 Creating Multiple SR-IOV Ethernet Virtual Functions

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

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

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

Destroying Ethernet Virtual Functions

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

How to Destroy an Ethernet SR-IOV Virtual Function

If you cannot use this dynamic method, use the static method instead. See "Static SR-IOV" on page 84.

1. **Identify the physical function device.**

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

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

   - **Dynamic method:**

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

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

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

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

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

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

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

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

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

- **Static method:**

  a. **Initiate a delayed reconfiguration.**

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

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

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

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

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

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

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

  c. **Reboot the root domain.**

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

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

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

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

Example 7–7 Destroying an Ethernet Virtual Function

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

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

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

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

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

Example 7–8 Destroying Multiple Ethernet SR-IOV Virtual Functions

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

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

Modifying Ethernet SR-IOV Virtual Functions

The `ldm set-io` *vf-name* command modifies the current configuration of a virtual function by changing the property values or by setting new properties. This command can modify both the network-specific properties and the device-specific properties. For information about device-specific properties, see "Advanced SR-IOV Topics: Ethernet SR-IOV" on page 101.

If you cannot use this dynamic method, use the static method instead. See "Static SR-IOV" on page 84.

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

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

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

- `pvid` and `vid`

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

- **Device-specific properties**

Use the `ldm list-io -d` *pf-name* command to view the list of valid device-specific properties. You can modify these properties for both the physical function and the virtual function. You must use the static method to modify device-specific properties. See "Static SR-IOV" on page 84. For more information about device-specific properties, see "Advanced SR-IOV Topics: Ethernet SR-IOV" on page 101.

▼ How to Modify Ethernet SR-IOV Virtual Function Properties

1 Identify the physical function device.

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

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

2 Modify a virtual function property.

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

Example 7–9 Modifying Ethernet Virtual Function Properties

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

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

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

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

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

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

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

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

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

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

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

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

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

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

If you cannot dynamically remove the virtual function, use the static method. See "Static SR-IOV" on page 84.

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

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

2 Add a virtual function dynamically or statically.

- **To dynamically add a virtual function:**

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

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

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

- **To statically add a virtual function:**

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

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

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

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

b. Reboot the root domain.

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

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

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

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

Example 7–10 Adding an Ethernet Virtual Function

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

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

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

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

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

If you cannot dynamically remove the virtual function, use the static method. See "Static SR-IOV" on page 84.

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

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

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

2 Remove a virtual function either dynammically or statically.

- **To dynamically remove a virtual function:**

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

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

- **To statically remove a virtual function:**

  a. **Stop the I/O domain.**

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

  b. **Remove the virtual function.**

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

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

  c. **Start the I/O domain.**

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

Example 7–11 Dynamically Removing an Ethernet Virtual Function

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

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

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

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

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

Advanced SR-IOV Topics: Ethernet SR-IOV

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

Advanced Network Configuration for Virtual Functions

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

- SR-IOV virtual functions can only use the MAC addresses that are assigned by the Logical Domains Manager. If you use other Oracle Solaris OS networking commands to change the MAC address on the I/O domain, the commands might fail or might not function properly.
- At this time, link aggregation of SR-IOV network virtual functions in the I/O domain is not supported. If you attempt to create a link aggregation, it might not function as expected.

- You can create virtual I/O services and assign them to I/O domains. These virtual I/O services can be created on the same physical function from which virtual functions are also created. For example, you can use an on-board 1-Gbps network device (`net0` or `igb0`) as a network back-end device for a virtual switch and also statically create virtual functions from the same physical function device.

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

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

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

SR-IOV Device-Specific Properties

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

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

`P` Applies to a physical function

`V` Applies to a virtual function

`R` Read-only or informative parameter only

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

Use the `ldm create-vf` or `ldm set-io` command to set the read-write properties for a physical function or a virtual function. Note that to set a device-specific property, you must use the static method. See "Static SR-IOV" on page 84.

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

```
primary# ldm list-io -d /SYS/MB/NET0/IOVNET.PF0
Device-specific Parameters
--------------------------
```

```
max-config-vfs
    Flags = PR
    Default = 7
    Descr = Max number of configurable VFs
max-vf-mtu
    Flags = VR
    Default = 9216
    Descr = Max MTU supported for a VF
max-vlans
    Flags = VR
    Default = 32
    Descr = Max number of VLAN filters supported
pvid-exclusive
    Flags = VR
    Default = 1
    Descr = Exclusive configuration of pvid required
unicast-slots
    Flags = PV
    Default = 0 Min = 0 Max = 24
    Descr = Number of unicast mac-address slots
```

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

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

Creating VNICs on SR-IOV Virtual Functions

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

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

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

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

```
primary# ldm start ldg1
```

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

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

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

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

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

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

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

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

Before You Begin Before you begin, ensure that you have enabled I/O virtualization for the PCIe bus that is the parent of the physical function from which you create virtual functions. See "How to Enable I/O Virtualization for a PCIe Bus" on page 87.

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

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

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

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

You can run this command for each virtual function that you want to create. You can also use the `-n` option to create more than one virtual function from the same physical function in a single command. See Example 7–6 and the `ldm`(1M) man page.

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

---

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

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

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

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

Note – If the OS in the target I/O domain does not support dynamic SR-IOV, you must use the static method. See "Static SR-IOV" on page 84.

---

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

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

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

Example 7–12 Dynamically Creating an I/O Domain by Assigning an SR-IOV Virtual Function to It

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

This example assumes that the following circumstances are true:

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

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

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

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

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

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

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

Example 7–13 Statically Creating an I/O Domain by Assigning an SR-IOV Virtual Function to It

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

This example assumes that the following circumstances are true:

- The OS on the `primary` domain does not support dynamic SR-IOV operations
- The `pci_0` bus is assigned to the `primary` domain and has not been initialized for I/O virtualization operations
- The `/SYS/MB/NET0/IOVNET.PF0` physical function belongs to the `pci_0` bus
- The `/SYS/MB/NET0/IOVNET.PF0` physical function does not have any existing virtual functions assigned to domains
- The `ldg1` domain is active and booted and its OS does not support dynamic SR-IOV operations
- The `ldg1` domain has the `auto-boot?` property set to `true` so that the domain boots automatically when the domain is started

First, initiate a delayed reconfiguration on the `primary` domain, enable I/O virtualization, and create the virtual function from the `/SYS/MB/NET0/IOVNET.PF0` physical function.

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

primary# ldm set-io iov=on pci_0
primary# ldm create-vf /SYS/MB/NET0/IOVNET.PF0

--------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
--------------------------------------------------------------------------------
Created new vf: /SYS/MB/NET0/IOVNET.PF0.VF0
```

Next, shut down the `primary` domain.

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

Stop the `ldg1` domain, add the virtual function, and start the domain.

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

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

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

```
/SYS/MB/NET2/IOVNET.PF1       PF    pci_1    primary
/SYS/MB/NET0/IOVNET.PF0.VF0   VF    pci_0    ldg1
```

Using InfiniBand SR-IOV Virtual Functions

Only the static SR-IOV feature is supported for InfiniBand SR-IOV devices.

To minimize downtime, run all of the SR-IOV commands as a group while the root domain is in delayed reconfiguration or a guest domain is stopped. The SR-IOV commands that are limited in this way are the `ldm create-vf`, `ldm destroy-vf`, `ldm add-io`, and `ldm remove-io` commands.

Typically, virtual functions are assigned to more than one guest domain. A reboot of the root domain affects all of the guest domains that have been assigned the root domain's virtual functions.

Because an unused InfiniBand virtual function has very little overhead, you can avoid downtime by creating the necessary virtual functions ahead of time, even if they are not used immediately.

InfiniBand SR-IOV Hardware Requirements

For information about the required PCIe InfiniBand SR-IOV hardware, see "SR-IOV Hardware and Software Requirements" on page 80.

For InfiniBand SR-IOV support, the root domain must be running at least the Oracle Solaris 11.1 SRU 10 OS. The I/O domains can run at least the Oracle Solaris 11.1 SRU 10 OS.

Creating and Destroying InfiniBand Virtual Functions

▼ How to Create an InfiniBand Virtual Function

This procedure describes how to create an InfiniBand SR-IOV virtual function.

1 Initiate a delayed reconfiguration on the root domain.

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

2 Enable I/O virtualization by setting `iov=on`.

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

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

3. **Create one or more virtual functions that are associated with the physical functions from that root domain.**

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

   You can run this command for each virtual function that you want to create. You can also use the `-n` option to create more than one virtual function from the same physical function in a single command. See Example 7–6 and the `ldm`(1M) man page.

4. **Reboot the root domain.**

   Run one of the following commands:

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

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

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

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

Example 7–14 Creating an InfiniBand Virtual Function

The following example shows information about the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0` physical function:

- This physical function is in PCIE slot 4.
- The `IOVIB` string indicates that the physical function is an InfiniBand SR-IOV device.

```
primary# ldm list-io
NAME                                      TYPE   BUS      DOMAIN   STATUS
----                                      ----   ---      ------   ------
pci_0                                     BUS    pci_0    primary
niu_0                                     NIU    niu_0    primary
/SYS/MB/RISER0/PCIE0                      PCIE   pci_0    primary  EMP
/SYS/MB/RISER1/PCIE1                      PCIE   pci_0    primary  EMP
/SYS/MB/RISER2/PCIE2                      PCIE   pci_0    primary  EMP
/SYS/MB/RISER0/PCIE3                      PCIE   pci_0    primary  OCC
/SYS/MB/RISER1/PCIE4                      PCIE   pci_0    primary  OCC
/SYS/MB/RISER2/PCIE5                      PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA0                           PCIE   pci_0    primary  OCC
/SYS/MB/SASHBA1                           PCIE   pci_0    primary  OCC
/SYS/MB/NET0                              PCIE   pci_0    primary  OCC
/SYS/MB/NET2                              PCIE   pci_0    primary  OCC
/SYS/MB/RISER0/PCIE3/IOVIB.PF0            PF     pci_0    primary
/SYS/MB/RISER1/PCIE4/IOVIB.PF0            PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF0                   PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF1                   PF     pci_0    primary
/SYS/MB/NET2/IOVNET.PF0                   PF     pci_0    primary
/SYS/MB/NET2/IOVNET.PF1                   PF     pci_0    primary
```

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

```
primary# ldm list-io -l /SYS/MB/RISER1/PCIE4/IOVIB.PF0
NAME                                    TYPE   BUS      DOMAIN   STATUS
----                                    ----   ---      ------   ------
/SYS/MB/RISER1/PCIE4/IOVIB.PF0          PF     pci_0    primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0]
    maxvfs = 64
```

The following example shows how to create a static virtual function. First, initiate a delayed reconfiguration on the `primary` domain and enable I/O virtualization on the `pci_0` PCIe bus. Because the `pci_0` bus has been assigned already to the `primary` root domain, use the `ldm set-io` command to enable I/O virtualization.

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

primary# ldm set-io iov=on pci_0
--------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
--------------------------------------------------------------------------
```

Now, use the `ldm create-vf` command to create a virtual function from the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0` physical function.

```
primary# ldm create-vf /SYS/MB/RISER1/PCIE4/IOVIB.PF0
--------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
--------------------------------------------------------------------------
Created new vf: /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0
```

Note that you can create more than one virtual function during the same delayed reconfiguration. The following command creates a second virtual function:

```
primary# ldm create-vf /SYS/MB/RISER1/PCIE4/IOVIB.PF0
--------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
--------------------------------------------------------------------------
Created new vf: /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1
```

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

```
primary# shutdown -i6 -g0 -y
Shutdown started.

Changing to init state 6 - please wait
...
```

How to Destroy an InfiniBand Virtual Function

This procedure describes how to destroy an InfiniBand SR-IOV virtual function.

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

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

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

2. **Destroy one or more virtual functions that are associated with the physical functions from that root domain.**

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

   You can run this command for each virtual function that you want to destroy. You can also use the `-n` option to destroy more than one virtual function from the same physical function in a single command. See Example 7–8 and the `ldm`(1M) man page.

3. **Reboot the root domain.**

   Run one of the following commands:

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

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

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

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

Example 7–15 Destroying an InfiniBand Virtual Function

The following example shows how to destroy a static InfiniBand virtual function, `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1`.

The `ldm list-io` command shows information about the buses, physical functions, and virtual functions.

```
primary# ldm list-io
NAME                                    TYPE   BUS      DOMAIN STATUS
----                                    ----   ---      ------ ------
pci_0                                   BUS    pci_0    primary  IOV
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0          PF     pci_0    primary
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0      VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1      VF     pci_0
```

You can obtain more details about the physical function and related virtual functions by using the `ldm list-io -l` command.

```
primary# ldm list-io -l /SYS/MB/RISER1/PCIE4/IOVIB.PF0
NAME                                     TYPE   BUS      DOMAIN STATUS
----                                     ----   ---      ------ ------
/SYS/MB/RISER1/PCIE4/IOVIB.PF0           PF     pci_0    primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0]
    maxvfs = 64
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0       VF     pci_0
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,1]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1       VF     pci_0
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,2]
```

A virtual function can be destroyed only if it is unassigned to a domain. The DOMAIN column of the `ldm list-io -l` output shows the name of any domain to which a virtual function is assigned. Also, virtual functions must be destroyed in the reverse order of their creation. Therefore, in this example, you must destroy the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1` virtual function before you can destroy the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0` virtual function.

After you identify the proper virtual function, you can destroy it. First, initiate a delayed reconfiguration.

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

primary# ldm destroy-vf /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1
------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
------------------------------------------------------------------------
```

You can issue more than one `ldm destroy-vf` command while in delayed reconfiguration. Thus, you could also destroy the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0`.

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

```
primary# shutdown -i6 -g0 -y
Shutdown started.

Changing to init state 6 - please wait
...
```

Adding and Removing InfiniBand Virtual Functions on I/O Domains

▼ How to Add an InfiniBand Virtual Function to an I/O Domain

This procedure describes how to add an InfiniBand SR-IOV virtual function to an I/O domain.

1 Stop the I/O domain.

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

2 Add one or more virtual functions to the I/O domain.

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

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

3 Start the I/O domain.

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

Example 7–16 Adding an InfiniBand Virtual Function

The following example shows how to add the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2` virtual function to the `iodom1` I/O domain.

First, identify the virtual function that you want to assign.

```
primary# ldm list-io
NAME                                    TYPE   BUS      DOMAIN STATUS
----                                    ----   ---      ------ ------
pci_0                                   BUS    pci_0    primary  IOV
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0          PF     pci_0    primary
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0      VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1      VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2      VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3      VF     pci_0
```

To add a virtual function to an I/O domain, it must be unassigned. The DOMAIN column indicates the name of the domain to which the virtual function is assigned. In this case, the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2` is not assigned to a domain.

To add a virtual function to a domain, the domain must be in the inactive or bound state.

```
primary# ldm list-domain
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
primary          active     -n-cv-  UART    32    64G      0.2%  0.2%  56m
iodom1           active     -n----  5000    8     8G        33%   33%  25m
```

The `ldm list-domain` output shows that the `iodom1` I/O domain is active, so it must be stopped.

```
primary# ldm stop iodom1
LDom iodom1 stopped
primary# ldm list-domain
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
primary          active     -n-cv-  UART    32    64G      0.0%  0.0%  57m
iodom1           bound      ------  5000    8     8G
```

Now you can add the virtual function to the I/O domain.

```
primary# ldm add-io /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2 iodom1
primary# ldm list-io
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2        VF     pci_0    iodom1
```

Note that you can add more than one virtual function while an I/O domain is stopped. For example, you might add other unassigned virtual functions such as `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3` to `iodom1`. After you add the virtual functions, you can restart the I/O domain.

```
primary# ldm start iodom1
LDom iodom1 started
primary# ldm list-domain
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
primary          active     -n-cv-  UART    32    64G      1.0%  1.0%  1h 18m
iodom1           active     -n----  5000    8     8G        36%   36%  1m
```

▼ How to Remove an InfiniBand Virtual Function From an I/O Domain

This procedure describes how to remove an InfiniBand SR-IOV virtual function from an I/O domain.

1 Stop the I/O domain.

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

2 Remove one or more virtual functions from the I/O domain.

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

Note – Before removing the virtual function from the I/O domain, ensure that it is not critical for booting that domain.

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

3 **Start the I/O domain.**

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

Example 7–17 Removing an InfiniBand Virtual Function

The following example shows how to remove the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2` virtual function from the `iodom1` I/O domain.

First, identify the virtual function that you want to remove.

```
primary# ldm list-io
NAME                                    TYPE   BUS      DOMAIN STATUS
----                                    ----   ---      ------ ------
pci_0                                   BUS    pci_0    primary  IOV
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0          PF     pci_0    primary
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0      VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1      VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2      VF     pci_0    iodom1
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3      VF     pci_0    iodom1
```

The DOMAIN column shows the name of the domain to which the virtual function is assigned. The `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2` virtual function is assigned to `iodom1`.

To remove a virtual function from an I/O domain, the domain must be inactive or bound state. Use the `ldm list-domain` command to determine the state of the domain.

```
primary# ldm list-domain
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
primary          active     -n-cv-  UART    32    64G      0.3%  0.3%  29m
iodom1           active     -n----  5000    8     8G        17%   17%  11m
```

In this case, the `iodom1` domain is active and so must be stopped.

```
primary# ldm stop iodom1
LDOM iodom1 stopped
primary# ldm list-domain
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
primary          active     -n-cv-  UART    32    64G      0.0%  0.0%  31m
iodom1           bound      ------  5000    8     8G
```

Now you can remove the /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2 virtual function from iodom1.

```
primary# ldm remove-io /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2 iodom1
primary# ldm list-io
NAME                                    TYPE   BUS      DOMAIN STATUS
----                                    ----   ---      ------ ------
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2      VF     pci_0
...
```

Note that the DOMAIN column for the virtual function is now empty.

You can remove more than one virtual function while an I/O domain is stopped. In this example, you could also remove the /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3 virtual function. After you remove the virtual functions, you can restart the I/O domain.

```
primary# ldm start iodom1
LDom iodom1 started
primary# ldm list-domain
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
primary          active     -n-cv-  UART    32    64G      0.3%  0.3%  39m
iodom1           active     -n----  5000    8     8G       9.4%  9.4%  5s
```

Adding and Removing InfiniBand Virtual Functions to Root Domains

How to Add an InfiniBand Virtual Function to a Root Domain

This procedure describes how to add an InfiniBand SR-IOV virtual function to a root domain.

1. **Initiate a delayed reconfiguration.**

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

2. **Add one or more virtual functions to the root domain.**

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

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

3. **Reboot the root domain.**

   Run one of the following commands:

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

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

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

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

▼ How to Remove an InfiniBand Virtual Function From a Root Domain

This procedure describes how to remove an InfiniBand SR-IOV virtual function from a root domain.

1. **Initiate a delayed reconfiguration.**

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

2. **Remove one or more virtual functions from the root domain.**

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

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

3. **Reboot the root domain.**

   Run one of the following commands:

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

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

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

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

Advanced SR-IOV Topics: InfiniBand SR-IOV

This section describes how to identify InfiniBand physical and virtual functions as well as to correlate the Logical Domains Manager and the Oracle Solaris view of InfiniBand physical and virtual functions.

Listing InfiniBand SR-IOV Virtual Functions

The following example shows different ways to display information about the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0` physical function. A physical function name that includes the `IOVIB` string indicates that it is an InfiniBand SR-IOV device.

```
primary# ldm list-io
NAME                                      TYPE   BUS      DOMAIN   STATUS
----                                      ----   ---      ------   ------
pci_0                                     BUS    pci_0    primary  IOV
niu_0                                     NIU    niu_0    primary
/SYS/MB/RISER0/PCIE0                      PCIE   pci_0    primary  EMP
```

```
/SYS/MB/RISER1/PCIE1                    PCIE   pci_0    primary  EMP
/SYS/MB/RISER2/PCIE2                    PCIE   pci_0    primary  EMP
/SYS/MB/RISER0/PCIE3                    PCIE   pci_0    primary  OCC
/SYS/MB/RISER1/PCIE4                    PCIE   pci_0    primary  OCC
/SYS/MB/RISER2/PCIE5                    PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA0                         PCIE   pci_0    primary  OCC
/SYS/MB/SASHBA1                         PCIE   pci_0    primary  OCC
/SYS/MB/NET0                            PCIE   pci_0    primary  OCC
/SYS/MB/NET2                            PCIE   pci_0    primary  OCC
/SYS/MB/RISER0/PCIE3/IOVIB.PF0          PF     pci_0    primary
/SYS/MB/RISER1/PCIE4/IOVIB.PF0          PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF0                 PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF1                 PF     pci_0    primary
/SYS/MB/NET2/IOVNET.PF0                 PF     pci_0    primary
/SYS/MB/NET2/IOVNET.PF1                 PF     pci_0    primary
/SYS/MB/RISER0/PCIE3/IOVIB.PF0.VF0      VF     pci_0    primary
/SYS/MB/RISER0/PCIE3/IOVIB.PF0.VF1      VF     pci_0    primary
/SYS/MB/RISER0/PCIE3/IOVIB.PF0.VF2      VF     pci_0    iodom1
/SYS/MB/RISER0/PCIE3/IOVIB.PF0.VF3      VF     pci_0    iodom1
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0      VF     pci_0    primary
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1      VF     pci_0    primary
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2      VF     pci_0    iodom1
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3      VF     pci_0    iodom1
```

The `ldm list-io -l` command provides more detailed information about the specified physical function device, `/SYS/MB/RISER1/PCIE4/IOVIB.PF0`. The `maxvfs` value shows that the maximum number of virtual functions supported by the physical device is 64. For each virtual function that is associated with the physical function, the output shows the following:

- Function name
- Function type
- Bus name
- Domain name
- Optional status of the function
- Device path

This `ldm list-io -l` output shows that `VF0` and `VF1` are assigned to the `primary` domain and that `VF2` and `VF3` are assigned to the `iodom1` I/O domain.

```
primary# ldm list-io -l /SYS/MB/RISER1/PCIE4/IOVIB.PF0
NAME                                    TYPE   BUS      DOMAIN   STATUS
----                                    ----   ---      ------   ------
/SYS/MB/RISER1/PCIE4/IOVIB.PF0          PF     pci_0    primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0]
    maxvfs = 64
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0      VF     pci_0    primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,1]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1      VF     pci_0    primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,2]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2      VF     pci_0    iodom1
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,3]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3      VF     pci_0    iodom1
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,4]
```

Identifying InfiniBand SR-IOV Functions

This section describes how to identify the InfiniBand SR-IOV devices.

Use the `ldm list-io -l` command to show the Oracle Solaris device path name that is associated with each physical function and virtual function.

```
primary# ldm list-io -l /SYS/MB/RISER1/PCIE4/IOVIB.PF0
NAME                                    TYPE   BUS      DOMAIN   STATUS
----                                    ----   ---      ------   ------
/SYS/MB/RISER1/PCIE4/IOVIB.PF0          PF     pci_0    primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0]
    maxvfs = 64
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0      VF     pci_0    primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,1]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1      VF     pci_0    primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,2]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2      VF     pci_0    iodom1
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,3]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF3      VF     pci_0    iodom1
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,4]
```

Use the `dladm show-phys -L` command to match each IP over InfiniBand (IPoIB) instance to its physical card. For example, the following command shows which IPoIB instances use the card in slot `PCIE4`, which is the same card shown in the previous `ldm list-io -l` example.

```
primary# dladm show-phys -L | grep PCIE4
net5              ibp0          PCIE4/PORT1
net6              ibp1          PCIE4/PORT2
net19             ibp8          PCIE4/PORT1
net9              ibp9          PCIE4/PORT2
net18             ibp4          PCIE4/PORT1
net11             ibp5          PCIE4/PORT2
```

Each InfiniBand host channel adapter (HCA) device has a globally unique ID (GUID). There are also GUIDs for each port (typically there are two ports to an HCA). An InfiniBand HCA GUID uniquely identifies the adapter. The port GUID uniquely identifies each HCA port and plays a role similar to a network device's MAC address. These 16-hexadecimal digit GUIDs are used by InfiniBand management tools and diagnostic tools.

Use the `dladm show-ib` command to obtain GUID information about the InfiniBand SR-IOV devices. Physical functions and virtual functions for the same device have related HCA GUID values. The 11th hexadecimal digit of the HCA GUID shows the relationship between a physical function and its virtual functions. Note that leading zeros are suppressed in the HCAGUID and PORTGUID columns.

For example, physical function `PF0` has two virtual functions, `VF0` and `VF1`, which are assigned to the `primary` domain. The 11th hexadecimal digit of each virtual function is incremented by one from the related physical function. So, if the GUID for the `PF0` is `8`, the GUIDs for `VF0` and `VF1` will be `9` and `A`, respectively.

The following `dladm show-ib` command output shows that the `net5` and `net6` links belong to the physical function `PF0`. The `net19` and `net9` links belong to `VF0` of the same device while the `net18` and `net11` links belong to `VF1`.

```
primary# dladm show-ib
LINK         HCAGUID         PORTGUID        PORT STATE  PKEYS
net6         2128**0**001A17F56  21280001A17F58  2    up     FFFF
net5         2128**0**001A17F56  21280001A17F57  1    up     FFFF
net19        2129**0**001A17F56  14050000000001  1    up     FFFF
net9         2129**0**001A17F56  14050000000008  2    up     FFFF
net18        212**A**0001A17F56  14050000000002  1    up     FFFF
net11        212**A**0001A17F56  14050000000009  2    up     FFFF
```

The device in the following `dladm show-phys` output shows the relationship between the links and the underlying InfiniBand port devices (`ibp`*X*).

```
primary# dladm show-phys
LINK              MEDIA                STATE      SPEED  DUPLEX    DEVICE
...
net6              Infiniband           up         32000  unknown   ibp1
net5              Infiniband           up         32000  unknown   ibp0
net19             Infiniband           up         32000  unknown   ibp8
net9              Infiniband           up         32000  unknown   ibp9
net18             Infiniband           up         32000  unknown   ibp4
net11             Infiniband           up         32000  unknown   ibp5
```

Use the `ls -l` command to show the actual InfiniBand port (IB port) device paths. An IB port device is a child of a device path that is shown in the `ldm list-io -l` output. A physical function has a one-part unit address such as `pciex15b3,673c@0` while virtual functions have a two-part unit address, `pciex15b3,1002@0,2`. The second part of the unit address is one higher than the virtual function number. (In this case, the second component is `2`, so this device is virtual function `1`.) The following output shows that `/dev/ibp0` is a physical function and `/dev/ibp5` is a virtual function.

```
primary# ls -l /dev/ibp0
lrwxrwxrwx   1 root     root          83 Apr 18 12:02 /dev/ibp0 ->
../devices/pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0/hermon@0/ibport@1,0,ipib:ibp0
primary# ls -l /dev/ibp5
lrwxrwxrwx   1 root     root          85 Apr 22 23:29 /dev/ibp5 ->
../devices/pci@400/pci@1/pci@0/pci@0/pciex15b3,1002@0,2/hermon@3/ibport@2,0,ipib:ibp5
```

You can use the OpenFabrics `ibv_devices` command to view the OpenFabrics device name and the node (HCA) GUID. When virtual functions are present, the Type column indicates whether the function is physical or virtual.

```
primary# ibv_devices
device                 node GUID            type
------              ----------------        ----
mlx4_4              0002c90300a38910         PF
mlx4_5              0021280001a17f56         PF
mlx4_0              0002cb0300a38910         VF
mlx4_1              0002ca0300a38910         VF
```

```
mlx4_2              00212a0001a17f56        VF
mlx4_3              0021290001a17f56        VF
```

Using Fibre Channel SR-IOV Virtual Functions

An SR-IOV Fibre Channel host bus adapter (HBA) might have one or more ports each of which appears as SR-IOV physical function. You can identify Fibre Channel physical functions by the `IOVFC` string in its device name.

Each Fibre Channel physical function has unique port and node world-wide name (WWN) values that are provided by the card manufacturer. When you create virtual functions from a Fibre Channel physical function, the virtual functions behave like a Fibre Channel HBA device. Each virtual function must have a unique identity that is specified by the port WWN and node WWN of the SAN fabric. You can use the Logical Domains Manager to automatically or manually assign the port and node WWNs. By assigning your own values, you can fully control the identity of any virtual function.

The Fibre Channel HBA virtual functions use the N_Port ID Virtualization (NPIV) method to log in to the SAN fabric. Because of this NPIV requirement, you must connect the Fibre Channel HBA port to an NPIV-capable Fibre Channel switch. The virtual functions are managed entirely by the hardware or the firmware of the SR-IOV card. Other than these exceptions, Fibre Channel virtual functions work and behave the same way as a non-SR-IOV Fibre Channel HBA device. The SR-IOV virtual functions have the same capabilities as the non-SR-IOV devices, so all types of SAN storage devices are supported in either configuration.

The virtual functions' unique port and node WWN values enable a SAN administrator to assign storage to the virtual functions in the same way as he would for any non-SR-IOV Fibre Channel HBA port. This mananagement includes zoning, LUN masking, and quality of service (QoS). You can configure the storage so that it is accessible exclusively to a specific logical domain without being visible to the physical function in the root domain.

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

Fibre Channel SR-IOV Hardware Requirements

For information about the required PCIe Fibre Channel SR-IOV hardware, see "SR-IOV Hardware and Software Requirements" on page 80.

- **Control domain.**
  - **QLogic cards.** At least the Oracle Solaris 11.2 OS
  - **Emulex cards.** At least the Oracle Solaris 11.1 SRU 17 OS

- **I/O domain.**
  - **QLogic cards.** At least the Oracle Solaris 11.2 OS
  - **Emulex cards.** At least the Oracle Solaris 11.1 SRU 17 OS

Fibre Channel SR-IOV Requirements and Limitations

The Fibre Channel SR-IOV feature has the following recommendations and limitations:

- The SR-IOV card must run the latest version of firmware that supports the SR-IOV feature.
- The Fibre Channel PCIe card must be connected to a Fibre Channel switch that supports NPIV and that is compatible with the PCIe card.
- The Logical Domains Manager properly autogenerates unique `port-wwn` and `node-wwn` property values by connecting the control domains of all systems to the same SAN fabric and by being part of the same multicast domain.

  If you cannot configure this environment, you must manually provide the `node-wwn` and `port-wwn` values when you create the virtual function. This behavior ensures that there are no naming conflicts. See "World-Wide Name Allocation for Fibre Channel Virtual Functions" on page 123.

Fibre Channel Device Class-Specific Properties

You can use the `ldm create-vf` or the `ldm set-io` commands to set the following Fibre Channel virtual function properties:

`bw-percent` Specifies the percentage of the bandwidth to be allocated to the Fibre Channel virtual function. Valid values are from 0 to 100. The total bandwidth value assigned to a Fibre Channel physical function's virtual functions cannot exceed 100. The default value is 0 so that the virtual function gets a fair share of the bandwidth that is not already reserved by other virtual functions that share the same physical function.

`node-wwn` Specifies the node world-wide name (WWN) for the Fibre Channel virtual function. Valid values are non-zero. By default, this value is allocated automatically. If you manually specify this value, you must also specify a value for the `port-wwn` property. For more information, see "World-Wide Name Allocation for Fibre Channel Virtual Functions" on page 123.

`port-wwn` Specifies the port WWN for the Fibre Channel virtual function. Valid values are non-zero. By default, this value is allocated automatically. If you manually specify this value, you must also specify a value for the `node-wwn` property. For more information, see "World-Wide Name Allocation for Fibre Channel Virtual Functions" on page 123.

You cannot modify the `node-wwn` or the `port-wwn` property values while the Fibre Channel virtual function is in use. However, you can modify the `bw-percent` property value dynamically even when the Fibre Channel virtual function is in use.

World-Wide Name Allocation for Fibre Channel Virtual Functions

The Logical Domains Manager supports both the automatic allocation and the manual assignment of world-wide names for the Fibre Channel virtual functions.

Automatic World-Wide Name Allocation

Logical Domains Manager allocates a unique MAC address from its automatic MAC address allocation pool and creates IEEE format `node-wwn` and `port-wwn` property values.

```
port-wwn = 10:00:XX:XX:XX:XX:XX:XX
node-wwn = 20:00:XX:XX:XX:XX:XX:XX
```

XX:*XX*:*XX*:*XX*:*XX*:*XX* is the automatically allocated MAC address.

This automatic allocation method produces unique WWNs when the control domains of all systems that are connected to the same Fibre Channel fabric are also connected by Ethernet and are part of the same multicast domain. If you cannot meet this requirement, you must manually assign unique WWNs, which is required on the SAN.

Manual World-Wide Name Allocation

You can construct unique WWNs by using any method. This section describes how to create WWNs from the Logical Domains Manager manual MAC address allocation pool. You must guarantee the uniqueness of the WWNs you allocate.

Logical Domains Manager has a pool of 256,000 MAC addresses that are available for manual allocation in the `00:14:4F:FC:00:00` - `00:14:4F:FF:FF:FF` range.

The following example shows the `port-wwn` and `node-wwn` property values based on the `00:14:4F:FC:00:01` MAC address:

```
port-wwn = 10:00:00:14:4F:FC:00:01
node-wwn = 20:00:00:14:4F:FC:00:01
```

`00:14:4F:FC:00:01` is the manually allocated MAC address. For information about automatic MAC address allocation, see "Assigning MAC Addresses Automatically or Manually" on page 241.

Note – It is best to manually assign the WWNs to ensure predictable configuration of the SAN storage.

You must use the manual WWN allocation method when all systems are not connected to the same multicast domain by Ethernet. You can also use this method to guarantee that the same WWNs are used when Fibre Channel virtual functions are destroyed and re-created.

Creating Fibre Channel SR-IOV Virtual Functions

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

How to Create a Fibre Channel SR-IOV Virtual Function

If you cannot use this dynamic method, use the static method instead. See "Static SR-IOV" on page 84.

1. **Identify the physical function device.**

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

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

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

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

   See "How to Enable I/O Virtualization for a PCIe Bus" on page 87.

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

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

   - **Dynamic method:**

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

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

       Use the `ldm create-vf -n max` command to create all the virtual functions for that physical function at one time. This command automatically allocates the port and node WWNs for each virtual function and sets the `bw-percent` property to the default value, which is `0`. This value specifies that fair share bandwidth is allocated to all virtual functions.

Tip – Create all virtual functions for the physical function at once. If you want to manually assign WWNs, first create all of the virtual functions and then use the `ldm set-io` command to manually assign your WWN values for each virtual function. This technique minimizes the number of state transitions when creating virtual functions from a physical function.

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

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

  `ldm create-vf [bw-percent=`*value*`] [port-wwn=`*value* `node-wwn=`*value*`]` *pf-name*

  You can also manually specify Fibre Channel class-specific property values.

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

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

- **Static method:**

  a. **Initiate a delayed reconfiguration.**

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

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

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

  c. **Reboot the root domain.**

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

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

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

       `primary# shutdown -i6 -g0 -y`

Example 7–18 Displaying Information About the Fibre Channel Physical Function

This example shows information about the `/SYS/MB/PCIE7/IOVFC.PF0` physical function:

- This physical function is from a board in a PCIe slot, `PCIE7`.
- The `IOVFC` string indicates that the physical function is a Fibre Channel SR-IOV device.

```
primary# ldm list-io
NAME                                      TYPE   BUS      DOMAIN   STATUS
----                                      ----   ---      ------   ------
pci_0                                     BUS    pci_0    primary  IOV
pci_1                                     BUS    pci_1    rootdom1 IOV
niu_0                                     NIU    niu_0    primary
niu_1                                     NIU    niu_1    primary
/SYS/MB/PCIE0                             PCIE   pci_0    primary  OCC
/SYS/MB/PCIE2                             PCIE   pci_0    primary  OCC
/SYS/MB/PCIE4                             PCIE   pci_0    primary  OCC
/SYS/MB/PCIE6                             PCIE   pci_0    primary  EMP
/SYS/MB/PCIE8                             PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA                            PCIE   pci_0    primary  OCC
/SYS/MB/NET0                              PCIE   pci_0    primary  OCC
/SYS/MB/PCIE1                             PCIE   pci_1    rootdom1 OCC
/SYS/MB/PCIE3                             PCIE   pci_1    rootdom1 OCC
/SYS/MB/PCIE5                             PCIE   pci_1    rootdom1 OCC
/SYS/MB/PCIE7                             PCIE   pci_1    rootdom1 OCC
/SYS/MB/PCIE9                             PCIE   pci_1    rootdom1 OCC
/SYS/MB/NET2                              PCIE   pci_1    rootdom1 OCC
/SYS/MB/NET0/IOVNET.PF0                   PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF1                   PF     pci_0    primary
/SYS/MB/PCIE5/IOVNET.PF0                  PF     pci_1    rootdom1
/SYS/MB/PCIE5/IOVNET.PF1                  PF     pci_1    rootdom1
/SYS/MB/PCIE7/IOVFC.PF0                   PF     pci_1    rootdom1
/SYS/MB/PCIE7/IOVFC.PF1                   PF     pci_1    rootdom1
/SYS/MB/NET2/IOVNET.PF0                   PF     pci_1    rootdom1
/SYS/MB/NET2/IOVNET.PF1                   PF     pci_1    rootdom1
```

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

```
primary# ldm list-io -l /SYS/MB/PCIE7/IOVFC.PF0
NAME                                      TYPE   BUS      DOMAIN   STATUS
----                                      ----   ---      ------   ------
/SYS/MB/PCIE7/IOVNET.PF0                  PF     pci_0    rootdom1
[pci@400/pci@1/pci@0/pci@6/SUNW,fcdev@0]
    maxvfs = 8
```

Example 7–19 Dynamically Creating a Fibre Channel Virtual Function Without Setting Optional Properties

This example dynamically creates a virtual function without setting any optional properties. In this case, the `ldm create-vf` command automatically allocates the default bandwidth percentage, port world-wide name (WWN), and node WWN values.

Ensure that I/O virtualization is enabled on the `pci_1` PCIe bus. See "How to Enable I/O Virtualization for a PCIe Bus" on page 87.

You can use the `ldm create-vf` command to create all the virtual functions from the `/SYS/MB/PCIE7/IOVFC.PF0` physical function.

```
primary# ldm create-vf -n max /SYS/MB/PCIE7/IOVFC.PF0
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF0
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF1
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF2
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF3
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF4
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF5
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF6
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF7
```

Example 7–20 Dynamically Creating a Fibre Channel Virtual Function and Setting Properties

This example dynamically creates a virtual function while setting the `bw-percent` property value to 25 and specifies the port and node WWNs.

```
primary# ldm create-vf port-wwn=10:00:00:14:4F:FC:00:01 \
node-wwn=20:00:00:14:4F:FC:00:01 bw-percent=25 /SYS/MB/PCIE7/IOVFC.PF0
```

Example 7–21 Statically Creating a Fibre Channel Virtual Function Without Setting Optional Properties

This example statically creates a virtual function without setting any optional properties. In this case, the `ldm create-vf` command automatically allocates the default bandwidth percentage, port world-wide name (WWN), and node WWN values.

First you initiate a delayed reconfiguration on the `rootdom1` domain. Then, enable I/O virtualization on the `pci_1` PCIe bus. Because the `pci_1` bus has already been assigned to the `rootdom1` root domain, use the `ldm set-io` command to enable I/O virtualization.

```
primary# ldm start-reconf rootdom1
Initiating a delayed reconfiguration operation on the rootdom1 domain.
All configuration changes for other domains are disabled until the rootdom1
domain reboots, at which time the new configuration for the rootdom1 domain
will also take effect.

primary# ldm set-io iov=on pci_1
```

Now, you can use the `ldm create-vf` command to create all the virtual functions from the `/SYS/MB/PCIE7/IOVFC.PF0` physical function.

```
primary# ldm create-vf -n max /SYS/MB/PCIE7/IOVFC.PF0

------------------------------------------------------------------------------
Notice: The rootdom1 domain is in the process of a delayed reconfiguration.
```

```
Any changes made to the rootdom1 domain will only take effect after it reboots.
-------------------------------------------------------------------------------

Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF0
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF1
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF2
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF3
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF4
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF5
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF6
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF7
```

Finally, reboot the `rootdom1` root domain to make the changes take effect in one of the following ways:

- `rootdom1` is a non-`primary` root domain

  ```
  primary# ldm stop-domain -r rootdom1
  ```

- `rootdom1` is the `primary` domain

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

Destroying Fibre Channel SR-IOV Virtual Functions

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

▼ How to Destroy a Fibre Channel SR-IOV Virtual Function

If you cannot use this dynamic method, use the static method instead. See "Static SR-IOV" on page 84.

1 Identify the physical function device.

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

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

- **Dynamic method:**

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

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

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

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

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

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

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

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

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

- **Static method:**

  a. **Initiate a delayed reconfiguration.**

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

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

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

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

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

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

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

  c. **Reboot the root domain.**

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

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

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

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

Example 7–22 Dynamically Destroying Multiple Fibre Channel SR-IOV Virtual Functions

This example shows the results of destroying all the virtual functions from the `/SYS/MB/PCIE5/IOVFC.PF1` physical function. The `ldm list-io` output shows that the physical function has eight virtual functions. The `ldm destroy-vf -n max` command destroys all the virtual functions, and the final `ldm list-io` output shows that none of the virtual functions remain.

```
primary# ldm list-io
...
/SYS/MB/PCIE5/IOVFC.PF1                  PF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF0              VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF1              VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF2              VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF3              VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF4              VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF5              VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF6              VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF7              VF     pci_1
primary# ldm destroy-vf -n max /SYS/MB/PCIE5/IOVFC.PF1
primary# ldm list-io
...
/SYS/MB/PCIE5/IOVFC.PF1                  PF     pci_1
```

Example 7–23 Destroying a Fibre Channel Virtual Function

This example shows how to statically destroy the virtual functions from the `/SYS/MB/PCIE7/IOVFC.PF0` physical function.

```
primary# ldm start-reconf rootdom1
Initiating a delayed reconfiguration operation on the rootdom1 domain.
All configuration changes for other domains are disabled until the rootdom1
domain reboots, at which time the new configuration for the rootdom1 domain
will also take effect.

primary# ldm destroy-vf -n max /SYS/MB/PCIE7/IOVFC.PF0
primary# ldm stop-domain -r rootdom1
```

Modifying Fibre Channel SR-IOV Virtual Functions

The `ldm set-io` command modifies the current configuration of a virtual function by changing the property values or by setting new properties.

If you cannot use this dynamic method, use the static method instead. See "Static SR-IOV" on page 84.

You can use the `ldm set-io` command to modify the `bw-percent`, `port-wwn`, and `node-wwn` properties.

You can dynamically change only the `bw-percent` propery while the virtual functions are assigned to a domain.

▼ How to Modify Fibre Channel SR-IOV Virtual Function Properties

1. **Identify the physical function device.**

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

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

2. **Modify a virtual function property.**

   ```
   ldm set-io [bw-percent=value] [port-wwn=value node-wwn=value] pf-name
   ```

   Unlike the `bw-percent` property value, which you can dynamically change at any time, you can dynamically modify the `port-wwn` and `node-wwn` property values only when the virtual function is not assigned to a domain.

Example 7–24 Modifying Fibre Channel SR-IOV Virtual Function Properties

This example modifies the properties of the specified virtual function, `/SYS/MB/PCIE7/IOVFC.PF0.VF0`, to specify the bandwidth percentage and the port and node WWN values.

```
primary# ldm set-io port-wwn=10:00:00:14:4f:fc:f4:7c \
node-wwn=20:00:00:14:4f:fc:f4:7c bw-percent=25 /SYS/MB/PCIE7/IOVFC.PF0.VF0
```

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

▼ How to Add a Fibre Channel SR-IOV Virtual Function to an I/O Domain

If you cannot dynamically remove the virtual function, use the static method. See "Static SR-IOV" on page 84.

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

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

2. **Add a virtual function either dynamically or statically.**

   - **To dynamically add a virtual function:**

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

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

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

- **To statically add a virtual function:**

  a. **Stop the domain and then add the virtual function.**

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

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

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

  b. **Restart the domain.**

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

Example 7–25 Adding a Fibre Channel Virtual Function

This example shows how to dynamically add the `/SYS/MB/PCIE7/IOVFC.PF0.VF0` virtual function to the `ldg2` domain.

```
primary# ldm add-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
```

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

```
primary# ldm stop-domain ldg2
primary# ldm add-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
primary# ldm start-domain ldg2
```

▼ How to Remove a Fibre Channel SR-IOV Virtual Function From an I/O Domain

If you cannot use this dynamic method, use the static method instead. See "Static SR-IOV" on page 84.

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

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

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

2 Remove a virtual function either dynamically or statically.

- **To dynamically remove a virtual function:**

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

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

- **To statically remove a virtual function:**

  a. **Stop the I/O domain.**

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

  b. **Remove the virtual function.**

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

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

  c. **Start the I/O domain.**

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

Example 7–26 Dynamically Removing a Fibre Channel Virtual Function

This example shows how to dynamically remove the `/SYS/MB/PCIE7/IOVFC.PF0.VF0` virtual function from the `ldg2` domain.

```
primary# ldm remove-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
```

If the command succeeds, the virtual function is removed from the `ldg2` domain. When `ldg2` is restarted, the specified virtual function no longer appears in that domain.

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

```
primary# ldm stop-domain ldg2
primary# ldm remove-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
primary# ldm start-domain ldg2
```

Advanced SR-IOV Topics: Fibre Channel SR-IOV

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

Accessing a Fibre Channel Virtual Function in a Guest Domain

The `ldg2` console log shows the operations of the assigned Fibre Channel virtual function device. Use the `fcadm` command to view and access the Fibre Channel virtual function device.

```
ldg2# fcadm hba-port
HBA Port WWN: 100000144ffb8a99
       Port Mode: Initiator
       Port ID: 13d02
       OS Device Name: /dev/cfg/c3
       Manufacturer: Emulex
       Model: 7101684
       Firmware Version: 7101684 1.1.60.1
       FCode/BIOS Version: Boot:1.1.60.1 Fcode:4.03a4
       Serial Number: 4925382+133400002R
       Driver Name: emlxs
       Driver Version: 2.90.15.0 (2014.01.22.14.50)
       Type: N-port
       State: online
       Supported Speeds: 4Gb 8Gb 16Gb
       Current Speed: 16Gb
       Node WWN: 200000144ffb8a99
       NPIV Not Supported
```

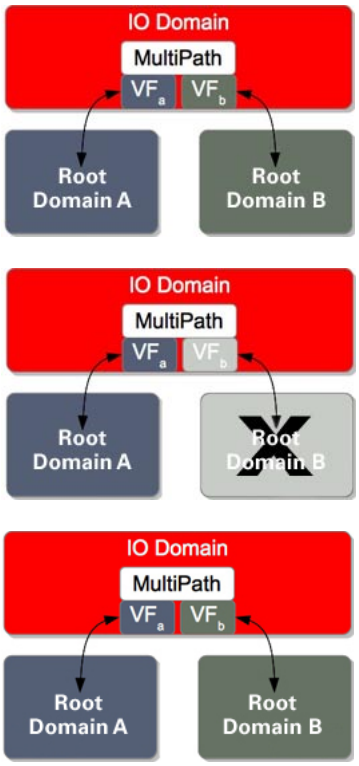
Use the `format` command to show the visible LUNs.

```
ldg2# format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
   0. c2d0 <Unknown-Unknown-0001-25.00GB>
      /virtual-devices@100/channel-devices@200/disk@0
   1. c3t21000024FF4C4BF8d0 <SUN-COMSTAR-1.0-10.00GB>
      /pci@340/pci@1/pci@0/pci@6/SUNW,emlxs@0,2/fp@0,0/ssd@w21000024ff4c4bf8,0
Specify disk (enter its number): ^D
ldg2#
```

I/O Domain Resiliency

I/O domain resiliency improves the availability and performance of an I/O domain by enabling it to continue to run even when one of its associated root domains is interrupted. When a root domain is interrupted, the I/O domains that use its services continue to run by enabling its affected devices to fail over to the alternate I/O path. When the root domain returns to service, the affected devices in the resilient I/O domain are also returned to service and the failover capabilities are restored.

The following diagrams show and describe what happens when one of the configured root domains fails and what happens when the root domain returns to service.



Each root domain provide a virtual function to the I/O
The I/O domain uses virtual device multipathing such a
for network devices and MPxIO for Fibre Channel devi

When root domain B is interrupted by a panic or reboo
function B is suspended in the I/O domain and then
multipathing engages to route all I/O through root dom

When root domain B is restored to service, virtual func
resumes operation in the I/O domain. The multipath gr
restored to full reduncancy.

In this configuration, the virtual function could be a virtual network device or a virtual storage device, which means that the I/O domain can be configured with any combination of virtual functions or virtual devices.

You can create a configuration where you have both resilient and non-resilient I/O domains. For an example, see "Example – Using Resilient and Non-Resilient Configurations" on page 140.

Resilient I/O Domain Requirements

Note – The Oracle Solaris 10 OS does not provide I/O domain resiliency.

A resilient I/O domain must meet the following requirements:

- Runs at least the Oracle Solaris 11.2 SRU 8 OS and its `primary` domain runs at least the Oracle VM Server for SPARC 3.2 software.
- Uses multipathing to create failover configurations for virtual functions and virtual devices. This configuration requires the virtual functions and the virtual devices to be of the same class: network or storage.
- Has the `master` property value set to the name of a root domain whose `failure-policy` property is set to `ignore`. Any other failure policy setting, such as `stop`, `reset`, or `panic`, supersedes I/O resiliency and the I/O domain is interrupted.
- Uses only SR-IOV virtual functions, virtual network devices, and virtual storage devices that support I/O domain resiliency. See `https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&id=1325454.1`.

I/O Domain Resiliency Limitations

- When you hotplug an SR-IOV card to the root domain and then assign virtual functions to it to an I/O domain, the I/O domain might fail to provide resiliency when the root domain fails. Therefore, you should only add the SR-IOV card while the root domain is down. Then assign the virtual functions after the root domain boots.
- If you have a resilient I/O domain and then assign a device in one of the following ways, then the I/O domain is no longer resilient:
  - Add a virtual function from a card that does not support I/O resiliency
  - Directly assign a device by using the direct I/O feature

  In that case, set the failure-policy from `ignore` to `reset` or `stop`.

Configuring Resilient I/O Domains

How to Configure a Resilient I/O Domain

Before You Begin Use only the PCIe cards that support the I/O domain resiliency feature. See `https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&id=1325454.1`.

Ensure that the I/O domain, root domain, service domain, and `primary` domain run at least the Oracle Solaris 11.2 SRU 8 OS and the Logical Domains Manager 3.2 software.

1 On the root domain, set the `failure-policy` property to `ignore`.

```
primary# ldm set-domain failure-policy=ignore root-domain-name
```

Note – If you add any devices to the I/O domain that are not supported for resiliency, that domain is no longer resilient. So, reset the `failure-policy` property value to `stop`, `reset`, or `panic`.

For information about domain dependencies, see "Configuring Domain Dependencies" on page 367.

2. **On the I/O domain, set the `master` property to the name of the root domain.**

   ```
   primary# ldm set-domain master=root-domain-name  I/O-domain-name
   ```

3. **Configure multipathing across the paths.**

   - **Ethernet. Use IPMP to configure multipathing across the paths.**

     For information about using IPMP to configure multipathing, see *Administering TCP/IP Networks, IPMI, and IP Tunnels in Oracle Solaris 11.2*.

   - **Fibre Channel. Use MPxIO to configure multipathing across the paths.**

     For information about using MPxIO to configure multipathing, see *Managing SAN Devices and Multipathing Oracle Solaris 11.2*.

Example 7–27 Using IPMP to Configure Multipathing With Ethernet SR-IOV Functions

This example shows how to use IPMP to configure network virtual-function devices for a resilient I/O domain. For more information, see *Administering TCP/IP Networks, IPMI, and IP Tunnels in Oracle Solaris 11.2*.

1. Identify two Ethernet SR-IOV physical functions that are assigned to different root domains.

   In this example, the `root-1` and `root-2` root domains have Ethernet SR-IOV physical functions.

   ```
   primary# ldm list-io | grep root-1 | grep PF
   /SYS/PCI-EM8/IOVNET.PF0                   PF     pci_1    root-1
   primary# ldm list-io | grep root-2 | grep PF
   /SYS/RIO/NET2/IOVNET.PF0                  PF     pci_2    root-2
   ```

2. Create two Ethernet virtual functions on each of the specified physical functions.

   ```
   primary# ldm create-vf /SYS/MB/NET0/IOVNET.PF0
   Created new vf: /SYS/PCI-EM8/IOVNET.PF0.VF0
   primary# ldm create-vf /SYS/RIO/NET2/IOVNET.PF0
   Created new vf: /SYS/RIO/NET2/IOVNET.PF0.VF0
   ```

3. Assign the Ethernet virtual functions to the `io-1` I/O domain.

   ```
   primary# ldm add-io /SYS/PCI-EM8/IOVNET.PF0.VF0 io-1
   primary# ldm add-io /SYS/RIO/NET2/IOVNET.PF0.VF0 io-1
   ```

4. Configure the Ethernet virtual functions into an IPMP group on the I/O domain.

a. Identify the newly added network devices, `net1` and `net2`, on the I/O domain.

```
i0-1# dladm show-phys
LINK        MEDIA        STATE    SPPED    DUPLEX    DEVICE
net0        Ethernet     up       0        unknown   vnet0
net1        Ethernet     up       1000     full      igbvf0
net2        Ethernet     up       1000     full      igbvf1
```

b. Create IP interfaces for the newly added network devices.

```
io-1# ipadm create-ip net1
io-1# ipadm create-ip net2
```

c. Create the `ipmp0` IPMP group for the two network interfaces.

```
io-1# ipadm create-ipmp -i net1 -i net2 ipmp0
```

d. Assign an IP address to the IPMP group.

This example configures the DHCP option.

```
io-1# ipadm create-addr -T dhcp ipmp0/v4
```

e. Check the status of the IPMP group interface.

```
io-1# ipmpstat -g
```

Example 7–28 Using MPxIO to Configure Multipathing With Fibre Channel SR-IOV Functions

This example shows how to use MPxIO to configure Fibre Channel virtual-function devices for a resilient I/O domain. For more information, see *Managing SAN Devices and Multipathing Oracle Solaris 11.2*.

1. Identify two Fibre Channel SR-IOV physical functions that are assigned to different root domains.

   In this example, the `root-1` and `root-2` root domains have Fibre Channel SR-IOV physical functions.

```
primary# ldm list-io | grep root-1 | grep PF
/SYS/PCI-EM4/IOVFC.PF0                    PF     pci_1    root-1
primary# ldm list-io | grep root-2 | grep PF
/SYS/PCI-EM15/IOVFC.PF0                   PF     pci_2    root-2
```

2. Create two virtual functions on each of the specified physical functions.

   For more information, see "How to Create a Fibre Channel SR-IOV Virtual Function" on page 124.

```
primary# ldm create-vf port-wwn=10:00:00:14:4f:fc:60:00 \
node-wwn=20:00:00:14:4f:fc:60:00 /SYS/PCI-EM4/IOVFC.PF0
Created new vf: /SYS/PCI-EM4/IOVFC.PF0.VF0
primary# ldm create-vf port-wwn=10:00:00:14:4f:fc:70:00 \
node-wwn=20:00:00:14:4f:fc:70:00 /SYS/PCI-EM15/IOVFC.PF0
Created new vf: /SYS/PCI-EM15/IOVFC.PF0.VF0
```

3. Add the newly created virtual functions to the `io-1` I/O domain.

```
primary# ldm add-io /SYS/PCI-EM4/IOVFC.PF0.VF0 io-1
primary# ldm add-io /SYS/PCI-EM15/IOVFC.PF0.VF0 io-1
```

4. Determine whether MPxIO is enabled on the I/O domain by using the `prtconf -v` command.

   If the output for the `fp` device includes the following device property setting, MPxIO is enabled:

   ```
   mpxio-disable="no"
   ```

   If the `mpxio-disable` property is set to `yes`, update the property value to `no` in the `/etc/driver/drv/fp.conf` file and then reboot the I/O domain.

   If the `mpxio-disable` device property does not appear in the `prtconf -v` output, add the `mpxio-disable="no"` entry to the `/etc/driver/drv/fp.conf` file and then reboot the I/O domain.

5. Check the status of MPxIO group.

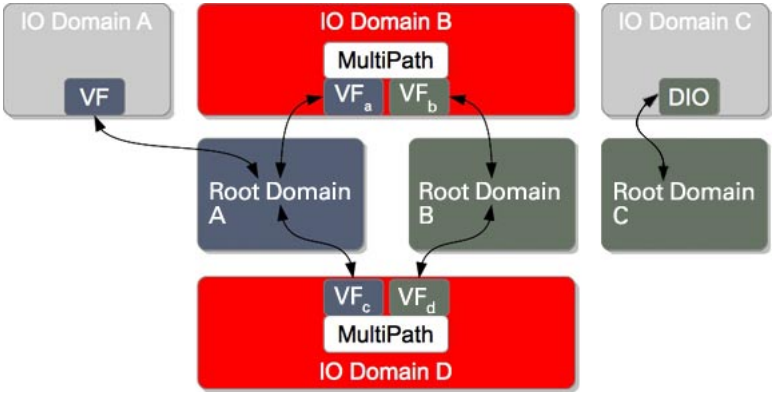
   ```
   io-1# mpathadm show LU

   Logical Unit:  /dev/rdsk/c0t600A0B80002A384600003D6B544EECD0d0s2
        mpath-support:  libmpscsi_vhci.so
        Vendor:  SUN
        Product:  CSM200_R
        Revision:  0660
        Name Type:  unknown type
        Name:  600a0b80002a384600003d6b544eecd0
        Asymmetric:  yes
        Current Load Balance:  round-robin
        Logical Unit Group ID:  NA
        Auto Failback:  on
        Auto Probing:  NA

        Paths:
                Initiator Port Name:  100000144ffc6000
                Target Port Name:  201700a0b82a3846
                Override Path:  NA
                Path State:  OK
                Disabled:  no

                Initiator Port Name:  100000144ffc7000
                Target Port Name:  201700a0b82a3846
                Override Path:  NA
                Path State:  OK
                Disabled:  no

        Target Port Groups:
                ID:  1
                Explicit Failover:  yes
                Access State:  active
                Target Ports:
                        Name:  201700a0b82a3846
                        Relative ID:  0
   ```

Example – Using Resilient and Non-Resilient Configurations

You can use configurations with both resilient and non-resilient domains.

The following figure shows that I/O domain A and I/O domain C are not resilient because neither use multipathing. I/O domain A has a virtual function and I/O domain C has a direct I/O device.

FIGURE 7–2 Configuration With Resilient and Non-Resilient I/O Domains



I/O domain B and I/O domain D are resilient. I/O domains A, B, and D depend on root domain A. I/O domains B and D depend on root domain B. I/O domain C depends on root domain C.

If root domain A is interrupted, I/O domain A is interrupted as well. I/O domains B and D, however, fail over to alternate paths and continue to run applications. If root domain C is interrupted, I/O domain C fails in the way specified by the `failure-policy` property value of root domain C.

Rebooting the Root Domain With Non-Resilient I/O Domains Configured

Note – If your I/O domain is resilient, it can continue to operate even when the root domain that services it is interrupted. For information about configuring resilient I/O domains, see "I/O Domain Resiliency" on page 134.

As with PCIe slots in the I/O domain, the concerns that are described in "Rebooting the Root Domain With PCIe Endpoints Configured" on page 149 also pertain to the virtual functions that are assigned to an I/O domain.

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

◆ ◆ ◆ CHAPTER 8

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

This chapter covers the following direct I/O topics:

Creating an I/O Domain by Assigning PCIe Endpoint Devices

You can assign an individual PCIe endpoint (or direct I/O-assignable) device to a domain. This use of PCIe endpoint devices increases the granularity of the device assignment to I/O domains. This capability is delivered by means of the direct I/O (DIO) feature.

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

A PCIe endpoint device can be one of the following:

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

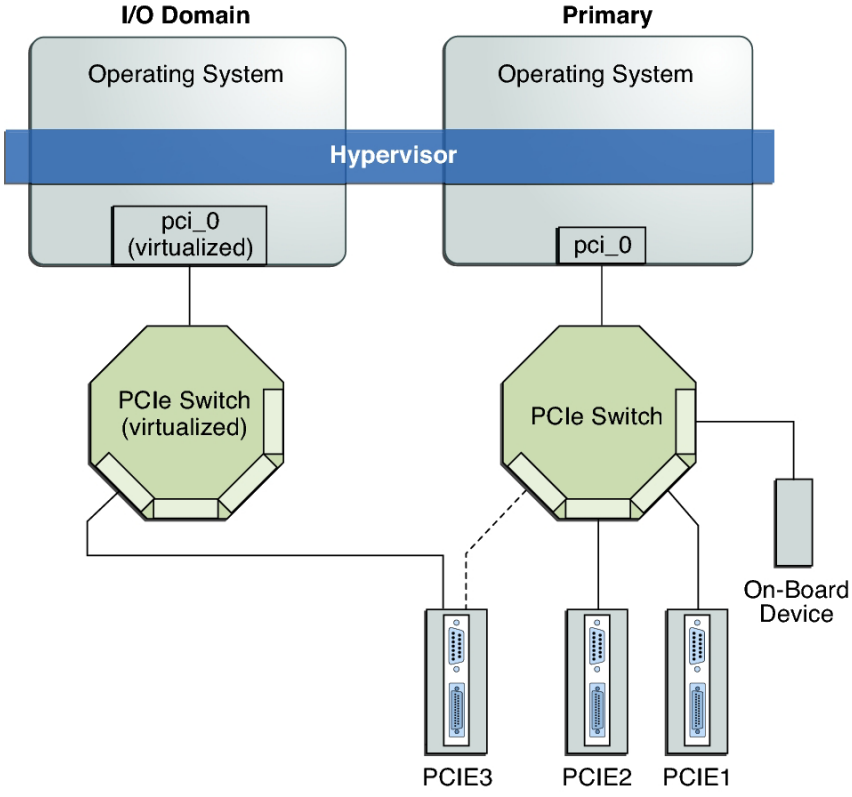
Note – Because root domains cannot have dependencies on other root domains, a root domain that owns a PCIe bus cannot have its PCIe endpoint devices or SR-IOV virtual functions assigned to another root domain. However, you *can* assign a PCIe endpoint device or virtual function from a PCIe bus to the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

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

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

Caution – You cannot use Oracle Solaris hot-plug operations to hot-remove a PCIe endpoint device after that device is removed from the `primary` domain by using the `ldm remove-io` command. For information about replacing or removing a PCIe endpoint device, see "Making PCIe Hardware Changes" on page 150.

FIGURE 8–1 Assigning a PCIe Endpoint Device to an I/O Domain



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

Though the DIO feature permits any PCIe card in a slot to be assigned to an I/O domain, only certain PCIe cards are supported. See "Direct I/O Hardware and Software Requirements" on page 145.

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

The following items describe important details about the DIO feature:

- This feature is enabled only when all the software requirements are met. See "Direct I/O Hardware and Software Requirements" on page 145.
- Only PCIe endpoints that are connected to a PCIe bus assigned to a root domain can be assigned to another domain with the DIO feature.
- I/O domains that use DIO have access to the PCIe endpoint devices only when the root domain is running.
- Rebooting the root domain affects I/O domains that have PCIe endpoint devices. See "Rebooting the Root Domain With PCIe Endpoints Configured" on page 149. The root domain also performs the following tasks:
  - Initializes and manages the PCIe bus.
  - Handles all bus errors that are triggered by the PCIe endpoint devices that are assigned to I/O domains. Note that only the `primary` domain receives all PCIe bus-related errors.

Direct I/O Hardware and Software Requirements

To successfully use the direct I/O (DIO) feature to assign direct I/O devices to domains, you must run the appropriate software and use supported PCIe cards.

- **Hardware Requirements.** Only certain PCIe cards can be used as a direct I/O endpoint device on an I/O domain. You can still use other cards in your Oracle VM Server for SPARC environment but they cannot be used with the DIO feature. Instead, they can be used for service domains and for I/O domains that have entire root complexes assigned to them.

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

Note – The SPARC T7 series server and SPARC M7 series server have an I/O controller that provides several PCIe buses and you can assign PCIe cards to different domains by using the PCIe bus assignment feature. For information, see Chapter 6, "Creating a Root Domain by Assigning PCIe Buses."

- **Software Requirements.** To use the DIO feature, the following domains must run the supported OS:
  - **Root domain.** At least the Oracle Solaris 11.3 OS.

    The recommended practice is for all domains to run at least the Oracle Solaris 10 1/13 OS plus the required patches in "Fully Qualified Oracle Solaris OS Versions" in *Oracle VM Server for SPARC 3.3 Installation Guide* or the Oracle Solaris 11.3 OS.
  - **I/O domain.** At least the Oracle Solaris 11 OS. Note that additional feature support is included in more recent Oracle Solaris 11 releases.

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

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

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

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

Current Direct I/O Feature Limitations

For information about how to work around the limitations, see "Planning PCIe Endpoint Device Configuration" on page 147.

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

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

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

Note – The direct I/O feature is not supported on the SPARC M7 series server and SPARC T7 series server. Instead, use the PCIe bus assignment feature. See Chapter 6, "Creating a Root Domain by Assigning PCIe Buses."

SPARC systems, up to and including the SPARC T5 and SPARC M6 platforms, provide a finite number of interrupts, so Oracle Solaris limits the number of interrupts that each device can use. The default limit should match the needs of a typical system configuration but you might need to adjust this value for certain system configurations. For more information, see "Adjusting the Interrupt Limit" on page 380.

Planning PCIe Endpoint Device Configuration

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

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

For an example, see "How to Create an I/O Domain by Assigning a PCIe Endpoint Device" on page 152.

You must take the following general steps to plan and perform a DIO device configuration:

1. Understand and record your system hardware configuration.

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

   Use the `ldm list-io -l` and `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 root domain, because reboots affect I/O domains.

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

   Note – After you have removed all the devices you want during a delayed reconfiguration, you need to reboot the root domain only one time to make all the changes take effect.

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

   Use the `ldm add-config` command.

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

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

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

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

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

   Use the `ldm add-io` command.

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

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

   See "Making PCIe Hardware Changes" on page 150.

Rebooting the Root Domain With PCIe Endpoints Configured

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

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

Note that if the I/O domain is resilient, it can continue to operate even if the root domain that is the owner of the PCIe bus becomes unavailable. See "I/O Domain Resiliency" on page 134.

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

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

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

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

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

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

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

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

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

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

EXAMPLE 8–1 Configuring Failure Policy Dependencies for a Configuration With a Non-primary Root Domain and I/O Domains

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

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

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

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

- If it is the non-`primary` root domain rebooting, this dependency relationship ensures that the I/O domain is stopped. Start the I/O domain after the non-`primary` root domain boots.

  ```
  primary# ldm start ldg2
  ```

- If it is the `primary` domain rebooting, this policy setting stops both the non-`primary` root domain and the dependent I/O domains. When the `primary` domain boots, you must start the non-`primary` root domain first. When the domain boots, start the I/O domain.

  ```
  primary# ldm start ldg1
  ```

  Wait for the `ldg1` domain to become active and then start the I/O domain.

  ```
  primary# ldm start ldg2
  ```

Making PCIe Hardware Changes

The following steps help you avoid misconfiguring the PCIe endpoint assignments. For platform-specific information about installing and removing specific hardware, see the documentation for your platform.

- No action is required if you are installing a PCIe card into an empty slot. This PCIe card is automatically owned by the domain that owns the PCIe bus.

  To assign the new PCIe card to an I/O domain, use the `ldm remove-io` command to first remove the card from the root domain. Then, use the `ldm add-io` command to assign the card to an I/O domain.

- No action is required if a PCIe card is removed from the system and assigned to the root domain.
- To remove a PCIe card that is assigned to an I/O domain, first remove the device from the I/O domain. Then, add the device to the root domain before you physically remove the device from the system.
- To replace a PCIe card that is assigned to an I/O domain, verify that the new card is supported by the DIO feature.

If so, no action is required to automatically assign the new card to the current I/O domain.

If not, first remove that PCIe card from the I/O domain by using the `ldm remove-io` command. Next, use the `ldm add-io` command to reassign that PCIe card to the root domain. Then, physically replace the PCIe card you assigned to the root domain with a different PCIe card. These steps enable you to avoid a configuration that is unsupported by the DIO feature.

Minimizing Guest Domain Outages When Removing a PCIe Card

While you remove or replace a PCIe card from a system that runs the Oracle VM Server for SPARC software, the domains that depend on this hardware are unavailable. To minimize such guest domain outages, you must prepare your system to use the hotplug capabilities to physically remove the card.

How to Minimize Guest Domain Outages When Removing a PCIe Card

This procedure enables you to avoid an outage to a guest domain that does not have direct I/O or SR-IOV device assigned to it and that has multiple paths configured. Note that this procedure requires two reboots of the `primary` domain.

Note – This procedure does not apply when the PCIe card is on a root complex owned by a non-`primary` root domain. Instead, see How to Replace PCIe Direct I/O Cards Assigned to an Oracle VM Server for SPARC Guest Domain (Doc ID 1684273.1) (`https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=226878266536565&id=1684273.1&_adf.ctrl-state=bo9fbmr1n_49`).

---

1. **Stop the guest domain that has the PCIe slot assigned to it.**

   ```
   primary# ldm stop domain-name
   ```

2. **Remove the PCIe slot from the guest domain.**

   ```
   primary# ldm remove-io PCIe-slot domain-name
   ```

3. **Stop the guest domains that have PCIe slots and SR-IOV virtual functions assigned to them.**

   ```
   primary# ldm stop domain-name
   ```

Note – You do not need to stop guest domains that have PCIe buses assigned to them because they might be providing alternate paths to network and disk devices to the guest domains.

---

4. **Initiate a delayed reconfiguration on the `primary` domain so that you can assign this slot to it.**

   ```
   primary# ldm start-reconf primary
   ```

5. **Add the PCIe slot to the `primary` domain.**

   ```
   primary# ldm add-io PCIe-slot domain-name
   ```

6. **Reboot the `primary` domain.**

   ```
   primary# shutdown -i6 -g0 -y
   ```

7. **Use the hotplug commands to replace the PCIe card.**

   For information about Oracle Solaris OS hotplug capabilities, see Chapter 2, "Dynamically Configuring Devices," in *Managing Devices in Oracle Solaris 11.3* .

8. **After the card is replaced, perform the following steps if you must reassign this same PCIe slot to the guest domain:**

   a. **Initiate a delayed reconfiguration on the `primary` domain.**

      ```
      primary# ldm start-reconf primary
      ```

   b. **Remove the PCIe slot from the `primary` domain.**

      ```
      primary# ldm remove-io PCIe-slot domain-name
      ```

   c. **Reboot the `primary` domain to cause the removal of the PCIe slot to take effect.**

      ```
      primary# shutdown -i6 -g0 -y
      ```

   d. **Reassign the PCIe slot to the guest domain.**

      ```
      primary# ldm add-io PCIe-slot domain-name
      ```

   e. **Start the guest domains to which you want to assign PCIe slots and SR-IOV virtual functions.**

      ```
      primary# ldm start domain-name
      ```

Creating an I/O Domain by Assigning a PCIe Endpoint Device

▼ How to Create an I/O Domain by Assigning a PCIe Endpoint Device

Plan all DIO deployments ahead of time to minimize downtime.

Caution – The `primary` domain loses access to the on-board DVD device if you assign the `/SYS/MB/SASHBA1` slot on a SPARC T3-1 or a SPARC T4-1 system to a DIO domain.

The SPARC T3-1 and SPARC T4-1 systems include two DIO slots for on-board storage, which are represented by the `/SYS/MB/SASHBA0` and `/SYS/MB/SASHBA1` paths. In addition to hosting multiheaded on-board disks, the `/SYS/MB/SASHBA1` slot hosts the on-board DVD device. So, if you assign `/SYS/MB/SASHBA1` to a DIO domain, the `primary` domain loses access to the on-board DVD device.

The SPARC T3-2 and SPARC T4-2 systems have a single `SASHBA` slot that hosts all on-board disks as well as the on-board DVD device. So, if you assign `SASHBA` to a DIO domain, the on-board disks *and* the on-board DVD device are loaned to the DIO domain and unavailable to the `primary` domain.

For an example of adding a PCIe endpoint device to create an I/O domain, see "Planning PCIe Endpoint Device Configuration" on page 147.

Note – In this release, use the `DefaultFixed` NCP to configure datalinks and network interfaces on Oracle Solaris 11 systems.

The Oracle Solaris 11 OS includes the following NCPs:

- `DefaultFixed` – Enables you to use the `dladm` or `ipadm` command to manage networking
- `Automatic` – Enables you to use the `netcfg` or `netadm` command to manage networking

Ensure that the `DefaultFixed` NCP is enabled by using the `netadm list` command. See Chapter 7, "Using Datalink and Interface Configuration Commands on Profiles," in *Oracle Solaris Administration: Network Interfaces and Network Virtualization* .

1 **Identify and archive the devices that are currently installed on the system.**

The output of the `ldm list-io -l` command shows how the I/O devices are currently configured. You can obtain more detailed information by using the `prtdiag -v` command.

Note – After the devices are assigned to I/O domains, the identity of the devices can be determined only in the I/O domains.

```
primary# ldm list-io -l
NAME                                      TYPE   BUS      DOMAIN   STATUS
----                                      ----   ---      ------   ------
niu_0                                     NIU    niu_0    primary
[niu@480]
niu_1                                     NIU    niu_1    primary
[niu@580]
```

```
pci_0                                    BUS    pci_0    primary
[pci@400]
pci_1                                    BUS    pci_1    primary
[pci@500]
/SYS/MB/PCIE0                            PCIE   pci_0    primary  OCC
[pci@400/pci@2/pci@0/pci@8]
    SUNW,emlxs@0/fp/disk
    SUNW,emlxs@0/fp/tape
    SUNW,emlxs@0/fp@0,0
    SUNW,emlxs@0,1/fp/disk
    SUNW,emlxs@0,1/fp/tape
    SUNW,emlxs@0,1/fp@0,0
/SYS/MB/PCIE2                            PCIE   pci_0    primary  OCC
[pci@400/pci@2/pci@0/pci@4]
    pci/scsi/disk
    pci/scsi/tape
    pci/scsi/disk
    pci/scsi/tape
/SYS/MB/PCIE4                            PCIE   pci_0    primary  OCC
[pci@400/pci@2/pci@0/pci@0]
    ethernet@0
    ethernet@0,1
    SUNW,qlc@0,2/fp/disk
    SUNW,qlc@0,2/fp@0,0
    SUNW,qlc@0,3/fp/disk
    SUNW,qlc@0,3/fp@0,0
/SYS/MB/PCIE6                            PCIE   pci_0    primary  EMP
[pci@400/pci@1/pci@0/pci@8]
/SYS/MB/PCIE8                            PCIE   pci_0    primary  EMP
[pci@400/pci@1/pci@0/pci@c]
/SYS/MB/SASHBA                           PCIE   pci_0    primary  OCC
[pci@400/pci@2/pci@0/pci@e]
    scsi@0/iport@1
    scsi@0/iport@2
    scsi@0/iport@4
    scsi@0/iport@8
    scsi@0/iport@80/cdrom@p7,0
    scsi@0/iport@v0
/SYS/MB/NET0                             PCIE   pci_0    primary  OCC
[pci@400/pci@1/pci@0/pci@4]
    network@0
    network@0,1
/SYS/MB/PCIE1                            PCIE   pci_1    primary  OCC
[pci@500/pci@2/pci@0/pci@a]
    SUNW,qlc@0/fp/disk
    SUNW,qlc@0/fp@0,0
    SUNW,qlc@0,1/fp/disk
    SUNW,qlc@0,1/fp@0,0
/SYS/MB/PCIE3                            PCIE   pci_1    primary  OCC
[pci@500/pci@2/pci@0/pci@6]
    network@0
    network@0,1
    network@0,2
    network@0,3
/SYS/MB/PCIE5                            PCIE   pci_1    primary  OCC
[pci@500/pci@2/pci@0/pci@0]
    network@0
    network@0,1
/SYS/MB/PCIE7                            PCIE   pci_1    primary  EMP
```

```
[pci@500/pci@1/pci@0/pci@6]
/SYS/MB/PCIE9                              PCIE   pci_1    primary  EMP
[pci@500/pci@1/pci@0/pci@0]
/SYS/MB/NET2                               PCIE   pci_1    primary  OCC
[pci@500/pci@1/pci@0/pci@5]
    network@0
    network@0,1
    ethernet@0,80
/SYS/MB/NET0/IOVNET.PF0                    PF     pci_0    primary
[pci@400/pci@1/pci@0/pci@4/network@0]
    maxvfs = 7
/SYS/MB/NET0/IOVNET.PF1                    PF     pci_0    primary
[pci@400/pci@1/pci@0/pci@4/network@0,1]
    maxvfs = 7
/SYS/MB/PCIE5/IOVNET.PF0                   PF     pci_1    primary
[pci@500/pci@2/pci@0/pci@0/network@0]
    maxvfs = 63
/SYS/MB/PCIE5/IOVNET.PF1                   PF     pci_1    primary
[pci@500/pci@2/pci@0/pci@0/network@0,1]
    maxvfs = 63
/SYS/MB/NET2/IOVNET.PF0                    PF     pci_1    primary
[pci@500/pci@1/pci@0/pci@5/network@0]
    maxvfs = 7
/SYS/MB/NET2/IOVNET.PF1                    PF     pci_1    primary
[pci@500/pci@1/pci@0/pci@5/network@0,1]
    maxvfs = 7
```

2 Determine the device path of the boot disk that must be retained.

See Step 2 in "How to Create a Root Domain by Assigning a PCIe Bus" on page 72.

3 Determine the physical device to which the block device is linked.

See Step 3 in "How to Create a Root Domain by Assigning a PCIe Bus" on page 72.

4 Determine the network interface that is used by the system.

See Step 4 in "How to Create a Root Domain by Assigning a PCIe Bus" on page 72.

5 Determine the physical device to which the network interface is linked.

The following command uses the `igb0` network interface:

```
primary# ls -l /dev/igb0
lrwxrwxrwx   1 root     root          46 Jul 30 17:29 /dev/igb0 ->
../devices/pci@500/pci@0/pci@8/network@0:igb0
```

In this example, the physical device for the network interface used by the `primary` domain is connected to the PCIe endpoint device (`pci@500/pci@0/pci@8`), which corresponds to the listing of `MB/NET0` in Step 1. So, you do not want to remove this device from the `primary` domain. You can safely assign all other PCIe devices to other domains because they are not used by the `primary` domain.

If the network interface used by the `primary` domain is on a bus that you want to assign to another domain, the `primary` domain would need to be reconfigured to use a different network interface.

6 Remove the PCIe endpoint devices that you might use in I/O domains.

In this example, you can remove the PCIE2, PCIE3, PCIE4, and PCIE5 endpoint devices because they are not being used by the primary domain.

a. Remove the PCIe endpoint devices.

Caution – Do not remove the devices that are used or required by the primary domain. Do not remove a bus that has devices that are used by a domain, such as network ports or usbecm devices.

If you mistakenly remove the wrong devices, use the ldm cancel-reconf primary command to cancel the delayed reconfiguration on the primary domain.

You can remove multiple devices at one time to avoid multiple reboots.

```
primary# ldm start-reconf primary
primary# ldm set-io iov=on pci_1
All configuration changes for other domains are disabled until the primary
domain reboots, at which time the new configuration for the primary domain
will also take effect.
primary# ldm remove-io /SYS/MB/PCIE1 primary
------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
------------------------------------------------------------------------------
primary# ldm remove-io /SYS/MB/PCIE3 primary
------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
------------------------------------------------------------------------------
primary# ldm remove-io /SYS/MB/PCIE5 primary
------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
------------------------------------------------------------------------------
```

b. Save the new configuration to the service processor (SP).

The following command saves the configuration in a file called dio:

```
primary# ldm add-config dio
```

c. Reboot the system to reflect the removal of the PCIe endpoint devices.

```
primary# shutdown -i6 -g0 -y
```

7 Log in to the primary domain and verify that the PCIe endpoint devices are no longer assigned to the domain.

```
primary# ldm list-io
NAME                                      TYPE   BUS      DOMAIN   STATUS
----                                      ----   ---      ------   ------
niu_0                                     NIU    niu_0    primary
```

```
niu_1                                    NIU    niu_1    primary
pci_0                                    BUS    pci_0    primary
pci_1                                    BUS    pci_1    primary  IOV
/SYS/MB/PCIE0                            PCIE   pci_0    primary  OCC
/SYS/MB/PCIE2                            PCIE   pci_0    primary  OCC
/SYS/MB/PCIE4                            PCIE   pci_0    primary  OCC
/SYS/MB/PCIE6                            PCIE   pci_0    primary  EMP
/SYS/MB/PCIE8                            PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA                           PCIE   pci_0    primary  OCC
/SYS/MB/NET0                             PCIE   pci_0    primary  OCC
/SYS/MB/PCIE1                            PCIE   pci_1             OCC
/SYS/MB/PCIE3                            PCIE   pci_1             OCC
/SYS/MB/PCIE5                            PCIE   pci_1             OCC
/SYS/MB/PCIE7                            PCIE   pci_1    primary  EMP
/SYS/MB/PCIE9                            PCIE   pci_1    primary  EMP
/SYS/MB/NET2                             PCIE   pci_1    primary  OCC
/SYS/MB/NET0/IOVNET.PF0                  PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF1                  PF     pci_0    primary
/SYS/MB/NET2/IOVNET.PF0                  PF     pci_1    primary
/SYS/MB/NET2/IOVNET.PF1                  PF     pci_1    primary
```

Note – The `ldm list-io -l` output might show `SUNW,assigned-device` for the PCIe endpoint devices that were removed. Actual information is no longer available from the `primary` domain, but the domain to which the device is assigned has this information.

8 Assign a PCIe endpoint device to a domain.

a. Add the `PCIE3` device to the `ldg1` domain.

```
primary# ldm add-io /SYS/MB/PCIE3 ldg1
```

b. Bind and start the `ldg1` domain.

```
primary# ldm bind ldg1
primary# ldm start ldg1
LDom ldg1 started
```

9 Log in to the `ldg1` domain and verify that the device is available for use.

Verify that the network device is available and then configure the network device for use in the domain.

```
primary# dladm show-phys
LINK              MEDIA               STATE      SPEED  DUPLEX    DEVICE
net0              Ethernet            unknown    0      unknown   nxge0
net1              Ethernet            unknown    0      unknown   nxge1
net2              Ethernet            unknown    0      unknown   nxge2
net3              Ethernet            unknown    0      unknown   nxge3
```

◆ ◆ ◆ CHAPTER 9

Using Non-primary Root Domains

This chapter covers the following non-`primary` root domain topics:

Non-primary Root Domains Overview

A *root domain* has a PCIe root complex assigned to it. This domain owns the PCIe fabric and provides all fabric-related services, such as fabric error handling. A root domain is also an I/O domain, as it owns and has direct access to physical I/O devices. The `primary` domain is the default root domain.

You can perform direct I/O and SR-IOV operations on PCIe buses that are assigned to any root domain. You can now perform the following operations for all root domains, including non-`primary` root domains:

- Show the status of PCIe slots
- Show the SR-IOV physical functions that are present
- Assign a PCIe slot to an I/O domain or a root domain
- Remove a PCIe slot from an I/O domain or a root domain
- Create a virtual function from its physical function
- Destroy a virtual function
- Assign a virtual function to another domain
- Remove a virtual function from another domain

The Logical Domains Manager obtains the PCIe endpoint devices and SR-IOV physical function devices from the Logical Domains agents that run in the non-`primary` root domains. This information is cached while the root domain is down after it is first discovered but only until the root domain is booted.

You can perform direct I/O and SR-IOV operations only when the root domain is active. Logical Domains Manager operates on the actual devices that are present at that time. The cached data might be refreshed when the following operations occur:

- The Logical Domains agent is restarted in the specified root domain
- A hardware change, such as a hot-plug operation, is performed in the specified root domain

Use the `ldm list-io` command to view the PCIe endpoint device status. The output also shows the sub-devices and physical function devices from the root complexes that are owned by each non-`primary` root domain.

You can use apply the following commands to any root domain:

- `ldm add-io`
- `ldm remove-io`
- `ldm set-io`
- `ldm create-vf`
- `ldm destroy-vf`
- `ldm start-reconf`
- `ldm cancel-reconf`

Delayed reconfiguration support has been extended to include non-`primary` root domains. However, it can be used *only* to run the `ldm add-io`, `ldm remove-io`, `ldm set-io`, `ldm create-vf` and `ldm destroy-vf` commands. The delayed reconfiguration can be used for any operation that cannot be completed by using dynamic operations such as the following:

- Performing direct I/O operations
- Creating and destroying virtual functions from a physical function that does not meet the dynamic SR-IOV configuration requirements.

Caution – Plan ahead to minimize the number of reboots of the root domain, which minimizes downtime.

Non-primary Root Domain Requirements

Non-`primary` root domains can be used in addition to the control domain to provide direct I/O and SR-IOV capabilities to other domains. This feature is supported on the SPARC T4 servers, SPARC T5 servers, SPARC T7 series servers, SPARC M5 servers, SPARC M6 servers, SPARC M7 series servers, and Fujitsu M10 servers.

- **Hardware Requirements.**

  In addition to the PCIe cards for the direct I/O and SR-IOV described in `https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&id=1325454.1`, other PCIe

cards can be used, but not for DIO and SR-IOV. To determine which cards you can use on your platform, see your platform's hardware documentation.

- **Firmware Requirements.**

  SPARC T4 platforms must run at least version 8.4.0.a of the system firmware.

  SPARC T5 servers, SPARC M5 servers, and SPARC M6 servers must run at least version 9.1.0.*x* of the system firmware.

  SPARC T7 series servers and SPARC M7 series servers must run at least version 9.4.3 of the system firmware.

  Fujitsu M10 servers must run at least version XCP2210 of the system firmware.

- **Software Requirements.**

  Non-`primary` domains must run at least the Oracle Solaris 11.2 OS.

Non-primary Root Domain Limitations

Use of the non-`primary` root domain has the following limitations:

- An I/O domain cannot start if the associated root domain is not running.
- Support for delayed reconfiguration has been extended to the non-`primary` root domains. Only the following commands can be run until that root domain has been rebooted or the delayed reconfiguration has been canceled:
  - `ldm add-io`
  - `ldm remove-io`
  - `ldm set-io`
  - `ldm create-vf`
  - `ldm destroy-vf`
- The root domain must be active and booted to perform the following operations:
  - Creating and destroying SR-IOV virtual functions
  - Adding and removing PCIe slots
  - Adding and removing SR-IOV virtual functions
- You must initiate a delayed reconfiguration on the root domain when you perform the `ldm add-io` and `ldm remove-io` direct I/O operations for PCIe slots.
- When your configuration does not meet the dynamic I/O virtualization requirements, you must use delayed reconfiguration for the following SR-IOV virtual function operations:
  - `ldm create-vf`
  - `ldm destroy-vf`
  - `ldm add-io`
  - `ldm remove-io`
  - `ldm set-io`

- The reboot of a root domain affects any I/O domain that has a device from the PCIe buses that the root domain owns. See "Rebooting the Root Domain With PCIe Endpoints Configured" on page 149.
- You cannot assign an SR-IOV virtual function or a PCIe slot from one root domain to another root domain. This limitation prevents circular dependencies.

Non-primary Root Domain Examples

The following examples describe how to enable I/O virtualization for a PCIe bus, manage direct I/O devices on non-`primary` root domains, and manage SR-IOV virtual functions on non-`primary` root domains.

Enabling I/O Virtualization for a PCIe Bus

The following example shows how to enable I/O virtualization by using the `ldm add-io` and `ldm set-io` commands.

The following SPARC T4-2 I/O configuration shows that bus `pci_1` already has been removed from the `primary` domain.

```
primary# ldm list-io
NAME                                    TYPE   BUS      DOMAIN   STATUS
----                                    ----   ---      ------   ------
pci_0                                   BUS    pci_0    primary  IOV
pci_1                                   BUS    pci_1
niu_0                                   NIU    niu_0    primary
niu_1                                   NIU    niu_1    primary
/SYS/MB/PCIE0                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE2                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE4                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE6                           PCIE   pci_0    primary  EMP
/SYS/MB/PCIE8                           PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA                          PCIE   pci_0    primary  OCC
/SYS/MB/NET0                            PCIE   pci_0    primary  OCC
/SYS/MB/PCIE1                           PCIE   pci_1             UNK
/SYS/MB/PCIE3                           PCIE   pci_1             UNK
/SYS/MB/PCIE5                           PCIE   pci_1             UNK
/SYS/MB/PCIE7                           PCIE   pci_1             UNK
/SYS/MB/PCIE9                           PCIE   pci_1             UNK
/SYS/MB/NET2                            PCIE   pci_1             UNK
/SYS/MB/NET0/IOVNET.PF0                 PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF1                 PF     pci_0    primary
```

The following listing shows that the guest domains are in the bound state:

```
primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
primary          active     -n-cv-  UART    8     8G       0.6%  0.6%  8m
```

```
rootdom1         bound      ------  5000    8     4G
ldg2             bound      ------  5001    8     4G
ldg3             bound      ------  5002    8     4G
```

The following `ldm add-io` command adds the `pci_1` bus to the `rootdom1` domain with I/O virtualization enabled for that bus. The `ldm start` command starts the `rootdom1` domain.

```
primary# ldm add-io iov=on pci_1 rootdom1
primary# ldm start rootdom1
LDom rootdom1 started
```

If a specified PCIe bus is assigned already to a root domain, use the `ldm set-io` command to enable I/O virtualization.

```
primary# ldm start-reconf rootdom1
primary# ldm set-io iov=on pci_1
primary# ldm stop-domain -r rootdom1
```

The root domain must be running its OS before you can configure the I/O devices. Connect to the console of the `rootdom1` guest domain and then boot the OS of the `rootdom1` root domain if your guest domains are not already set to autoboot.

```
primary# telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Connecting to console "rootdom1" in group "rootdom1" ....
Press ~? for control options ..
ok> boot
...
primary#
```

The following command shows that the `pci_1` PCIe bus and its children are now owned by the `rootdom1` root domain.

```
primary# ldm list-io
NAME                                      TYPE   BUS      DOMAIN   STATUS
----                                      ----   ---      ------   ------
pci_0                                     BUS    pci_0    primary  IOV
pci_1                                     BUS    pci_1    rootdom1     IOV
niu_0                                     NIU    niu_0    primary
niu_1                                     NIU    niu_1    primary
/SYS/MB/PCIE0                             PCIE   pci_0    primary  OCC
/SYS/MB/PCIE2                             PCIE   pci_0    primary  OCC
/SYS/MB/PCIE4                             PCIE   pci_0    primary  OCC
/SYS/MB/PCIE6                             PCIE   pci_0    primary  EMP
/SYS/MB/PCIE8                             PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA                            PCIE   pci_0    primary  OCC
/SYS/MB/NET0                              PCIE   pci_0    primary  OCC
/SYS/MB/PCIE1                             PCIE   pci_1    rootdom1     OCC
/SYS/MB/PCIE3                             PCIE   pci_1    rootdom1     OCC
/SYS/MB/PCIE5                             PCIE   pci_1    rootdom1     OCC
/SYS/MB/PCIE7                             PCIE   pci_1    rootdom1     OCC
/SYS/MB/PCIE9                             PCIE   pci_1    rootdom1     EMP
```

```
/SYS/MB/NET2                              PCIE   pci_1    rootdom1     OCC
/SYS/MB/NET0/IOVNET.PF0                   PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF1                   PF     pci_0    primary
/SYS/MB/PCIE5/IOVNET.PF0                  PF     pci_1    rootdom1
/SYS/MB/PCIE5/IOVNET.PF1                  PF     pci_1    rootdom1
/SYS/MB/NET2/IOVNET.PF0                   PF     pci_1    rootdom1
/SYS/MB/NET2/IOVNET.PF1                   PF     pci_1    rootdom1
```

Managing Direct I/O Devices on Non-primary Root Domains

The following example shows how to manage direct I/O devices on non-`primary` root domains.

The following command produces an error because it attempts to remove a slot from the root domain while it is still active:

```
primary# ldm remove-io /SYS/MB/PCIE7 ldg1
Dynamic I/O operations on PCIe slots are not supported.
Use start-reconf command to trigger delayed reconfiguration and make I/O
changes statically.
```

The following command shows the correct method of removing a slot by first initiating a delayed reconfiguration on the root domain.

```
primary# ldm start-reconf ldg1
Initiating a delayed reconfiguration operation on the ldg1 domain.
All configuration changes for other domains are disabled until the ldg1
domain reboots, at which time the new configuration for the ldg1 domain
will also take effect.
primary# ldm remove-io /SYS/MB/PCIE7 ldg1
--------------------------------------------------------------------------------
Notice: The ldg1 domain is in the process of a delayed reconfiguration.
Any changes made to the ldg1 domain will only take effect after it reboots.
--------------------------------------------------------------------------------
primary# ldm stop-domain -r ldg1
```

The following `ldm list-io` command verifies that the `/SYS/MB/PCIE7` slot is no longer on the root domain.

```
primary# ldm list-io
NAME                                      TYPE   BUS      DOMAIN   STATUS
----                                      ----   ---      ------   ------
pci_0                                     BUS    pci_0    primary  IOV
pci_1                                     BUS    pci_1    ldg1     IOV
niu_0                                     NIU    niu_0    primary
niu_1                                     NIU    niu_1    primary
/SYS/MB/PCIE0                             PCIE   pci_0    primary  OCC
/SYS/MB/PCIE2                             PCIE   pci_0    primary  OCC
/SYS/MB/PCIE4                             PCIE   pci_0    primary  OCC
/SYS/MB/PCIE6                             PCIE   pci_0    primary  EMP
/SYS/MB/PCIE8                             PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA                            PCIE   pci_0    primary  OCC
```

```
/SYS/MB/NET0                          PCIE   pci_0    primary  OCC
/SYS/MB/PCIE1                         PCIE   pci_1    ldg1     OCC
/SYS/MB/PCIE3                         PCIE   pci_1    ldg1     OCC
/SYS/MB/PCIE5                         PCIE   pci_1    ldg1     OCC
/SYS/MB/PCIE7                         PCIE   pci_1             OCC
/SYS/MB/PCIE9                         PCIE   pci_1    ldg1     EMP
/SYS/MB/NET2                          PCIE   pci_1    ldg1     OCC
/SYS/MB/NET0/IOVNET.PF0               PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF1               PF     pci_0    primary
/SYS/MB/PCIE5/IOVNET.PF0              PF     pci_1    ldg1
/SYS/MB/PCIE5/IOVNET.PF1              PF     pci_1    ldg1
/SYS/MB/NET2/IOVNET.PF0               PF     pci_1    ldg1
/SYS/MB/NET2/IOVNET.PF1               PF     pci_1    ldg1
```

The following commands assign the `/SYS/MB/PCIE7` slot to the `ldg2` domain. The `ldm start` command starts the `ldg2` domain.

```
primary# ldm add-io /SYS/MB/PCIE7 ldg2
primary# ldm start ldg2
LDom ldg2 started
```

Managing SR-IOV Virtual Functions on Non-primary Root Domains

These commands create two virtual functions from each of the two physical functions that belong to the non-`primary` root domain.

```
primary# ldm create-vf /SYS/MB/PCIE5/IOVNET.PF0
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF0
primary# ldm create-vf /SYS/MB/PCIE5/IOVNET.PF0
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF1
primary# ldm create-vf /SYS/MB/NET2/IOVNET.PF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF0
primary# ldm create-vf /SYS/MB/NET2/IOVNET.PF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF1
```

You can also use the `-n` option to create the two virtual functions by using the following two commands:

```
primary# ldm create-vf -n 2 /SYS/MB/PCIE5/IOVNET.PF0
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF0
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF1
primary# ldm create-vf -n 2 /SYS/MB/NET2/IOVNET.PF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF0
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF1
```

If you were unable to dynamically create the virtual functions on a given physical function, initiate a delayed reconfiguration to create them statically.

```
primary# ldm start-reconf ldg1
primary# ldm create-vf /SYS/MB/PCIE5/IOVNET.PF0
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF0
```

```
primary# ldm create-vf /SYS/MB/PCIE5/IOVNET.PF0
Created new vf: /SYS/MB/PCIE5/IOVNET.PF0.VF1
primary# ldm create-vf /SYS/MB/NET2/IOVNET.PF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF0
primary# ldm create-vf /SYS/MB/NET2/IOVNET.PF1
Created new vf: /SYS/MB/NET2/IOVNET.PF1.VF1
```

Then reboot the root domain, `ldg1`, to effect the changes.

```
primary# ldm stop-domain -r ldg1
```

The following command shows the new virtual functions.

```
primary# ldm list-io
NAME                                    TYPE   BUS      DOMAIN   STATUS
----                                    ----   ---      ------   ------
pci_0                                   BUS    pci_0    primary  IOV
pci_1                                   BUS    pci_1    ldg1     IOV
niu_0                                   NIU    niu_0    primary
niu_1                                   NIU    niu_1    primary
/SYS/MB/PCIE0                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE2                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE4                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE6                           PCIE   pci_0    primary  EMP
/SYS/MB/PCIE8                           PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA                          PCIE   pci_0    primary  OCC
/SYS/MB/NET0                            PCIE   pci_0    primary  OCC
/SYS/MB/PCIE1                           PCIE   pci_1    ldg1     OCC
/SYS/MB/PCIE3                           PCIE   pci_1    ldg1     OCC
/SYS/MB/PCIE5                           PCIE   pci_1    ldg1     OCC
/SYS/MB/PCIE7                           PCIE   pci_1    ldg2     OCC
/SYS/MB/PCIE9                           PCIE   pci_1    ldg1     EMP
/SYS/MB/NET2                            PCIE   pci_1    ldg1     OCC
/SYS/MB/NET0/IOVNET.PF0                 PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF1                 PF     pci_0    primary
/SYS/MB/PCIE5/IOVNET.PF0                PF     pci_1    ldg1
/SYS/MB/PCIE5/IOVNET.PF1                PF     pci_1    ldg1
/SYS/MB/NET2/IOVNET.PF0                 PF     pci_1    ldg1
/SYS/MB/NET2/IOVNET.PF1                 PF     pci_1    ldg1
/SYS/MB/PCIE5/IOVNET.PF0.VF0            VF     pci_1
/SYS/MB/PCIE5/IOVNET.PF0.VF1            VF     pci_1
/SYS/MB/NET2/IOVNET.PF1.VF0             VF     pci_1
/SYS/MB/NET2/IOVNET.PF1.VF1             VF     pci_1
```

The following command dynamically adds the `/SYS/MB/PCIE5/IOVNET.PF0.VF1` virtual function to the `ldg1` non-`primary` root domain:

```
primary# ldm add-io /SYS/MB/PCIE5/IOVNET.PF0.VF1 ldg1
```

The following command dynamically adds the `/SYS/MB/NET2/IOVNET.PF1.VF0` virtual function to the `ldg2` domain:

```
primary# ldm add-io /SYS/MB/NET2/IOVNET.PF1.VF0 ldg2
```

The following command adds the `/SYS/MB/NET2/IOVNET.PF1.VF1` virtual function to the bound `ldg3` domain:

```
primary# ldm add-io /SYS/MB/NET2/IOVNET.PF1.VF1 ldg3
primary# ldm start ldg3
LDom ldg3 started
```

Connect to the console of the `ldg3` domain and then boot its OS.

The following output shows that all the assignments appear as expected. One virtual function is unassigned so it can be assigned dynamically to the `ldg1`, `ldg2`, or `ldg3` domain.

```
# ldm list-io
NAME                                    TYPE   BUS      DOMAIN   STATUS
----                                    ----   ---      ------   ------
pci_0                                   BUS    pci_0    primary  IOV
pci_1                                   BUS    pci_1    ldg1     IOV
niu_0                                   NIU    niu_0    primary
niu_1                                   NIU    niu_1    primary
/SYS/MB/PCIE0                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE2                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE4                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE6                           PCIE   pci_0    primary  EMP
/SYS/MB/PCIE8                           PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA                          PCIE   pci_0    primary  OCC
/SYS/MB/NET0                            PCIE   pci_0    primary  OCC
/SYS/MB/PCIE1                           PCIE   pci_1    ldg1     OCC
/SYS/MB/PCIE3                           PCIE   pci_1    ldg1     OCC
/SYS/MB/PCIE5                           PCIE   pci_1    ldg1     OCC
/SYS/MB/PCIE7                           PCIE   pci_1    ldg2     OCC
/SYS/MB/PCIE9                           PCIE   pci_1    ldg1     EMP
/SYS/MB/NET2                            PCIE   pci_1    ldg1     OCC
/SYS/MB/NET0/IOVNET.PF0                 PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF1                 PF     pci_0    primary
/SYS/MB/PCIE5/IOVNET.PF0                PF     pci_1    ldg1
/SYS/MB/PCIE5/IOVNET.PF1                PF     pci_1    ldg1
/SYS/MB/NET2/IOVNET.PF0                 PF     pci_1    ldg1
/SYS/MB/NET2/IOVNET.PF1                 PF     pci_1    ldg1
/SYS/MB/PCIE5/IOVNET.PF0.VF0            VF     pci_1
/SYS/MB/PCIE5/IOVNET.PF0.VF1            VF     pci_1    ldg1
/SYS/MB/NET2/IOVNET.PF1.VF0             VF     pci_1    ldg2
/SYS/MB/NET2/IOVNET.PF1.VF1             VF     pci_1    ldg3
```

◆ ◆ ◆ CHAPTER 10

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 is sent. The virtual disk back end is exported from a service domain by the virtual disk server (`vds`) driver. The `vds` driver communicates with the virtual disk client (`vdc`) driver in the guest domain through the hypervisor using a logical domain channel (LDC). Finally, a virtual disk appears as `/dev/[r]dsk/cXdYsZ` devices in the guest domain.

Note – You can refer to a disk either by using `/dev/dsk` or `/dev/rdsk` as part of the disk path name. Either reference produces the same result.

Caution – Do not use the `d0` device to represent the entire disk. This device represents the entire disk only when the disk has an EFI label and not a VTOC label. Using the `d0` device results in the virtual disk being a single-slice disk, which might cause you to corrupt the disk label if you write the beginning of the disk.

Instead, use the `s2` slice to virtualize the entire disk. The `s2` slice is independent of the label.

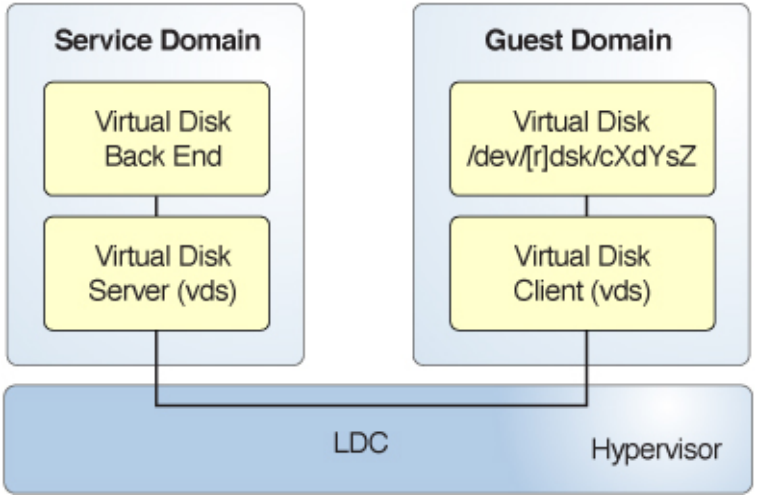
The virtual disk back end can be physical or logical. Physical devices can include the following:

- Physical disk or disk logical unit number (LUN)
- Physical disk slice

Logical devices can be any of the following:

- A file on a local file system, such as ZFS or UFS, or on a remote file system that is made available by means of NFS
- A logical volume from a volume manager, such as ZFS, VxVM, or Solaris Volume Manager
- Any disk pseudo device accessible from the service domain

FIGURE 10–1 Virtual Disks With Oracle VM Server for SPARC



Virtual Disk Identifier and Device Name

When you use the `ldm add-vdisk` command to add a virtual disk to a domain, you can specify its device number by setting the `id` property.

`ldm add-vdisk [id=`*disk-id*`]` *disk-name volume-name*`@`*service-name domain-name*

Each virtual disk of a domain has a unique device number that is assigned when the domain is bound. If a virtual disk is added with an explicit device number (by setting the `id` property), the specified device number is used. Otherwise, the system automatically assigns the lowest device number available. In that case, the device number assigned depends on how virtual disks were added to the domain. The device number eventually assigned to a virtual disk is visible in the output of the `ldm list-bindings` command when a domain is bound.

When a domain with virtual disks is running the Oracle Solaris OS, each virtual disk appears in the domain as a `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.

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 175 for a description of virtual disk options. See "Virtual Disk Timeout" on page 190 for a description of the virtual disk timeout.

A virtual disk back end can be exported multiple times either through the same or different virtual disk servers. Each exported instance of the virtual disk back end can then be assigned to either the same or different guest domains.

When a virtual disk back end is exported multiple times, it should not be exported with the exclusive (`excl`) option. Specifying the `excl` option will only allow exporting the back end once. The back end can be safely exported multiple times as a read-only device with the `ro` option.

Assigning a virtual disk device to a domain creates an implicit dependency on the domain providing the virtual disk service. You can view these dependencies or view domains that depend on the virtual disk service by using the `ldm list-dependencies` command. See "Listing Domain I/O Dependencies" on page 382.

How to Add a Virtual Disk

1. **Export the virtual disk back end from a service domain.**

   ```
   ldm add-vsdev [-fq] [options={ro,slice,excl}] [mpgroup=mpgroup] \
   backend volume-name@service-name
   ```

2. **Assign the back end to a guest domain.**

```
ldm add-vdisk [timeout=seconds] [id=disk-id] disk-name volume-name@service-name domain-name
```

You can specify a custom ID of a new virtual disk device by setting the `id` property. By default, ID values are automatically generated, so set this property if you need to match an existing device name in the OS. See "Virtual Disk Identifier and Device Name" on page 171.

Note – A back end is actually exported from the service domain and assigned to the guest domain when the guest domain (*domain-name*) is bound.

How to Export a Virtual Disk Back End Multiple Times

Caution – When a virtual disk back end is exported multiple times, applications running on guest domains and using that virtual disk are responsible for coordinating and synchronizing concurrent write access to ensure data coherency.

The following example describes how to add the same virtual disk to two different guest domains through the same virtual disk service.

1. **Export the virtual disk back end two times from a service domain.**

   ```
   ldm add-vsdev [options={ro,slice}] backend volume1@service-name
   # ldm add-vsdev -f [options={ro,slice}] backend volume2@service-name
   ```

   Note that the second `ldm add-vsdev` command uses the `-f` option to force the second export of the back end. Use this option when using the same back-end path for both commands and when the virtual disk servers are located on the same service domain.

2. **Assign the exported back end to each guest domain.**

   The *disk-name* can be different for `ldom1` and `ldom2`.

   ```
   ldm add-vdisk [timeout=seconds] disk-name volume1@service-name ldom1
   # ldm add-vdisk [timeout=seconds] disk-name volume2@service-name ldom2
   ```

How to Change Virtual Disk Options

For more information about virtual disk options, see "Virtual Disk Back End Options" on page 175.

- **After a back end is exported from the service domain, you can change the virtual disk options.**

  ```
  primary# ldm set-vdsdev options=[{ro,slice,excl}] volume-name@service-name
  ```

How to Change the Timeout Option

For more information about virtual disk options, see "Virtual Disk Back End Options" on page 175.

- **After a virtual disk is assigned to a guest domain, you can change the timeout of the virtual disk.**

  ```
  primary# ldm set-vdisk timeout=seconds disk-name domain-name
  ```

How to Remove a Virtual Disk

1. **Remove a virtual disk from a guest domain.**

   ```
   primary# ldm rm-vdisk disk-name domain-name
   ```

2. **Stop exporting the corresponding back end from the service domain.**

   ```
   primary# ldm rm-vdsdev volume-name@service-name
   ```

Virtual Disk Appearance

When a back end is exported as a virtual disk, it can appear in the guest domain either as a full disk or as a single-slice disk. The way it appears depends on the type of the back end and on the options used to export it.

Caution – The SPARC T7 series server and SPARC M7 series server introduce Non-Volatile Memory Express (NVMe) storage, which can be a disk drive or a Flash Accelerator F160 PCIe card. This disk type cannot be used as a full disk to build a virtual disk back end.

When using such a disk, use a slice of the disk or the `slice` option when creating the `vdsdev`. For example, after you partition the disk, use one of the following commands:

```
ldm add-vdsdev dev/dsk/c14t1d0s6 volume3@primary-vds4
```

Or:

```
ldm add-vdsdev options=slice /dev/dsk/c14t1d0s2 volume3@primary-vds4
```

Caution – Single-slice disks do not have device IDs. If a device ID is required, use a full physical disk backend.

Full Disk

When a back end is exported to a domain as a full disk, it appears in that domain as a regular disk with eight slices (`s0` to `s7`). This type of disk is visible with the `format`(1M) command. The disk's partition table can be changed using either the `fmthard` or `format` command.

A full disk is also visible to the OS installation software and can be selected as a disk onto which the OS can be installed.

Any back end can be exported as a full disk except physical disk slices that can be exported only as single-slice disks.

Single-Slice Disk

When a back end is exported to a domain as a single-slice disk, it appears in that domain as a regular disk with eight slices (`s0` to `s7`). However, only the first slice (`s0`) is usable. This type of disk is visible with the `format`(1M) command, but the disk's partition table cannot be changed.

A single-slice disk is also visible from the OS installation software and can be selected as a disk onto which you can install the OS. In that case, if you install the OS using the UNIX File System (UFS), then only the root partition (`/`) must be defined, and this partition must use all the disk space.

Any back end can be exported as a single-slice disk except physical disks that can only be exported as full disks.

Note – Prior to the Oracle Solaris 10 10/08 OS release, a single-slice disk appeared as a disk with a single partition (`s0`). This type of disk was not visible with the `format` command. The disk also was not visible from the OS installation software and could not be selected as a disk device onto which the OS could be installed.

Virtual Disk Back End Options

Different options can be specified when exporting a virtual disk back end. These options are indicated in the `options=` argument of the `ldm add-vdsdev` command as a comma-separated list. The valid options are: `ro`, `slice`, and `excl`.

Read-only (ro) Option

The read-only (`ro`) option specifies that the back end is to be exported as a read-only device. In that case, the virtual disk assigned to the guest domain can be accessed only for read operations, and any write operation to the virtual disk will fail.

Exclusive (excl) Option

The exclusive (`excl`) option specifies that the back end in the service domain has to be opened exclusively by the virtual disk server when it is exported as a virtual disk to another domain. When a back end is opened exclusively, it is not accessible by other applications in the service domain. This restriction prevents the applications running in the service domain from inadvertently using a back end that is also being used by a guest domain.

Note – Some drivers do not honor the `excl` option and will disallow some virtual disk back ends from being opened exclusively. The `excl` option is known to work with physical disks and slices, but the option does not work with files. It might work with pseudo devices, such as disk volumes. If the driver of the back end does not honor the exclusive open, the back end `excl` option is ignored, and the back end is not opened exclusively.

Because the `excl` option prevents applications running in the service domain from accessing a back end exported to a guest domain, do not set the `excl` option in the following situations:

- When guest domains are running, if you want to be able to use commands such as `format` or `luxadm` to manage physical disks, then do not export these disks with the `excl` option.
- When you export a Solaris Volume Manager volume, such as a RAID or a mirrored volume, do not set the `excl` option. Otherwise, this can prevent Solaris Volume Manager from starting some recovery operation in case a component of the RAID or mirrored volume fails. See "Using Virtual Disks With Solaris Volume Manager" on page 197 for more information.
- If the Veritas Volume Manager (VxVM) is installed in the service domain and Veritas Dynamic Multipathing (VxDMP) is enabled for physical disks, then physical disks have to be exported without the (non-default) `excl` option. Otherwise, the export fails, because the virtual disk server (`vds`) is unable to open the physical disk device. See "Using Virtual Disks When VxVM Is Installed" on page 197 for more information.
- If you are exporting the same virtual disk back end multiple times from the same virtual disk service, see "How to Export a Virtual Disk Back End Multiple Times" on page 172 for more information.

By default, the back end is opened non-exclusively. That way the back end still can be used by applications running in the service domain while it is exported to another domain.

Slice (slice) Option

A back end is normally exported either as a full disk or as a single-slice disk depending on its type. If the `slice` option is specified, then the back end is forcibly exported as a single-slice disk.

This option is useful when you want to export the raw content of a back end. For example, if you have a ZFS or Solaris Volume Manager volume where you have already stored data and you want your guest domain to access this data, then you should export the ZFS or Solaris Volume Manager volume using the `slice` option.

For more information about this option, see "Virtual Disk Back End" on page 176.

Virtual Disk Back End

The virtual disk back end is the location where data of a virtual disk are stored. The back end can be a disk, a disk slice, a file, or a volume, such as ZFS, Solaris Volume Manager, or VxVM. A back end appears in a guest domain either as a full disk or as single-slice disk, depending on whether the `slice` option is set when the back end is exported from the service domain. By default, a virtual disk back end is exported non-exclusively as a readable-writable full disk.

Physical Disk or Disk LUN

A physical disk or disk LUN is always exported as a full disk. In that case, virtual disk drivers (`vds` and `vdc`) forward I/O from the virtual disk and act as a pass-through to the physical disk or disk LUN.

A physical disk or disk LUN is exported from a service domain by exporting the device that corresponds to the slice 2 (`s2`) of that disk without setting the `slice` option. If you export the slice 2 of a disk with the `slice` option, only this slice is exported and not the entire disk.

▼ How to Export a Physical Disk as a Virtual Disk

Caution – When configuring virtual disks, ensure that each virtual disk references a distinct physical (back-end) resource, such as a physical disk, a disk slice, a file, or a volume.

Some disks, such as FibreChannel and SAS, have a "dual-ported" nature, which means that the same disk can be referenced by two different paths. Ensure that the paths you assign to different domains do not refer to the same physical disk.

1 Export a physical disk as a virtual disk.

For example, to export the physical disk `c1t48d0` as a virtual disk, you must export slice 2 of that disk (`c1t48d0s2`).

```
primary# ldm add-vdsdev /dev/dsk/c1t48d0s2 c1t48d0@primary-vds0
```

2 Assign the disk to a guest domain.

For example, assign the disk (`pdisk`) to guest domain `ldg1`.

```
primary# ldm add-vdisk pdisk c1t48d0@primary-vds0 ldg1
```

3 After the guest domain is started and running the Oracle Solaris OS, verify that the disk is accessible and is a full disk.

A full disk is a regular disk that has eight (8) slices.

For example, the disk being checked is `c0d1`.

```
ldg1# ls -1 /dev/dsk/c0d1s*
/dev/dsk/c0d1s0
/dev/dsk/c0d1s1
/dev/dsk/c0d1s2
/dev/dsk/c0d1s3
/dev/dsk/c0d1s4
/dev/dsk/c0d1s5
/dev/dsk/c0d1s6
/dev/dsk/c0d1s7
```

Physical Disk Slice

A physical disk slice is always exported as a single-slice disk. In that case, virtual disk drivers (`vds` and `vdc`) forward I/O from the virtual disk and act as a pass-through to the physical disk slice.

A physical disk slice is exported from a service domain by exporting the corresponding slice device. If the device is different from slice 2 then it is automatically exported as a single-slice disk regardless of whether you specify the `slice` option. If the device is the slice 2 of the disk, you must set the `slice` option to export only slice 2 as a single-slice disk. Otherwise, the entire disk is exported as full disk.

Caution – The SPARC T7 series server and SPARC M7 series server introduce Non-Volatile Memory Express (NVMe) storage, which can be a disk drive or a Flash Accelerator F160 PCIe card. This disk type cannot be used as a full disk to build a virtual disk back end.

When using such a disk, use a slice of the disk or the `slice` option when creating the `vdsdev`.

▼ How to Export a Physical Disk Slice as a Virtual Disk

1 Export a slice of a physical disk as a virtual disk.

For example, to export slice 0 of the physical disk `c1t57d0` as a virtual disk, you must export the device that corresponds to that slice (`c1t57d0s0`) as follows.

```
primary# ldm add-vdsdev /dev/dsk/c1t57d0s0 c1t57d0s0@primary-vds0
```

You do not need to specify the `slice` option because a slice is always exported as a single-slice disk.

2 Assign the disk to a guest domain.

For example, assign the disk (`pslice`) to guest domain `ldg1`.

```
primary# ldm add-vdisk pslice c1t57d0s0@primary-vds0 ldg1
```

3 After the guest domain is started and running the Oracle Solaris OS, you can list the disk (`c0d13`, for example) and see that the disk is accessible.

```
ldg1# ls -1 /dev/dsk/c0d13s*
/dev/dsk/c0d13s0
/dev/dsk/c0d13s1
/dev/dsk/c0d13s2
/dev/dsk/c0d13s3
/dev/dsk/c0d13s4
/dev/dsk/c0d13s5
/dev/dsk/c0d13s6
/dev/dsk/c0d13s7
```

Although there are eight devices, because the disk is a single-slice disk, only the first slice (`s0`) is usable.

▼ How to Export Slice 2

- **To export slice 2 (disk `c1t57d0s2`, for example) you must specify the `slice` option. Otherwise, the entire disk is exported.**

```
primary# ldm add-vdsdev options=slice /dev/dsk/c1t57d0s2 c1t57d0s2@primary-vds0
```

File and Volume Exporting

A file or volume (for example from ZFS or Solaris Volume Manager) is exported either as a full disk or as single-slice disk depending on whether the `slice` option is set.

File or Volume Exported as a Full Disk

If you do not set the `slice` option, a file or volume is exported as a full disk. In that case, virtual disk drivers (`vds` and `vdc`) forward I/O from the virtual disk and manage the partitioning of the virtual disk. The file or volume eventually becomes a disk image containing data from all slices of the virtual disk and the metadata used to manage the partitioning and disk structure.

When a blank file or volume is exported as full disk, it appears in the guest domain as an unformatted disk; that is, a disk with no partition. Then you need to run the `format` command in the guest domain to define usable partitions and to write a valid disk label. Any I/O to the virtual disk fails while the disk is unformatted.

Note – You must run the `format` command in the guest domain to create partitions.

▼ How to Export a File as a Full Disk

1. **From the service domain, create a file (`fdisk0` for example) to use as the virtual disk.**

```
service# mkfile 100m /ldoms/domain/test/fdisk0
```

The size of the file defines the size of the virtual disk. This example creates a 100-Mbyte blank file to get a 100-Mbyte virtual disk.

2. **From the control domain, export the file as a virtual disk.**

```
primary# ldm add-vdsdev /ldoms/domain/test/fdisk0 fdisk0@primary-vds0
```

In this example, the `slice` option is not set, so the file is exported as a full disk.

3. **From the control domain, assign the disk to a guest domain.**

For example, assign the disk (`fdisk`) to guest domain `ldg1`.

```
primary# ldm add-vdisk fdisk fdisk0@primary-vds0 ldg1
```

4 **After the guest domain is started and running the Oracle Solaris OS, verify that the disk is accessible and is a full disk.**

A full disk is a regular disk with eight slices.

The following example shows how to list the disk, `c0d5`, and verify that it is accessible and is a full disk.

```
ldg1# ls -1 /dev/dsk/c0d5s*
/dev/dsk/c0d5s0
/dev/dsk/c0d5s1
/dev/dsk/c0d5s2
/dev/dsk/c0d5s3
/dev/dsk/c0d5s4
/dev/dsk/c0d5s5
/dev/dsk/c0d5s6
/dev/dsk/c0d5s7
```

▼ How to Export a ZFS Volume as a Full Disk

1 **Create a ZFS volume to use as a full disk.**

The following example shows how to create a ZFS volume, `zdisk0`, to use as a full disk:

```
service# zfs create -V 100m ldoms/domain/test/zdisk0
```

The size of the volume defines the size of the virtual disk. This example creates a 100-Mbyte volume to result in a 100-Mbyte virtual disk.

2 **From the control domain, export the corresponding device to that ZFS volume.**

```
primary# ldm add-vdsdev /dev/zvol/dsk/ldoms/domain/test/zdisk0 \
zdisk0@primary-vds0
```

In this example, the `slice` option is not set so the file is exported as a full disk.

3 **From the control domain, assign the volume to a guest domain.**

The following example shows how to assign the volume, `zdisk0`, to the guest domain `ldg1`:

```
primary# ldm add-vdisk zdisk0 zdisk0@primary-vds0 ldg1
```

4 **After the guest domain is started and running the Oracle Solaris OS, verify that the disk is accessible and is a full disk.**

A full disk is a regular disk with eight slices.

The following example shows how to list the disk, `c0d9`, and verify that it is accessible and is a full disk:

```
ldg1# ls -1 /dev/dsk/c0d9s*
/dev/dsk/c0d9s0
/dev/dsk/c0d9s1
/dev/dsk/c0d9s2
/dev/dsk/c0d9s3
/dev/dsk/c0d9s4
/dev/dsk/c0d9s5
```

```
/dev/dsk/c0d9s6
/dev/dsk/c0d9s7
```

File or Volume Exported as a Single-Slice Disk

If the `slice` option is set, then the file or volume is exported as a single-slice disk. In that case, the virtual disk has only one partition (`s0`), which is directly mapped to the file or volume back end. The file or volume only contains data written to the virtual disk with no extra data like partitioning information or disk structure.

When a file or volume is exported as a single-slice disk, the system simulates a fake disk partitioning which makes that file or volume appear as a disk slice. Because the disk partitioning is simulated, you do not create partitioning for that disk.

▼ How to Export a ZFS Volume as a Single-Slice Disk

1 Create a ZFS volume to use as a single-slice disk.

The following example shows how to create a ZFS volume, `zdisk0`, to use as a single-slice disk.

```
service# zfs create -V 100m ldoms/domain/test/zdisk0
```

The size of the volume defines the size of the virtual disk. This example creates a 100-Mbyte volume to get a 100-Mbyte virtual disk.

2 From the control domain, export the corresponding device to that ZFS volume, and set the `slice` option so that the volume is exported as a single-slice disk.

```
primary# ldm add-vdsdev options=slice /dev/zvol/dsk/ldoms/domain/test/zdisk0 \
zdisk0@primary-vds0
```

3 From the control domain, assign the volume to a guest domain.

The following shows how to assign the volume, `zdisk0`, to guest domain `ldg1`.

```
primary# ldm add-vdisk zdisk0 zdisk0@primary-vds0 ldg1
```

4 After the guest domain is started and running the Oracle Solaris OS, you can list the disk (`c0d9`, for example) and see that the disk is accessible and is a single-slice disk (`s0`).

```
ldg1# ls -1 /dev/dsk/c0d9s*
/dev/dsk/c0d9s0
/dev/dsk/c0d9s1
/dev/dsk/c0d9s2
/dev/dsk/c0d9s3
/dev/dsk/c0d9s4
/dev/dsk/c0d9s5
/dev/dsk/c0d9s6
/dev/dsk/c0d9s7
```

Exporting Volumes and Backward Compatibility

If you have a configuration exporting volumes as virtual disks, volumes are now exported as full disks instead of single-slice disks. To preserve the old behavior and to have your volumes exported as single-slice disks, you need to do either of the following:

- Use the `ldm set-vdsdev` command in Oracle VM Server for SPARC 3.3 software, and set the `slice` option for all volumes you want to export as single-slice disks. See the `ldm(1M)` man page.
- Add the following line to the `/etc/system` file on the service domain.

  ```
  set vds:vd_volume_force_slice = 1
  ```

  For information about correctly creating or updating `/etc/system` property values, see "Updating Property Values in the `/etc/system` File" on page 362.

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
Volume, including ZFS, Solaris Volume Manager, or VxVM	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.

Guidelines for Exporting Files and Disk Slices as Virtual Disks

This section includes guidelines for exporting a file and a disk slice as a virtual disk.

Using the Loopback File (lofi) Driver

Using the loopback file (`lofi`) driver to export a file as a virtual disk adds an extra driver layer and affects performance of the virtual disk. Instead, you can directly export a file as a full disk or as a single-slice disk. See "File and Volume Exporting" on page 179.

Directly or Indirectly Exporting a Disk Slice

To export a slice as a virtual disk either directly or indirectly (for example through a Solaris Volume Manager volume), ensure that the slice does not start on the first block (block 0) of the physical disk by using the `prtvtoc` command.

If you directly or indirectly export a disk slice which starts on the first block of a physical disk, you might overwrite the partition table of the physical disk and make all partitions of that disk inaccessible.

Configuring Virtual Disk Multipathing

Virtual disk multipathing enables you to configure a virtual disk on a guest domain to access its back-end storage by more than one path. The paths lead through different service domains that provide access to the same back-end storage, such as a disk LUN. This feature enables a virtual disk in a guest domain to remain accessible even if one of the service domains goes down. For example, you might set up a virtual disk multipathing configuration to access a file on a network file system (NFS) server. Or, you can use this configuration to access a LUN from shared storage that is connected to more than one service domain. So, when the guest domain accesses the virtual disk, the virtual disk driver goes through one of the service domains to access the back-end storage. If the virtual disk driver cannot connect to the service domain, the virtual disk attempts to reach the back-end storage through a different service domain.

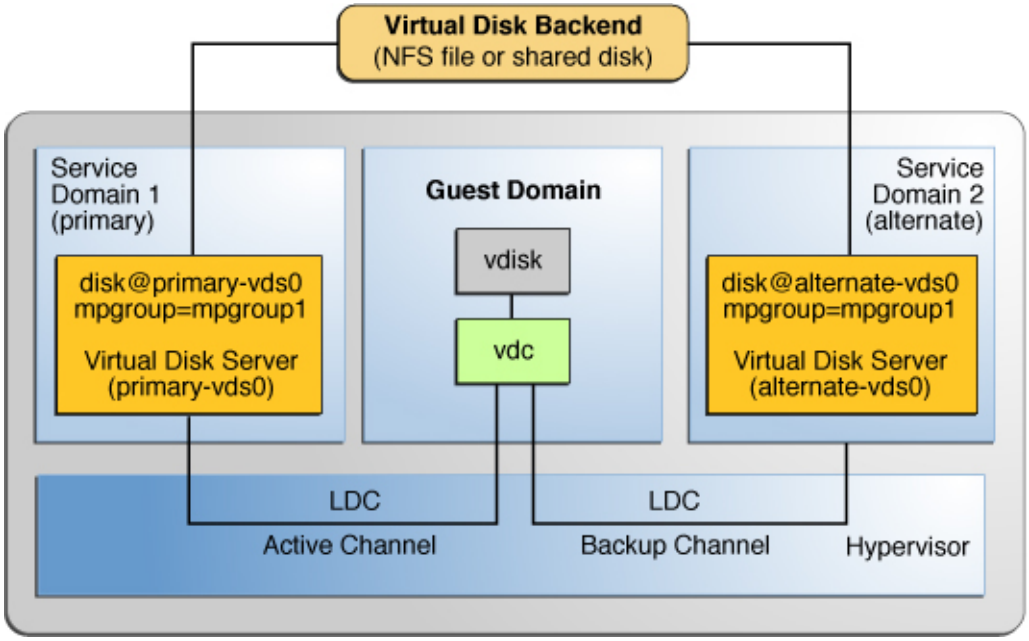
Note – You cannot use mpgroups and SCSI reservation together.

Note that the virtual disk multipathing feature can detect when the service domain cannot access the back-end storage. In such an instance, the virtual disk driver attempts to access the back-end storage by another path.

To enable virtual disk multipathing, you must export a virtual disk back end from each service domain and add the virtual disk to the same multipathing group (`mpgroup`). The `mpgroup` is identified by a name and is configured when you export the virtual disk back end.

The following figure shows a virtual disk multipathing configuration that is used as an example in the procedure "How to Configure Virtual Disk Multipathing" on page 184. In this example, a multipathing group named `mpgroup1` is used to create a virtual disk, whose back end is accessible from two service domains: `primary` and `alternate`.

FIGURE 10–2 Configuring Virtual Disk Multipathing



Virtual Disk Multipathing and Virtual Disk Timeout

With virtual disk multipathing, the path that is used to access the back end automatically changes if the back end becomes inaccessible by means of the currently active path. This path change occurs independently of the value of the virtual disk `timeout` property.

The virtual disk `timeout` property specifies the amount of time after which an I/O fails when no service domain is available to process the I/O. This timeout applies to all virtual disks, even those that use virtual disk multipathing.

As a consequence, setting a virtual disk timeout when virtual disk multipathing is configured can prevent multipathing from working correctly, especially with a small timeout value. So, avoid setting a virtual disk timeout for virtual disks that are part of a multipathing group.

For more information, see "Virtual Disk Timeout" on page 190.

▼ How to Configure Virtual Disk Multipathing

See Figure 10–2.

1 Export the virtual disk back end from the `primary` service domain.

```
primary# ldm add-vdsdev mpgroup=mpgroup1 backend-path1 volume@primary-vds0
```

backend-path1 is the path to the virtual disk back end from the `primary` domain.

2 Export the same virtual disk back end from the `alternate` service domain.

```
primary# ldm add-vdsdev mpgroup=mpgroup1 backend-path2 volume@alternate-vds0
```

backend-path2 is the path to the virtual disk back end from the `alternate` domain.

Note – *backend-path1* and *backend-path2* are paths to the same virtual disk back end, but from two different domains (`primary` and `alternate`). These paths might be the same or different, depending on the configuration of the `primary` and `alternate` domains. The *volume* name is a user choice. It might be the same or different for both commands.

---

3 Export the virtual disk to the guest domain.

```
primary# ldm add-vdisk disk-name volume@primary-vds0 domain-name
```

Note – Although the virtual disk back end is exported several times through different service domains, you assign only one virtual disk to the guest domain and associate it with the virtual disk back end through any of the service domains.

Example 10–1 Using an Mpgroup to Add a LUN to the Virtual Disk Service of Both Primary and Alternate Domains

The following shows how to create a LUN and add it to the virtual disk service for both primary and alternate domains by using the same mpgroup:

To determine which domain to use first when accessing the LUN, specify the associated path when adding the disk to the domain.

- Create the virtual disk devices:

```
primary# ldm add-vdsdev mpgroup=ha lun1@primary-vds0
primary# ldm add-vdsdev mpgroup=ha lun1@alternate-vds0
```

- To use the LUN from `primary-vds0` first, perform the following command:

```
primary# ldm add-vdisk disk1 lun1@primary-vds0 gd0
```

- To use the LUN from `alternate-vds0` first, perform the following command:

```
primary# ldm add-vdisk disk1 lun1@alternate-vds0 gd0
```

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 one of the service domains it has been associated with. If this service domain becomes unavailable, the virtual disk attempts to access its back end through another service domain that is part of the same multipathing group.

Caution – When defining a multipathing group (`mpgroup`), ensure that the virtual disk back ends that are part of the same `mpgroup` are effectively the same virtual disk back end. If you add different back ends into the same `mpgroup`, you might see some unexpected behavior, and you can potentially lose or corrupt data stored on the back ends.

Dynamic Path Selection

You can dynamically select the path to be used for a virtual disk on guest domains that run at least the Oracle Solaris 11.2 SRU 1 OS.

Dynamic path selection occurs when the first path in an mpgroup disk is changed by using the `ldm set-vdisk` command to set the `volume` property to a value in the form *volume-name*@*service-name*. An active domain that supports dynamic path selection can switch to only the selected path. If the updated drivers are not running, this path is selected when the Oracle Solaris OS reloads the disk instance or at the next domain reboot.

The dynamic path selection feature enables you to dynamically perform the following steps while the disk is in use:

- Specify the disk path to be tried first by the guest domain when attaching the disk
- Change the currently active path to the one that is indicated for already attached multipathing disks

Using the `ldm add-vdisk` command with an mpgroup disk now specifies the path indicated by *volume-name*@*service-name* as the selected path with which to access the disk.

The selected disk path is listed first in the set of paths provided to the guest domain irregardless of its rank when the associated mpgroup was created.

You can use the `ldm set-vdisk` command on bound, inactive, and active domains. When used on active domains, this command permits you to choose only the selected path of the mpgroup disk.

The `ldm list-bindings` command shows the following information:

- The `STATE` column for each mpgroup path indicates one of the following values:
  - `active` – Current active path of the mpgroup
  - `standby` – Path is not currently used
  - `unknown` – Domain does not support dynamic path selection, the device is not attached, or an error prevents the path state from being retrieved
- The disk paths are listed in the order that is used for choosing the active path
- The volume that is associated with the disk is the selected path for the mpgroup and is listed first.

The following example shows that the selected path is `vol-ldg2@opath-ldg2` and that the currently used active path is going through the `ldg1` domain. You might see this situation if the selected path could not be used and the second possible path was used instead. Even if the selected path comes online, the non-selected path continues to be used. To make the first path active again, re-issue the `ldm set-vdisk` command to set the `volume` property to the name of the path you want.

```
DISK

NAME            VOLUME                  TOUT ID DEVICE SERVER  MPGROUP
disk            disk-ldg4@primary-vds0       0  disk@0 primary
tdiskgroup      vol-ldg2@opath-ldg2          1  disk@1 ldg2    testdiskgroup
    PORT MPGROUP VOLUME        MPGROUP SERVER   STATE
    2    vol-ldg2@opath-ldg2   ldg2            standby
    0    vol-ldg1@opath-vds    ldg1            active
    1    vol-prim@primary-vds0 primary         standby
```

If you use the `ldm set-vdisk` command on an mpgroup disk of a bound domain that does not run at least the Oracle Solaris 11.2 SRU 1 OS, the operation changes the order of the path priorities and the new path can be used first during next disk attach or reboot or if the OBP needs to access it.

CD, DVD and ISO Images

You can export a compact disc (CD) or digital versatile disc (DVD) the same way you export any regular disk. To export a CD or DVD to a guest domain, export slice 2 of the CD or DVD device as a full disk; that is, without the `slice` option.

Note – You cannot export the CD or DVD drive itself. You can export only the CD or DVD that is inside the CD or DVD drive. Therefore, a CD or DVD must be present inside the drive before you can export it. Also, to be able to export a CD or DVD, that CD or DVD cannot be in use in the service domain. In particular, the Volume Management file system, `volfs` service must not use the CD or DVD. See "How to Export a CD or DVD From the Service Domain to the Guest Domain" on page 188 for instructions on how to remove the device from use by `volfs`.

If you have an International Organization for Standardization (ISO) image of a CD or DVD stored in file or on a volume and export that file or volume as a full disk, then it appears as a CD or DVD in the guest domain.

When you export a CD, DVD, or an ISO image, it automatically appears as a read-only device in the guest domain. However, you cannot perform any CD control operations from the guest domain; that is, you cannot start, stop, or eject the CD from the guest domain. If the exported CD, DVD, or ISO image is bootable, the guest domain can be booted on the corresponding virtual disk.

For example, if you export a Oracle Solaris OS installation DVD, you can boot the guest domain on the virtual disk that corresponds to that DVD and install the guest domain from that DVD. To do so, when the guest domain reaches the `ok` prompt, use the following command.

```
ok boot /virtual-devices@100/channel-devices@200/disk@n:f
```

Where *n* is the index of the virtual disk representing the exported DVD.

Note – If you export a Oracle Solaris OS installation DVD and boot a guest domain on the virtual disk that corresponds to that DVD to install the guest domain, then you cannot change the DVD during the installation. So, you might need to skip any step of the installation requesting a different CD/DVD, or you will need to provide an alternate path to access this requested media.

▼ How to Export a CD or DVD From the Service Domain to the Guest Domain

1. **From the service domain, check whether the volume management daemon, `vold`, is running and online.**

   ```
   service# svcs volfs
   STATE          STIME    FMRI
   online         12:28:12 svc:/system/filesystem/volfs:default
   ```

2. **If the volume management daemon is running and online, as in the example in Step 1, do the following:**

   a. **In the `/etc/vold.conf` file, comment out the line starting with the following words:**

      ```
      use cdrom drive....
      ```

      See the `vold.conf`(4) man page.

   b. **Insert the CD or DVD in the CD or DVD drive.**

   c. **From the service domain, restart the volume management file system service.**

      ```
      service# svcadm refresh volfs
      service# svcadm restart volfs
      ```

3. **From the service domain, find the disk path for the CD-ROM device.**

```
service# cdrw -l
Looking for CD devices...
   Node                  Connected Device                Device type
-â-â-â-â-â-â-â-â-â-â+â-â-â-â-â-â-â-â-â-â-â-â-â-â-â-+â-â-â-â-â-â-â-â-â
/dev/rdsk/c1t0d0s2    | MATSHITA CD-RW  CW-8124   DZ13 | CD Reader/Writer
```

4. **Export the CD or DVD disk device as a full disk.**

```
primary# ldm add-vdsdev /dev/dsk/c1t0d0s2 cdrom@primary-vds0
```

5. **Assign the exported CD or DVD to the guest domain.**

The following command shows how to assign the exported CD or DVD to domain `ldg1`:

```
primary# ldm add-vdisk cdrom cdrom@primary-vds0 ldg1
```

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 "How to Export a Virtual Disk Back End Multiple Times" on page 172 for more information.

▼ How to Export an ISO Image From the Control Domain to Install a Guest Domain

Before You Begin

This procedure assumes that both the `primary` domain and the guest domain are configured.

For example, the following `ldm list` shows that both the `primary` and `ldom1` domains are configured:

```
primary# ldm list
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active   -n-cv   SP      8     8G       0.3%  15m
ldom1            active   -t-â-   5000    4     1G        25%  8m
```

1. **Add a virtual disk server device to export the ISO image.**

In this example, the ISO image is `/export/images/sol-10-u8-ga-sparc-dvd.iso`.

```
primary# ldm add-vdsdev /export/images/sol-10-u8-ga-sparc-dvd.iso dvd-iso@primary-vds0
```

2. **Stop the guest domain.**

In this example, the logical domain is `ldom1`.

```
primary# ldm stop-domain ldom1
LDom ldom1 stopped
```

3. **Add the virtual disk for the ISO image to the logical domain.**

In this example, the logical domain is `ldom1`.

```
primary# ldm add-vdisk s10-dvd dvd-iso@primary-vds0 ldom1
```

4. **Restart the guest domain.**

In this example, the logical domain is `ldom1`.

```
primary# ldm start-domain ldom1
LDom ldom1 started
# ldm list
```

```
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active   -n-cv   SP      8     8G       0.4%  25m
ldom1            active   -t-â-   5000    4     1G       0.0%  0s
```

In this example, the `ldm list` command shows that the `ldom1` domain has just been started.

5 Connect to the guest domain.

```
primary# telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Connecting to console "ldom1" in group "ldom1" ....
Press ~? for control options ..
```

6 Verify the existence of the ISO image as a virtual disk.

```
{0} ok show-disks
a) /virtual-devices@100/channel-devices@200/disk@1
b) /virtual-devices@100/channel-devices@200/disk@0
q) NO SELECTION
Enter Selection, q to quit: q
```

In this example, the newly added device is `/virtual-devices@100/channel-devices@200/disk@1`.

7 Boot the guest domain to install from the ISO image.

In this example, boot from the `f` slice of the `/virtual-devices@100/channel-devices@200/disk@1` disk.

```
{0} ok boot /virtual-devices@100/channel-devices@200/disk@1:f
```

Virtual Disk Timeout

By default, if the service domain providing access to a virtual disk back end is down, all I/O from the guest domain to the corresponding virtual disk is blocked. The I/O automatically is resumed when the service domain is operational and is servicing I/O requests to the virtual disk back end.

However, in some cases, file systems or applications might not want the I/O operation to block but rather to fail and report an error if the service domain is down for too long. You can now set a connection timeout period for each virtual disk, which can then be used to establish a connection between the virtual disk client on a guest domain and the virtual disk server on the service domain. When that timeout period is reached, any pending I/O and any new I/O will fail as long as the service domain is down and the connection between the virtual disk client and server is not reestablished.

Set this timeout by using one of the following methods:

- Using the `ldm add-vdisk` command.

```
ldm add-vdisk timeout=seconds disk-name volume-name@service-name domain-name
```

- Using the `ldm set-vdisk` command.

  ```
  ldm set-vdisk timeout=seconds disk-name domain-name
  ```

- Adding the following line to the `/etc/system` file on the guest domain.

  ```
  set vdc:vdc_timeout=seconds
  ```

  For information about correctly creating or updating `/etc/system` property values, see "Updating Property Values in the /etc/system File" on page 362.

Note – If this tunable is set, it overwrites any timeout setting done using the `ldm` CLI. Also, the tunable sets the timeout for all virtual disks in the guest domain.

Specify the timeout in seconds. If the timeout is set to `0`, the timeout is disabled and I/O is blocked while the service domain is down (this is the default setting and behavior).

Virtual Disk and SCSI

If a physical SCSI disk or LUN is exported as a full disk, the corresponding virtual disk supports the user SCSI command interface, `uscsi` and multihost disk control operations `mhd`. Other virtual disks, such as virtual disks having a file or a volume as a back end, do not support these interfaces.

Note – You cannot use mpgroups and SCSI reservation together.

As a consequence, applications or product features using SCSI commands (such as Solaris Volume Manager `metaset` or Oracle Solaris Cluster `shared devices`) can be used in guest domains only with virtual disks having a physical SCSI disk as a back end.

Note – SCSI operations are effectively executed by the service domain, which manages the physical SCSI disk or LUN used as a virtual disk back end. In particular, SCSI reservations are done by the service domain. Therefore, applications running in the service domain and in guest domains should not issue SCSI commands to the same physical SCSI disks. Doing so can lead to an unexpected disk state.

Virtual Disk and the format Command

The format command recognizes all virtual disks that are present in a domain. However, for virtual disks that are exported as single-slice disks, the format command cannot change the partition table of the virtual disk. Commands such as label will fail unless you try to write a disk label similar to the one that is already associated with the virtual disk.

Virtual disks whose back ends are SCSI disks support all format(1M) subcommands. Virtual disks whose back ends are not SCSI disks do not support some format(1M) subcommands, such as repair and defect. In that case, the behavior of format(1M) is similar to the behavior of Integrated Drive Electronics (IDE) disks.

Using ZFS With Virtual Disks

This section describes using the Zettabyte File System (ZFS) to store virtual disk back ends exported to guest domains. ZFS provides a convenient and powerful solution to create and manage virtual disk back ends. ZFS enables you to do the following:

- Store disk images in ZFS volumes or ZFS files
- Use snapshots to back up disk images
- Use clones to duplicate disk images and provision additional domains

Refer to *Oracle Solaris ZFS Administration Guide* for more information about using ZFS.

In the following descriptions and examples, the primary domain is also the service domain where disk images are stored.

Configuring a ZFS Pool in a Service Domain

To store the disk images, first create a ZFS storage pool in the service domain. For example, this command creates the ZFS storage pool ldmpool containing the disk c1t50d0 in the primary domain.

```
primary# zpool create ldmpool c1t50d0
```

Storing Disk Images With ZFS

The following command creates a disk image for guest domain ldg1. A ZFS file system for this guest domain is created, and all disk images of this guest domain will be stored on that file system.

```
primary# zfs create ldmpool/ldg1
```

Disk images can be stored on ZFS volumes or ZFS files. Creating a ZFS volume, whatever its size, is quick using the `zfs create -V` command. On the other hand, ZFS files have to be created by using the `mkfile` command. This command can take some time to complete, especially if the file to be created is quite large, which is often the case when creating a disk image.

Both ZFS volumes and ZFS files can take advantage of ZFS features such as the snapshot and clone features, but a ZFS volume is a pseudo device while a ZFS file is a regular file.

If the disk image is to be used as a virtual disk onto which an OS is installed, the disk image must be large enough to accommodate the OS installation requirements. This size depends on the version of the OS and on the type of installation performed. If you install the Oracle Solaris OS, you can use a disk size of 20 Gbytes to accommodate any type of installation of any version of the Oracle Solaris OS.

Examples of Storing Disk Images With ZFS

The following examples show how to store disk images using a ZFS volume or a ZFS file. The syntax to export a ZFS volume or file is the same but the path to the back end is different.

When the guest domain is started, the ZFS volume or file appears as a virtual disk on which the Oracle Solaris OS can be installed.

EXAMPLE 10–2 Storing a Disk Image Using a ZFS Volume

First, create a 20-Gbyte image on a ZFS volume.

```
primary# zfs create -V 20gb ldmpool/ldg1/disk0
```

Then, export the ZFS volume as a virtual disk.

```
primary# ldm add-vdsdev /dev/zvol/dsk/ldmpool/ldg1/disk0 ldg1_disk0@primary-vds0
```

Assign the ZFS volume to the `ldg1` guest domain.

```
primary# ldm add-vdisk disk0 ldg1_disk0@primary-vds0 ldg1
```

EXAMPLE 10–3 Storing a Disk Image Using a ZFS File

First, create a 20-Gbyte disk image on a ZFS volume and create the ZFS file.

```
primary# zfs create ldmpool/ldg1/disk0
primary# mkfile 20g /ldmpool/ldg1/disk0/file
```

Then, export the ZFS file as a virtual disk.

```
primary# ldm add-vdsdev /ldmpool/ldg1/disk0/file ldg1_dis0@primary-vds0
```

Assign the ZFS file to the `ldg1` guest domain.

EXAMPLE 10–3 Storing a Disk Image Using a ZFS File *(Continued)*

```
primary# ldm add-vdisk disk0 ldg1_disk0@primary-vds0 ldg1
```

Creating a Snapshot of a Disk Image

When your disk image is stored on a ZFS volume or on a ZFS file, you can create snapshots of this disk image by using the ZFS `snapshot` command.

Before you create a snapshot of the disk image, ensure that the disk is not currently in use in the guest domain to ensure that data currently stored on the disk image are coherent. You can ensure that a disk is not in use in a guest domain in one of the following ways:

- Stop and unbind the guest domain. This solution is the safest, and is the only solution available if you want to create a snapshot of a disk image used as the boot disk of a guest domain.
- Unmount any slices of the disk you want to snapshot that are used in the guest domain, and ensure that no slice is in use in the guest domain.

In this example, because of the ZFS layout, the command to create a snapshot of the disk image is the same whether the disk image is stored on a ZFS volume or on a ZFS file.

EXAMPLE 10–4 Creating a Snapshot of a Disk Image

This example creates a snapshot of the disk image that was created for the `ldg1` domain.

```
primary# zfs snapshot ldmpool/ldg1/disk0@version_1
```

Using Clone to Provision a New Domain

Once you have created a snapshot of a disk image, you can duplicate this disk image by using the ZFS `clone` command. The cloned image then can be assigned to another domain. Cloning a boot disk image quickly creates a boot disk for a new guest domain without having to perform the entire Oracle Solaris OS installation process.

For example, if the `disk0` created was the boot disk of domain `ldg1`, do the following to clone that disk to create a boot disk for domain `ldg2`.

```
primary# zfs create ldmpool/ldg2
primary# zfs clone ldmpool/ldg1/disk0@version_1 ldmpool/ldg2/disk0
```

Then `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 Oracle Solaris 11 `sysconfig unconfigure` command or the Oracle Solaris 10 `sys-unconfig` command. To avoid this problem, you can also create a snapshot of a disk image of an unconfigured system.

If the original domain was configured with the Dynamic Host Configuration Protocol (DHCP), then a new domain using the cloned image also uses DHCP. In that case, you do not need to change the network configuration of the new domain because it automatically receives an IP address and its network configuration as it boots.

Note – The host ID of a domain is not stored on the boot disk, but rather is assigned by the Logical Domains Manager when you create a domain. Therefore, when you clone a disk image, the new domain does not keep the host ID of the original domain.

▼ How to Create a Snapshot of a Disk Image of an Unconfigured System

1. **Bind and start the original domain.**
2. **Unconfigure the system.**
   - **Oracle Solaris 11 OS: Run the `sysconfig unconfigure` command.**
   - **Oracle Solaris 10 OS: Run the `sys-unconfig` command.**

   When this operation completes, the domain halts.
3. **Stop and unbind the domain, do *not* reboot it.**
4. **Take a snapshot of the domain boot disk image.**

   For example:

   ```
   primary# zfs snapshot ldmpool/ldg1/disk0@unconfigured
   ```

At this point, you have the snapshot of the boot disk image of an unconfigured system.

5 **Clone this image to create a new domain which, when first booted, asks for the configuration of the system.**

Using Volume Managers in an Oracle VM Server for SPARC Environment

This section describes using volume managers in an Oracle VM Server for SPARC environment.

Using Virtual Disks With Volume Managers

Any ZFS, Solaris Volume Manager, or Veritas Volume Manager (VxVM) volume can be exported from a service domain to a guest domain as a virtual disk. A volume can be exported either as a single-slice disk (if the `slice` option is specified with the `ldm add-vdsdev` command) or as a full disk.

Note – The remainder of this section uses a Solaris Volume Manager volume as an example. However, the discussion also applies to ZFS and VxVM volumes.

The following examples show how to export a volume as a single-slice disk.

The virtual disk in the guest domain (for example, `/dev/dsk/c0d2s0`) is directly mapped to the associated volume (for example, `/dev/md/dsk/d0`), and data stored onto the virtual disk from the guest domain are directly stored onto the associated volume with no extra metadata. Data stored on the virtual disk from the guest domain can therefore also be directly accessed from the service domain through the associated volume.

Examples

- If the Solaris Volume Manager volume `d0` is exported from the `primary` domain to `domain1`, then the configuration of `domain1` requires some extra steps.

```
primary# metainit d0 3 1 c2t70d0s6 1 c2t80d0s6 1 c2t90d0s6
primary# ldm add-vdsdev options=slice /dev/md/dsk/d0 vol3@primary-vds0
primary# ldm add-vdisk vdisk3 vol3@primary-vds0 domain1
```

- After `domain1` has been bound and started, the exported volume appears as `/dev/dsk/c0d2s0`, for example, and you can use it.

```
domain1# newfs /dev/rdsk/c0d2s0
domain1# mount /dev/dsk/c0d2s0 /mnt
domain1# echo test-domain1 > /mnt/file
```

- After `domain1` has been stopped and unbound, data stored on the virtual disk from `domain1` can be directly accessed from the primary domain through Solaris Volume Manager volume `d0`.

```
primary# mount /dev/md/dsk/d0 /mnt
primary# cat /mnt/file
test-domain1
```

Using Virtual Disks With Solaris Volume Manager

When a RAID or mirror Solaris Volume Manager volume is used as a virtual disk by another domain, then it has to be exported without setting the exclusive (`excl`) option. Otherwise, if there is a failure on one of the components of the Solaris Volume Manager volume, then the recovery of the Solaris Volume Manager volume using the `metareplace` command or using a hot spare does not start. The `metastat` command sees the volume as resynchronizing, but the resynchronization does not progress.

For example, `/dev/md/dsk/d0` is a RAID Solaris Volume Manager volume exported as a virtual disk with the `excl` option to another domain, and `d0` is configured with some hot-spare devices. If a component of `d0` fails, Solaris Volume Manager replaces the failing component with a hot spare and resynchronizes the Solaris Volume Manager volume. However, the resynchronization does not start. The volume is reported as resynchronizing, but the resynchronization does not progress.

```
primary# metastat d0
d0: RAID
    State: Resyncing
    Hot spare pool: hsp000
    Interlace: 32 blocks
    Size: 20097600 blocks (9.6 GB)
Original device:
    Size: 20100992 blocks (9.6 GB)
Device                                          Start Block  Dbase   State Reloc
c2t2d0s1                                               330  No       Okay  Yes
c4t12d0s1                                              330  No       Okay  Yes
/dev/dsk/c10t600C0FF0000000000015153295A4B100d0s1  330  No  Resyncing  Yes
```

In such a situation, the domain using the Solaris Volume Manager volume as a virtual disk has to be stopped and unbound to complete the resynchronization. Then the Solaris Volume Manager volume can be resynchronized using the `metasync` command.

```
# metasync d0
```

Using Virtual Disks When VxVM Is Installed

When the VxVM is installed on your system and Veritas Dynamic Multipathing (DMP) is enabled on a physical disk or partition you want to export as virtual disk, then you have to export that disk or partition without setting the (non-default) `excl` option. Otherwise, you receive an error in `/var/adm/messages` while binding a domain that uses such a disk.

```
vd_setup_vd():  ldi_open_by_name(/dev/dsk/c4t12d0s2) = errno 16
vds_add_vd():  Failed to add vdisk ID 0
```

You can check whether Veritas DMP is enabled by checking the multipathing information in the `vxdisk list` output. For example:

```
# vxdisk list Disk_3
Device:    Disk_3
devicetag: Disk_3
type:      auto
info:      format=none
flags:     online ready private autoconfig invalid
pubpaths:  block=/dev/vx/dmp/Disk_3s2 char=/dev/vx/rdmp/Disk_3s2
guid:      -
udid:      SEAGATE%5FST336753LSUN36G%5FDISKS%5F30323339483031443 04E0000
site:      -
Multipathing information:
numpaths:  1
c4t12d0s2  state=enabled
```

Alternatively, if Veritas DMP is enabled on a disk or a slice that you want to export as a virtual disk with the `excl` option set, then you can disable DMP using the `vxdmpadm` command. For example:

```
# vxdmpadm -f disable path=/dev/dsk/c4t12d0s2
```

Using Volume Managers With Virtual Disks

This section describes using volume managers with virtual disks.

Using ZFS With Virtual Disks

Any virtual disk can be used with ZFS. A ZFS storage pool (`zpool`) can be imported in any domain that sees all the storage devices that are part of this `zpool`, regardless of whether the domain sees all these devices as virtual devices or real devices.

Using Solaris Volume Manager With Virtual Disks

Any virtual disk can be used in the Solaris Volume Manager local disk set. For example, a virtual disk can be used for storing the Solaris Volume Manager metadevice state database, `metadb`, of the local disk set or for creating Solaris Volume Manager volumes in the local disk set.

Any virtual disk whose back end is a SCSI disk can be used in a Solaris Volume Manager shared disk set, `metaset`. Virtual disks whose back ends are not SCSI disks cannot be added into a Solaris Volume Manager share disk set. Trying to add a virtual disk whose back end is not a SCSI disk into a Solaris Volume Manager shared disk set fails with an error similar to the following.

```
# metaset -s test -a c2d2
metaset: domain1: test: failed to reserve any drives
```

Using VxVM With Virtual Disks

For VxVM support in guest domains, refer to the VxVM documentation from Symantec.

Virtual Disk Issues

The following section describe issues that you might encounter when using virtual disks.

In Certain Conditions, a Guest Domain's Solaris Volume Manager Configuration or Metadevices Can Be Lost

If a service domain is running a version of Oracle Solaris 10 OS prior to Oracle Solaris 10 1/13 OS and is exporting a physical disk slice as a virtual disk to a guest domain, then this virtual disk will appear in the guest domain with an inappropriate device ID. If that service domain is then upgraded to Oracle Solaris 10 1/13 OS, the physical disk slice exported as a virtual disk will appear in the guest domain with no device ID.

This removal of the device ID of the virtual disk can cause problems to applications attempting to reference the device ID of virtual disks. In particular, Solaris Volume Manager might be unable to find its configuration or to access its metadevices.

Workaround: After upgrading a service domain to Oracle Solaris 10 1/13 OS, if a guest domain is unable to find its Solaris Volume Manager configuration or its metadevices, perform the following procedure.

▼ How to Find a Guest Domain's Solaris Volume Manager Configuration or Metadevices

1. **Boot the guest domain.**
2. **Disable the `devid` feature of Solaris Volume Manager by adding the following lines to the `/kernel/drv/md.conf` file:**

   ```
   md_devid_destroy=1;
   md_keep_repl_state=1;
   ```

3. **Reboot the guest domain.**

   After the domain has booted, the Solaris Volume Manager configuration and metadevices should be available.

4. **Check the Solaris Volume Manager configuration and ensure that it is correct.**

5 **Re-enable the Solaris Volume Manager `devid` feature by removing from the `/kernel/drv/md.conf` file the two lines that you added in Step 2.**

6 **Reboot the guest domain.**

During the reboot, you will see messages similar to this:

```
NOTICE: mddb: unable to get devid for 'vdc', 0x10
```

These messages are normal and do not report any problems.

Oracle Solaris Boot Disk Compatibility

Historically, the Oracle Solaris OS has been installed on a boot disk configured with an SMI VTOC disk label. Starting with the Oracle Solaris 11.1 OS, the OS is installed on a boot disk that is configured with an extensible firmware interface (EFI) GUID partition table (GPT) disk label by default. If the firmware does not support EFI, the disk is configured with an SMI VTOC disk label instead. This situation applies only to SPARC T4 servers that run at least system firmware version 8.4.0, to SPARC T5 servers, SPARC M5 servers, and SPARC M6 servers that run at least system firmware version 9.1.0, to SPARC T7 series servers and SPARC M7 series servers that run at least system firmware version 9.4.3, and to Fujitsu M10 servers that run at least system firmware version XCP2230.

The following servers cannot boot from a disk that has an EFI GPT disk label:

- UltraSPARC T2, UltraSPARC T2 Plus, and SPARC T3 servers no matter which system firmware version is used
- SPARC T4 servers that run system firmware versions prior to 8.4.0
- SPARC T5 servers, SPARC M5 servers, and SPARC M6 servers that run system firmware versions prior to 9.1.0
- SPARC T7 series servers and SPARC M7 series servers that run system firmware versions prior to 9.4.3

So, an Oracle Solaris 11.1 boot disk that is created on an up-to-date SPARC T4 server, SPARC T5 server, SPARC T7 series server, SPARC M5 server, SPARC M6 server, or SPARC M7 series server cannot be used on older servers or on servers that run older firmware.

This limitation restrains the ability to use either cold or live migration to move a domain from a recent server to an older server. This limitation also prevents you from using an EFI GPT boot disk image on an older server.

To determine whether an Oracle Solaris 11.1 boot disk is compatible with your server and its firmware, ensure that the Oracle Solaris 11.1 OS is installed on a disk that is configured with an SMI VTOC disk label.

To maintain backward compatibility with systems that run older firmware, use one of the following procedures. Otherwise, the boot disk uses the EFI GPT disk label by default. These procedures show how to ensure that the Oracle Solaris 11.1 OS is installed on a boot disk with an SMI VTOC disk label on a SPARC T4 server with at least system firmware version 8.4.0, on a SPARC T5 server, SPARC M5 server, or SPARC M6 server with at least system firmware version 9.1.0, and on a SPARC T7 series server or SPARC M7 series server with at least system firmware version 9.4.3.

- **Solution 1:** Remove the `gpt` property so that the firmware does not report that it supports EFI.

  1. From the OpenBoot PROM prompt, disable automatic booting and reset the system to be installed.

     ```
     ok setenv auto-boot? false
     ok reset-all
     ```

     After the system resets, it returns to the `ok` prompt.

  2. Change to the `/packages/disk-label` directory and remove the `gpt` property.

     ```
     ok cd /packages/disk-label
     ok " gpt" delete-property
     ```

  3. Begin the Oracle Solaris 11.1 OS installation.

     For example, perform a network installation:

     ```
     ok boot net - install
     ```

- **Solution 2:** Use the `format -e` command to write an SMI VTOC label on the disk to be installed with the Oracle Solaris 11.1 OS.

  1. Write an SMI VTOC label on the disk.

     For example, select the `label` option and specify the SMI label:

     ```
     # format -e c1d0
     format> label
     [0] SMI Label
     [1] EFI Label
     Specify Label type[1]: 0
     ```

  2. Configure the disk with a slice 0 and slice 2 that cover the entire disk.

     The disk should have no other partitions. For example:

     ```
     format> partition

     partition> print
     Current partition table (unnamed):
     Total disk cylinders available: 14087 + 2 (reserved cylinders)

     Part      Tag    Flag     Cylinders         Size            Blocks
       0       root    wm       0 - 14086      136.71GB    (14087/0/0) 286698624
       1 unassigned    wu       0                0         (0/0/0)             0
       2     backup    wu       0 - 14086      136.71GB    (14087/0/0) 286698624
       3 unassigned    wm       0                0         (0/0/0)             0
       4 unassigned    wm       0                0         (0/0/0)             0
     ```

```
  5 unassigned    wm       0               0         (0/0/0)           0
  6 unassigned    wm       0               0         (0/0/0)           0
  7 unassigned    wm       0               0         (0/0/0)           0
```

3. Re-write the SMI VTOC disk label.

```
partition> label
[0] SMI Label
[1] EFI Label
Specify Label type[0]: 0
Ready to label disk, continue? y
```

4. Configure your Oracle Solaris Automatic Installer (AI) to install the Oracle Solaris OS on slice 0 of the boot disk.

   Change the `<disk>` excerpt in the AI manifest as follows:

```
<target>
   <disk whole_disk="true">
        <disk_keyword key="boot_disk"/>
        <slice name="0" in_zpool="rpool"/>
   </disk>
[...]
</target>
```

5. Perform the installation of the Oracle Solaris 11.1 OS.

◆ ◆ ◆ CHAPTER 11

Using Virtual SCSI Host Bus Adapters

This chapter describes how to use virtual SCSI Host Bus Adapters (HBAs) with Oracle VM Server for SPARC software.

This chapter covers the following topics:

Introduction to Virtual SCSI Host Bus Adapters

A virtual SCSI host bus adapter (HBA) is composed of two components: a virtual HBA in the guest domain and a virtual storage area network (SAN) in the service domain. The virtual HBA and virtual SAN instances cooperate to implement a SCSI HBA interface for SCSI target drivers that execute in the guest domain. The vSAN service is implemented by the `vsan` driver, which passes SCSI I/O requests to the physical SCSI HBA driver that executes in the service domain. The `vhba` driver sends I/O requests to `vsan` by using a hypervisor-managed logical domain channel (LDC).

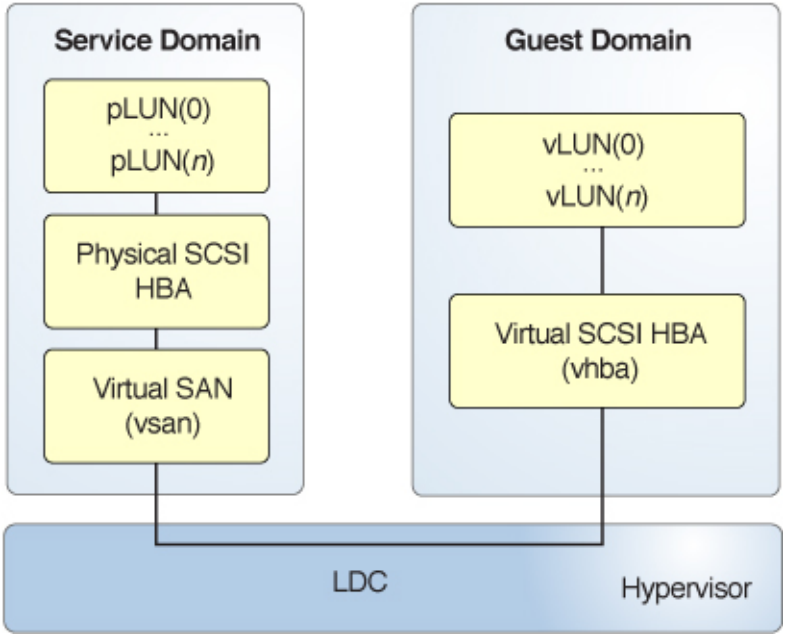
A vHBA instance provides access to all SCSI devices that are reachable by a specific vSAN instance. A vHBA can recognize any SCSI device type such as disk, CD, DVD, or tape. The set of reachable SCSI devices changes based on the set of physical SCSI devices that is currently

known to the virtual SAN's associated physical HBA driver. The identity and number of SCSI devices known to a specific vHBA are not known until runtime, which also occurs with a physical HBA driver.

The vHBA has virtual LUNs (vLUNs) as its child devices and they behave the same way as physical LUNs. For example, you can use the native Oracle Solaris multipathing solution (MPxIO) with a vHBA instance and its vLUNs. The device path for a vLUN uses the full c*X*t*Y*d*Z*s*N* notation: `/dev/[r]dsk/c`*X*`t`*Y*`d`*Z*`s`*N*. The `t`*Y* part of the device name indicates the SCSI target device.

After you configure the virtual SAN and virtual SCSI HBA, you can perform operations such as booting a virtual LUN from the OpenBoot prompt or viewing all the virtual LUNs by using the `format` command.

FIGURE 11–1 Virtual SCSI HBAs With Oracle VM Server for SPARC



A virtual SAN exists in a service domain and is implemented by the `vsan` kernel module, whereas a virtual SCSI HBA exists in a guest domain and is implemented by the `vhba` module. A virtual SAN is associated with a specific physical SCSI HBA initiator port, whereas a virtual SCSI HBA is associated with a specific virtual SAN.

The vhba module exports a SCSA-compliant interface to receive I/O requests from any SCSA-compliant SCSI target driver. The vhba module translates the I/O requests into virtual I/O protocol messages that are sent through an LDC to the service domain.

The vsan module translates the virtual I/O messages sent by vhba into I/O requests. These requests are sent to a SCSA-compliant physical SCSI HBA driver. The vsan module returns the I/O payload and status to vhba through the LDC. Finally, the vhba forwards this I/O response to the originator of the I/O request.

Operational Model for Virtual SCSI HBAs

The operational model for using virtual SCSI HBAs is qualitatively different than for other types of Oracle VM Server for SPARC virtual devices because only the virtual SCSI HBA and virtual SAN instances are known to the Logical Domains Manager. The virtual LUNs that appear in the guest domain and the physical LUNs that appear in the service domain are not known until they are discovered at runtime. The virtual LUNs and physical LUNs are discovered implicitly when the associated LDC connection is reset and explicitly by using the `ldm rescan-vhba` command.

Although you use the `ldm` command to name a virtual disk explicitly, a virtual LUN in a guest domain derives its identity from the identity of the associated physical LUN in the service domain. See the `ldm(1M)` man page.

For example, the physical LUN and the virtual LUN share the text shown in bold in the following device paths:

- Physical LUN in the service domain:

  ```
  /pci@0/pci@0/pci@8/pci@0/pci@2/SUNW,qlc@0/fp@0,0/ssd@w216000c0ff8089d5,0
  ```

- Virtual LUN in the guest domain:

  ```
  /virtual-devices@100/channel-devices@200/scsi@1/iport@0/disk@w216000c0ff8089d5,0
  ```

Note – The guest domain device path is present only when MPxIO is disabled in the guest domain. If MPxIO is enabled, the `scsi_vhci` module creates the device path in the guest domain and it has a different syntax.

Note that the `scsi@1` component in the virtual LUN's device path denotes the virtual SCSI HBA instance of which this virtual LUN is a member.

Because a virtual SCSI HBA's set of virtual LUNs is derived from the service domain at runtime, virtual LUNs cannot be added or removed explicitly from the guest domain. Instead, you must add or remove the underlying physical LUNs so that the guest domain's virtual LUN membership can be altered. An event such as a domain reboot or a domain migration, might cause the guest domains' virtual LUN membership to change. This change occurs because the

virtual LUNs are rediscovered automatically whenever the virtual SCSI HBA's LDC connection is reset. If a virtual LUN's underlying physical LUN is not found on a future discovery, the virtual LUN is marked as unavailable and accesses to the virtual LUN return an error similar to the following:

```
WARNING: .../scsi@1/iport@0/disk@w216000c0ff8089d5,0 (sd6): ... Command failed to complete...Device is gone
```

A virtual SCSI HBA instance is managed by the `vhba` driver, but a virtual LUN is managed by a SCSI target driver based on the device type of the underlying physical LUN. The following output confirms that the `vhba` driver manages the virtual SCSI HBA instance and that the `sd` SCSI disk driver manages the virtual LUN:

```
# prtconf -a -D /dev/dsk/c2t216000C0FF8089D5d0
SUNW,SPARC-Enterprise-T5220 (driver name: rootnex)
    virtual-devices, instance #0 (driver name: vnex)
        channel-devices, instance #0 (driver name: cnex)
            scsi, instance #0 (driver name: vhba)
                iport, instance #3 (driver name: vhba)
                    disk, instance #30 (driver name: sd)
```

Virtual SCSI HBA Identifier and Device Name

When you use the `ldm add-vhba` command to add a virtual SCSI HBA to a domain, you can specify its device number by setting the `id` property.

`ldm add-vhba [id=`*vHBA-ID*`]` *vHBA-name vSAN-name domain-name*

Each virtual SCSI HBA of a domain has a unique device number that is assigned when the domain is bound. If a virtual SCSI HBA is added with an explicit device number (by setting the `id` property to a decimal value), 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 the virtual SCSI HBAs were added to the domain. When a domain is bound, the device number eventually assigned to a virtual SCSI HBA is visible in the output of the `ldm list-bindings` and `ldm list -o hba` commands.

The `ldm list-bindings`, `ldm list -o hba`, and `ldm add-vhba id=`*id* commands all show and specify the `id` property value as a decimal value. The Oracle Solaris OS shows the virtual SCSI HBA `id` value as hexadecimal.

The following example shows that the `vhba@0` device is the device name for the `vh1` virtual SCSI HBA on the `gdom` domain.

```
primary# ldm list -o hba gdom
NAME
gdom

VHBA
```

```
NAME            VSAN                    DEVICE  TOUT SERVER
vh1             vs1                     vhba@0  0    svcdom
```

Caution – If a device number is not assigned explicitly to a virtual SCSI HBA, its device number can change when the domain is unbound and is later re-bound. In that case, the device name assigned by the OS running in the domain might also change and break the existing configuration of the system. This might happen, for example, when a virtual SCSI HBA is removed from the configuration of the domain.

Managing Virtual SCSI HBAs

This section covers the following tasks:

For more information about the commands that are shown in this section, see the `ldm(1M)` man page.

Obtaining Physical SCSI HBA Information

Before you configure a virtual SCSI HBA, you must obtain information about the physical SCSI HBAs that are attached to the service domain. For more information about configuring HBA cards in I/O domains, see Chapter 5, "Configuring I/O Domains."

Note – If at least the Oracle Solaris 11.3 OS is installed in the `primary` domain, the service domain can be the control domain.

The `ldm list-hba` command lists the physical SCSI HBA initiator ports for the specified active domain. After identifying a logical domain's SCSI HBA initiator ports, you can specify a particular initiator port on the `ldm add-vsan` command line to create a virtual SAN.

```
ldm list-hba [-d] [-l] [-p] [-t] domain-name
```

The following example shows the initiator ports for the SCSI HBAs that are attached to the `svcdom` service domain. The `-l` option shows detailed information.

```
primary# ldm list-hba -l svcdom
NAME                                                    VSAN
----                                                    ----
/SYS/MB/SASHBA0/HBA0/PORT1
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@1]
/SYS/MB/SASHBA0/HBA0/PORT2
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@2]
/SYS/MB/SASHBA0/HBA0/PORT4
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@4]
/SYS/MB/SASHBA0/HBA0/PORT8
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@8]
/SYS/MB/PCIE1/HBA0/PORT0,0
[/pci@300/pci@1/pci@0/pci@4/SUNW,emlxs@0/fp@0,0]
/SYS/MB/PCIE1/HBA0,1/PORT0,0
[/pci@300/pci@1/pci@0/pci@4/SUNW,emlxs@0,1/fp@0,0]
```

If the LUNs you expect to see for an initiator port do not appear in the `ldm list-hba` output, verify that multipathing is disabled in the referenced service domain for the referenced initiator port. See *Managing SAN Devices and Multipathing in Oracle Solaris 11.3*.

If you add a virtual SAN to a bound service domain, sometimes the connection to the guest domain is not established. You might see the following symptoms:

- Disks from the virtual SCSI HBA do not appear in `format` output
- LDC errors appear when booting from a virtual LUN

To resolve this issue, add a line that says `ddi-no-autodetach="yes"` to the `/etc/driver/drv/vsan.conf` file, and then reboot the service domain.

Creating a Virtual Storage Area Network

After you obtain the initiator port of the physical SCSI HBA, you must create the virtual storage area network (SAN) on the service domain. The virtual SAN manages all SCSI devices that are reachable from the specified SCSI HBA initiator port.

`ldm add-vsan [-q]` *iport-path vSAN-name domain-name*

The vSAN name is unique to the control domain and not to the specified domain name. The domain name identifies the domain in which the SCSI HBA initiator port is configured. You can create multiple virtual SANs that reference the same initiator port path.

You can configure an initiator port path into one or more virtual SANs by using the `ldm add-vsan` command. This configuration enables multiple service domains in the control domain to use the same initiator port path.

Note – When the Oracle Solaris 11.3 OS is running on the service domain, the `ldm add-vsan` command verifies that the initiator port path is a valid device path. If the specified service domain is not active when you run the `ldm add-vsan` command, the specified initiator port path cannot be verified by the service domain. If the initiator port path does not correspond to an installed physical SCSI HBA initiator port that is part of the service domain, a warning message is written to the service domain's system log after the service domain becomes active.

In this example, you associate the `/SYS/MB/PCIE1/HBA0,1/PORT0,0` initiator port on the `svcdom` service domain with a virtual SAN. You can choose the name of the virtual SAN. In this example, `port0` is the name of the virtual SAN.

```
primary# ldm add-vsan /SYS/MB/PCIE1/HBA0,1/PORT0,0 port0 svcdom
/SYS/MB/PCIE1/HBA0,1/PORT0,0 resolved to device:
/pci@300/pci@1/pci@0/pci@4/SUNW,emlxs@0,1/fp@0,0
```

Creating a Virtual SCSI Host Bus Adapter

After the virtual SAN has been defined, you can use the `ldm add-vhba` command to create a virtual SCSI HBA in a guest domain. The virtual SCSI HBA sends I/O requests to the physical SCSI devices in the virtual SAN.

`ldm add-vhba [id=`*vHBA-ID*`]` *vHBA-name vSAN-name domain-name*

In this example, you create the `port0_vhba` virtual SCSI HBA on the `gdom` guest domain that communicates with the `port0` virtual SAN.

```
primary# ldm add-vhba port0_vhba port0 gdom
```

Verifying the Presence of a Virtual SCSI HBA

Use the `ldm list` command to verify the presence of the newly created virtual SCSI HBA and virtual SAN devices on the service domain and the guest domain.

`ldm list -o san,hba [`*domain-name* `...]`

In this example, the service domain that has the virtual SAN is `svcdom` and the guest domain that has the virtual SCSI HBA is `gdom`. Note that the virtual HBA identifier is not allocated in this example because the `gdom` domain is not yet bound.

```
primary# ldm list -o san,hba svcdom gdom
NAME
svcdom
```

```
VSAN
    NAME             TYPE   DEVICE IPORT
    port0            VSAN [/pci@300/pci@1/pci@0/pci@4/SUNW,emlxs@0,1/fp@0,0]

------------------------------------------------------------------------------
NAME
gdom

VHBA
    NAME             VSAN                        DEVICE TOUT SERVER
    port0_vhba       port0                              0    svcdom
```

Setting the Virtual SCSI HBA Timeout Option

The `ldm set-vhba` command enables you to specify a timeout value for the virtual SCSI HBA on the specified logical domain. The `timeout` property specifies how long, in seconds, the specified virtual SCSI HBA instance waits before declaring that an LDC connection cannot be made with the virtual SAN. See "Virtual SCSI HBA Timeout" on page 218.

The default timeout value of zero causes `vhba` to wait indefinitely to create the LDC connection with the virtual SAN.

```
ldm set-vhba [timeout=seconds] vHBA-name  domain-name
```

In this example, you set a timeout of 90 sections for the `port0_vhba` virtual SCSI HBA on the `gdom` guest domain.

```
primary# ldm set-vhba timeout=90 port0_vhba gdom
```

Removing a Virtual SCSI Host Bus Adapter

You can use the `ldm remove-vhba` command to remove a virtual SCSI HBA from a specified guest domain.

Ensure that neither the OS nor any applications are actively using the virtual SCSI HBA before you attempt to remove it. If the virtual SCSI HBA is in use, the `ldm remove-vhba` command fails.

```
ldm remove-vhba vHBA-name  domain-name
```

In this example, you remove the `port0_vhba` virtual SCSI HBA from the `gdom` guest domain.

```
primary# ldm remove-vhba port0_vhba gdom
```

Removing a Virtual Storage Area Network

You can use the `ldm remove-vsan` command to remove a virtual SAN.

First, remove the virtual SCSI HBA that is associated with the virtual SAN. Then, use the `ldm remove-vsan` command to remove the virtual SAN.

`ldm remove-vsan` *vSAN-name*

In this example, you remove the `port0` virtual SAN:

```
primary# ldm remove-vsan port0
```

Adding or Removing a LUN

You cannot add or remove a virtual LUN directly from a virtual SCSI HBA. You must first add or remove a physical LUN and then run the `ldm rescan-vhba` command to synchronize the set of SCSI devices that are seen by the virtual SCSI HBA and virtual SAN. The commands to add or remove a physical LUN are specific to the topology of the virtual SAN's associated initiator port. For example, if the initiator port communicates with a physical SAN, you must use SAN administration commands to add a LUN to or remove a LUN from a SAN element.

`ldm rescan-vhba` *vHBA-name domain-name*

For example, the following command synchronizes the SCSI devices for the `port0_vhba` virtual SCSI HBA on the `gdom` domain:

```
primary# ldm rescan-vhba port0_vhba gdom
```

Appearance of Virtual LUNs in a Guest Domain

The virtual LUN or LUNs that are associated with a virtual SCSI HBA behave as if they are physical LUNs.

A virtual LUN that represents a SCSI disk, for example, appears in the guest domain as a regular disk under `/dev/[r]dsk`. The virtual LUN is visible in the output of the `format` command because the underlying associated physical LUN is of type `disk`. The device path of the virtual LUN can be operated on by other commands, such as `prtpicl` and `prtconf`.

If the virtual LUN represents a CD or a DVD, its device path is defined in `/dev/[r]dsk`. If the virtual LUN represents a tape device, its device path is defined in `/dev/rmt`. For commands that can operate on the virtual LUN, see *Managing Devices in Oracle Solaris 11.3* .

Virtual SCSI HBA and Virtual SAN Configurations

Configuring virtual SCSI HBAs and virtual SANs is very flexible. The physical SCSI HBA initiator port that is used by the `ldm add-vsan` command can drive any type of bus that supports SCSI, such as Fibre Channel, SAS, or SATA. The virtual SCSI HBA and virtual SAN can execute in the same domain. Also, a virtual SAN can execute in an I/O domain after using the `ldm add-io` command to add a physical SCSI HBA card to a service domain.

Although a virtual SAN is conceptually associated with a physical SAN, it does not need to be. You can create a virtual SAN that comprises one or more of a server's local disks. For example, some systems have disks that are reachable from the motherboard's SAS HBA, which is shown as `SASHBA` in the following `ldm list-hba` output:

```
primary# ldm list-hba -d primary
NAME MY_VSAN

---- ----
/SYS/MB/SASHBA0/HBA0/PORT4
    c5t5000CCA0564DEF39d0s0
/SYS/MB/SASHBA0/HBA0/PORT1
    c3t5000CCA0564F1A7Dd0s0
/SYS/MB/SASHBA0/HBA0/PORT2
   c4t5000CCA0564F6B89d0s0
/SYS/MB/SASHBA0/HBA0/PORT8
    c6t5000CCA0564FCF6Dd0s0
```

If you define a virtual SAN that encapsulates a server's local disks, be sure to use the following `zpool` command so that you do not mistakenly create a virtual LUN for the disk from which the `primary` domain boots. For example, the following `zpool` command confirms that the root rpool is mounted on disk `c4t5000CCA0564F6B89d0`, which is reachable from the `/SYS/MB/SASHBA0/HBA/PORT2` initiator port:

```
# zpool iostat -v
                          capacity     operations    bandwidth
pool                    alloc   free   read  write   read  write
----------------------  -----  -----  -----  -----  -----  -----
rpool                   25.0G   531G      0     10    257  81.7K
c4t5000CCA0564F6B89d0   25.0G   531G      0     10    257  81.7K
----------------------  -----  -----  -----  -----  -----  -----
```

When configuring a virtual SAN, note that only SCSI target devices with a LUN 0 have their physical LUNs visible in the guest domain. This constraint is imposed by an Oracle Solaris OS implementation that requires a target's LUN 0 to respond to the SCSI `REPORT LUNS` command.

Configuring Virtual SCSI HBA Multipathing

The virtual SCSI HBA subsystem supports multipathing in the guest domain by leveraging the native Oracle Solaris multipathing implementation (MPxIO). For more information, see *Managing SAN Devices and Multipathing in Oracle Solaris 11.3*.

As in native multipathing, a specific back-end SCSI device can be accessed by one or more paths. For the virtual SCSI HBA subsystem, each path is associated with one virtual LUN. The `scsi_vhci` module implements the native multipathing behavior, which sends I/O requests to the set of virtual LUNs based on arguments passed to the associated `mpathadm` administrative command. For more information, see the `scsi_vhci(7D)` and `mpathadm(1M)` man pages.

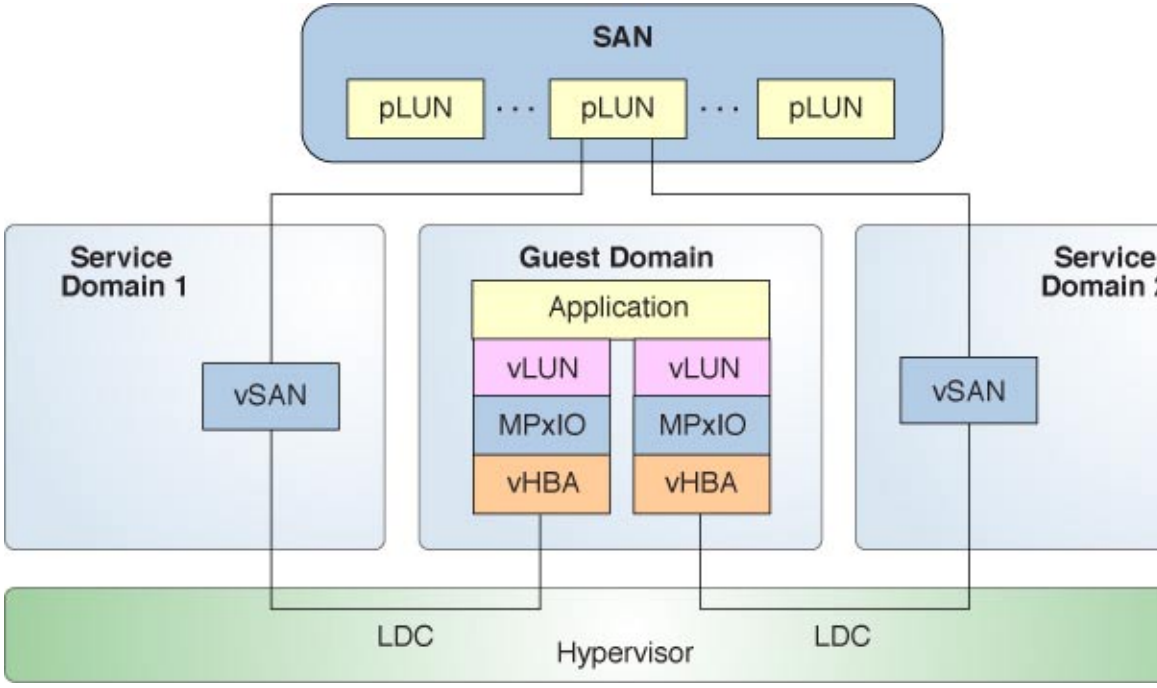
To configure multipathing, you must configure two or more distinct paths from the guest domain to the same back-end device. Note that multipathing still operates with one configured path. However, the expected configuration has two or more paths that send their I/O requests through distinct physical SCSI HBA initiator ports that reside on distinct service domains.

1. Execute a pair of `ldm add-vhba` and `ldm add-vsan` commands for each separate path to the back-end storage.
2. Enable native multipathing in the guest domain for the initiator ports that are managed by the `vhba` virtual HBA module.

Note – The virtual SCSI HBA subsystem does not support the enabling of native multipathing in the service domain. If you expect to see LUNs and they are not shown in the `ldm list-hba` output for a specific service domain, verify that MPxIO is disabled for the LUNs' associated physical SCSI HBA initiator port. If necessary, use the `stmsboot` command to disable MPxIO for the initiator port.

The following figure is an example of a multipathing configuration. It shows one physical LUN of a SAN that is accessed by two paths that are managed by MPxIO. For a procedure that describes how to create the configuration shown in this figure, see "How to Configure Virtual SCSI HBA Multipathing" on page 214.

FIGURE 11–2 Configuring Virtual SCSI HBA Multipathing



How to Configure Virtual SCSI HBA Multipathing

This procedure describes how to create the virtual SCSI HBA multipathing configuration that is shown in Figure 11–2. This example is just one of many possible multipathing configurations.

1. **Create an I/O domain with the physical SCSI HBA assigned to it.**

   See Chapter 5, "Configuring I/O Domains."

2. **Export the virtual SAN for the first initiator port path from the first service domain.**

   `ldm add-vsan` *vSAN-path1 vSAN-name domain-name*

   vSAN-path1 is the first initiator port path to the SAN.

   For example, the following command exports the `vsan-mpxio1` virtual SAN from the `svcdom1` domain:

   ```
   primary# ldm add-vsan /SYS/MB/RISER0/PCIE1/HBA0/PORT0 vsan-mpxio1 svcdom1
   ```

3. **Export the virtual SAN for the second initiator port path from the second service domain.**

   `ldm add-vsan` *vSAN-path2 vSAN-name domain-name*

   vSAN-path2 is the second initiator port path to the SAN.

For example, the following command exports the `vsan-mpxio2` virtual SAN from the `svcdom2` domain:

```
primary# ldm add-vsan /SYS/MB/RISER0/PCIE3/HBA0/PORT0 vsan-mpxio2 svcdom2
```

4. **Export the virtual SCSI HBAs to the guest domain.**

   `ldm add-vhba` *vHBA-name vSAN-name domain-name*

   For example, the following commands export the `vhba-mpxio1` and `vhba-mpxio2` virtual SCSI HBAs to the `gdom` domain:

```
primary# ldm add-vhba vhba-mpxio1 vsan-mpxio1 gdom
primary# ldm add-vhba vhba-mpxio2 vsan-mpxio2 gdom
```

5. **Specify the `timeout` property value for the virtual SCSI HBAs on the guest domain.**

   `ldm set-vhba timeout=`*seconds vHBA-name domain-name*

   For example, the following commands set the `timeout` property value to `30` for the `vsan-mpxio1` and `vsan-mpxio2` virtual SCSI HBAs on the `gdom` domain:

```
primary# ldm set-vhba timeout=30 vhba-mpxio1 gdom
primary# ldm set-vhba timeout=30 vhba-mpxio2 gdom
```

6. **Reboot the guest domain.**

How to Enable Multipathing for Virtual SCSI HBAs

Perform this task on the guest domain.

1. **Copy the `/platform/sun4v/kernel/drv/vhba.conf` file to the `/etc/driver/drv` directory.**

```
# cp /platform/sun4v/kernel/drv/vhba.conf /etc/driver/drv
```

2. **Enable MPxIO by editing the `/etc/driver/drv/vhba.conf` file to set the `mpxio-disable` property to `no`.**

3. **Reboot the guest domain.**

How to Disable Multipathing for Virtual SCSI HBAs

Perform this task on the guest domain.

1. **Copy the `/platform/sun4v/kernel/drv/vhba.conf` file to the `/etc/driver/drv` directory.**

```
# cp /platform/sun4v/kernel/drv/vhba.conf /etc/driver/drv
```

2. **Disable MPxIO by editing the `/etc/driver/drv/vhba.conf` file to set the `mpxio-disable` property to `yes`.**

3. **Reboot the guest domain.**

Booting From SCSI Devices

The following sections describe how to boot from SCSI devices:

- "Booting From a Virtual LUN" on page 216
- "Booting From a SCSI DVD Device" on page 217

Booting From a Virtual LUN

You can boot any virtual LUN whose associated physical LUN references a SCSI device type that is bootable by OBP, such as CD, DVD, or disk.

Before you issue the `boot` command at the OpenBoot PROM prompt, run the `probe-scsi-all` command to find the guest domain's virtual SCSI HBAs and associated virtual LUNs.

The following annotated example highlights the relevant parts of the output:

```
{0} ok probe-scsi-all

/virtual-devices@100/channel-devices@200/scsi@0              Line 1

vHBA

TPORT-PHYS: w200200110d214900                                Line 2
  LUN: 1   Disk     VLUN          2097152 Blocks, 1073 MB
  LUN: 0   Disk     VLUN          32768000 Blocks, 16 GB     Line 3
```

This example `probe-scsi-all` output shows one virtual SCSI HBA instance (`scsi@0`) that has two LUNs that are of type `disk`.

To boot from a specific virtual LUN, manually compose the device path to pass to the `boot` command. The device path has this syntax:

vhba-device-path`/disk@`*target-port*`,`*lun*`:`*slice*

To boot from the LUN on Line 3, you must compose the device path as follows:

- Take the value of *target-port* from Line 2
- Take the value of *vhba-device-path* from Line 1

The following is the resulting device path:

```
/virtual-devices@100/channel-devices@200/scsi@0/disk@w200200110d214900,0
```

You can pass this device path to the OBP `boot` command as follows:

```
{0} ok boot /virtual-devices@100/channel-devices@200/scsi@0/disk@w200200110d214900,0
```

Booting From a SCSI DVD Device

You can boot from a SCSI digital versatile disc (DVD) drive to install the guest domain from that DVD.

The following example shows the configuration of a virtual SCSI HBA that has a SCSI DVD device attached to the `primary` domain.

```
primary# ldm list-hba -t -d primary
IPORT                                                VSAN
-----                                                ----
[...]
/SYS/MB/SASHBA1/HBA0/PORT40
        init-port w50800200008f4329
        Transport Protocol SAS
  c2t3d0s0
        lun 0
        removable media 1

primary# ldm add-vsan /SYS/MB/SASHBA1/HBA0/PORT40 dvd_vsan primary
/SYS/MB/SASHBA1/HBA0/PORT40 resolved to device: /pci@400/pci@2/pci@0/pci@4/scsi@0/iport@40
primary# ldm add-vhba dvd_vhba dvd_vsan gdom
```

From the `gdom` domain console, probe the SCSI devices and boot from the DVD.

```
{0} ok probe-scsi-all
/virtual-devices@100/channel-devices@200/scsi@0

vHBA

TPORT-PHYS: p3
   LUN: 0   Removable Read Only device     TEAC    DV-W28SS-V 1.0B

{0} ok boot /virtual-devices@100/channel-devices@200/scsi@0/disk@p3
...
```

Installing a Virtual LUN

You can install an OS on any virtual LUN whose associated physical LUN references a SCSI device whose type is supported by the installation program. You can then boot from the specified virtual LUN.

Virtual SCSI HBA Timeout

By default, if the service domain that provides access to a virtual SAN is unavailable, all I/O from the guest domain to the corresponding virtual SCSI HBA is blocked. The I/O is resumed automatically when the service domain becomes operational and restores service to the virtual SAN.

Sometimes, file systems or applications might require an I/O operation to fail and report an error if the service domain is unavailable for too long. You can set a connection timeout period for each virtual SCSI HBA to establish a connection between the virtual SCSI HBA on a guest domain and the virtual SAN on the service domain. When that timeout period is reached, any pending I/O and any new I/O operations fail as long as the service domain is unavailable and the connection between the virtual SCSI HBA and the virtual SAN is not re-established.

Other circumstances in which you might want to specify the timeout value include the following:

- If you want MPxIO to fail over to another configured path, you must set the timeout for each virtual SCSI HBA involved.
- If you perform a live migration, set the `timeout` property value to `0` for each virtual SCSI HBA in the guest domain to be migrated. After the migration completes, reset the timeout property to the original setting for each virtual SCSI HBA.

To find out how to set the timeout value, see "Setting the Virtual SCSI HBA Timeout Option" on page 210.

Virtual SCSI HBA and SCSI

The `vhba` module proxies SCSI commands to the physical SCSI HBA driver that is associated with the virtual SAN's SCSI initiator port.

The `scsi_vhci` driver, which implements native Oracle Solaris multipathing, handles reservation persistency during path failover for both SCSI-2 reservations and SCSI-3 reservations. The `vhba` module plugs in to the Oracle Solaris I/O framework and thus supports SCSI reservations by leveraging the `scsi_vhci` support.

Supporting Highly Fragmented I/O Buffers in the Guest Domain

As with other `sun4v` virtual devices, the `vhba` module operates on an I/O buffer that is created by higher layers in the software stack. If the I/O buffer is an aggregation of too many fragments of physical memory, the `vhba` module issues the following fatal warning message when processing the I/O request:

```
WARNING: ... ldc_mem_bind_hdl: ncookies(max, actual) = (8, 9)
```

Each physical memory fragment is associated with one cookie. If the actual number of cookies cannot be supported by the maximum number of cookies, the I/O request will fail.

The error message shows the actual number of cookies that are required. To eliminate the error, change the `vhba_desc_ncookies` value in the `/etc/system` file, which specifies the number of per-I/O buffer cookies to use, to be at least as large as the actual value. Also, increase the value of the `vhba_desc_max_ncookies` property, which specifies the allowable maximum number of cookies.

For information about correctly creating or updating `/etc/system` property values, see "Updating Property Values in the `/etc/system` File" on page 362.

You then re-create the virtual SCSI HBA's connection by performing a sequence of `ldm remove-vhba` and `ldm add-vhba` commands or by rebooting the guest domain.

For example, to set the `vhba_desc_max_ncookies` property value to `12`, add the following line to the `/etc/system` file:

```
set vhba:vhba_desc_ncookies = 12
```

◆ ◆ ◆ CHAPTER 12

Using Virtual Networks

This chapter describes how to use a virtual network with Oracle VM Server for SPARC software, and covers the following topics:

Oracle Solaris OS networking changed a great deal between the Oracle Solaris 10 OS and the Oracle Solaris 11 OS. For information about issues to consider, see "Oracle Solaris 10 Networking Overview" on page 225, "Oracle Solaris 11 Networking Overview" on page 222, and "Oracle Solaris 11 Networking-Specific Feature Differences" on page 277.

Introduction to a Virtual Network

A virtual network enables domains to communicate with each other without using any external physical networks. A virtual network also can enable domains to use the same physical network interface to access a physical network and communicate with remote systems. A virtual network is created by having a virtual switch to which you can connect virtual network devices.

Oracle Solaris networking differs greatly between the Oracle Solaris 10 OS and the Oracle Solaris 11 OS. The following two sections provide overview information about networking for each OS.

Note – Oracle Solaris 10 networking behaves the same as it would on a domain or a system. The same is true for Oracle Solaris 11 networking. For more information about Oracle Solaris OS networking, see Oracle Solaris 10 Documentation and Oracle Solaris 11.3 Documentation.

The feature differences between Oracle Solaris 10 and Oracle Solaris 11 networking are described in "Oracle Solaris 11 Networking Overview" on page 222.

Oracle Solaris 11 Networking Overview

The Oracle Solaris 11 OS introduced many new networking features, which are described in the Oracle Solaris 11 networking documentation at Oracle Solaris 11.3 Documentation.

The following Oracle Solaris 11 networking features are important to understand when you use the Oracle VM Server for SPARC software:

- All network configuration is performed by the `ipadm` and `dladm` commands.
- The "vanity name by default" feature generates generic link names, such as `net0`, for all physical network adapters. This feature also generates generic names for virtual switches (`vsw`*n*) and virtual network devices (`vnet`*n*), which appear like physical network adapters to the OS. To identify the generic link name that is associated with a physical network device, use the `dladm show-phys` command.

  By default in Oracle Solaris 11, physical network device names use generic "vanity" names. Generic names, such as `net0`, are used instead of device driver names, such as `nxge0`, which were used in Oracle Solaris 10.

  The following command creates a virtual switch for the `primary` domain by specifying the generic name, `net0`, instead of a driver name, such as `nxge0`:

  ```
  primary# ldm add-vsw net-dev=net0 primary-vsw0 primary
  ```
- The Oracle Solaris 11 OS uses virtual network interface cards (VNICs) to create internal virtual networks.

  A VNIC is a virtual instantiation of a physical network device that can be created from the physical network device and assigned to a zone.

- Use the Oracle Solaris 11 `DefaultFixed` network configuration profile (NCP) when configuring the Oracle VM Server for SPARC software.

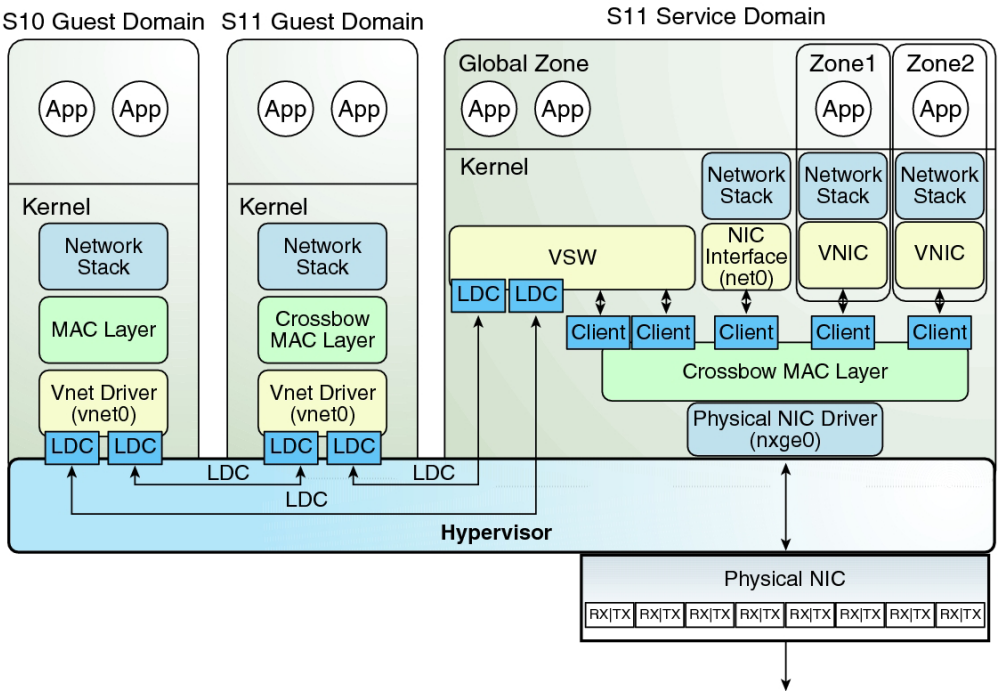
  For Oracle Solaris 11 domains, use the `DefaultFixed` NCP. You can enable this profile during or after installation. During an Oracle Solaris 11 installation, select the Manual networking configuration.
- Do not replace the primary network interface with the virtual switch (`vsw`) interface. The service domain can use the existing primary network interface to communicate with the guest domains that have virtual network devices connected to the same virtual switch.
- Do not use the physical network adapter's MAC address for the virtual switch because using the physical adapter's MAC address for the virtual switch conflicts with the primary network interface.

Note – In this release, use the `DefaultFixed` NCP to configure datalinks and network interfaces on Oracle Solaris 11 systems by using the `dladm` or `ipadm` command.

Ensure that the `DefaultFixed` NCP is enabled by using the `netadm list` command. See Chapter 7, "Using Datalink and Interface Configuration Commands on Profiles," in *Oracle Solaris Administration: Network Interfaces and Network Virtualization* .

The following diagram shows that a guest domain that runs the Oracle Solaris 10 OS is fully compatible with an Oracle Solaris 11 service domain. The only differences are features added or enhanced in the Oracle Solaris 11 OS.

FIGURE 12–1 Oracle VM Server for SPARC Network Overview for the Oracle Solaris 11 OS



The diagram shows that network device names, such as `nxge0` and `vnet0`, can be represented by generic link names, such as `net`*n* in Oracle Solaris 11 domains. Also note the following:

- The virtual switch in the service domain is connected to the guest domains, which enables guest domains to communicate with each other.
- The virtual switch is also connected to the physical network device `nxge0`, which enables guest domains to communicate with the physical network.

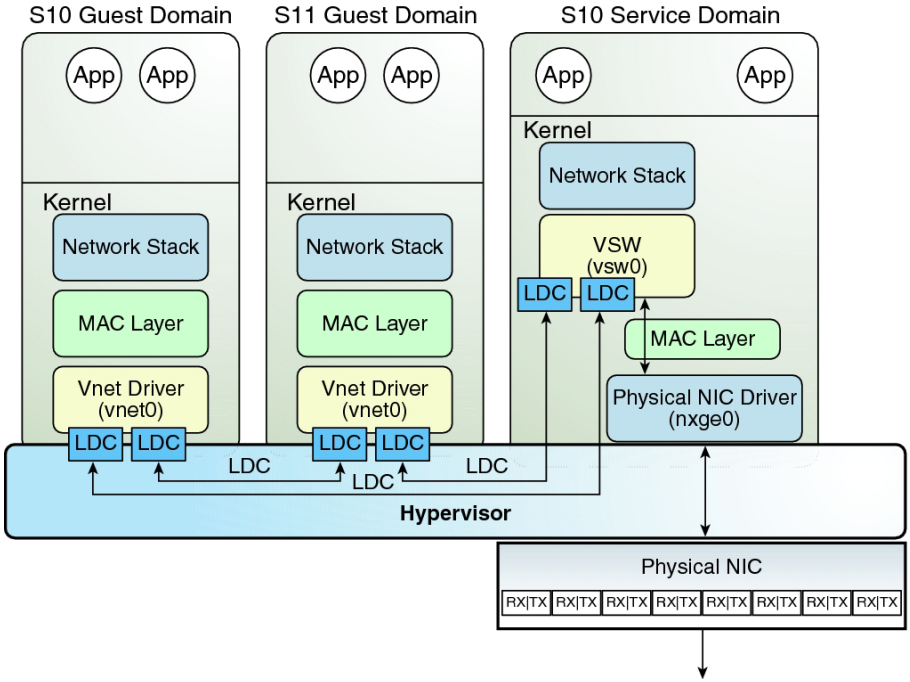
  The virtual switch also enables guest domains to communicate with the service domain network interface `net0` and with VNICs on the same physical network device as `nxge0`. This includes communication between the guest domains and the Oracle Solaris 11 service domain. Do not configure the virtual switch itself (the `vsw`*n* device) as a network device, as this functionality has been deprecated in Oracle Solaris 11 and is no longer supported.
- The virtual network device `vnet0` in an Oracle Solaris 10 guest domain can be configured as a network interface by using the `ifconfig` command.
- The virtual network device `vnet0` in an Oracle Solaris 11 guest domain might appear with a generic link name, such as `net0`. It can be configured as a network interface by using the `ipadm` command.

A virtual switch behaves like a regular physical network switch and switches network packets between the different systems to which it is connected. A system can be a guest domain, a service domain, or a physical network.

Oracle Solaris 10 Networking Overview

The following diagram shows that a guest domain that runs the Oracle Solaris 11 OS is fully compatible with an Oracle Solaris 10 service domain. The only differences are features added or enhanced in the Oracle Solaris 11 OS.

FIGURE 12–2 Oracle VM Server for SPARC Network Overview for the Oracle Solaris 10 OS



The previous diagram shows interface names such as `nxge0`, `vsw0`, and `vnet0` which apply to the Oracle Solaris 10 OS only. Also note the following:

- The virtual switch in the service domain is connected to the guest domains, which enables guest domains to communicate with each other.
- The virtual switch is also connected to the physical network interface `nxge0`, which enables guest domains to communicate with the physical network.

- The virtual switch network interface `vsw0` is created in the service domain, which enables the two guest domains to communicate with the service domain.
- The virtual switch network interface `vsw0` in the service domain can be configured by using the Oracle Solaris 10 `ifconfig` command.
- The virtual network device `vnet0` in an Oracle Solaris 10 guest domain can be configured as a network interface by using the `ifconfig` command.
- The virtual network device `vnet0` in an Oracle Solaris 11 guest domain might appear with a generic link name, such as `net0`. It can be configured as a network interface by using the `ipadm` command.

The virtual switch behaves like a regular physical network switch and switches network packets between the different systems, such as guest domains, the service domain, and the physical network, to which it is connected. The `vsw` driver provides the network device functionality that enables the virtual switch to be configured as a network interface.

Maximizing Virtual Network Performance

You can achieve high transfer rates for guest and external networks and for guest-to-guest communications when you configure your platform and the domains as described in this section. The virtual network stack introduces support for large segment offload (LSO), which produces high TCP performance without requiring the use of jumbo frames.

Hardware and Software Requirements

Meet the following requirements to maximize the network performance for your domains:

- **Hardware requirements.** These performance improvements are available only for the SPARC T4 server, SPARC T5 server, SPARC T7 series server, SPARC M5 server, SPARC M6 server, and SPARC M7 series server.
- **System firmware requirements.** These SPARC systems must run the latest system firmware. See "Fully Qualified System Firmware Versions" in *Oracle VM Server for SPARC 3.3 Installation Guide*.
- **Oracle Solaris OS requirements.** Ensure that the service domain and guest domain run the following Oracle Solaris OS versions.

Note – Running the fully qualified Oracle Solaris OS version provides you with access to new features. See "Fully Qualified Oracle Solaris OS Versions" in *Oracle VM Server for SPARC 3.3 Installation Guide*.

  - **Service domain.** At least the Oracle Solaris 11.1 SRU 9 OS or the Oracle Solaris 10 OS with the 150031-03 patch.

  - **Guest domain.** At least the Oracle Solaris 11.1 SRU 9 OS or the Oracle Solaris 10 OS with the 150031-03 patch.
- **CPU and memory requirements.** Ensure that you assign sufficient CPU and memory resources to the service domain and the guest domains.
  - **Service domain.** Because the service domain acts as a data proxy for the guest domains, assign at least 2 CPU cores and at least 4 Gbytes of memory to the service domain.
  - **Guest domain.** Configure each guest domain to be able to drive at least 10-Gbps performance. Assign at least 2 CPU cores and at least 4 Gbytes of memory to each guest domain.

Configuring Your Domains to Maximize the Performance of Your Virtual Network

In previous versions of Oracle VM Server for SPARC and the Oracle Solaris OS, you could improve your network performance by configuring jumbo frames. This configuration is no longer required and unless required for another reason, using the standard MTU value of 1500 for your service and guest domains is best.

To achieve the improved networking performance, set the `extended-mapin-space` property to `on` for the service domain and the guest domains. This is the default behavior.

```
primary# ldm set-domain extended-mapin-space=on domain-name
```

To check the `extended-mapin-space` property value, run the following command:

```
primary# ldm list -l domain-name |grep extended-mapin
extended-mapin-space=on
```

Note – A change to the `extended-mapin-space` property value triggers a delayed reconfiguration on the `primary` domain. This situation requires a `primary` domain reboot. You also must first stop the guest domains before you change this property value.

Virtual Switch

A virtual switch (`vsw`) is a component running in a service domain and managed by the virtual switch driver. A virtual switch can be connected to some guest domains to enable network communications between those domains. In addition, if the virtual switch is also associated with a physical network interface, network communication is permitted between guest domains and the physical network over the physical network interface. When running in an Oracle Solaris 10 service domain, a virtual switch also has a network interface, `vsw`*n*, which

permits the service domain to communicate with the other domains that are connected to that virtual switch. The virtual switch can be used like any regular network interface and configured with the Oracle Solaris 10 `ifconfig` command.

Assigning a virtual network device to a domain creates an implicit dependency on the domain providing the virtual switch. You can view these dependencies or view domains that depend on this virtual switch by using the `ldm list-dependencies` command. See "Listing Domain I/O Dependencies" on page 382.

In an Oracle Solaris 11 service domain, the virtual switch cannot be used as a regular network interface. If the virtual switch is connected to a physical network interface, communication with the service domain is possible by using this physical interface. If configured without a physical interface, you can enable communication with the service domain by using an `etherstub` as the network device (`net-dev`) that is connected with a VNIC.

To determine which network device to use as the back-end device for the virtual switch, search for the physical network device in the `dladm show-phys` output or use the `ldm list-netdev` command to list the network devices for logical domains.

Note – When a virtual switch is added to an Oracle Solaris 10 service domain, its network interface is not created. So, by default, the service domain is unable to communicate with the guest domains connected to its virtual switch. To enable network communications between guest domains and the service domain, the network interface of the associated virtual switch must be created and configured in the service domain. See "Enabling Networking Between the Oracle Solaris 10 Service Domain and Other Domains" on page 52 for instructions.

This situation occurs *only* for the Oracle Solaris 10 OS and *not* for the Oracle Solaris 11 OS.

You can add a virtual switch to a domain, set options for a virtual switch, and remove a virtual switch by using the `ldm add-vsw`, `ldm set-vsw`, and `ldm rm-vsw` commands, respectively. See the `ldm(1M)` man page.

When you create a virtual switch on a VLAN tagged instance of a NIC or an aggregation, you must specify the NIC (`nxge0`), the aggregation (`aggr3`), or the vanity name (`net0`) as the value of the `net-dev` property when you use the `ldm add-vsw` or `ldm set-vsw` command.

Note – Starting with the Oracle Solaris 11.2 SRU 1 OS, you can dynamically update the `net-dev` property value by using the `ldm set-vsw` command. In previous Oracle Solaris OS releases, using the `ldm set-vsw` command to update the `net-dev` property value in the `primary` domain causes the `primary` domain to enter a delayed reconfiguration.

You cannot add a virtual switch on top of an InfiniBand IP-over-InfiniBand (IPoIB) network device. Although the `ldm add-vsw` and `ldm add-vnet` commands appear to succeed, no data

will flow because these devices transport IP packets by means of the InfiniBand transport layer. The virtual switch only supports Ethernet as a transport layer.

The following command creates a virtual switch on a physical network adapter called `net0`:

```
primary# ldm add-vsw net-dev=net0 primary-vsw0 primary
```

The following example uses the `ldm list-netdev -b` command to show only the valid virtual switch back-end devices for the `svcdom` service domain.

```
primary# ldm list-netdev -b svcdom
DOMAIN
svcdom

NAME              CLASS MEDIA STATE   SPEED OVER   LOC
----              ----- ----- -----   ----- ----   ---
net0              PHYS  ETHER up      10000 ixgbe0 /SYS/MB/RISER1/PCIE
net1              PHYS  ETHER unknown 0     ixgbe1 /SYS/MB/RISER1/PCIE4
net2              ESTUB ETHER unknown 0     --     --
net3              ESTUB ETHER unknown 0     --     --
ldoms-estub.vsw0 ESTUB ETHER unknown 0     --     --
```

Virtual Network Device

A virtual network device is a virtual device that is defined in a domain connected to a virtual switch. A virtual network device is managed by the virtual network driver, and it is connected to a virtual network through the hypervisor using logical domain channels (LDCs).

A virtual network device can be used as a network interface with the name `vnet`*n*, which can be used like any regular network interface and configured with the Oracle Solaris 10 `ifconfig` command or the Oracle Solaris 11 `ipadm` command.

Note – For Oracle Solaris 11, the devices are assigned generic names, so `vnet`*n* would use a generic name, such as `net0`.

You can add a virtual network device to a domain, set options for an existing virtual network device, and remove a virtual network device by using the `ldm add-vnet`, `ldm set-vnet`, and `ldm rm-vnet` commands, respectively. See the `ldm(1M)` man page.

See the information about Oracle VM Server for SPARC networking for Oracle Solaris 10 and Oracle Solaris 11 in Figure 12–2 and Figure 12–1, respectively.

Inter-Vnet LDC Channels

By default, the Logical Domains Manager would assign LDC channels in the following manner:

- An LDC channel would be assigned between the virtual network devices and the virtual switch device.
- An LDC channel would be assigned between each pair of virtual network devices that are connected to the same virtual switch device (inter-vnet).

The inter-vnet LDC channels are configured so that virtual network devices can communicate directly to achieve high guest-to-guest communications performance. However, as the number of virtual network devices in a virtual switch device increases, the number of required LDC channels for inter-vnet communications increases quadratically.

You can choose to enable or disable inter-vnet LDC channel allocation for all virtual network devices attached to a given virtual switch device. By disabling this allocation, you can reduce the consumption of LDC channels, which are limited in number.

Disabling this allocation is useful in the following situations:

- When guest-to-guest communications performance is not of primary importance
- When a large number of virtual network devices are required in a virtual switch device

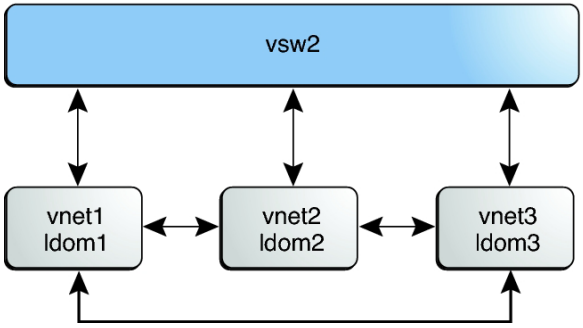
By not assigning inter-vnet channels, more LDC channels are available for use to add more virtual I/O devices to a guest domain.

Note – If guest-to-guest performance is of higher importance than increasing the number of virtual network devices in the system, do not disable inter-vnet LDC channel allocation.

You can use the `ldm add-vsw` and the `ldm set-vsw` commands to specify a value of `on` or `off` for the `inter-vnet-link` property.

The following figure shows a typical virtual switch that has three virtual network devices. The `inter-vnet-link` property is set to `on`, which means that inter-vnet LDC channels are allocated. The guest-to-guest communications between `vnet1` and `vnet2` is performed directly without going through the virtual switch.

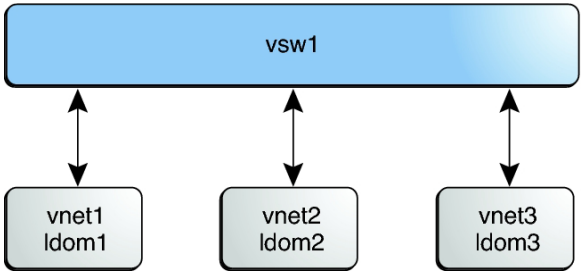
FIGURE 12–3 Virtual Switch Configuration That Uses Inter-Vnet Channels



The following figure shows the same virtual switch configuration with the `inter-vnet-link` property set to `off`. The inter-vnet LDC channels are not allocated. Fewer LDC channels are used than when the `inter-vnet-link` property is set to `on`. In this configuration, guest-to-guest communications between `vnet1` and `vnet2` must go through `vsw1`.

Note – Disabling the assignment of inter-vnet LDC channels does not prevent guest-to-guest communications. Instead, all guest-to-guest communications traffic goes through the virtual switch rather than directly from one guest domain to another guest domain.

FIGURE 12–4 Virtual Switch Configuration That Does Not Use Inter-Vnet Channels



Viewing Network Device Configurations and Statistics

The `ldm list-netdev` and `ldm list-netstat` commands enable you to view information about the network devices in the system and the network statistics, respectively. As a result, you have a centralized view of the network devices and statistics in a given physical domain.

To use these commands, you must run at least the Oracle Solaris 11.2 SRU 1 OS in the guest domain.

EXAMPLE 12–1 Listing Network Device Configuration Information

The following example shows a short listing of the network devices for the `ldg1` domain by using the `ldm list-netdev` command.

```
primary# ldm list-netdev ldg1

DOMAIN
ldg1

NAME              CLASS MEDIA STATE   SPEED OVER      LOC
----              ----- ----- -----   ----- ----      ---
net0              VNET  ETHER up      0     --        primary-vsw0/vne t0_ldg1
net3              PHYS  ETHER up      10000 --        /SYS/MB/RISER1/PCIE4
net4              VSW   ETHER up      10000 --        ldg1-vsw1
net1              PHYS  ETHER up      10000 --        /SYS/MB/RISER1/PCIE4
net5              VNET  ETHER up      0     --        ldg1-vsw1/vnet1_ldg1
net6              VNET  ETHER up      0     --        ldg1-vsw1/vnet2_ldg1
aggr2             AGGR  ETHER unknown 0     net1,net3 --
ldoms-vsw0.vport3 VNIC  ETHER unknown 0     --        ldg1-vsw1/vnet2_ldg1
ldoms-vsw0.vport2 VNIC  ETHER unknown 0     --        ldg1-vsw1/vnet1_ldg1
ldoms-vsw0.vport1 VNIC  ETHER unknown 0     --        ldg1-vsw1/vnet2_ldg3
ldoms-vsw0.vport0 VNIC  ETHER unknown 0     --        ldg1-vsw1/vnet2_ldg2
```

EXAMPLE 12–2 Listing Detailed Network Device Configuration Information

The following example shows a detailed listing of the network devices for the `ldg1` domain by using the `ldm list-netdev -l` command.

```
primary# ldm list-netdev -l ldg1

-------------------------------------------------------------------
DOMAIN
ldg1

NAME              CLASS    MEDIA    STATE   SPEED OVER LOC
----              -----    -----    -----   ----- ---- ---
net0              VNET     ETHER    up      0     --   primary-vsw0/vnet0_ldg1
    [/virtual-devices@100/channel-devices@200/network@0]
    MTU       : 1500 [1500-1500]
    IPADDR    : 10.129.241.200/255.255.255.0
    MAC_ADDRS : 00:14:4f:fb:9c:df

net3              PHYS     ETHER    up      10000 --   /SYS/MB/RISER1/PCIE4
    [/pci@400/pci@1/pci@0/pci@0/network@0]
    MTU       : 1500 [576-15500]
    MAC_ADDRS : a0:36:9f:0a:c5:d2

net4              VSW      ETHER    up      10000 --   ldg1-vsw1
    [/virtual-devices@100/channel-devices@200/virtual-network-switch@0]
    MTU       : 1500 [1500-1500]
    IPADDR    : 192.168.1.2/255.255.255.0
    MAC_ADDRS : 00:14:4f:fb:61:6e

net1              PHYS     ETHER    up      10000 --   /SYS/MB/RISER1/PCIE4
    [/pci@400/pci@1/pci@0/pci@0/network@0,1]
    MTU       : 1500 [576-15500]
```

EXAMPLE 12–2 Listing Detailed Network Device Configuration Information *(Continued)*

```
    MAC_ADDRS : a0:36:9f:0a:c5:d2

net5               VNET     ETHER    up      0     --   ldg1-vsw1/vnet1_ldg1
    [/virtual-devices@100/channel-devices@200/network@1]
    MTU       : 1500 [1500-1500]
    IPADDR    : 0.0.0.0  /255.0.0.0
              : fe80::214:4fff:fef8:5062/ffc0::
    MAC_ADDRS : 00:14:4f:f8:50:62

net6               VNET     ETHER    up      0     --   ldg1-vsw1/vnet2_ldg1
    [/virtual-devices@100/channel-devices@200/network@2]
    MTU       : 1500 [1500-1500]
    IPADDR    : 0.0.0.0  /255.0.0.0
              : fe80::214:4fff:fef8:af92/ffc0::
    MAC_ADDRS : 00:14:4f:f8:af:92

aggr2              AGGR     ETHER    unknown  0    net1,net3 --
    MODE      : TRUNK
    POLICY    : L2,L3
    LACP_MODE : ACTIVE
    MEMBER    : net1 [PORTSTATE = attached]
    MEMBER    : net3 [PORTSTATE = attached]
    MAC_ADDRS : a0:36:9f:0a:c5:d2

ldoms-vsw0.vport3  VNIC     ETHER    unknown  0    -- ldg1-vsw1/vnet2_ldg1
    MTU       : 1500 [576-1500]
    MAC_ADDRS : 00:14:4f:f8:af:92

ldoms-vsw0.vport2  VNIC     ETHER    unknown  0    -- ldg1-vsw1/vnet1_ldg1
    MTU       : 1500 [576-1500]
    MAC_ADDRS : 00:14:4f:f8:50:62

ldoms-vsw0.vport1  VNIC     ETHER    unknown  0    -- ldg1-vsw1/vnet2_ldg3
    MTU       : 1500 [576-1500]
    MAC_ADDRS : 00:14:4f:f9:d3:88

ldoms-vsw0.vport0  VNIC     ETHER    unknown  0    -- ldg1-vsw1/vnet2_ldg2
    MTU       : 1500 [576-1500]
    MAC_ADDRS : 00:14:4f:fa:47:f4
              : 00:14:4f:f9:65:b5
              : 00:14:4f:f9:60:3f
```

EXAMPLE 12–3 Listing Network Device Statistics

The `ldm list-netstat` command shows network statistics for one or more domains in the system.

The following example shows the default network statistics for all domains in the system.

```
primary# ldm list-netstat

DOMAIN
primary
```

EXAMPLE 12–3 Listing Network Device Statistics *(Continued)*

```
NAME              IPACKETS     RBYTES       OPACKETS     OBYTES
----              --------     ------       --------     ------
net3              0            0            0            0
net0              2.72M        778.27M      76.32K       6.01M
net4              2.72M        778.27M      76.32K       6.01M
net6              2            140          1.30K        18.17K
net7              0            0            0            0
net2              0            0            0            0
net1              0            0            0            0
aggr1             0            0            0            0
ldoms-vsw0.vport0 935.40K      74.59M       13.15K       984.43K
ldoms-vsw0.vport1 933.26K      74.37M       11.42K       745.15K
ldoms-vsw0.vport2 933.24K      74.37M       11.46K       747.66K
ldoms-vsw1.vport1 202.26K      17.99M       179.75K      15.69M
ldoms-vsw1.vport0 202.37K      18.00M       189.00K      16.24M
------------------------------------------------------------------------
DOMAIN
ldg1

NAME              IPACKETS     RBYTES       OPACKETS     OBYTES
----              --------     ------       --------     ------
net0              5.19K        421.57K      68           4.70K
net3              0            0            2.07K        256.93K
net4              0            0            4.37K        560.17K
net1              0            0            2.29K        303.24K
net5              149          31.19K       78           17.00K
net6              147          30.51K       78           17.29K
aggr2             0            0            0            0
ldoms-vsw0.vport3 162          31.69K       52           14.11K
ldoms-vsw0.vport2 163          31.74K       51           13.76K
ldoms-vsw0.vport1 176          42.99K       25           1.50K
ldoms-vsw0.vport0 158          40.19K       45           4.42K
------------------------------------------------------------------------
DOMAIN
ldg2

NAME              IPACKETS     RBYTES       OPACKETS     OBYTES
----              --------     ------       --------     ------
net0              5.17K        418.90K      71           4.88K
net1              2.70K        201.67K      2.63K        187.01K
net2              132          36.40K       1.51K        95.07K
------------------------------------------------------------------------
DOMAIN
ldg3

NAME              IPACKETS     RBYTES       OPACKETS     OBYTES
----              --------     ------       --------     ------
net0              5.16K        417.43K      72           4.90K
net1              2.80K        206.12K      2.67K        190.36K
net2              118          35.00K       1.46K        87.78K
```

Controlling the Amount of Physical Network Bandwidth That Is Consumed by a Virtual Network Device

The bandwidth resource control feature enables you to limit the physical network bandwidth consumed by a virtual network device. This feature is supported on a service domain that runs at least the Oracle Solaris 11 OS and is configured with a virtual switch. Oracle Solaris 10 service domains silently ignore network bandwidth settings. This feature ensures that one guest domain does not take over the available physical network bandwidth and leave none for the others.

Use the `ldm add-vnet` and `ldm set-vnet` commands to specify the bandwidth limit by providing a value for the `maxbw` property. Use the `ldm list-bindings` or the `ldm list-domain -o network` command to view the `maxbw` property value for an existing virtual network device. The minimum bandwidth limit is 10 Mbps.

Network Bandwidth Limitations

Note – This feature is not supported by a Hybrid I/O-enabled virtual network device. The `maxbw` property is not enforced for Hybrid mode virtual networks because the Hybrid I/O assigns a specific unit of hardware resource that cannot be changed to limit bandwidth. To limit a virtual network device's bandwidth, you must disable the Hybrid mode.

The bandwidth resource control applies only to the traffic that goes through the virtual switch. Thus, inter-vnet traffic is not subjected to this limit. If you do not have a physical backend device configured, you can ignore bandwidth resource control.

The minimum supported bandwidth limit depends on the Oracle Solaris network stack in the service domain. The bandwidth limit can be configured with any desired high value. There is no upper limit. The bandwidth limit ensures only that the bandwidth does not exceed the configured value. Thus, you can configure a bandwidth limit with a value greater than the link speed of the physical network device that is assigned to the virtual switch.

Setting the Network Bandwidth Limit

Use the `ldm add-vnet` command to create a virtual network device and specify the bandwidth limit by providing a value for the `maxbw` property.

```
primary# ldm add-vnet maxbw=limit if-name vswitch-name domain-name
```

Use the `ldm set-vnet` command to specify the bandwidth limit for an existing virtual network device.

```
primary# ldm set-vnet maxbw=limit if-name domain-name
```

You can also clear the bandwidth limit by specifying a blank value for the `maxbw` property:

```
primary# ldm set-vnet maxbw= if-name domain-name
```

The following examples show how to use the `ldm` command to specify the bandwidth limit. The bandwidth is specified as an integer with a unit. The unit is `M` for megabits-per-second or `G` for gigabits-per-second. The unit is megabits-per-second if you do not specify a unit.

EXAMPLE 12–4 Setting the Bandwidth Limit When Creating a Virtual Network Device

The following command creates a virtual network device (`vnet0`) that has a bandwidth limit of 100 Mbps.

```
primary# ldm add-vnet maxbw=100M vnet0 primary-vsw0 ldg1
```

The following command would issue an error message when attempting to set a bandwidth limit below the minimum value, which is 10 Mbps.

```
primary# ldm add-vnet maxbw=1M vnet0 primary-vsw0 ldg1
```

EXAMPLE 12–5 Setting the Bandwidth Limit on an Existing Virtual Network Device

The following commands sets the bandwidth limit to 200 Mbps on the existing `vnet0` device.

Depending on the real-time network traffic pattern, the amount of bandwidth might not reach the specified limit of 200 Mbps. For example, the bandwidth might be 95 Mbps, which does not exceed the 200 Mbps limit.

```
primary# ldm set-vnet maxbw=200M vnet0 ldg1
```

The following command sets the bandwidth limit to 2 Gbps on the existing `vnet0` device.

Because there is no upper limit on bandwidth in the MAC layer, you can still set the limit to be 2 Gbps even if the underlying physical network speed is less than 2 Gbps. In such a case, there is no bandwidth limit effect.

```
primary# ldm set-vnet maxbw=2G vnet0 ldg1
```

EXAMPLE 12–6 Clearing the Bandwidth Limit on an Existing Virtual Network Device

The following command clears the bandwidth limit on the specified virtual network device (`vnet0`). By clearing this value, the virtual network device uses the maximum bandwidth available, which is provided by the underlying physical device.

```
primary# ldm set-vnet maxbw= vnet0 ldg1
```

EXAMPLE 12–7 Viewing the Bandwidth Limit of an Existing Virtual Network Device

The `ldm list-bindings` command shows the value of the `maxbw` property for the specified virtual network device, if defined.

The following command shows that the `vnet0` virtual network device has a bandwidth limit of 15 Mbps. If no bandwidth limit is set, the `MAXBW` field is blank.

```
primary# ldm list-bindings
...
VSW
  NAME         MAC               NET-DEV   ID   DEVICE   LINKPROP
   primary-vsw0 00:14:4f:f9:95:97 net0      0    switch@0 1

   DEFAULT-VLAN-ID PVID VID       MTU   MODE   INTER-VNET-LINK
   1               1              1500         on

   PEER       MAC               PVID VID MTU  MAXBW  LINKPROP INTERVNETLINK
   vnet0@ldg1 00:14:4f:fb:b8:c8 1        1500 15

...

   NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
   ldg1             bound      ------  5000    8     2G

   NETWORK
      NAME             SERVICE                     ID   DEVICE
      vnet0            primary-vsw0@primary        0    network@0

      MAC               MODE   PVID VID     MTU   MAXBW      LINKPROP
      00:14:4f:fb:b8:c8 1                   1500  15

      PEER                       MAC               MODE   PVID VID
      primary-vsw0@primary       00:14:4f:f9:95:97  1

      MTU   MAXBW      LINKPROP
      1500
```

You can also use the `dladm show-linkprop` command to view the `maxbw` property value as follows:

```
# dladm show-linkprop -p maxbw
LINK              PROPERTY PERM VALUE  EFFECTIVE DEFAULT POSSIBLE
...
ldoms-vsw0.vport0 maxbw    rw   15     15        --      --
```

Virtual Device Identifier and Network Interface Name

When you add a virtual switch or virtual network device to a domain, you can specify its device number by setting the `id` property.

```
primary# ldm add-vsw [id=switch-id] vswitch-name domain-name
primary# ldm add-vnet [id=network-id] if-name vswitch-name domain-name
```

Each virtual switch and virtual network device of a domain has a unique device number that is assigned when the domain is bound. If a virtual switch or virtual network device was added with an explicit device number (by setting the `id` property), the specified device number is used. Otherwise, the system automatically assigns the lowest device number available. In that case, the device number assigned depends on how virtual switch or virtual network devices were added to the system. The device number eventually assigned to a virtual switch or virtual network device is visible in the output of the `ldm list-bindings` command when a domain is bound.

The following example shows that the `primary` domain has one virtual switch, `primary-vsw0`. This virtual switch has a device number of `0` (`switch@0`).

```
primary# ldm list-bindings primary
...
VSW
    NAME         MAC               NET-DEV DEVICE   DEFAULT-VLAN-ID PVID VID MTU MODE
    primary-vsw0 00:14:4f:fb:54:f2 net0    switch@0 1               1    5,6 1500
...
```

The following example shows that the `ldg1` domain has two virtual network devices: `vnet` and `vnet1`. The `vnet` device has a device number of `0` (`network@0`) and the `vnet1` device has a device number of `1` (`network@1`).

```
primary# ldm list-bindings ldg1
...
NETWORK
    NAME  SERVICE              DEVICE    MAC               MODE   PVID VID MTU
    vnet  primary-vsw0@primary network@0 00:14:4f:fb:e0:4b hybrid 1        1500
    ...
    vnet1 primary-vsw0@primary network@1 00:14:4f:f8:e1:ea        1        1500
...
```

Similarly, when a domain with a virtual network device is running the Oracle Solaris OS, the virtual network device has a network interface, `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*.

Note – On Oracle Solaris 11 systems, generic link names in the form of `net`*n* are assigned to both `vsw`*n* and `vnet`*n*. Use the `dladm show-phys` command to identify which `net`*n* names map to the `vsw`*n* and `vnet`*n* devices.

Caution – The Oracle Solaris OS preserves the mapping between the name of a network interface and a virtual switch or a virtual network device based on the device number. If a device number is not explicitly assigned to a virtual switch or virtual network device, its device number can change when the domain is unbound and is later bound again. In that case, the network interface name assigned by the OS running in the domain can also change and make the existing system configuration unusable. This situation might happen, for example, when a virtual switch or a virtual network interface is removed from the configuration of the domain.

You cannot use the `ldm list-*` commands to directly determine the Oracle Solaris OS network interface name that corresponds to a virtual switch or virtual network device. However, you can obtain this information by using a combination of the output from `ldm list -l` command and from the entries under `/devices` on the Oracle Solaris OS.

Finding the Oracle Solaris 11 Network Interface Name

On Oracle Solaris 11 systems, you can use the `ldm list-netdev` command to find the Oracle Solaris OS network interface names. For more information, see the `ldm(1M)` man page.

The following example shows the `ldm list-netdev` and `ldm list -o network` commands. The `ldm list -o network` command shows the virtual network devices in the `NAME` field. The `ldm list-netdev` output shows the corresponding OS interface name in the `NAME` column.

```
primary# ldm list -o network ldg1
....
NETWORK
    NAME        SERVICE              ID DEVICE    MAC               MODE
    PVID VID MTU   MAXBW LINKPROP
    vnet0-ldg1  primary-vsw0@primary 0  network@0 00:14:4f:fa:eb:4e 1
             1500
    vnet1-ldg1  svcdom-vsw0@svcdom   1  network@1 00:14:4f:f8:53:45 4
             1500
           PVLAN :400,community

primary# ldm list-netdev ldg1
DOMAIN
ldg1

NAME CLASS MEDIA STATE   SPEED OVER  LOC
---- ----- ----- -----   ----- ----  ---
net0 VNET  ETHER up      0     vnet0 primary-vsw0/vnet0-ldg1
```

```
net1 VNET  ETHER up      0     vnet1 svcdom-vsw0/vnet1-ldg1
net2 VNET  ETHER unknown 0     vnet2 svcdom-vsw1/vnet2-ldg1
```

To verify that the `ldm list-netdev` output is correct, run the `dladm show-phys` and `dladm show-linkprop -p mac-address` commands from the `ldg1`:

```
ldg1# dladm show-phys
LINK     MEDIA     STATE      SPEED  DUPLEX         DEVICE
net0     Ethernet  up         0      unknown        vnet0
net1     Ethernet  up         0      unknown        vnet1
net2     Ethernet  unknown    0      unknown        vnet2

ldg1# dladm show-linkprop -p mac-address
LINK PROPERTY    PERM VALUE            EFFECTIVE        DEFAULT          POSSIBLE
net0 mac-address rw   0:14:4f:fa:eb:4e 0:14:4f:fa:eb:4e 0:14:4f:fa:eb:4e --
net1 mac-address rw   0:14:4f:f8:53:45 0:14:4f:f8:53:45 0:14:4f:f8:53:45 --
```

How to Find the Oracle Solaris OS Network Interface Name

This procedure describes how to find the Oracle Solaris OS network interface name in `ldg1` that corresponds to `net-c`. This example also shows differences if you are looking for the network interface name of a virtual switch instead of a virtual network device. In this example procedure, guest domain `ldg1` contains two virtual network devices, `net-a` and `net-c`.

1 Use the `ldm` command to find the virtual network device number for `net-c`.

```
primary# ldm list -l ldg1
...
NETWORK
NAME         SERVICE                     DEVICE       MAC
net-a        primary-vsw0@primary        network@0    00:14:4f:f8:91:4f
net-c        primary-vsw0@primary        network@2    00:14:4f:f8:dd:68
...
```

The virtual network device number for `net-c` is 2 (`network@2`).

To determine the network interface name of a virtual switch, find the virtual switch device number, *n*, as `switch@`*n*.

2 Find the corresponding network interface on `ldg1` by logging into `ldg1` and finding the entry for this device number under `/devices`.

```
ldg1# uname -n
ldg1
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. **Verify that `vnet1` has the MAC address `00:14:4f:f8:dd:68` as shown in the `ldm list -l` output for `net-c` in Step 1.**

   - **Oracle Solaris 11 OS.**

     a. **Determine the name of the interface to specify for `vnet1`.**

     ```
     ldg1# dladm show-phys |grep vnet1
     net2      Ethernet      up      0      unknown   vnet1
     ```

     b. **Determine the MAC address of `net2`.**

     ```
     # dladm show-linkprop -p mac-address net2
     LINK PROPERTY    PERM VALUE             EFFECTIVE         DEFAULT POSSIBLE
     net2 mac-address rw   00:14:4f:f8:dd:68 00:14:4f:f8:dd:68 --      --
     ```

     This example MAC address matches the output of the `ldm list -l` command for `net-c` in Step 1.

   - **Oracle Solaris 10 OS.**

     ```
     ldg1# 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, and a virtual switch, or you can manually assign MAC addresses from your own pool of assigned MAC addresses. The `ldm` subcommands that set MAC addresses are `add-domain`, `add-vsw`, `set-vsw`, `add-vnet`, and `set-vnet`. If you do not specify a MAC address in these subcommands, the Logical Domains Manager assigns one automatically.

The advantage to having the Logical Domains Manager assign the MAC addresses is that it uses the block of MAC addresses dedicated for use with logical domains. Also, the Logical Domains Manager detects and prevents MAC address collisions with other Logical Domains Manager instances on the same subnet. This behavior frees you from having to manually manage your pool of MAC addresses.

MAC address assignment happens as soon as a logical domain is created or a network device is configured into a domain. In addition, the assignment is persistent until the device, or the logical domain itself, is removed.

Range of MAC Addresses Assigned to Domains

Domains have been assigned the following block of 512K MAC addresses:

```
00:14:4F:F8:00:00 ~ 00:14:4F:FF:FF:FF
```

The lower 256K addresses are used by the Logical Domains Manager for automatic MAC address allocation, and you cannot manually request an address in this range:

```
00:14:4F:F8:00:00 - 00:14:4F:FB:FF:FF
```

You can use the upper half of this range for manual MAC address allocation:

```
00:14:4F:FC:00:00 - 00:14:4F:FF:FF:FF
```

Note – In Oracle Solaris 11, the allocation of MAC addresses for VNICs uses addresses outside these ranges.

Automatic Assignment Algorithm

When you do not specify a MAC address when creating a logical domain or a network device, the Logical Domains Manager automatically allocates and assigns a MAC address to that logical domain or network device.

To obtain this MAC address, the Logical Domains Manager iteratively attempts to select an address and then checks for potential collisions. The MAC address is randomly selected from the 256K range of addresses set aside for this purpose. The MAC address is selected randomly to lessen the chance of a duplicate MAC address being selected as a candidate.

The address selected is then checked against other Logical Domains Managers on other systems to prevent duplicate MAC addresses from actually being assigned. The algorithm employed is described in "Duplicate MAC Address Detection" on page 242. If the address is already assigned, the Logical Domains Manager iterates, choosing another address and again checking for collisions. This process continues until a MAC address is found that is not already allocated or a time limit of 30 seconds has elapsed. If the time limit is reached, then the creation of the device fails, and an error message similar to the following is shown.

```
Automatic MAC allocation failed.  Please set the vnet MAC address manually.
```

Duplicate MAC Address Detection

To prevent the same MAC address from being allocated to different devices, the Logical Domains Manager checks with other Logical Domains Managers on other systems by sending a multicast message over the control domain's default network interface, including the address

that the Logical Domains Manager wants to assign to the device. The Logical Domains Manager attempting to assign the MAC address waits for one second for a response. If a different device on another Oracle VM Server for SPARC-enabled system has already been assigned that MAC address, the Logical Domains Manager on that system sends a response containing the MAC address in question. If the requesting Logical Domains Manager receives a response, it notes the chosen MAC address has already been allocated, chooses another, and iterates.

By default, these multicast messages are sent only to other managers on the same subnet. The default time-to-live (TTL) is `1`. The TTL can be configured using the Service Management Facilities (SMF) property `ldmd/hops`.

Each Logical Domains Manager is responsible for the following:

- Listening for multicast messages
- Keeping track of MAC addresses assigned to its domains
- Looking for duplicates
- Responding so that duplicates do not occur

If the Logical Domains Manager on a system is shut down for any reason, duplicate MAC addresses could occur while the Logical Domains Manager is down.

Automatic MAC allocation occurs at the time the logical domain or network device is created and persists until the device or the logical domain is removed.

Note – A detection check for duplicate MAC addresses is performed when the logical domain or network device is created, and the logical domain is started.

Using Network Adapters With Domains That Run Oracle Solaris 10

In an Oracle Solaris 10 logical domains environment, the virtual switch service running in a service domain can directly interact with GLDv3-compliant network adapters. Though non-GLDv3 compliant network adapters can be used in these systems, the virtual switch cannot interface with them directly. See "Configuring a Virtual Switch and the Service Domain for NAT and Routing" on page 244 for information about how to use non-GLDv3 compliant network adapters.

Note – GLDv3 compliance is not an issue for Oracle Solaris 11 environments.

For more information about using link aggregation, see "Using Link Aggregation With a Virtual Switch" on page 271.

▼ How to Determine Whether a Network Adapter Is GLDv3-Compliant

This procedure applies only to Oracle Solaris 10 domains.

- **Determine whether the network adapter is GLDv3-compliant.**

  The following example uses `bge0` as the network device name.

  ```
  # dladm show-link bge0
  bge0            type: non-vlan   mtu: 1500      device: bge0
  ```

  The value of the `type:` field is one of the following:

  - GLDv3-compliant drivers have a type of `non-vlan` or `vlan`.
  - Non-GLDv3-compliant drivers have a type of `legacy`.

Configuring a Virtual Switch and the Service Domain for NAT and Routing

In the Oracle Solaris 10 OS, the virtual switch (`vsw`) is a layer-2 switch, which also can be used as a network device in the service domain. The virtual switch can be configured to act only as a switch between the virtual network devices in the various logical domains but with no connectivity to a network outside the box through a physical device. In this mode, creating the `vsw` as a network device and enabling IP routing in the service domain enables virtual networks to communicate outside the box using the service domain as a router. This mode of operation is essential to provide external connectivity to the domains when the physical network adapter is not GLDv3-compliant.

The advantages of this configuration are:

- The virtual switch does not need to use a physical device directly and can provide external connectivity even when the underlying device is not GLDv3-compliant.
- The configuration can take advantage of the IP routing and filtering capabilities of the Oracle Solaris OS.

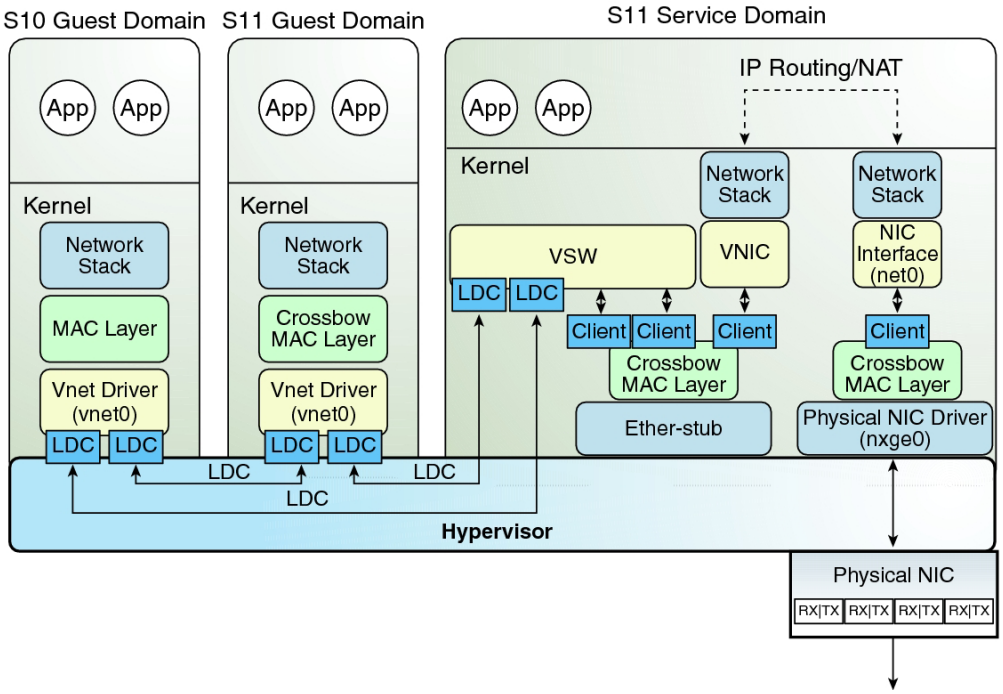
Configuring NAT on an Oracle Solaris 11 System

The Oracle Solaris 11 network virtualization features include `etherstub`, which is a pseudo network device. This device provides functionality similar to physical network devices but only for private communications with its clients. This pseudo device can be used as a network back-end device for a virtual switch that provides the private communications between virtual networks. By using the `etherstub` device as a back-end device, guest domains can also communicate with VNICs on the same `etherstub` device. Using the `etherstub` device in this

way enables guest domains to communicate with zones in the service domain. Use the `dladm create-etherstub` command to create an `etherstub` device.

The following diagram shows how virtual switches, `etherstub` devices, and VNICs can be used to set up Network Address Translation (NAT) in a service domain.

FIGURE 12–5 Virtual Network Routing



You might consider using persistent routes. For more information, see "Troubleshooting Issues When Adding a Persistent Route" in *Troubleshooting Network Administration Issues in Oracle Solaris 11.3* and "Creating Persistent (Static) Routes" in *Configuring and Managing Network Components in Oracle Solaris 11.3* .

How to Set Up a Virtual Switch to Provide External Connectivity to Domains (Oracle Solaris 11)

1 Create an Oracle Solaris 11 `etherstub` device.

```
primary# dladm create-etherstub stub0
```

2 **Create a virtual switch that uses `stub0` as the physical back-end device.**

```
primary# ldm add-vsw net-dev=stub0 primary-stub-vsw0 primary
```

3 **Create a VNIC on the `stub0` device.**

```
primary# dladm create-vnic -l stub0 vnic0
```

4 **Configure `vnic0` as the network interface.**

```
primary# ipadm create-ip vnic0
primary# ipadm create-addr -T static -a 192.168.100.1/24  vnic0/v4static
```

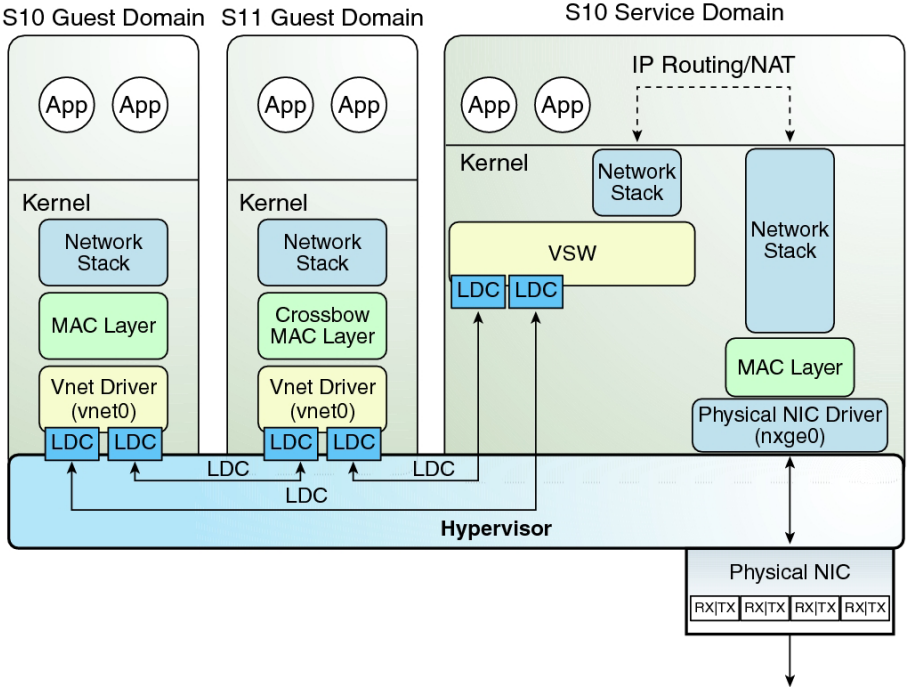
5 **Enable IPv4 forwarding and create NAT rules.**

See "Customizing IP Interface Properties and Addresses" in *Configuring and Managing Network Components in Oracle Solaris 11.3* and "Packet Forwarding and Routing on IPv4 Networks" in *Oracle Solaris Administration: IP Services* .

Configuring NAT on an Oracle Solaris 10 System

The following diagram shows how a virtual switch can be used to configure Network Address Translation (NAT) in a service domain to provide external connectivity for guest domains.

FIGURE 12–6 Virtual Network Routing



How to Set Up a Virtual Switch on Oracle Solaris 10 Service Domains to Provide External Connectivity to Domains

1. **Create a virtual switch that does not have an associated physical device.**

   If assigning an address, ensure that the virtual switch has a unique MAC address.

   ```
   primary# ldm add-vsw [mac-addr=xx:xx:xx:xx:xx:xx] ldg1-vsw0 ldg1
   ```

2. **Create the virtual switch as a network device in addition to the physical network device being used by the domain.**

   See "How to Configure the Virtual Switch as the Primary Interface" on page 52 for more information about creating the virtual switch.

3. **Configure the virtual switch device for DHCP, if needed.**

   See "How to Configure the Virtual Switch as the Primary Interface" on page 52 for more information about configuring the virtual switch device for DHCP.

4. **Create the `/etc/dhcp.vsw` file, if needed.**

5 **Configure IP routing in the service domain, and set up required routing tables in all the domains.**

For more information about IP routing, see "Packet Forwarding and Routing on IPv4 Networks" in *Oracle Solaris Administration: IP Services* .

Configuring IPMP in an Oracle VM Server for SPARC Environment

The Oracle VM Server for SPARC software supports link-based IP network multipathing (IPMP) with virtual network devices. When configuring an IPMP group with virtual network devices, configure the group to use link-based detection. If you are using older versions of the Oracle VM Server for SPARC (Logical Domains) software, you can only configure probe-based detection with virtual network devices.

Configuring Virtual Network Devices Into an IPMP Group in an Oracle Solaris 11 Domain

Figure 12–7 shows two virtual networks (`vnet0` and `vnet1`) connected to separate virtual switch instances (`vsw0` and `vsw1`) in the service domain, which in turn use two different physical interfaces. The physical interfaces are `net0` and `net1` in the Oracle Solaris 11 service domain.

If a physical link failure occurs in the service domain, the virtual switch device that is bound to that physical device detects the link failure. Then, the virtual switch device propagates the failure to the corresponding virtual network device that is bound to this virtual switch. The virtual network device sends notification of this link event to the IP layer in the guest `LDom_A`, which results in failover to the other virtual network device in the IPMP group.

FIGURE 12–7 Two Virtual Networks Connected to Separate Virtual Switch Instances (Oracle Solaris 11)

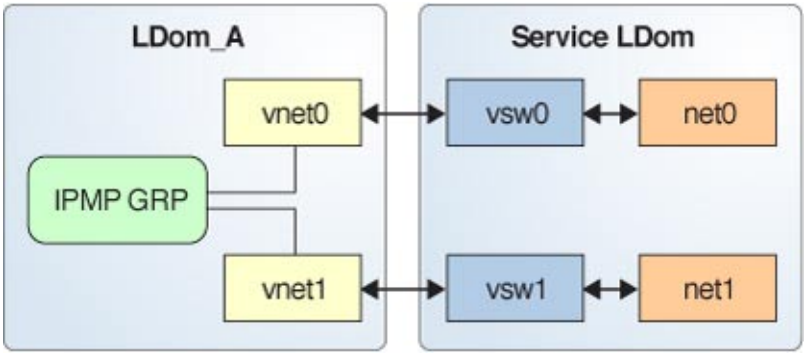
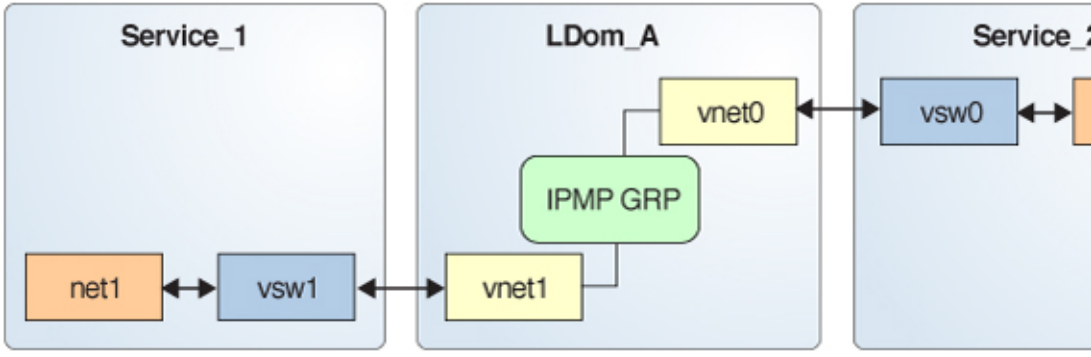


Figure 12–8 shows that you can achieve further reliability in the logical domain by connecting each virtual network device (`vnet0` and `vnet1`) to virtual switch instances in different service domains. In this case, in addition to physical network failure, `LDom_A` can detect virtual network failure and trigger a failover following a service domain crash or shutdown.

FIGURE 12–8 Virtual Network Devices Each Connected to Different Service Domains (Oracle Solaris 11)



For more information, see "Establishing an Oracle Solaris Network" in the Oracle Solaris 11.3 Information Library.

Configuring Virtual Network Devices Into an IPMP Group in an Oracle Solaris 10 Domain

Figure 12–9 shows two virtual networks (`vnet0` and `vnet1`) connected to separate virtual switch instances (`vsw0` and `vsw1`) in the service domain, which in turn use two different physical interfaces. The physical interfaces are `nxge0` and `nxge1` in the Oracle Solaris 10 service domain.

If a physical link failure occurs in the service domain, the virtual switch device that is bound to that physical device detects the link failure. Then, the virtual switch device propagates the failure to the corresponding virtual network device that is bound to this virtual switch. The virtual network device sends notification of this link event to the IP layer in the guest `LDom_A`, which results in failover to the other virtual network device in the IPMP group.

FIGURE 12–9 Two Virtual Networks Connected to Separate Virtual Switch Instances (Oracle Solaris 10)

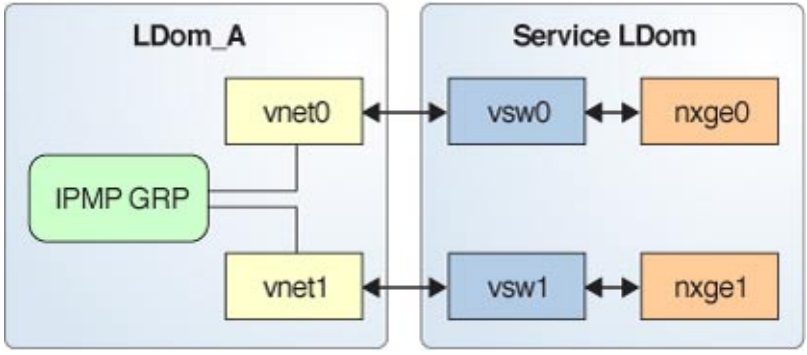
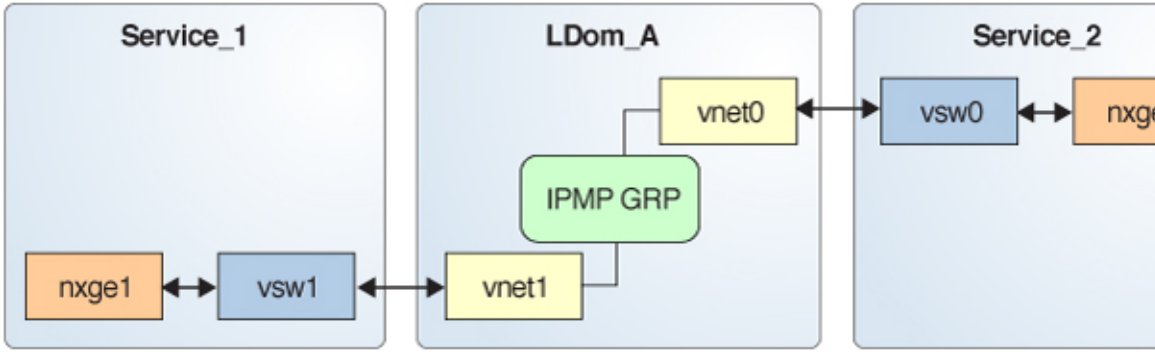


Figure 12–10 shows that you can achieve further reliability in the logical domain by connecting each virtual network device (`vnet0` and `vnet1`) to virtual switch instances in different service domains. In this case, in addition to physical network failure, `LDom_A` can detect virtual network failure and trigger a failover following a service domain crash or shutdown.

FIGURE 12–10 Virtual Network Devices Each Connected to Different Service Domains (Oracle Solaris 10)



For more information, see *Oracle Solaris Administration: IP Services* .

Configuring and Using IPMP in the Service Domain

On an Oracle Solaris 11 system, you can configure IPMP in a service domain by configuring physical interfaces into a group in the same way as on a system that has no virtual network or domains. On an Oracle Solaris 10 system, you can configure IPMP in the service domain by configuring virtual switch interfaces into a group. Figure 12–11 and Figure 12–12 show two virtual switch instances (`vsw0` and `vsw1`) that are bound to two different physical devices. The two virtual switch interfaces can then be created and configured into an IPMP group. In the event of a physical link failure, the virtual switch device that is bound to that physical device detects the link failure. Then, the virtual switch device sends notification of this link event to the IP layer in the service domain, which results in a failover to the other virtual switch device in the IPMP group. The two physical interfaces are `net0` and `net1` in Oracle Solaris 11 and `nxge0` and `nxge1` in Oracle Solaris 10.

FIGURE 12–11 Two Physical NICs Configured as Part of an IPMP Group (Oracle Solaris 11)

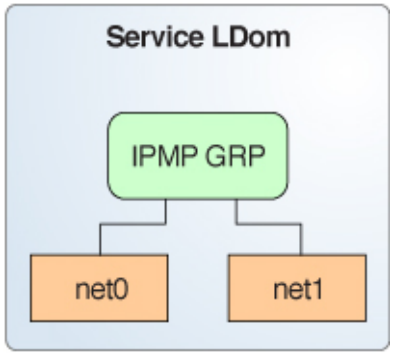
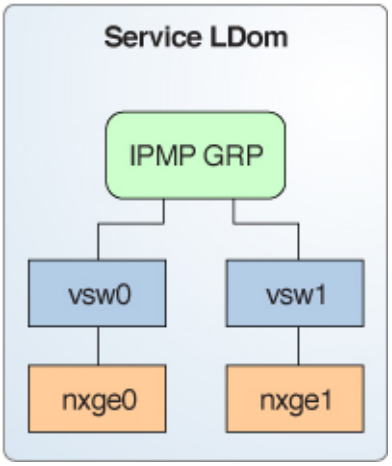


FIGURE 12–12 Two Virtual Switch Interfaces Configured as Part of an IPMP Group (Oracle Solaris 10)



Using Link-Based IPMP in Oracle VM Server for SPARC Virtual Networking

The virtual network and virtual switch devices support link status updates to the network stack. By default, a virtual network device reports the status of its virtual link (its LDC to the virtual switch). This configuration is enabled by default and does not require you to perform additional configuration steps.

Sometimes detecting physical network link state changes might be necessary. For instance, if a physical device has been assigned to a virtual switch, even if the link from a virtual network device to its virtual switch device is up, the physical network link from the service domain to the external network might be down. In such a case, you might need to obtain and report the physical link status to the virtual network device and its stack.

You can use the `linkprop=phys-state` option to configure physical link state tracking for virtual network devices as well as for virtual switch devices. When this option is enabled, the virtual device (virtual network or virtual switch) reports its link state based on the physical link state while it is created as an interface in the domain. You can use standard Oracle Solaris network administration commands such as `dladm` and `ifconfig` to check the link status. In addition, the link status is also logged in the `/var/adm/messages` file.

For Oracle Solaris 10, see the `dladm(1M)` and `ifconfig(1M)` man pages. For Oracle Solaris 11, see the `dladm(1M)`, `ipadm(1M)`, and `ipmpstat(1M)` man pages.

Note – You can run both link-state-unaware and link-state-aware `vnet` and `vsw` drivers concurrently on an Oracle VM Server for SPARC system. However, if you intend to configure link-based IPMP, you must install the link-state-aware driver. If you intend to enable physical link state updates, upgrade both the `vnet` and `vsw` drivers to the Oracle Solaris 10 1/13 OS, and run at least version 1.3 of the Logical Domains Manager.

How to Configure Physical Link Status Updates

This procedure shows how to enable physical link status updates for virtual network devices.

You can also enable physical link status updates for a virtual switch device by following similar steps and specifying the `linkprop=phys-state` option to the `ldm add-vsw` and `ldm set-vsw` commands.

Note – You need to use the `linkprop=phys-state` option only if the virtual switch device itself is created as an interface. If `linkprop=phys-state` is specified and the physical link is down, the virtual network device reports its link status as down, even if the connection to the virtual switch is up. This situation occurs because the Oracle Solaris OS does not currently provide interfaces to report two distinct link states, such as virtual-link-state and physical-link-state.

1 Become an administrator.

For Oracle Solaris 11.3, see Chapter 1, "About Using Rights to Control Users and Processes," in *Securing Users and Processes in Oracle Solaris 11.3* .

2 Enable physical link status updates for the virtual device.

You can enable physical link status updates for a virtual network device in the following ways:

- Create a virtual network device by specifying `linkprop=phys-state` when running the `ldm add-vnet` command.

  Specifying the `linkprop=phys-state` option configures the virtual network device to obtain physical link state updates and report them to the stack.

  Note – If `linkprop=phys-state` is specified and the physical link is down (even if the connection to the virtual switch is up), the virtual network device reports its link status as `down`. This situation occurs because the Oracle Solaris OS does not currently provide interfaces to report two distinct link states, such as virtual-link-state and physical-link-state.

  ```
  primary# ldm add-vnet linkprop=phys-state if-name vswitch-name domain-name
  ```

  The following example enables physical link status updates for `ldom1_vnet0` connected to `primary-vsw0` on the logical domain `ldom1`:

  ```
  primary# ldm add-vnet linkprop=phys-state ldom1_vnet0 primary-vsw0 ldom1
  ```

- Modify an existing virtual network device by specifying `linkprop=phys-state` when running the `ldm set-vnet` command.

  ```
  primary# ldm set-vnet linkprop=phys-state if-name domain-name
  ```

  The following example enables physical link status updates for `vnet0` on the logical domain `ldom1`:

  ```
  primary# ldm set-vnet linkprop=phys-state ldom1_vnet0 ldom1
  ```

To disable physical link state updates, specify `linkprop=` by running the `ldm set-vnet` command.

The following example disables physical link status updates for `ldom1_vnet0` on the logical domain `ldom1`:

```
primary# ldm set-vnet linkprop= ldom1_vnet0 ldom1
```

Example 12–8 Configuring Link-Based IPMP

The following examples show how to configure link-based IPMP both with and without enabling physical link status updates:

- The following example configures two virtual network devices on a domain. Each virtual network device is connected to a separate virtual switch device on the service domain to use link-based IPMP.

  Note – Test addresses are not configured on these virtual network devices. Also, you do not need to perform additional configuration when you use the `ldm add-vnet` command to create these virtual network devices.

  The following commands add the virtual network devices to the domain. Note that because `linkprop=phys-state` is not specified, only the link to the virtual switch is monitored for state changes.

  ```
  primary# ldm add-vnet ldom1_vnet0 primary-vsw0 ldom1
  primary# ldm add-vnet ldom1_vnet1 primary-vsw1 ldom1
  ```

  The following commands configure the virtual network devices on the guest domain and assign them to an IPMP group. Note that test addresses are not configured on these virtual network devices because link-based failure detection is being used.

  - **Oracle Solaris 10 OS:** Use the `ifconfig` command.

    ```
    # ifconfig vnet0 plumb
    # ifconfig vnet1 plumb
    # ifconfig vnet0 group ipmp0
    # ifconfig vnet1 group ipmp0
    ```

    The second and third commands configure the `ipmp0` interface with the IP address, as appropriate.

- **Oracle Solaris 11 OS:** Use the `ipadm` command.

  Note that `net0` and `net1` are the Oracle Solaris 11 vanity names for `vnet0` and `vnet1`, respectively.

  ```
  # ipadm create-ip net0
  # ipadm create-ip net1
  # ipadm create-ipmp ipmp0
  # ipadm add-ipmp -i net0 -i net1 ipmp0
  ```

- The following example configures two virtual network devices on a domain. Each domain is connected to a separate virtual switch device on the service domain to use link-based IPMP. The virtual network devices are also configured to obtain physical link state updates.

  ```
  primary# ldm add-vnet linkprop=phys-state ldom1_vnet0 primary-vsw0 ldom1
  primary# ldm add-vnet linkprop=phys-state ldom1_vnet1 primary-vsw1 ldom1
  ```

  Note – The virtual switch must have a physical network device assigned for the domain to successfully bind. If the domain is already bound and the virtual switch does not have a physical network device assigned, the `ldm add-vnet` commands will fail.

  The following commands create the virtual network devices and assign them to an IPMP group:

  - **Oracle Solaris 10 OS:** Use the `ifconfig` command.

    ```
    # 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
    ```

  - **Oracle Solaris 11 OS:** Use the `ipadm` command.

    Note that `net0` and `net1` are the vanity names for `vnet0` and `vnet1`, respectively.

    ```
    # ipadm create-ip net0
    # ipadm create-ip net1
    # ipadm create-ipmp ipmp0
    # ipadm add-ipmp -i net0 -i net1 ipmp0
    # ipadm create-addr -T static -a 192.168.1.1/24 ipmp0/v4addr1
    # ipadm create-addr -T static -a 192.168.1.2/24 ipmp0/v4addr2
    ```

Using VLAN Tagging

The Oracle VM Server for SPARC software supports 802.1Q VLAN-Tagging in the network infrastructure.

The virtual switch (`vsw`) and virtual network (`vnet`) devices support switching of Ethernet packets based on the virtual local area network (VLAN) identifier (ID) and handle the necessary tagging or untagging of Ethernet frames.

You can create multiple VLAN interfaces over a virtual network device in a guest domain. Use the Oracle Solaris 10 `ifconfig` command or the Oracle Solaris 11 `dladm` and `ipadm` commands to create a VLAN interface over a virtual network device. The creation method is the same as the method used to configure a VLAN interface over any other physical network device. The additional requirement in the Oracle VM Server for SPARC environment is that you must use the `ldm` command to assign VLANs to a `vsw` or `vnet` virtual network device. See the `ldm(1M)` man page.

Similarly, you can configure VLAN interfaces over a virtual switch device in an Oracle Solaris 10 service domain. VLAN IDs 2 through 4094 are valid. VLAN ID 1 is reserved as the `default-vlan-id`. You do not have to configure VLAN IDs on a virtual switch in an Oracle Solaris 11 service domain.

When you create a virtual network device on a guest domain, you must assign it to the required VLANs by specifying a port VLAN ID and zero or more VLAN IDs for this virtual network using the `pvid=` and `vid=` arguments to the `ldm add-vnet` command. This information configures the virtual switch to support multiple VLANs in the Oracle VM Server for SPARC network and switch packets using both MAC address and VLAN IDs in the network.

Any VLANs to be used by an Oracle Solaris 10 service domain must be configured on the `vsw` device. Use the `ldm add-vsw` or `ldm set-vsw` command to specify the `pvid` and `vid` property values.

You can change the VLANs to which a device belongs using `ldm set-vnet` or `ldm set-vsw` command.

Port VLAN ID

The Port VLAN ID (PVID) specifies the VLAN of which the virtual network device must be a member in untagged mode. In this case, the `vsw` device provides the necessary tagging or untagging of frames for the `vnet` device over the VLAN specified by its PVID. Any outbound frames from the virtual network that are untagged are tagged with its PVID by the virtual switch. Inbound frames tagged with this PVID are untagged by the virtual switch, before sending it to the `vnet` device. Thus, assigning a PVID to a virtual network implicitly means that the corresponding virtual network port on the virtual switch is marked untagged for the VLAN specified by the PVID. You can have only one PVID for a virtual network device.

The corresponding virtual network interface, when configured without a VLAN ID and using only its device instance, results in the interface being implicitly assigned to the VLAN specified by the virtual network's PVID.

For example, if you were to create virtual network instance `0` using one of the following commands and if the `pvid=` argument for the `vnet` has been specified as `10`, the `vnet0` interface would be implicitly assigned to belong to VLAN 10. Note that the following commands show the `vnet0` interface names, which pertain to Oracle Solaris 10. For Oracle Solaris 11, use the generic name instead, such as `net0`.

- **Oracle Solaris 10 OS:** Use the `ifconfig` command.

  ```
  # ifconfig vnet0 plumb
  ```

- **Oracle Solaris 11 OS:** Use the `ipadm` command.

  ```
  # ipadm create-ip net0
  ```

VLAN ID

The VID ID (VID) specifies the VLAN of which a virtual network device or virtual switch must be a member in tagged mode. The virtual network device sends and receives tagged frames over the VLANs specified by its VIDs. The virtual switch passes any frames that are tagged with the specified VID between the virtual network device and the external network.

Assigning and Using VLANs

The example devices used in the following tasks use an instance number of 0 in the domains. The VLANs are mapped to the following subnets:

- VLAN 20 Subnet `192.168.1.0` (netmask: `255.255.255.0`)
- VLAN 21 Subnet `192.168.2.0` (netmask: `255.255.255.0`)
- VLAN 22 Subnet `192.168.3.0` (netmask: `255.255.255.0`)

▼ How to Assign and Use VLANs in an Oracle Solaris 11 Service Domain

1 Assign the virtual switch (`vsw`).

```
primary# ldm add-vsw net-dev=net0 primary-vsw0 primary
```

2 Create the VLAN interface in the service domain.

Note that the `-T static` option of the `ipadm create-addr` command is required only if running an Oracle Solaris 11 OS older than the Oracle Solaris 11.1 OS. Starting with the Oracle Solaris 11 OS, `-T static` is the default behavior.

```
# ipadm create-ip net0
# ipadm create-addr -T static -a 192.168.2.100/24 net0
# dladm create-vlan -l net0 -v 20 vlan20
# ipadm create-ip vlan20
# ipadm create-addr -T static -a 192.168.2.100/24 vlan20
```

For more information about how to configure VLAN interfaces in the Oracle Solaris 11 OS, refer to Chapter 3, "Configuring Virtual Networks by Using Virtual Local Area Networks," in *Managing Network Datalinks in Oracle Solaris 11.3*.

How to Assign and Use VLANs in an Oracle Solaris 10 Service Domain

1 Assign the virtual switch (`vsw`) to two VLANs.

For example, configure VLAN 21 as untagged and VLAN 20 as tagged. Note that the service domain is not configured to access VLAN ID 22.

```
primary# ldm add-vsw net-dev=nxge0 pvid=21 vid=20 primary-vsw0 primary
```

2 Create the VLAN interface in the service domain.

```
# ifconfig vsw0 plumb
# ifconfig vsw0 192.168.2.100 netmask 0xffffff00 broadcast + up
# ifconfig vsw20000 plumb
# ifconfig vsw20000 192.168.1.100 netmask 0xffffff00 broadcast + up
```

How to Assign and Use VLANs in an Oracle Solaris 11 Guest Domain

After you complete this task, the `ldom1` guest domain can communicate with the `primary` service domain and with remote and external systems that use externally tagged VLAN ID 21 and IP addresses on 192.168.2.0/24. The `ldom1` guest domain can also communicate with the service domain and external systems that use tagged VLAN ID 20 and IP addresses on 192.168.1.0/24. The `ldom1` guest domain can communicate only with external systems but not with the service domain that uses VLAN 22 and IP addresses on 192.168.3.0/24.

1 Assign the virtual network (`vnet`) to two VLANs.

For example, configure VLAN 21 as untagged and VLAN 20 as tagged.

```
primary# ldm add-vnet pvid=21 vid=20,22 vnet0 primary-vsw0 ldom1
ldom1# ipadm create-ip net0
ldom1# ipadm create-addr -t 192.168.2.101/24 net0
```

2 Create the VLAN interface in the guest domain.

```
ldom1# dladm create-vlan -l net0 -v 20 vlan20
ldom1# ipadm create-ip vlan20
ldom1# ipadm create-addr -t 192.168.1.101/24 vlan20

ldom1# dladm create-vlan -l net0 -v 22 vlan22
ldom1# ipadm create-ip vlan22
ldom1# ipadm create-addr -t 192.168.3.101/24 vlan22
```

How to Assign and Use VLANs in an Oracle Solaris 10 Guest Domain

After you complete this task, the `ldom1` guest domain can communicate with the `primary` service domain and with remote and external systems that use externally tagged VLAN ID 21 and IP addresses on 192.168.2.0/24. The `ldom1` guest domain can also communicate with the service domain and external systems that use tagged VLAN ID 20 and IP addresses on 192.168.1.0/24. The `ldom1` guest domain can communicate only with external systems but not with the service domain that uses VLAN 22 and IP addresses on 192.168.3.0/24.

1. **Assign the virtual network (`vnet`) to two VLANs.**

   For example, configure VLAN 21 as untagged and VLAN 20 as tagged.

   ```
   primary# ldm add-vnet pvid=21 vid=20,22 vnet0 primary-vsw0 ldom1
   ldom1# ifconfig vnet0 plumb
   ldom1# ifconfig vnet0 192.168.2.101 netmask 0xffffff00 broadcast + up
   ```

2. **Create the VLAN interface in the guest domain.**

   ```
   ldom1# ifconfig vnet20000 plumb
   ldom1# ifconfig vnet20000 192.168.1.102 netmask 0xffffff00 broadcast + up
   ldom1# ifconfig vnet22000 plumb
   ldom1# ifconfig vnet22000 192.168.3.102 netmask 0xffffff00 broadcast + up
   ```

How to Install a Guest Domain When the Install Server Is in a VLAN

Be careful when using the Oracle Solaris JumpStart feature to install a guest domain over the network when the installation server is in a VLAN. This feature is supported only on Oracle Solaris 10 systems.

For more information about using the Oracle Solaris JumpStart feature to install a guest domain, see "How to Use the Oracle Solaris JumpStart Feature on an Oracle Solaris 10 Guest Domain" on page 64.

1. **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
   ```

   Do not configure any tagged VLANs (`vid`) for that virtual network device. You must do this because the OpenBoot PROM (OBP) is not aware of VLANs and cannot handle VLAN-tagged network packets.

2. **After the installation is complete and the Oracle Solaris OS boots, configure the virtual network in tagged mode.**

   ```
   primary# ldm set-vnet pvid= vid=21, 22, 23 vnet01 primary-vsw0 ldom1
   ```

   You can now add the virtual network device to additional VLANs in tagged mode.

Using Private VLANs

The private VLAN (PVLAN) mechanism enables you to divide a regular VLAN into sub-VLANs to isolate network traffic. The PVLAN mechanism is defined in RFC 5517 (`http://tools.ietf.org/html/rfc5517`). Usually, a regular VLAN is a single broadcast domain, but when configured with PVLAN properties, the single broadcast domain is partitioned into smaller broadcast subdomains while keeping the existing Layer 3 configuration. When you configure a PVLAN, the regular VLAN is called the *primary VLAN* and the sub-VLANs are called *secondary VLANs*.

When two virtual networks use the same VLAN ID on a physical link, all broadcast traffic is passed between the two virtual networks. However, when you create virtual networks that use PVLAN properties, the packet-forwarding behavior might not apply to all situations.

The following table shows the broadcast packet-forwarding rules for isolated and community PVLANs.

TABLE 12–1 Broadcast Packet-Forwarding Rules

PVLAN Type	Isolated	Community A	Community B
Isolated	No	No	No
Community A	No	Yes	No
Community B	No	No	Yes

For example, when both the `vnet0` and `vnet1` virtual networks are isolated on the `net0` network, `net0` does not pass broadcast traffic between the two virtual networks. However, when the `net0` network receives traffic from an isolated VLAN, the traffic is not passed to the isolated ports that are related to the VLAN. This situation occurs because the isolated virtual network accepts only traffic from the primary VLAN.

The inter-vnet-links feature supports the communication restrictions of isolated and community PVLANs. Inter-vnet-links are disabled for isolated PVLANs and are enabled only for virtual networks that are in the same community for community PVLANs. Direct traffic from other virtual networks outside of the community is not permitted.

Note – If a target service domain does not support the PVLAN feature, the migration of a guest domain that is configured for PVLAN might fail.

PVLAN Requirements

You can configure PVLANs by using the `ldm add-vnet` and `ldm set-vnet` commands. Use these commands to set the `pvlan` property. Note that you must also specify the `pvid` property to successfully configure the PVLAN.

This feature requires at least the Oracle Solaris 11.2 SRU 4 OS.

To configure a PVLAN, you must specify the following information:

- **Primary VLAN ID.** The primary VLAN ID is the port VLAN ID (PVID) that is used to configure a PVLAN for a single virtual network device. This configuration ensures that a guest domain does receive VLAN packets. Note that you cannot configure VIDs with a PVLAN. This value is represented by the `pvid` property.
- **Secondary VLAN ID.** A secondary VLAN ID is used by a particular VLAN to provide PVLAN functionality. You specify this information as the *secondary-vid* part of the `pvlan` value. *secondary-vid* is an integer value in the range of 1-4094. A primary VLAN can have many secondary VLANs with the following restrictions:
  - Neither the primary VLAN ID nor the secondary VLAN ID can be the same as the default VLAN ID.
  - The primary VLAN ID and the secondary VLAN ID cannot have the same values for both isolated and community PVLAN types.
  - Each primary VLAN can configure only one isolated PVLAN. So, you cannot create two isolated PVLANs that use the same primary VLAN ID.
  - A primary VLAN can have multiple community VLANs with the following restrictions:
    - A primary VLAN ID cannot be used as secondary VLAN ID create another community PVLAN.

      For example, if you have a community PVLAN with a primary VLAN ID of `3` and a secondary VLAN ID of `100`, you cannot create another community PVLAN that uses `3` as the secondary VLAN ID.
    - A secondary VLAN ID cannot be used as primary VLAN ID to create a community PVLAN.

      For example, if you have a community PVLAN with a primary VLAN ID of `3` and a secondary VLAN ID of `100`, you cannot create another community PVLAN that uses `100` as the primary VLAN ID.
    - The secondary VLAN ID cannot already be used as a VLAN ID for regular virtual networks or VNICs.

Caution – The Logical Domains Manager can validate only the configuration of the virtual networks on a particular virtual switch. If a PVLAN configuration is set up for Oracle Solaris VNICs on the same back-end device, ensure that the same requirements are met across all VNICs and virtual networks.

- **PVLAN type.** You specify this information as the *pvlan-type* part of the `pvlan` value. *pvlan-type* is one of the following values:
  - `isolated`**.** The ports that are associated with an isolated PVLAN are isolated from all of the peer virtual networks and Oracle Solaris virtual NICs on the back-end network device. The packets reach only the external network based on the values you specified for the PVLAN.
  - `community`**.** The ports that are associated with a community PVLAN can communicate with other ports that are in the same community PVLAN but are isolated from all other ports. The packets reach the external network based on the values you specified for the PVLAN.

Configuring PVLANs

This section includes tasks that describes how to create PVLANs and list information about PVLANs.

Creating a PVLAN

You can configure a PVLAN by setting the `pvlan` property value by using the `ldm add-vnet` or `ldm set-vnet` command. See the `ldm(1M)` man page.

You can use the following commands to create or remove a PVLAN:

- Use `ldm add-vnet` to create a PVLAN:

  ```
  ldm add-vnet pvid=port-VLAN-ID pvlan=secondary-vid,pvlan-type \
  if-name vswitch-name domain-name
  ```

  The following command shows how to create a virtual network with a PVLAN that has a primary *vlan-id* of `4`, a secondary *vlan-id* of `200`, and a *pvlan-type* of `isolated`.

  ```
  primary# ldm add-vnet pvid=4 pvlan=200,isolated vnet1 primary-vsw0 ldg1
  ```

- Use `ldm set-vnet` to create a PVLAN:

  ```
  ldm set-vnet pvid=port-VLAN-ID pvlan=secondary-vid,pvlan-type if-name domain-name
  ```

  The following command shows how to create a virtual network with a PVLAN that has a primary *vlan-id* of `3`, a secondary *vlan-id* of `300`, and a *pvlan-type* of `community`.

  ```
  primary# ldm add-vnet pvid=3 pvlan=300,community vnet1 primary-vsw0 ldg1
  ```

- Use `ldm set-vnet` to remove a PVLAN:

  `ldm set-vnet pvlan=` *if-name vswitch-name domain-name*

  The following command removes the PVLAN configuration for the `vnet0` virtual network. The result of this command is that the specified virtual network is a regular VLAN that uses the *vlan-id* that you specified when you configured the PVLAN.

  ```
  primary# ldm set-vnet pvlan= vnet0 primary-vsw0 ldg1
  ```

Viewing PVLAN Information

You can view information about a PVLAN by using several of the Logical Domains Manager listing subcommands. See the `ldm(1M)` man page.

You can use the following commands to view PVLAN information.

- Use `ldm list-domain -o network` to list PVLAN information:

  `ldm list-domain [-e] [-l] -o network [-p] [`*domain-name*`...]`

  The following examples show information about PVLAN configuration on the `ldg1` domain by using the `ldm list-domain -o network` command.

  - The following `ldm list-domain` command shows information about the PVLAN configuration on the `ldg1` domain.

    ```
    primary# ldm list-domain -o network ldg1
    NAME
    ldg1

    MAC
        00:14:4f:fa:bf:0f

    NETWORK
    NAME   SERVICE              ID DEVICE    MAC
    vnet0  primary-vsw0@primary 0  network@0 00:14:4f:f8:03:ed
    MODE   PVID VID             MTU  MAXBW LINKPROP
           1    3               1500 1700
           PVLAN : 200,community
    ```

  - The following `ldm list-domain` command shows PVLAN configuration information in a parseable form for the `ldg1` domain.

    ```
    primary# ldm list-domain -o network -p ldg1
    VERSION 1.13
    DOMAIN|name=ldg1|
    MAC|mac-addr=00:14:4f:fa:bf:0f
    VNET|name=vnet0|dev=network@0|service=primary-vsw0@primary
    |mac-addr=00:14:4f:f8:03:ed|mode=|pvid=1|vid=3|mtu=1500|linkprop=|id=0
    |alt-mac-addrs=|maxbw=1700|protect=|priority=|cos=|pvlan=200,community
    ```

- Use `ldm list-bindings` to list PVLAN information:

  `ldm list-bindings [-e] [-p] [`*domain-name*`...]`

The following examples show information about PVLAN configuration on the `ldg1` domain by using the `ldm list-bindingsnetwork` command.

- The following `ldm list-bindings` command shows information about the PVLAN configuration on the `ldg1` domain.

```
primary# ldm list-bindings
...
NETWORK
NAME    SERVICE              ID DEVICE    MAC
vnet0   primary-vsw0@primary 0  network@0 00:14:4f:f8:03:ed
MODE   PVID VID   MTU  MAXBW LINKPROP
       1    3     1500 1700
       PVLAN :200,community
PEER                MAC               MODE  PVID VID  MTU  MAXBW LINKPROP
primary-vsw0@primary 00:14:4f:f8:fe:5e 1
```

- The following `ldm list-bindings` command shows PVLAN configuration information in a parseable form for the `ldg1` domain.

```
primary# ldm list-bindings -p
...
VNET|name=vnet0|dev=network@0|service=primary-vsw0@primary
|mac-addr=00:14:4f:f8:03:ed|mode=|pvid=1|vid=3|mtu=1500|linkprop=
|id=0|alt-mac-addrs=|maxbw=1700|protect=|priority=|cos=|pvlan=200,community
|peer=primary-vsw0@primary|mac-addr=00:14:4f:f8:fe:5e|mode=|pvid=1|vid=
|mtu=1500|maxbw=
```

- Use `ldm list-constraints` to list PVLAN information:

```
ldm list-constraints [-x] [domain-name...]
```

The following shows the output generated by running the `ldm list-constraints` command:

```
primary# ldm list-constraints -x ldg1
...
<Section xsi:type="ovf:VirtualHardwareSection_Type">
  <Item>
    <rasd:OtherResourceType>network</rasd:OtherResourceType>
    <rasd:Address>auto-allocated</rasd:Address>
    <gprop:GenericProperty key="vnet_name">vnet0</gprop:GenericProperty>
    <gprop:GenericProperty key="service_name">primary-vsw0</gprop:GenericProperty>
    <gprop:GenericProperty key="pvid">1</gprop:GenericProperty>
    <gprop:GenericProperty key="vid">3</gprop:GenericProperty>
    <gprop:GenericProperty key="pvlan">200,community</gprop:GenericProperty>
    <gprop:GenericProperty key="maxbw">1700000000</gprop:GenericProperty>
    <gprop:GenericProperty key="device">network@0</gprop:GenericProperty>
    <gprop:GenericProperty key="id">0</gprop:GenericProperty>
  </Item>
```

Tuning Packet Throughput Performance

You can use the `ldm add-vnet` and `ldm set-vnet` commands to set the following data link property values to tune packet throughput performance:

`priority`	Specifies the CPU packet-processing priority
`cos`	Specifies the IEEE 802.1p link service class of the link
`protection`	Specifies packet traffic security type

For information about valid and default property values, see the `ldm(1M)` man page.

EXAMPLE 12–9 Setting and Viewing Data Link Packet Properties

The following example shows how to use the `ldm set-vnet` command to set the `priority`, `protection`, and `cos` property values in a single command. You can also use the `ldm add-vnet` command to add a new virtual network that uses the specified data link property values.

```
primary# ldm set-vnet allowed-ips=192.168.100.1,192.168.100.2 \
allowed-dhcpcids=oracle1@system1.company.com, \
00:14:4f:f9:d3:88,system2,00:14:4f:fb:61:6e cos=7 priority=high \
protection=restricted,mac-nospoof,ip-nospoof,dhcp-nospoof vnet3_ldg3 ldg3
```

The `ldm list -o network` command shows the data link property values on the `ldg3` domain that you set with the previous `ldm set-vnet` command. The protection values are `mac-nospoof`, `restricted`, `ip-nospoof` for the `192.168.100.1,192.168.100.2` MAC address, and `dhcp-nospoof` for `system1@company.com,00:14:4f:f9:d3:88,system2,00:14:4f:fb:61:6e`. The `priority` is set to `high` and the class of service (`cos`) is set to 7.

```
primary# ldm list -o network ldg3
NAME
ldg3

MAC
00:14:4f:fa:3b:49

VSW
NAME      MAC               NET-DEV ID DEVICE   LINKPROP DEFAULT-VLAN-ID
PVID VID MTU  MODE INTER-VNET-LINK
ldg3_vsw1 00:14:4f:f9:65:b5 net3    0  switch@0 1        1
         1500      on

NETWORK
  NAME       SERVICE              ID DEVICE    MAC               MODE PVID VID
  MTU  MAXBW LINKPROP
  vnet3      primary-vsw0@primary 0  network@0 00:14:4f:fa:47:f4      1
  1500
  vnet1_ldg3 primary-vsw1@primary 1  network@1 00:14:4f:fb:90:b7      1
  1500
  vnet2_ldg3 ldg1_vsw1@ldg1       2  network@2 00:14:4f:fb:61:6e      1
```

EXAMPLE 12–9 Setting and Viewing Data Link Packet Properties *(Continued)*

```
1500
vnet3_ldg3 ldg1_vsw1@ldg1       3  network@3 00:14:4f:fb:25:70      1
                                              00:14:4f:f8:7a:8d
                                              00:14:4f:fb:6e:32
                                              00:14:4f:fb:1a:2f
1500 1000
     PROTECTION: mac-nospoof
                 restricted
                 ip-nospoof/192.168.100,1,192.168.100.2
                 dhcp-nospoof/system1@oracle.us.oracle.com,
                 00:14:4f:f9:d3:88,system2,00:14:4f:fb:61:6e
     PRIORITY  : high
     COS       : 7
```

Using NIU Hybrid I/O

The virtual I/O framework implements a *hybrid* I/O model for improved functionality and performance. The hybrid I/O model combines direct and virtualized I/O to enable flexible deployment of I/O resources to virtual machines. It is particularly useful when direct I/O does not provide full capability for the virtual machine, or direct I/O is not persistently or consistently available to the virtual machine due to resource availability or virtual machine migration.

The hybrid I/O architecture is well-suited for the Network Interface Unit (NIU) on Oracle Sun UltraSPARC T2, SPARC T3, and SPARC T4 platforms. An NIU is a network I/O interface that is integrated on the chip. This architecture enables the dynamic assignment of Direct Memory Access (DMA) resources to virtual networking devices and, thereby, provides consistent performance to applications in the domain.

Note – The NIU Hybrid I/O feature is deprecated in favor of SR-IOV. Oracle VM Server for SPARC 3.3 is the last software release to include this feature.

NIU hybrid I/O is available for Oracle Sun UltraSPARC T2, SPARC T3, and SPARC T4 platforms. This feature is enabled by an optional hybrid mode that provides for a virtual network (`vnet`) device where the DMA hardware resources are loaned to a `vnet` device in a guest domain for improved performance. In the hybrid mode, a vnet device in a guest domain can send and receive unicast traffic from an external network directly into the guest domain using the DMA hardware resources. The broadcast or multicast traffic and unicast traffic to the other guest domains in the same system continue to be sent using the virtual I/O communication mechanism.

Note – NIU hybrid I/O is not available on UltraSPARC T2 Plus platforms.

Figure 12–13 and Figure 12–14 show hybrid I/O configurations for Oracle Solaris 11 and Oracle Solaris 10, respectively.

FIGURE 12–13 Hybrid Virtual Networking (Oracle Solaris 11)

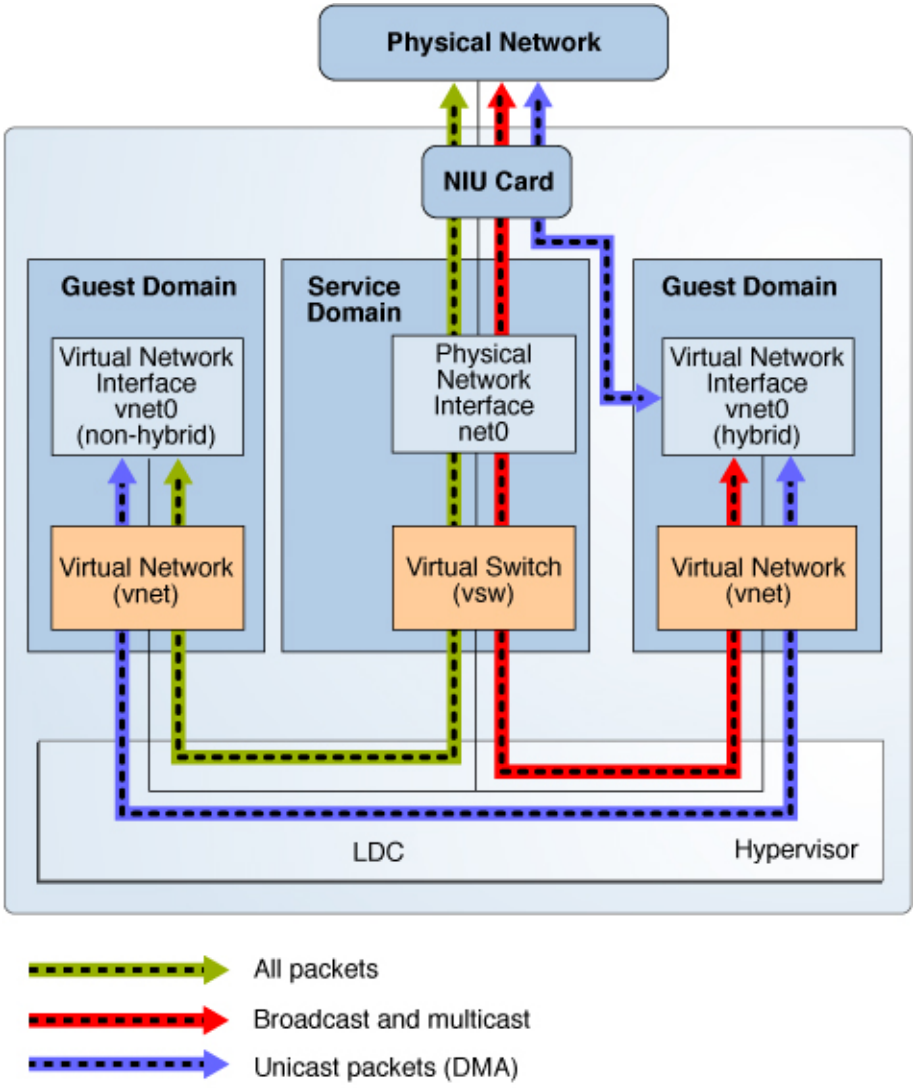
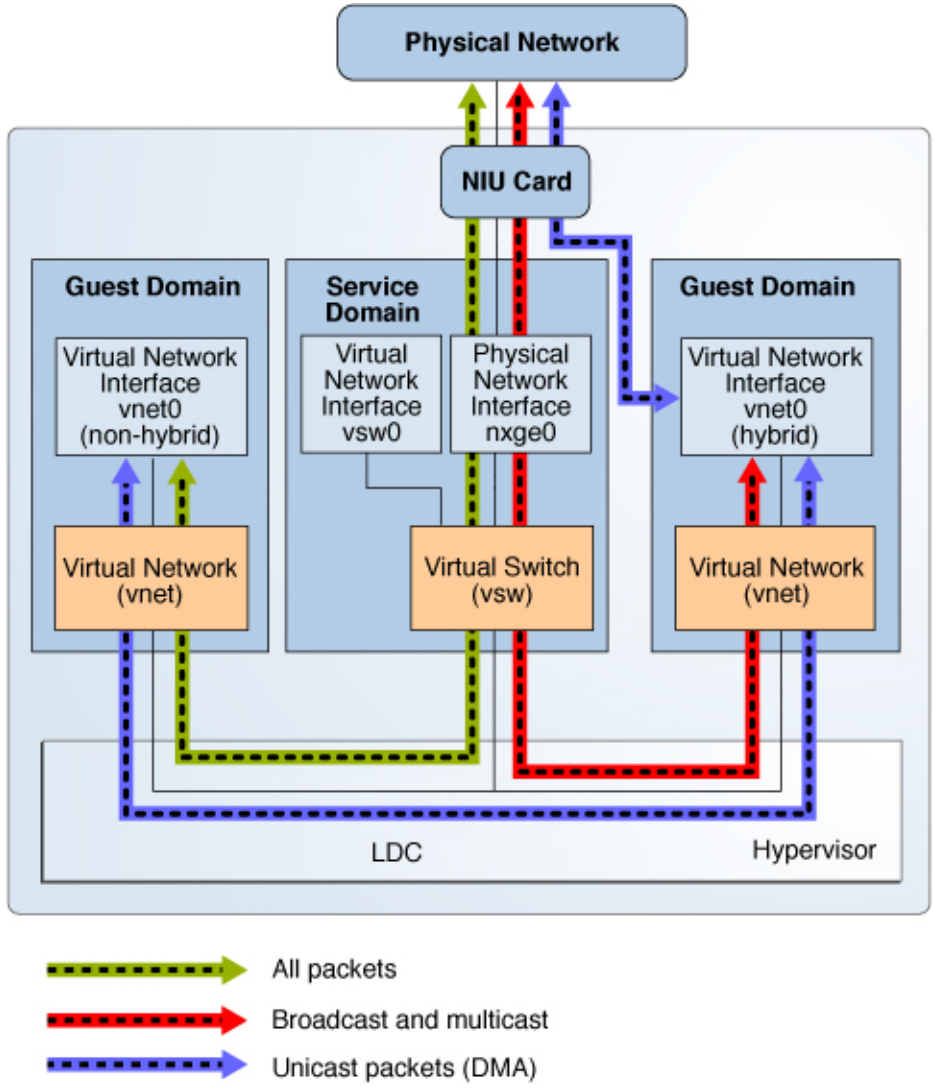


FIGURE 12–14 Hybrid Virtual Networking (Oracle Solaris 10)



The hybrid mode applies only for the `vnet` devices that are associated with a virtual switch (`vsw`) configured to use an NIU network device. Because the shareable DMA hardware resources are limited, up to only three `vnet` devices per `vsw` can have DMA hardware resources assigned at a given time. If more than three `vnet` devices have the hybrid mode enabled, the assignment is done on a first-come, first-served basis. Because there are two NIU network devices in a system, there can be a total of six `vnet` devices on two different virtual switches with DMA hardware resources assigned.

Note the following points when using this feature:

- Hybrid mode option for a `vnet` device is treated as a suggestion only, so DMA resources are assigned only when they are available and the device is capable of using them.
- Logical Domains Manager CLI commands do not validate the hybrid mode option; that is, you can set the hybrid mode on any `vnet` or any number of `vnet` devices.
- Guest domains and the service domain need to run Oracle Solaris 10 10/08 OS at a minimum.
- Up to a maximum of only three `vnet` devices per `vsw` can have DMA hardware resources loaned at a given time. Because there are two NIU network devices, there can be a total of six `vnet` devices with DMA hardware resources loaned.

  Note – Set the hybrid mode only for three `vnet` devices per `vsw` so that they are guaranteed to have DMA hardware resources assigned.

- The hybrid mode option cannot be changed dynamically while the guest domain is active.
- The DMA hardware resources are assigned only when an active `vnet` device is created in the guest domain.
- The NIU 10-gigabit Ethernet driver (`nxge`) is used for the NIU card. The same driver is also used for other 10-gigabit network cards. However, the NIU hybrid I/O feature is available for NIU network devices only.

▼ How to Configure a Virtual Switch With an NIU Network Device

1 Determine an NIU network device.

The following example shows the output on an UltraSPARC T2 server.

```
primary# grep nxge /etc/path_to_inst
"/niu@80/network@0" 0 "nxge"
"/niu@80/network@1" 1 "nxge"
```

The following example shows the output on a SPARC T3-1 or SPARC T4-1 server.

```
primary# grep nxge /etc/path_to_inst
"/niu@480/network@0" 0 "nxge"
"/niu@480/network@1" 1 "nxge"
```

2. **Oracle Solaris 11 OS only: Identify the link name that corresponds to the NIU network device, such as `nxge0`.**

```
primary# dladm show-phys -L |grep nxge0
net2              nxge0         /SYS/MB
```

3. **Configure a virtual switch.**

   The following example uses `net2` instead of `nxge0`.

```
primary# ldm add-vsw net-dev=net2 primary-vsw0 primary
```

▼ How to Enable or Disable Hybrid Mode

Hybrid mode is disabled by default for a `vnet` device and must be explicitly enabled.

- **Use the `ldm` command to enable and disable hybrid mode.**
  - **To enable a hybrid mode for a `vnet` device while it is being created:**

```
primary# ldm add-vnet mode=hybrid vnet01 primary-vsw0 ldom01
```

  - **To disable hybrid mode for a `vnet` device:**

```
primary# ldm set-vnet mode= vnet01 ldom01
```

Using Link Aggregation With a Virtual Switch

A virtual switch can be configured to use a link aggregation. A link aggregation is used as the virtual switch's network device to connect to the physical network. This configuration enables the virtual switch to leverage the features provided by the IEEE 802.3ad Link Aggregation Standard. Such features include increased bandwidth, load balancing, and failover. For information about how to configure link aggregation, see "Creating a Link Aggregation" in *Managing Network Datalinks in Oracle Solaris 11.3*.

After you create a link aggregation, you can assign it to the virtual switch. Making this assignment is similar to assigning a physical network device to a virtual switch. Use the `ldm add-vswitch` or `ldm set-vswitch` command to set the `net-dev` property.

When the link aggregation is assigned to the virtual switch, traffic to and from the physical network flows through the aggregation. Any necessary load balancing or failover is handled transparently by the underlying aggregation framework. Link aggregation is completely transparent to the virtual network (`vnet`) devices that are on the guest domains and that are bound to a virtual switch that uses an aggregation.

Note – You cannot group the virtual network devices (`vnet` and `vsw`) into a link aggregation.

You can create and use the virtual switch that is configured to use a link aggregation in the service domain. See "How to Configure the Virtual Switch as the Primary Interface" on page 52.

Figure 12–15 and Figure 12–16 show a virtual switch configured to use an aggregation, `aggr1`, over physical interfaces `net0` and `net1`, and `nxge0` and `nxge1`, respectively.

FIGURE 12–15 Configuring a Virtual Switch to Use a Link Aggregation (Oracle Solaris 11)

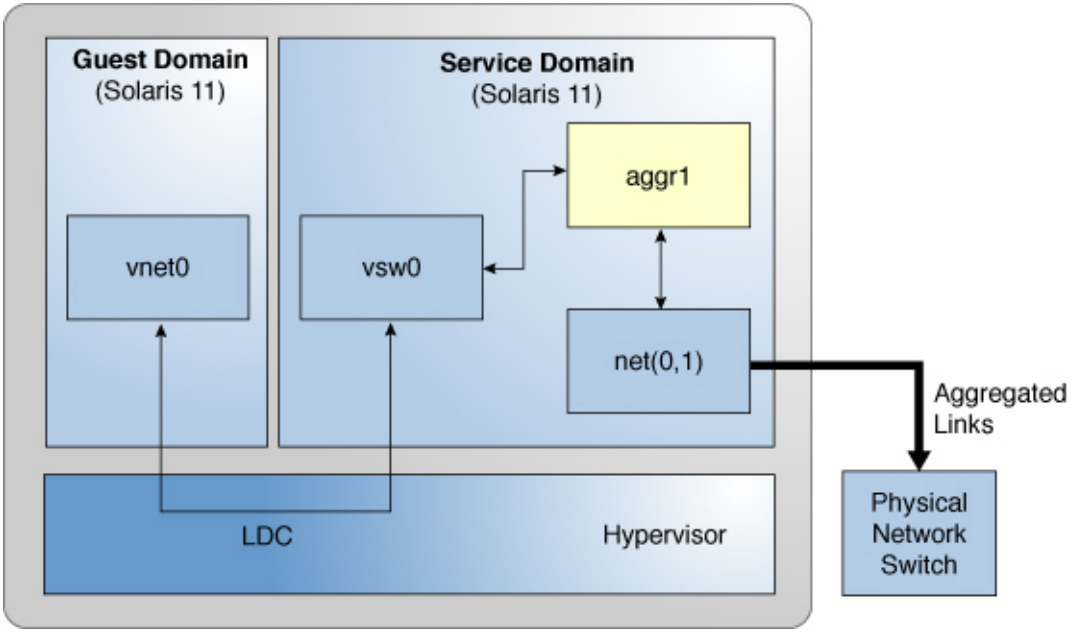
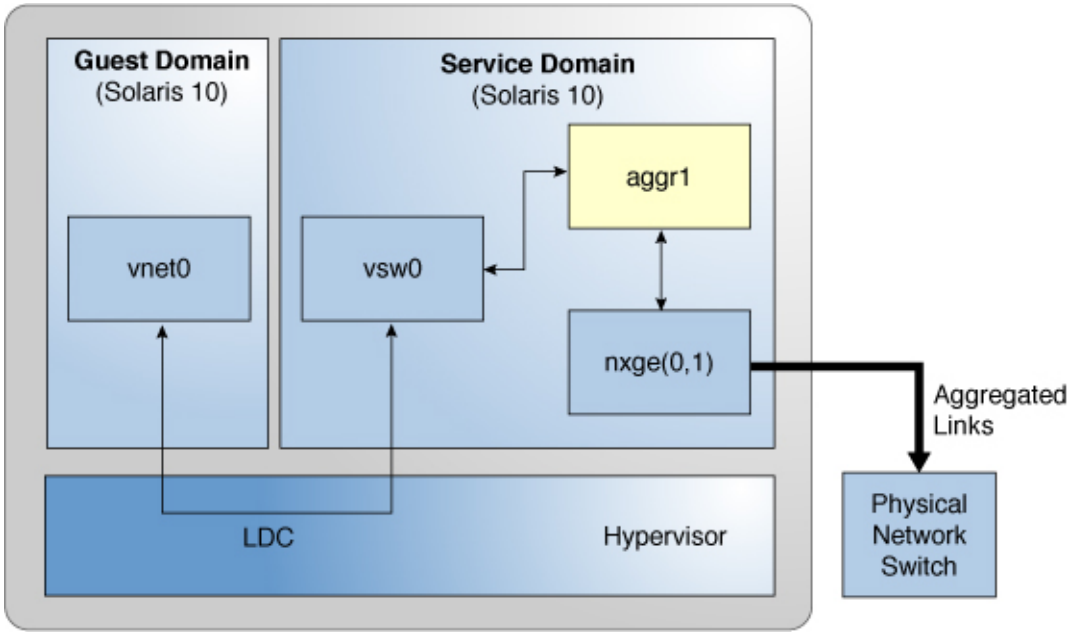


FIGURE 12–16 Configuring a Virtual Switch to Use a Link Aggregation (Oracle Solaris 10)



Configuring Jumbo Frames

The Oracle VM Server for SPARC virtual switch (`vsw`) and virtual network (`vnet`) devices can now support Ethernet frames with payload sizes larger than 1500 bytes. These drivers are therefore now able to increase network throughput.

You enable jumbo frames by specifying the maximum transmission unit (MTU) for the virtual switch device. In such cases, the virtual switch device and all virtual network devices that are bound to the virtual switch device use the specified MTU value.

If the required MTU value for the virtual network device should be less than that supported by the virtual switch, you can specify an MTU value directly on a virtual network device.

Note – Only on the Oracle Solaris 10 5/09 OS, the MTU of a physical device must be configured to match the MTU of the virtual switch. For information about configuring particular drivers, see the man page that corresponds to that driver in Section 7D of the Oracle Solaris reference manual. For example, to obtain information about the Oracle Solaris 10 `nxge` driver, see the `nxge(7D)` man page.

In rare circumstances, you might need to use the `ldm add-vnet` or `ldm set-vnet` command to specify an MTU value for a virtual network device that differs from the MTU value of the virtual

switch. For example, you might change the virtual network device's MTU value if you configure VLANs over a virtual network device and the largest VLAN MTU is less than the MTU value on the virtual switch. A `vnet` driver that supports jumbo frames might not be required for domains where only the default MTU value is used. However, if the domains have virtual network devices bound to a virtual switch that uses jumbo frames, ensure that the `vnet` driver supports jumbo frames.

If you use the `ldm set-vnet` command to specify an `mtu` value on a virtual network device, future updates to the MTU value of the virtual switch device are not propagated to that virtual network device. To re-enable the virtual network device to obtain the MTU value from the virtual switch device, run the following command:

```
primary# ldm set-vnet mtu= vnet-name domain-name
```

Note that enabling jumbo frames for a virtual network device automatically enables jumbo frames for any hybrid I/O resource that is assigned to that virtual network device.

On the control domain, the Logical Domains Manager updates the MTU values that are initiated by the `ldm set-vsw` and `ldm set-vnet` commands as delayed reconfiguration operations. To make MTU updates to domains other than the control domain, you must stop a domain prior to running the `ldm set-vsw` or `ldm set-vnet` command to modify the MTU value.

How to Configure Virtual Network and Virtual Switch Devices to Use Jumbo Frames

1. **Log in to the control domain.**

2. **Become an administrator.**

   For Oracle Solaris 11.3, see Chapter 1, "About Using Rights to Control Users and Processes," in *Securing Users and Processes in Oracle Solaris 11.3* .

3. **Determine the value of MTU that you want to use for the virtual network.**

   You can specify an MTU value from 1500 to 16000 bytes. The specified MTU must match the MTU of the physical network device that is assigned to the virtual switch.

4. **Specify the MTU value of a virtual switch device or virtual network device.**

   Do one of the following:

   - Enable jumbo frames on a new virtual switch device in the service domain by specifying its MTU as a value of the `mtu` property.

     ```
     primary# ldm add-vsw net-dev=device mtu=value vswitch-name ldom
     ```

In addition to configuring the virtual switch, this command updates the MTU value of each virtual network device that will be bound to this virtual switch.

- Enable jumbo frames on an existing virtual switch device in the service domain by specifying its MTU as a value of the `mtu` property.

  ```
  primary# ldm set-vsw net-dev=device mtu=value vswitch-name
  ```

  In addition to configuring the virtual switch, this command updates the MTU value of each virtual network device that will be bound to this virtual switch.

Example 12–10 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, `ldg1-vsw0`, with an MTU value of `9000`. Note that instance 0 of the network device `net0` is specified as a value of the `net-dev` property.

  ```
  primary# ldm add-vsw net-dev=net0 mtu=9000 ldg1-vsw0 ldg1
  ```

  Next, the `ldm add-vnet` command adds a client virtual network device to this virtual switch, `ldg1-vsw0`. Note that the MTU of the virtual network device is implicitly assigned from the virtual switch to which it is bound. As a result, the `ldm add-vnet` command does not require that you specify a value for the `mtu` property.

  ```
  primary# ldm add-vnet vnet01 ldg1-vsw0 ldg1
  ```

  Depending on the version of the Oracle Solaris OS that is running, do the following:

  - **Oracle Solaris 11 OS:** Use the `ipadm` command to view the `mtu` property value of the primary interface.

    ```
    # ipadm show-ifprop -p mtu net0
    IFNAME PROPERTY PROTO PERM CURRENT PERSISTENT DEFAULT POSSIBLE
    net0   mtu      ipv4  rw   9000    --           9000    68-9000
    ```

    The `ipadm` command creates the virtual network interface in the guest domain, `ldg1`. The `ipadm show-ifprop` command output shows that the value of the `mtu` property is `9000`.

    ```
    ldg1# ipadm create-ip net0
    ldg1# ipadm create-addr -T static -a 192.168.1.101/24 net0/ipv4
    ldg1# ipadm show-ifprop -p mtu net0
    IFNAME PROPERTY PROTO PERM CURRENT PERSISTENT DEFAULT POSSIBLE
    net0   mtu      ipv4  rw   9000    --         9000    68-9000
    ```

  - **Oracle Solaris 10 OS:** The `ifconfig` command creates the virtual switch interface in the service domain, `ldg1`. The `ifconfig vsw0` command output shows that the value of the `mtu` property is `9000`.

```
ldg1# ifconfig vsw0 plumb
ldg1# ifconfig vsw0 192.168.1.100/24 up
ldg1# ifconfig vsw0
vsw0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 9000 index 5
        inet 192.168.1.100 netmask ffffff00 broadcast 192.168.1.255
        ether 0:14:4f:fa:0:99
```

The `ifconfig` command creates the virtual network interface in the guest domain, `ldg1`. The `ifconfig vnet0` command output shows that the value of the `mtu` property is `9000`.

```
ldg1# ifconfig vnet0 plumb
ldg1# ifconfig vnet0 192.168.1.101/24 up
ldg1# 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 change the MTU of the interface to `4000`.

  Note that the MTU of an interface can only be changed to a value that is less than the MTU of the device that is assigned by the Logical Domains Manager. This method is useful when VLANs are configured and each VLAN interface requires a different MTU.

  - **Oracle Solaris 11 OS:** Use the `ipadm` command.

    ```
    primary# ipadm set-ifprop -p mtu=4000 net0
    primary# ipadm show-ifprop -p mtu net0
    IFNAME PROPERTY PROTO PERM CURRENT PERSISTENT DEFAULT POSSIBLE
    net0   mtu      ipv4  rw   4000    --         9000    68-9000
    ```

- **Oracle Solaris 10 OS:** Use the `ifconfig` command.

  ```
  primary# ifconfig vnet0 mtu 4000
  primary# ifconfig vnet0
  vnet0: flags=1201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS,FIXEDMTU>
  mtu 4000 index 4
          inet 192.168.1.101 netmask ffffff00 broadcast 192.168.1.255
          ether 0:14:4f:f9:c4:13
  ```

Compatibility With Older (Jumbo-Unaware) Versions of the `vnet` and `vsw` Drivers (Oracle Solaris 10)

Note – This section applies only to the Oracle Solaris 10 OS.

Drivers that support jumbo frames can interoperate with drivers that do not support jumbo frames on the same system. This interoperability is possible as long as jumbo frame support is not enabled when you create the virtual switch.

Note – Do not set the `mtu` property if any guest or service domains that are associated with the virtual switch do not use Oracle VM Server for SPARC drivers that support jumbo frames.

Jumbo frames can be enabled by changing the `mtu` property of a virtual switch from the default value of 1500. In this instance, older driver versions ignore the `mtu` setting and continue to use the default value. Note that the `ldm list` output will show the MTU value you specified and not the default value. Any frames larger than the default MTU are not sent to those devices and are dropped by the new drivers. This situation might result in inconsistent network behavior with those guests that still use the older drivers. This limitation applies to both client guest domains and the service domain.

Therefore, while jumbo frames are enabled, ensure that all virtual devices in the Oracle VM Server for SPARC network are upgraded to use the new drivers that support jumbo frames. You must be running at least Logical Domains 1.2 to configure jumbo frames.

Oracle Solaris 11 Networking-Specific Feature Differences

Some of the Oracle VM Server for SPARC networking features work differently when a domain runs the Oracle Solaris 10 OS as compared to the Oracle Solaris 11 OS. The feature differences for the Oracle VM Server for SPARC virtual network device and virtual switch when the Oracle Solaris 11 OS is run in a domain are as follows:

- **Configuring the `vsw`*n* device as the primary network interface to enable a service domain to communicate with guest domains**

  This configuration is required only for domains that run the Oracle Solaris 10 OS. For Oracle Solaris 11, a virtual switch uses the Oracle Solaris 11 network stack, automatically enabling its virtual network devices to communicate with the network interface that corresponds to its back-end device, such as `net0`. Configuring the `vsw`*n* device as a network interface on Oracle Solaris 11 is not supported.

- **Using an Oracle Solaris 11 `etherstub` device as a back-end device to create a private virtual switch**

  If not connected to a back-end device, a virtual switch provides communication only between guest domains and not between guest domains and the service domain. Using an `etherstub` as a back-end device enables a guest domain to communicate with a zone (including the global zone) that is configured in an Oracle Solaris 11 service domain. This configuration is accomplished by using a VNIC connected to that `etherstub`.

- **Using generic names for the virtual switch and virtual network devices**

The Oracle Solaris 11 OS assigns generic names for `vsw`*n* and `vnet`*n* devices. Ensure that you do not create a virtual switch with the back-end device that is another `vsw` or `vnet` device. Use the `dladm show-phys` command to see the actual physical devices that are associated with generic network device names.

- **Using an Oracle Solaris 11 VNIC to create a VLAN on an Ethernet stub**

  Do not configure VLANs on the virtual switch interface for Oracle Solaris 11 service domains because this configuration is not supported. Instead, create the VLAN on the interface that corresponds to the virtual switch's `net-dev` property value.

  For Oracle Solaris 10, you can set the `net-dev` property with a null value to create a routed virtual switch. However, this method is not supported for Oracle Solaris 11. Instead, configure the VNIC over the Ethernet stub device to be part of the VLAN.

  The following example shows how to create VNICs on an Ethernet stub. The `dladm create-etherstub` command creates an Ethernet stub, `estub100`, which is a backing device used by the `ldm add-vsw` command to create the virtual switch. The `ldm add-vsw` command creates the virtual switch. The `dladm create-vnic` command creates a VNIC on top of the etherstub to create the VLAN for that virtual switch.

  ```
  primary# dladm create-etherstub estub100
  primary# ldm add-vsw net-dev=estub100 vid=100 inter-vnet-link=off \
  primary-vsw100 primary
  primary# dladm create-vnic -l estub100 -m auto -v 100 vnic100
  ```

  The following `ldm add-vnet` commands create two VNICs that enable communication between the `ldg1` and `ldg2` domains over VLAN 100.

  ```
  primary# ldm add-vnet vid=100 ldg1-vnet100 primary-vsw100 ldg1
  primary# ldm add-vnet vid=100 ldg2-vnet100 primary-vsw100 ldg2
  ```

  In the following example, the `dladm` commands create VLANs on the `ldg1` and `ldg2` guest domains. The `ipadm` commands create IP addresses for the VNICs that you created on the `ldg1` and `ldg2` domains.

  ```
  ldg1# dladm create-vlan -l net1 -v 100 vlan100
  ldg1# ipadm create-ip vlan100
  ldg1# ipadm create-ipaddr -T static -a 192.168.100.10/24 vlan100/v4
  ldg2# dladm create-vlan -l net1 -v 100 vlan100
  ldg2# ipadm create-ip vlan100
  ldg2# ipadm create-ipaddr -T static -a 192.168.100.20/24 vlan100/v4
  ```

- **Using generic names for the virtual switch and virtual network devices**

  The Oracle Solaris 11 OS assigns generic names for `vsw`*n* and `vnet`*n* devices. Ensure that you do not create a virtual switch with the back-end device that is another `vsw` or `vnet` device. Use the `dladm show-phys` command to see the actual physical devices that are associated with generic network device names.

- **Using VNICs on the virtual switch and virtual network devices**

You *cannot* use VNICs on `vsw`*n* devices. An attempt to create a VNIC on `vsw`*n* fails. See "Oracle Solaris 11: Zones Configured With an Automatic Network Interface Might Fail to Start" in *Oracle VM Server for SPARC 3.3 Release Notes*.

- **Using the network observability commands on Oracle Solaris 11 guest domains**

  You can use the `ldm list-netdev` and `ldm list-netstat` commands to obtain information about Oracle Solaris 11 guest domains.

Using Virtual NICs on Virtual Networks

The Oracle Solaris 11 OS enables you to define virtual networks that consist of virtual network interface cards (VNICs), virtual switches, and etherstubs. Oracle Solaris Zones virtualize operating system services and provide isolated and secure environments for running applications within the same Oracle Solaris OS instance of a logical domain.

Oracle Solaris 11 improves on the Oracle Solaris 10 "shared IP" zone model in which zones inherit network properties from the global zone and cannot set their own network address or other properties. Now, by using zones with virtual network devices, you can configure multiple isolated virtual NICs, associate zones with each virtual network, and establish rules for isolation, connectivity, and quality of service (QoS).

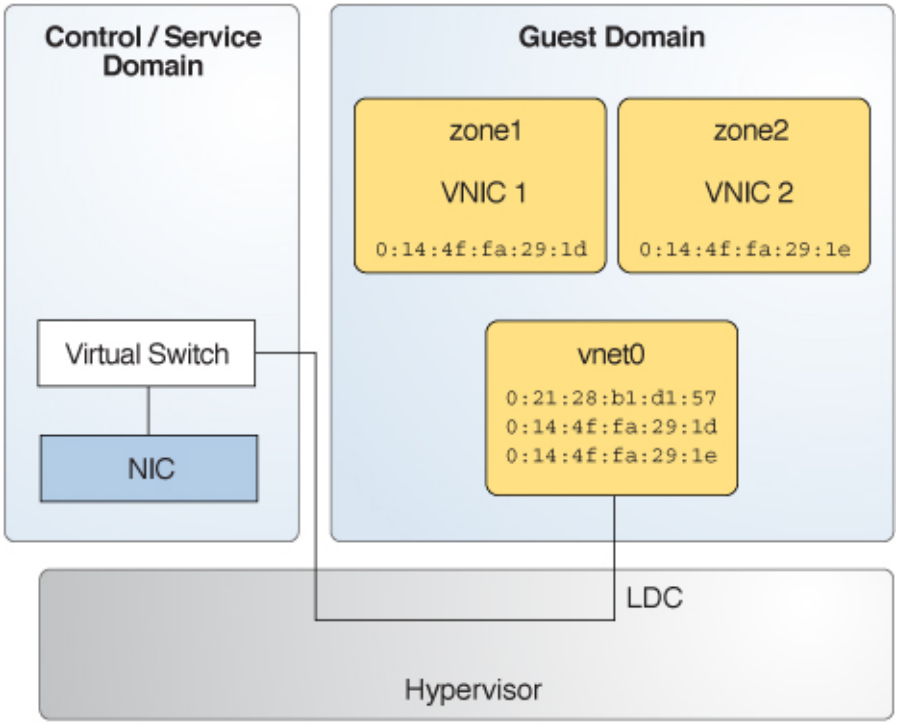
For more information, see the networking books in the Oracle Solaris 11.3 information library (`http://docs.oracle.com/cd/E53394_01/`).

A virtual network device in a logical domain can support multiple Oracle Solaris 11 virtual NICs. The virtual network device must be configured to support multiple MAC addresses, one for each virtual NIC it will support. Oracle Solaris zones in the logical domain connect to the virtual NICs.

Figure 12–17 shows a logical domain, `domain1`, that provides a single virtual network device called `vnet1` to the Oracle Solaris OS. This virtual network device can host multiple Oracle Solaris 11 virtual network devices, each of which has its own MAC address and can be assigned individually to a zone.

Within the `domain1` domain are three Oracle Solaris 11 zones: `zone1`, `zone2`, and `zone3`. Each zone is connected to the network by a virtual NIC based on the `vnet1` virtual network device.

FIGURE 12–17 Virtual NICs on Virtual Network Devices



The following sections describe the configuring of virtual NICs on virtual network devices and the creating of zones in the domain with the virtual NICs:

- "Configuring Virtual NICs on Virtual Network Devices" on page 280
- "Creating Oracle Solaris 11 Zones in a Domain" on page 281

For information about using virtual NICs on Ethernet SR-IOV virtual functions, see the following sections:

- "Creating Ethernet Virtual Functions" on page 90
- "Modifying Ethernet SR-IOV Virtual Functions" on page 97
- "Creating VNICs on SR-IOV Virtual Functions" on page 103

Configuring Virtual NICs on Virtual Network Devices

To configure virtual NICs on virtual network devices, the control domain must run at least Oracle Solaris 11.1 SRU 4 OS and the guest domain must run at least the Oracle Solaris 11.1 OS.

To configure a virtual network device to host multiple MAC addresses, use the `ldm add-vnet` or `ldm set-vnet` command to specify one or more comma-separated values for the `alt-mac-addrs` property. Valid values are an octet MAC address and `auto`. The `auto` value indicates that the system generates the MAC address.

For example, you can specify three system-generated alternate MAC addresses for a virtual network device in either of the following ways:

- By using the `ldm add-vnet` command. The following `ldm add-vnet` command creates the `vnet1` virtual network device on the `domain1` domain and makes three system-generated MAC addresses available to the device.

  ```
  primary# ldm add-vnet alt-mac-addrs=auto,auto,auto vnet1 primary-vsw0 domain1
  ```

- By using a combination of the `ldm add-vnet` and `ldm set-vnet` commands. The following `ldm add-vnet` and `ldm set-vnet` commands show how to create a virtual network device and subsequently assign more MAC addresses to the existing virtual network device.

  The first command uses the `ldm add-vnet` command to create the `vnet1` virtual network device on the `domain1` domain. The second command uses the `ldm set-vnet` command to make three system-generated MAC addresses available to the `vnet1` virtual network device.

  ```
  primary# ldm add-vnet vnet1 primary-vsw0 domain1
  primary# ldm set-vnet alt-mac-addrs=auto,auto,auto vnet1 domain1
  ```

Creating Oracle Solaris 11 Zones in a Domain

After creating the virtual NICs in "Configuring Virtual NICs on Virtual Network Devices" on page 280, create a zone that is associated with an available MAC address. For information about Oracle Solaris Zones, see *Creating and Using Oracle Solaris Zones* .

Use the `zonecfg` command to specify a MAC address to use for a zone:

```
zonecfg:zone-name> set mac-address=[MAC-address,auto]
```

You can either specify a value of `auto` to choose one of the available MAC addresses automatically or provide a specific alternate MAC address that you created with the `ldm set-vnet` command.

◆ ◆ ◆ CHAPTER 13

Migrating Domains

This chapter describes how to migrate domains from one host machine to another host machine.

This chapter covers the following topics:

Note – To use the migration features described in this chapter, you must be running the most recent versions of the Logical Domains Manager, system firmware, and Oracle Solaris OS. For information about migration using previous versions of Oracle VM Server for SPARC, see *Oracle VM Server for SPARC 3.3 Release Notes* and related versions of the administration guide.

Introduction to Domain Migration

Domain migration enables you to migrate a guest domain from one host machine to another host machine. The machine on which the migration is initiated is the *source machine*. The machine to which the domain is migrated is the *target machine*.

While a migration operation is in progress, the *domain to be migrated* is transferred from the source machine to the *migrated domain* on the target machine.

The *live migration* feature provides performance improvements that enable an active domain to be migrated while it continues to run. In addition to live migration, you can migrate bound or inactive domains, which is called *cold migration*.

You might use domain migration to perform tasks such as the following:

- Balancing the load between machines
- Performing hardware maintenance while a guest domain continues to run

To achieve the best migration performance, ensure that both the source machine and the target machine are running the latest version of the Logical Domains Manager.

Overview of a Migration Operation

The Logical Domains Manager on the source machine accepts the request to migrate a domain and establishes a secure network connection with the Logical Domains Manager that runs on the target machine. The migration occurs after this connection has been established. The migration operation is performed in the following phases:

Phase 1: After the source machine connects with the Logical Domains Manager that runs in the target machine, information about the source machine and the domain to be migrated are transferred to the target machine. This information is used to perform a series of checks to determine whether a migration is possible. The checks to perform are based on the state of the domain to be migrated. For example, if the domain to be migrated is active, a different set of checks are performed than if that domain is bound or inactive.

Phase 2: When all checks in Phase 1 have passed, the source and target machines prepare for the migration. On the target machine, a domain is created to receive the domain to be migrated. If the domain to be migrated is inactive or bound, the migration operation proceeds to Phase 5.

Phase 3: If the domain to be migrated is active, its runtime state information is transferred to the target machine. The domain to be migrated continues to run, and the Logical Domains Manager simultaneously tracks the modifications being made by the OS to this domain. This information is retrieved from the hypervisor on the source machine and installed in the hypervisor on the target machine.

Phase 4: The domain to be migrated is suspended. At this time, all of the remaining modified state information is recopied to the target machine. In this way, there is little or no perceivable interruption to the domain. The amount of interruption depends on the workload.

Phase 5: A handoff occurs from the Logical Domains Manager on the source machine to the Logical Domains Manager on the target machine. The handoff occurs when the migrated domain resumes execution (if the domain to be migrated was active) and the domain on the source machine is destroyed. From this point forward, the migrated domain is the sole version of the domain running.

Software Compatibility

For a migration to occur, both the source and target machines must be running compatible software, as follows:

- The Logical Domains Manager version running on both machines must be either the current version or the most recent previously released version.
- Both the source and target machines must have a compatible version of firmware installed to support live migration. Both machines must be running at least the minimum version of the firmware supported with this release of the Oracle VM Server for SPARC software.

For more information, see "Version Restrictions for Migration" on page 289.

Security for Migration Operations

Oracle VM Server for SPARC provides the following security features for migration operations:

- **Authentication.** Because the migration operation executes on two machines, a user must be authenticated on both the source and target machines in some cases. In particular, a user other than superuser must use the `LDoms Management` rights profile. However, if you perform a migration with SSL certificates, users are not required to be authenticated on both the target and source machines and you cannot specify another user.

  The `ldm migrate-domain` command permits you to optionally specify an alternate user name for authentication on the target machine. If this alternate user name is not specified, the user name of the user who is executing the migration command is used. See Example 13–3. In either case, the user is prompted for a password for the target machine, unless the `-p` option is used to initiate a non-interactive migration. See "Performing Non-Interactive Migrations" on page 292.
- **Encryption.** Oracle VM Server for SPARC uses SSL to encrypt migration traffic to protect sensitive data from exploitation and to eliminate the requirement for additional hardware and dedicated networks.

On platforms that have cryptographic units, the speed of the migration operation increases when the `primary` domain on the source and target machines has cryptographic units assigned. This increase in speed occurs because the SSL operations can be off-loaded to the cryptographic units.

The speed of a migration operation is automatically improved on platforms that have cryptographic instructions in the CPU. This improvement occurs because the SSL operations can be carried out by the cryptographic instructions rather than in software.

- **FIPS 140-2.** You can configure your system to perform domain migrations to use the Oracle Solaris FIPS 140-2 certified OpenSSL libraries. See "FIPS 140-2 Mode for Domain Migration" on page 287.

Configuring SSL Certificates for Migration

To perform certificate-based authentication, use the `-c` option with the `ldm migrate-domain` command. This option is mutually exclusive with the password file and alternate user options. If the `-c` option is not specified, the migration operation performs password authentication.

How to Configure SSL Certificates for Migration

To configure SSL certificates, you must perform the steps in this task on the control domain of both the source machine and the target machine.

1. **Create the `/var/share/ldomsmanager/trust` directory if it does not already exist.**

2. **Securely copy the remote `ldmd` certificate to the local `ldmd` trusted certificate directory.**

   The remote `ldmd` certificate is the `/var/share/ldomsmanager/server.crt` on the remote host. The local `ldmd` trusted certificate directory is `/var/share/ldomsmanager/trust`. Call the remote certificate file *remote-hostname*`.pem`.

3. **Create a symbolic link from the certificate in the `ldmd` trusted certificate directory to `/etc/certs/CA`.**

   Set the `REMOTE` variable to *remote-host*.

   ```
   localhost# ln -s /var/share/ldomsmanager/trust/${REMOTE}.pem /etc/certs/CA/
   ```

4. **Restart the `svc:/system/ca-certificates` service.**

   ```
   localhost# svcadm restart svc:/system/ca-certificates
   ```

5. **Verify that the configuration is operational.**

   ```
   localhost# openssl verify  /var/share/ldomsmanager/trust/${REMOTE}.pem
   /var/share/ldomsmanager/trust/remote-hostname.pem: OK
   ```

6. **Restart the `ldmd` daemon.**

   ```
   localhost# svcadm restart ldmd
   ```

Removing SSL Certificates

If you remove a `.pem` file from the `/var/share/ldomsmanager/trust` and `/etc/certs/CA` directories, you must restart the `svc:/system/ca-certificates` service and then the `ldmd` service. Note that any migrations that use that `.pem` file are still permitted until the services restart.

```
localhost# svcadm restart svc:/system/ca-certificates
localhost# svcadm restart ldmd
```

FIPS 140-2 Mode for Domain Migration

You can configure your Logical Domains Manager to perform domain migrations that use the Oracle Solaris FIPS 140-2 certified OpenSSL libraries. When the Logical Domains Manager is in FIPS 140-2 mode, you can use it only to migrate domains to another system that has the Logical Domains Manager running in FIPS 140-2 mode. Attempts to migrate to a non-FIPS system are rejected. If the Logical Domains Manager is not in FIPS 140-2 mode, it cannot migrate to a Logical Domains Manager that is in FIPS 140-2 mode.

To successfully start the Logical Domains Manager in FIPS 140-2 mode, you must enable the FIPS mediator. For step-by-step instructions, see "How to Run Logical Domains Manager in FIPS 140-2 Mode" on page 287.

For more information and showing how to use the FIPS 140-capable OpenSSL implementation, see "How to Switch to the FIPS 140-Capable OpenSSL Implementation" in *Managing Encryption and Certificates in Oracle Solaris 11.3* and "Example of Enabling Two Applications in FIPS 140 Mode on an Oracle Solaris System" in *Oracle SuperCluster M7-8 Owner's Guide: Overview* .

▼ How to Run Logical Domains Manager in FIPS 140-2 Mode

Before You Begin Before you can run the Logical Domains Manager in FIPS 140-2 mode, ensure that you are running at least version 3.2 of the Logical Domains Manager and that the `primary` domain runs at least the Oracle Solaris 11.2 OS.

1 Install and enable the FIPS 140-2 OpenSSL mediator.

a. Install the FIPS 140-2 OpenSSL mediator if necessary.

This package should be installed by default when you install the Oracle Solaris 11.2 OS.

```
# pkg install openssl-fips-140
```

b. **List the current OpenSSL mediator.**

```
# pkg mediator openssl
MEDIATOR VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
openssl  vendor            local      default
```

c. **List the available OpenSSL mediators.**

```
# pkg mediator -a openssl
MEDIATOR VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
openssl  vendor            vendor     default
openssl  system            system     fips-140
```

Caution – The OpenSSL implementation to which you switch must exist in the system. If you switch to an implementation that is not in the system, the system might become unusable.

d. **Enable the FIPS 140-2 mediator.**

```
# pkg set-mediator -I fips-140 openssl
```

e. **Reboot.**

```
# reboot
```

f. **Confirm that the FIPS 140-2 mediator is set.**

```
# pkg mediator openssl
    MEDIATOR     VER. SRC. VERSION IMPL. SRC. IMPLEMENTATION
    openssl      system            local      fips-140
```

2 **Configure the `ldmd` Daemon to use FIPS 140-2 mode.**

a. **Put the `ldmd` daemon in FIPS 140-2 mode.**

```
# svccfg -s ldoms/ldmd setprop ldmd/fips1402_enabled = true
```

b. **Restart the `ldmd` daemon.**

```
# svcadm refresh ldmd
# svcadm restart ldmd
```

▼ How to Revert the Logical Domains Manager to the Default Mode From FIPS 140-2 Mode

1 **Stop using the FIPS 140-2 OpenSSL mediator by reverting to the default OpenSSL mediator.**

Only perform this step if the FIPS 140-2 mediator is not required for other applications.

```
# pkg set-mediator -I default openssl
```

2 **Reboot.**

```
# reboot
```

3. **Configure the `ldmd` daemon to use the default mode.**

```
# svccfg -s ldoms/ldmd setprop ldmd/fips1402_enabled = false
```

4. **Restart the `ldmd` daemon.**

```
# svcadm refresh ldmd
# svcadm restart ldmd
```

Domain Migration Restrictions

The following sections describe restrictions for domain migration. The Logical Domains Manager software and the system firmware versions must be compatible to permit migrations. Also, you must meet certain CPU requirements to ensure a successful domain migration.

Live migration is not qualified and supported on all combinations of the source and target platforms and system firmware versions. For those combinations that cannot perform a live migration, you can perform a cold migration instead.

Version Restrictions for Migration

This section describes version restrictions for performing live migrations.

- **Logical Domains Manager version.** You can perform a live migration in either direction when one system runs the latest version of the Logical Domains Manager and the other system runs at least the immediately preceding version of the Logical Domains Manager.
- **Oracle Solaris OS version.** You can perform a live migration of a guest domain that runs at least the Oracle Solaris 10 9/10 OS. You *cannot* perform a live migration of a guest domain that runs the Oracle Solaris 10 10/09 OS or earlier Oracle Solaris OS versions. You can still boot these older Oracle Solaris OS versions and perform cold migrations of such domains.
- **System firmware version.** In general, you can perform a live migration between two systems when both the source and target machines support the appropriate minimum system firmware versions. See "Minimum System Firmware Versions" in *Oracle VM Server for SPARC 3.3 Installation Guide*.

However, some specific platform and firmware combinations do not support live migration. Attempting to perform the live migration of a domain from a system that runs at least system firmware version 8.4 or XCP2210 to a system that runs an older system firmware version fails. The failure occurs because of a hypervisor API mismatch between the newer and older system firmware versions. In this instance, the following message is issued:

```
primary# ldm migrate ldg1 root@target-name
Target Password:
Domain ldg1 is using features of the system firmware that are not supported in
the version of the firmware running on the target machine.
Domain Migration of LDom ldg1 failed
```

Note that you can perform the live migration of a domain from a system that runs system firmware version 8.3 to a system that runs at least system firmware version 8.4 unless the target machine is a SPARC M5-32 system. For more information, see "Domain Migrations From SPARC T4 Systems That Run System Firmware 8.3 to SPARC T5, SPARC M5, or SPARC M6 Systems Are Erroneously Permitted" in *Oracle VM Server for SPARC 3.3 Release Notes*.

Cross-CPU Restrictions for Migration

You cannot perform live migration operations between an UltraSPARC T2, UltraSPARC T2 Plus, or SPARC T3 system and a SPARC T5, SPARC M5, or SPARC M6 system.

CPU Restrictions for Migration

If the domain to be migrated is running an Oracle Solaris OS version older than the Oracle Solaris 10 1/13 OS, you might see the following message during the migration:

```
Domain domain-name is not running an operating system that is
compatible with the latest migration functionality.
```

The following CPU requirements and restrictions apply when you run an OS prior to the Oracle Solaris 10 1/13 OS:

- Full cores must be allocated to the migrated domain. If the number of threads in the domain to be migrated is less than a full core, the extra threads are unavailable to any domain until after the migrated domain is rebooted.
- After a migration, CPU dynamic reconfiguration (DR) is disabled for the migrated domain until it has been rebooted. At that time, you can use CPU DR on the migrated domain.
- The target machine must have enough free full cores to provide the number of threads that are required for the migrated domain. After the migration, if a full core is only partially used by the migrated domain, any extra threads are unavailable to any domain until after the migrated domain is rebooted.

These restrictions also apply when you attempt to migrate a domain that is running in OpenBoot or in the kernel debugger. See "Migrating a Domain From the OpenBoot PROM or a Domain That Is Running in the Kernel Debugger" on page 298.

Migration Restrictions for Setting `perf-counters`

Take care when performing migrations of domains that have the `perf-counters` property value set.

Before you perform the migration of a domain that has the `perf-counters` property value set to `global`, ensure that no other domain on the target machine has the `perf-counters` property set to `global`.

During a migration operation, the `perf-counters` property is treated differently based on whether the performance access capability is available on the source machine, the target machine, or both.

The `perf-counters` property value is treated as follows:

- **Source machine only.** The `perf-counters` property value is not propagated to the target machine.
- **Target machine only.** The `perf-counters` property value on the machine to be migrated is updated to be equivalent to `perf-counters=`.
- **Source and target machines.** The `perf-counters` property value is propogated from the domain to be migrated to the migrated domain on the target machine.

For more information about the `perf-counters` property, see "Using Perf-Counter Properties" on page 339 and the `ldm`(1M) man page.

Migrating a Domain

You can use the `ldm migrate-domain` command to initiate the migration of a domain from one host machine to another host machine.

Note – If you migrate a domain, any named resources that you assigned by using the `cid` and `mblock` properties are dropped. Instead, the domain uses anonymous resources on the target system.

For information about migrating an active domain while it continues to run, see "Migrating an Active Domain" on page 292. For information about migrating a bound or inactive domain, see "Migrating Bound or Inactive Domains" on page 298.

For information about the migration options and operands, see the `ldm`(1M) man page.

Note – After a domain migration completes, save a new configuration to the SP of both the source and target systems. As a result, the state of the migrated domain is correct if either the source or target system undergoes a power cycle.

Performing a Dry Run

When you provide the `-n` option to the `ldm migrate-domain` command, migration checks are performed but the domain is not migrated. Any requirement that is not satisfied is reported as an error. The dry run results enable you to correct any configuration errors before you attempt an actual migration.

Note – Because of the dynamic nature of logical domains, a dry run could succeed and an actual migration fail, and vice versa.

Performing Non-Interactive Migrations

Use the SSL certificate method to perform non-interactive migration operations. Although the use of the legacy `ldm migrate-domain -p` *filename* command to initiate a non-interactive migration operation is deprecated, you can still use it.

The file name you specify as an argument to the `-p` option must have the following characteristics:

- The first line of the file must contain the password.
- The password must be plain text.
- The password must not exceed 256 characters in length.

A newline character at the end of the password and all lines that follow the first line are ignored.

The file in which you store the target machine's password must be properly secured. If you plan to store passwords in this manner, ensure that the file permissions are set so that only the root owner or a privileged user can read or write the file (400 or 600).

Migrating an Active Domain

Certain requirements and restrictions are imposed on the domain to be migrated, the source machine, and the target machine when you attempt to migrate an active domain. For more information, see "Domain Migration Restrictions" on page 289.

Tip – You can reduce the overall migration time by adding more virtual CPUs to the `primary` domain on both the source and target machines. Having at least two whole cores in each `primary` domain is recommended but not required.

A domain "loses time" during the migration process. To mitigate this time-loss issue, synchronize the domain to be migrated with an external time source, such as a Network Time Protocol (NTP) server. When you configure a domain as an NTP client, the domain's date and time are corrected shortly after the migration completes.

To configure a domain as an Oracle Solaris 10 NTP client, see "Managing Network Time Protocol (Tasks)" in *System Administration Guide: Network Services* . To configure a domain as an Oracle Solaris 11 NTP client, see "Managing Network Time Protocol (Tasks)" in *Introduction to Oracle Solaris 11 Network Services* .

Note – During the suspend phase at the end of a migration, a guest domain might experience a slight delay. This delay should not result in any noticeable interrupt to network communications, especially if the protocol includes a retry mechanism such as TCP or if a retry mechanism exists at the application level such as NFS over UDP. However, if the guest domain runs a network-sensitive application such as Routing Information Protocol (RIP), the domain might experience a short delay or interrupt when attempting an operation. This delay would occur during the short period when the guest network interface is being torn down and re-created during the suspension phase.

Domain Migration Requirements for CPUs

Following are the requirements and restrictions on CPUs when you perform a migration:

- The target machine must have sufficient free virtual CPUs to accommodate the number of virtual CPUs in use by the domain to be migrated.
- Setting the `cpu-arch` property on the guest domain enables you to migrate the domain between systems that have different processor types. Note that the guest domain must be in a bound or inactive state to change the `cpu-arch` value.

  The supported `cpu-arch` property values are as follows:

  - `native` uses CPU-specific hardware features to enable a guest domain to migrate *only* between platforms that have the same CPU type. `native` is the default value.
  - `migration-class1` is a cross-CPU migration family for SPARC platforms starting with the SPARC T4. These platforms support hardware cryptography during and after these migrations so that there is a lower bound to the supported CPUs.

    This value is not compatible with UltraSPARC T2, UltraSPARC T2 Plus, or SPARC T3 platforms, or Fujitsu M10 platforms.
  - `sparc64-class1` is a cross-CPU migration family for SPARC64 platforms. Because the `sparc64-class1` value is based on SPARC64 instructions, it has a greater number of instructions than the `generic` value. Therefore, it does not have a performance impact compared to the `generic` value.

    This value is compatible only with Fujitsu M10 servers.
  - `generic` uses the lowest common CPU hardware features that are used by all platforms to enable a guest domain to perform a CPU-type-independent migration.

  The following `isainfo -v` commands show the instructions that are available on a system when `cpu-arch=generic` and when `cpu-arch=migration-class1`.

  - `cpu-arch=generic`

    ```
    # isainfo -v
    64-bit sparcv9 applications
            asi_blk_init vis2 vis popc
    ```

```
32-bit sparc applications
        asi_blk_init vis2 vis popc v8plus div32 mul32
```

- `cpu-arch=migration-class1`

```
# isainfo -v
64-bit sparcv9 applications
        crc32c cbcond pause mont mpmul sha512 sha256 sha1 md5
        camellia des aes ima hpc vis3 fmaf asi_blk_init vis2
        vis popc
32-bit sparc applications
        crc32c cbcond pause mont mpmul sha512 sha256 sha1 md5
        camellia des aes ima hpc vis3 fmaf asi_blk_init vis2
        vis popc v8plus div32 mul32
```

Using the `generic` value might result in reduced performance of the guest domain compared to using the `native` value. The possible performance degradation occurs because the guest domain uses only generic CPU features that are available on all supported CPU types instead of using the native hardware features of a particular CPU. By not using these features, the `generic` value enables the flexibility of migrating the domain between systems that use CPUs that support different features.

If migrating a domain between at least SPARC T4 systems, you can set `cpu-arch=migration-class1` to improve the guest domain performance. While the performance is improved from using the `generic` value, the `native` value still provides the best performance for the guest domain.

Use the `psrinfo -pv` command when the `cpu-arch` property is set to `native` to determine the processor type, as follows:

```
# psrinfo -pv
The physical processor has 2 virtual processors (0 1)
  SPARC-T5 (chipid 0, clock 3600 MHz)
```

Note that when the `cpu-arch` property is set to a value other than `native`, the `psrinfo -pv` output does not show the platform type. Instead, the command shows that the `sun4v-cpu` CPU module is loaded.

```
# psrinfo -pv
The physical processor has 2 cores and 13 virtual processors (0-12)
  The core has 8 virtual processors (0-7)
  The core has 5 virtual processors (8-12)
    sun4v-cpu (chipid 0, clock 3600 MHz)
```

Migration Requirements for Memory

The target machine memory requirements are as follows:

- Sufficient free memory to accommodate the migration of a domain
- The free memory must be available in a compatible layout

Compatibility requirements differ for each SPARC platform. However, at a minimum the real address and physical address alignment relative to the largest supported page size must be preserved for each memory block in the migrated domain.

Use the `pagesize` commmand to determine the largest page size that is supported on the target machine.

For a guest domain that runs at least the Oracle Solaris 11.3 OS, the migrated domain's memory blocks might be automatically split up during the migration so that the migrated domain can fit into smaller available free memory blocks. Memory blocks can only be split up on boundaries aligned with the largest page size.

Other memory layout requirements for operating systems, firmware, or platforms might prevent memory blocks from being split during a given migration. This situation could cause the migration to fail even when the total amount of free memory available is sufficient for the domain.

Migration Requirements for Physical I/O Devices

Domains that have direct access to physical devices cannot be migrated. For example, you cannot migrate I/O domains. However, virtual devices that are associated with physical devices can be migrated.

For more information, see "Migration Requirements for PCIe Endpoint Devices" on page 296 and "Migration Requirements for PCIe SR-IOV Virtual Functions" on page 296.

Migration Requirements for Virtual I/O Devices

All virtual I/O services that are used by the domain to be migrated must be available on the target machine. In other words, the following conditions must exist:

- Each virtual disk back end that is used in the domain to be migrated must be defined on the target machine. This shared storage can be a SAN disk, or storage that is available by means of the NFS or iSCSI protocols. The virtual disk back end you define must have the same volume and service names as on the source machine. Paths to the back end might be different on the source and target machines, but they *must* refer to the same back end.

Caution – A migration will succeed even if the paths to a virtual disk back end on the source and target machines do not refer to the same storage. However, the behavior of the domain on the target machine will be unpredictable, and the domain is likely to be unusable. To remedy the situation, stop the domain, correct the configuration issue, and then restart the domain. If you do not perform these steps, the domain might be left in an inconsistent state.

- Each virtual network device in the domain to be migrated must have a corresponding virtual network switch on the target machine. Each virtual network switch must have the same name as the virtual network switch to which the device is attached on the source machine.

  For example, if `vnet0` in the domain to be migrated is attached to a virtual switch service named `switch-y`, a domain on the target machine must provide a virtual switch service named `switch-y`.

  Note – The physical network on the target machine must be correctly configured so that the migrated domain can access the network resources it requires. Otherwise, some network services might become unavailable on the domain after the migration completes.

  For example, you might want to ensure that the domain can access the correct network subnet. Also, you might want to ensure that gateways, routers, or firewalls are properly configured so that the domain can reach the required remote systems from the target machine.

  MAC addresses used by the domain to be migrated that are in the automatically allocated range must be available for use on the target machine.
- A virtual console concentrator (`vcc`) service must exist on the target machine and have at least one free port. Explicit console constraints are ignored during the migration. The console for the migrated domain is created by using the migrated domain name as the console group and by using any available port on any available `vcc` device in the control domain. If no available ports are available in the control domain, the console is created by using an available port on an available `vcc` device in a service domain. The migration fails if there is a conflict with the default group name.
- Each virtual SAN that is used by the domain to be migrated must be defined on the target machine.

Migration Requirements for PCIe Endpoint Devices

You cannot perform a domain migration on an I/O domain that is configured with PCIe endpoint devices.

For information about the direct I/O feature, see "Creating an I/O Domain by Assigning PCIe Endpoint Devices" on page 143.

Migration Requirements for PCIe SR-IOV Virtual Functions

You cannot perform a domain migration on an I/O domain that is configured with PCIe SR-IOV virtual functions.

For information about the SR-IOV feature, see Chapter 7, "Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions."

Migration Requirements for NIU Hybrid I/O

You can migrate a domain that uses NIU Hybrid I/O resources. A constraint that specifies NIU Hybrid I/O resources is not a hard requirement of a domain. If such a domain is migrated to a machine that does not have available NIU resources, the constraint is preserved but not fulfilled.

Note that the NIU Hybrid I/O feature is deprecated in favor of SR-IOV. Oracle VM Server for SPARC 3.3 is the last software release to include this feature.

Migration Requirements for Cryptographic Units

On platforms that have cryptographic units, you can migrate a guest domain that has bound cryptographic units if it runs an operating system that supports cryptographic unit dynamic reconfiguration (DR).

At the start of the migration, the Logical Domains Manager determines whether the domain to be migrated supports cryptographic unit DR. If supported, the Logical Domains Manager attempts to remove any cryptographic units from the domain. After the migration completes, the cryptographic units are re-added to the migrated domain.

Note – If the constraints for cryptographic units cannot be met on the target machine, the migration operation will not be blocked. In such a case, the migrated domain might have fewer cryptographic units than it had prior to the migration operation.

Delayed Reconfiguration in an Active Domain

Any active delayed reconfiguration operations on the source or target machine prevent a migration from starting. You are not permitted to initiate a delayed reconfiguration operation while a migration is in progress.

Migrating While an Active Domain Has the Power Management Elastic Policy in Effect

You can perform a live migration when the power management (PM) elastic policy is in effect on either the source machine or the target machine.

Operations on Other Domains

While a migration is in progress on a machine, any operation that might result in the modification of the state or configuration of the domain being migrated is blocked. All operations on the domain itself, as well as operations such as bind and stop on other domains on the machine, are blocked.

Migrating a Domain From the OpenBoot PROM or a Domain That Is Running in the Kernel Debugger

Performing a domain migration requires coordination between the Logical Domains Manager and the Oracle Solaris OS that is running in the domain to be migrated. When a domain to be migrated is running in OpenBoot or in the kernel debugger (`kmdb`), this coordination is not possible. As a result, the migration attempt fails.

When a domain to be migrated is running in OpenBoot, you will see the following message:

```
primary# ldm migrate ldg1 system2
Migration is not supported while the domain ldg1 is in the 'OpenBoot Running' state
Domain Migration of LDom ldg1 failed
```

When a domain to be migrated is running in the kernel debugger (`kmdb`), you will see the following message:

```
primary# ldm migrate ldg1 system2
Migration is not supported while the domain ldg1 is in the 'Solaris debugging' state
Domain Migration of LDom ldg1 failed
```

Migrating Bound or Inactive Domains

Only a few domain migration restrictions apply to a bound or inactive domain because such domains are not executing at the time of the migration. Therefore, you can migrate between different platform types, such as SPARC T3 to SPARC T5 platforms or Fujitsu M10 platforms, because no runtime state is being copied across.

The migration of a bound domain requires that the target machine is able to satisfy the CPU, memory, and I/O constraints of the domain to be migrated. If these constraints cannot be met, the migration will fail.

Caution – When you migrate a bound domain, the virtual disk back-end `options` and `mpgroup` values are not checked because no runtime state information is exchanged with the target machine. This check *does* occur when you migrate an active domain.

The migration of an inactive domain does not have such requirements. However, the target machine must satisfy the migrated domain's constraints when a bind is later attempted or the domain binding will fail.

Note – After a domain migration completes, save a new configuration to the SP of both the source and target systems. As a result, the state of the migrated domain is correct if either the source or target system undergoes a power cycle.

Migration Requirements for Virtual I/O Devices

For an inactive domain, no checks are performed of the virtual I/O (VIO) constraints. Therefore, the VIO servers do not need to exist for the migration to succeed. As with any inactive domain, the VIO servers must exist and be available at the time the domain is bound.

Migration Requirements for PCIe Endpoint Devices

You cannot perform a domain migration on an I/O domain that is configured with PCIe endpoint devices. This requirement applies to bound domains but not to inactive domains.

For information about the direct I/O (DIO) feature, see "Creating an I/O Domain by Assigning PCIe Endpoint Devices" on page 143.

Migration Requirements for PCIe SR-IOV Virtual Functions

You cannot perform a domain migration on an I/O domain that is configured with PCIe SR-IOV virtual functions. This requirement applies to bound domains but not to inactive domains.

For information about the SR-IOV feature, see Chapter 7, "Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions."

Monitoring a Migration in Progress

When a migration is in progress, the domain being migrated and the migrated domain are shown differently in the status output. The output of the `ldm list` command indicates the state of the migrating domain.

The sixth column in the `FLAGS` field shows one of the following values:

- `s` – The domain that is the source of the migration.
- `+` – The migrated domain that is the target of the migration.
- `e` – An error has occurred that requires user intervention.

The following command shows that the `ldg-src` domain is the source of the migration:

```
# ldm list ldg-src
NAME       STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg-src    suspended  -n---s          1     1G       0.0%  2h 7m
```

The following command shows that the `ldg-tgt` domain is the target of the migration:

```
# ldm list ldg-tgt
NAME       STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg-tgt    bound      -----t  5000    1     1G
```

The long form of the status output shows additional information about the migration. On the source machine, the status output shows the completion percentage of the operation as well as the names of the target machine and the migrated domain. Similarly, on the target machine, the status output shows the completion percentage of the operation as well as the names of the source machine and the domain being migrated.

The following command shows the progress of a migration operation for the `ldg-src` domain:

```
# ldm list -o status ldg-src
NAME
ldg-src

STATUS
    OPERATION    PROGRESS    TARGET
    migration    17%         system2
```

Canceling a Migration in Progress

After a migration starts, the migration operation is terminated if the `ldm` command is interrupted by a KILL signal. When the migration operation is terminated, the migrated domain is destroyed and the domain to be migrated is resumed if it was active. If the controlling shell of the `ldm` command is lost, the migration continues in the background.

A migration operation can also be canceled externally by using the `ldm cancel-operation` command. This command terminates the migration in progress and the domain being

migrated resumes as the active domain. The `ldm cancel-operation` command must be initiated from the source machine. On a given machine, any migration-related command affects the migration operation that was started from that machine. A target machine cannot control a migration operation.

Recovering From a Failed Migration

The migration operation terminates if the network connection is lost after the domain being migrated has completed sending all the runtime state information to the migrated domain but before the migrated domain can acknowledge that the domain has been resumed.

You must determine whether the migration completed successfully by taking the following steps:

1. Determine whether the migrated domain has successfully resumed operations. The migrated domain will be in one of two states:
   - If the migration completed successfully, the migrated domain is in the normal state.
   - If the migration failed, the target machine cleans up and destroys the migrated domain.
2. If the migrated domain successfully resumed operations, you can safely destroy the domain on the source machine that is in the error state. However, if the migrated domain is not present, the domain on the source machine is still the master version of the domain and must be recovered. To recover this domain, run the `ldm cancel-operation` command on the source machine. This command clears the error state and restores the domain to its original condition.

Migration Examples

EXAMPLE 13–1 Using SSL Certificates to Perform a Guest Domain Migration

This example shows how to migrate the `ldg1` domain to a machine called `system2`. Before the migration operation starts, you must have configured the SSL certificates on both the source and target machines. See "How to Configure SSL Certificates for Migration" on page 286.

```
# ldm migrate-domain -c ldg1 system2
```

EXAMPLE 13–2 Migrating a Guest Domain

This example shows how to migrate the `ldg1` domain to a machine called `system2`.

```
# ldm migrate-domain ldg1 system2
Target Password:
```

To perform this migration without being prompted for the target machine password, use the following command:

EXAMPLE 13–2 Migrating a Guest Domain *(Continued)*

```
# ldm migrate-domain -p pfile ldg1 system2
```

The `-p` option takes a file name as an argument. The specified file contains the superuser password for the target machine. In this example, `pfile` contains the password for the target machine, `system2`.

EXAMPLE 13–3 Migrating and Renaming a Guest Domain

This example shows how to rename a domain as part of the migration operation. The `ldg-src` domain on the source machine is renamed to `ldg-tgt` on the target machine (`system2`) as part of the migration. In addition, the `ldm-admin` user is used for authentication on the target machine.

```
# ldm migrate ldg-src ldm-admin@system2:ldg-tgt
Target Password:
```

EXAMPLE 13–4 Migration Failure Message

This example shows the error message that you might see if the target machine does not support the latest migration functionality.

```
# ldm migrate ldg1 dt212-346
Target Password:
The target machine is running an older version of the domain
manager that does not support the latest migration functionality.

Upgrading to the latest software will remove restrictions on
a migrated domain that are in effect until it is rebooted.
Consult the product documentation for a full description of
these restrictions.

The target machine is running an older version of the domain manager
that is not compatible with the version running on the source machine.

Domain Migration of LDom ldg1 failed
```

EXAMPLE 13–5 Obtaining the Migration Status for the Domain on the Target Machine

This example shows how to obtain the status on a migrated domain while a migration is in progress. In this example, the source machine is `t5-sys-1`.

```
# ldm list -o status ldg-tgt
NAME
ldg-tgt

STATUS
    OPERATION    PROGRESS    SOURCE
    migration    55%         t5-sys-1
```

EXAMPLE 13–6 Obtaining the Parseable Migration Status for the Domain on the Source Machine

This example shows how to obtain the parseable status on the domain being migrated while a migration is in progress. In this example, the target machine is `system2`.

```
# ldm list -o status -p ldg-src
VERSION 1.6
DOMAIN|name=ldg-src|
STATUS
|op=migration|progress=42|error=no|target=system2
```

◆ ◆ ◆ CHAPTER 14

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 and the Oracle Solaris 11 OS
- **Virtual I/O devices –** Supported in at least the Oracle Solaris 10 10/08 OS and the Oracle Solaris 11 OS
- **Cryptographic units –** Supported in at least the Oracle Solaris 10 1/13 OS and the Oracle Solaris 11 OS
- **Memory –** See "Using Memory Dynamic Reconfiguration" on page 322
- **CPU whole cores –** See "Fully Qualified Oracle Solaris OS Versions" in *Oracle VM Server for SPARC 3.3 Installation Guide*
- **Physical I/O devices –** Not supported

To use the DR capability, the Logical Domains DR daemon, `drd`, must be running in the domain that you want to change. See the `drd`(1M) man page.

Delayed Reconfiguration

In contrast to DR operations that take place immediately, delayed reconfiguration operations take effect in the following circumstances:

- After the next reboot of the OS
- After a stop and start of a logical domain if no OS is running

In general, delayed reconfiguration operations are restricted to the control domain. For all other domains, you must stop the domain to modify the configuration unless the resource can be dynamically reconfigured.

Delayed reconfiguration operations are restricted to the control domain. You can run a limited number of commands while a delayed reconfiguration on the root domain is in progress to support operations that cannot be completed dynamically. These subcommands are `add-io`, `set-io`, `remove-io`, `create-vf`, and `destroy-vf`. You can also run the `ldm start-reconf` command on the root domain. For all other domains, you must stop the domain to modify the configuration unless the resource can be dynamically reconfigured.

While a delayed reconfiguration is in progress, other reconfiguration requests for that domain are deferred until it is rebooted or stopped and started.

The `ldm cancel-reconf` command cancels delayed reconfiguration operations on the domain. For more information about how to use the delayed reconfiguration feature, see the `ldm(1M)` man page.

Note – You cannot use the `ldm cancel-reconf` command if any other `ldm remove-*` commands have already performed a delayed reconfiguration operation on virtual I/O devices. The `ldm cancel-reconf` command fails in this circumstance.

You can use delayed reconfiguration to decrease resources on the control domain. To remove a large number of CPUs from the control domain, see "Removing a Large Number of CPUs From a Domain Might Fail" in *Oracle VM Server for SPARC 3.3 Release Notes*. To remove large amounts of memory from the control domain, see "Decrease the Control Domain's Memory" on page 323.

Note – When the `primary` domain is in a delayed reconfiguration state, resources that are managed by Oracle VM Server for SPARC are power-managed *only* after the `primary` domain reboots. Resources that are managed directly by the OS, such as CPUs that are managed by the Solaris Power Aware Dispatcher, are not affected by this state.

Only One CPU Configuration Operation Is Permitted to Be Performed During a Delayed Reconfiguration

Do not attempt to perform more than one CPU configuration operation on the `primary` domain while it is in a delayed reconfiguration. If you attempt more CPU configuration requests, they will be rejected.

Workaround: Perform one of the following actions:

- Cancel the delayed reconfiguration, start another one, and request the configuration changes that were lost from the previous delayed reconfiguration.
- Reboot the control domain with the incorrect CPU count and then make the allocation corrections after the domain reboots.

Resource Allocation

The resource allocation mechanism uses resource allocation constraints to assign resources to a domain at bind time.

A *resource allocation constraint* is a hard requirement that the system must meet when you assign a resource to a domain. If the constraint cannot be met, both the resource allocation and the binding of the domain fail.

Caution – Do not create a circular dependency between two domains in which each domain provides services to the other. Such a configuration creates a single point of failure condition where an outage in one domain causes the other domain to become unavailable. Circular dependency configurations also prevent you from unbinding the domains after they have been bound initially.

The Logical Domains Manager does not prevent the creation of circular domain dependencies.

If the domains cannot be unbound due to a circular dependency, remove the devices that cause the dependency and then attempt to unbind the domains.

CPU Allocation

When you run threads from the same core in separate domains, you might experience unpredictable and poor performance. The Oracle VM Server for SPARC software uses the CPU affinity feature to optimize CPU allocation during the logical domain binding process, which occurs before you can start the domain. This feature attempts to keep threads from the same core allocated to the same logical domain because this type of allocation improves cache sharing between the threads within the same core.

CPU affinity attempts to avoid the sharing of cores among domains unless there is no other recourse. When a domain has been allocated a partial core and requests more strands, the strands from the partial core are bound first, and then another free core is located to fulfill the request, if necessary.

The CPU allocation mechanism uses the following constraints for CPU resources:

- **Whole-core constraint.** This constraint specifies that CPU cores are allocated to a domain rather than virtual CPUs. As long as the domain does not have the max-cores constraint enabled, the whole-core constraint can be added or removed by using the `ldm set-core` or `ldm set-vcpu` command, respectively. The domain can be inactive, bound, or active. However, enough cores must be available to satisfy the request to apply the constraint. As a worst-case example, if a domain that shares cores with another domain requests the whole-core constraint, cores from the free list would need to be available to satisfy the request. As a best-case example, all the virtual CPUs in the core are already on core boundaries, so the constraint is applied without changes to CPU resources.
- **Maximum number of cores (max-cores) constraint.** This constraint specifies the maximum number of cores that can be assigned to a bound or active domain.

▼ How to Apply the Whole-Core Constraint

Ensure that the domain has the whole-core constraint enabled prior to setting the max-cores constraint.

1. **Apply the whole-core constraint on the domain.**

   ```
   primary# ldm set-core 1 domain-name
   ```

2. **Verify that the domain has the whole-core constraint enabled.**

   ```
   primary# ldm ls -o resmgmt domain-name
   ```

   Notice that `max-cores` is set to `unlimited`. The domain cannot be used in conjunction with hard partitioning until the max-cores constraint is enabled.

Example 14–1 Applying the Whole-Core Constraint

This example shows how to apply the whole-core constraint on the `ldg1` domain. The first command applies the constraint, while the second command verifies that it is enabled.

```
primary# ldm set-core 1 ldg1
primary# ldm ls -o resmgmt ldg1
NAME
ldg1

CONSTRAINT
    cpu=whole-core
    max-cores=unlimited
```

▼ How to Apply the Max-Cores Constraint

Ensure that the domain has the whole-core constraint enabled prior to setting the max-cores constraint.

You can only enable, modify, or disable the max-cores constraint on an inactive domain, not on a domain that is bound or active. Before you update the max-cores constraint on the control domain, you must first initiate a delayed reconfiguration.

1. **Enable the max-cores constraint on the domain.**

   ```
   primary# ldm set-domain max-cores=max-number-of-CPU-cores domain-name
   ```

   Note – The cryptographic units that are associated with those cores are unaffected by core additions. So, the system does not automatically add the associated cryptographic units to the domain. However, a cryptographic unit is automatically removed *only* when the last virtual CPU of the core is being removed. This action prevents a cryptographic unit from being "orphaned."

2. **Verify that the whole-core constraint is enabled.**

   ```
   primary# ldm ls -o resmgmt domain-name
   ```

3 Bind and restart the domain.

```
primary# ldm bind domain-name
primary# ldm start domain-name
```

Now, you can use the domain with hard partitioning.

Example 14–2 Applying the Max-Cores Constraint

This example shows how to constrain max-cores to three cores by setting the `max-cores` property, and verifying that the constraint is enabled:

```
primary# ldm set-domain max-cores=3 ldg1
primary# ldm ls -o resmgmt ldg1
NAME
ldg1

CONSTRAINT
    cpu=whole-core
    max-cores=3
```

Now, you can use the domain with hard partitioning.

The following example removes the max-cores constraint from the unbound and inactive `ldg1` domain, but leaves the whole-core constraint as-is.

```
primary# ldm stop ldg1
primary# ldm unbind ldg1
primary# ldm set-domain max-cores=unlimited ldg1
```

Alternately, to remove both the max-cores constraint and the whole-core constraint from the `ldg1` domain, assign virtual CPUs instead of cores, as follows:

```
primary# ldm set-vcpu 8 ldg1
```

In either case, bind and restart the domain.

```
primary# ldm bind ldg1
primary# ldm start ldg1
```

Interactions Between the Whole-Core Constraint and Other Domain Features

This section describes the interactions between the whole-core constraint and the following features:

- “CPU Dynamic Reconfiguration” on page 311
- “Dynamic Resource Management” on page 311

CPU Dynamic Reconfiguration

The whole-core constraint is fully compatible with CPU dynamic reconfiguration (DR). When a domain is defined with the whole-core constraint, you can use the `ldm add-core`, `ldm set-core`, or `ldm remove-core` command to change the number of cores on an active domain.

However, if a bound or active domain is not in delayed reconfiguration mode, its number of cores cannot exceed the maximum number of cores. This maximum is set with the maximum core constraint, which is automatically enabled when the whole-core constraint is enabled. Any CPU DR operation that does not satisfy the maximum core constraint fails.

Dynamic Resource Management

The whole-core constraint is fully compatible with dynamic resource management (DRM).

The expected interactions between the whole-core constraint and DRM are as follows:

- While a DRM policy exists for a domain, you cannot switch the domain from being whole-core constrained to whole-core unconstrained or from being whole-core unconstrained to whole-core constrained. For example:
  - When a domain is whole-core constrained, you cannot use the `ldm set-vcpu` command to specify a number of virtual CPUs and to remove the whole-core constraint.
  - When a domain is not whole-core constrained, you cannot use the `ldm set-core` command to specify a number of whole cores and to add the whole-core constraint.
- When a domain is whole-core constrained and you specify the `attack`, `decay`, `vcpu-min`, or `vcpu-max` value, the value must be a whole-core multiple.

Configuring the System With Hard Partitions

This section describes hard partitioning with the Oracle VM Server for SPARC software, and how to use hard partitioning to conform to the Oracle CPU licensing requirements.

For information about Oracle's hard partitioning requirements for software licenses, see Partitioning: Server/Hardware Partitioning (`http://www.oracle.com/us/corporate/pricing/partitioning-070609.pdf`).

- **CPU cores and CPU threads.** The Oracle VM Server for SPARC software runs on SPARC T-Series and SPARC M-Series platforms, and Fujitsu M10 platforms. The processors that are used in these systems have multiple CPU cores, each of which contains multiple CPU threads.
- **Hard partitioning and CPU whole cores.** Beginning with the Oracle VM Server for SPARC 2.0 release, hard partitioning is enforced by using CPU whole-core configurations. A CPU whole-core configuration has domains that are allocated CPU whole cores instead of individual CPU threads. By default, a domain is configured to use CPU threads.

When binding a domain in a whole-core configuration, the system provisions the specified number of CPU cores and all its CPU threads to the domain. Using a CPU whole-core configuration limits the number of CPU cores that can be dynamically assigned to a bound or active domain.

- **Oracle hard partition licensing.** To conform to the Oracle hard partition licensing requirement, you must use at least the Oracle VM Server for SPARC 2.0 release. You must also use CPU whole cores as follows:
  - A domain that runs applications that use Oracle hard partition licensing must be configured with CPU whole cores.
  - A domain that does not run applications that use Oracle hard partition licensing is not required to be configured with CPU whole cores. For example, if you do not run any Oracle applications in the control domain, that domain is not required to be configured with CPU whole cores.

Checking the Configuration of a Domain

You use the `ldm list-o` command to determine whether a domain is configured with CPU whole cores and how to list the CPU cores that are assigned to a domain.

- To determine whether the domain is configured with CPU whole cores:

  ```
  primary# ldm list -o resmgmt domain-name
  ```

  Verify that the whole-core constraint appears in the output and that the `max-cores` property specifies the maximum number of CPU cores that are configured for the domain. See the `ldm(1M)` man page.

  The following command shows that the `ldg1` domain is configured with CPU whole cores and a maximum of five cores:

  ```
  primary# ldm list -o resmgmt ldg1
  NAME
  ldg1

  CONSTRAINT
     whole-core
     max-cores=5
  ```

- When a domain is bound, CPU cores are assigned to the domain. To list the CPU cores that are assigned to a domain:

  ```
  primary# ldm list -o core domain-name
  ```

  The following command shows the cores that are assigned to the `ldg1` domain:

  ```
  primary# ldm list -o core ldg1
  NAME
  ldg1
  ```

```
CORE
CID    PCPUSET
1    (8, 9, 10, 11, 12, 13, 14, 15)
2    (16, 17, 18, 19, 20, 21, 22, 23)
```

Configuring a Domain With CPU Whole Cores

The tasks in this section explain how to create a new domain with CPU whole cores, how to configure an existing domain with CPU whole cores, and how to configure the `primary` domain with CPU whole cores.

Use the following command to configure a domain to use CPU whole cores:

`ldm set-core` *number-of-CPU-cores domain*

This command also specifies the maximum number of CPU cores for the domain, which is the CPU cap. See the `ldm(1M)` man page.

The CPU cap and the allocation of CPU cores is handled by separate commands. By using these commands, you can independently allocate CPU cores, set a cap, or both. The allocation unit can be set to cores even when no CPU cap is in place. However, running the system in this mode is *not* acceptable for configuring hard partitioning on your Oracle VM Server for SPARC system.

- Allocate the specified number of CPU cores to a domain by using the `add-core`, `set-core`, or `rm-core` subcommand.
- Set the CPU cap by using the `create-domain` or `set-domain` subcommand to specify the `max-cores` property value.

  You must set the cap if you want to configure hard partitioning on your Oracle VM Server for SPARC system.

▼ How to Create a New Domain With CPU Whole Cores

Note – You only need to stop and unbind the domain if you optionally set the max-cores constraint.

1 Create the domain.

```
primary# ldm create domain-name
```

2 Set the number of CPU whole cores for the domain.

```
primary# ldm set-core number-of-CPU-cores domain
```

3 (Optional) Set the `max-cores` property for the domain.

```
primary# ldm set-domain max-cores=max-number-of-CPU-cores domain
```

4. **Configure the domain.**

   During this configuration, ensure that you use the `ldm add-core`, `ldm set-core`, or `ldm rm-core` command.

5. **Bind and start the domain.**

```
primary# ldm bind domain-name
primary# ldm start domain-name
```

Example 14–3 Creating a New Domain With Two CPU Whole Cores

This example creates a domain, `ldg1`, with two CPU whole cores. The first command creates the `ldg1` domain. The second command configures the `ldg1` domain with two CPU whole cores.

At this point, you can perform further configuration on the domain, subject to the restrictions described in Step 3 in "How to Create a New Domain With CPU Whole Cores" on page 313.

The third and fourth commands show how to bind and start the `ldg1` domain, at which time you can use the `ldg1` domain.

```
primary# ldm create ldg1
primary# ldm set-core 2 ldg1
...
primary# ldm bind ldg1
primary# ldm start ldg1
```

▼ How to Configure an Existing Domain With CPU Whole Cores

If a domain already exists and is configured to use CPU threads, you can change its configuration to use CPU whole cores.

1. **(Optional) Stop and unbind the domain.**

   This step is required only if you also set the max-cores constraint.

```
primary# ldm stop domain-name
primary# ldm unbind domain-name
```

2. **Set the number of CPU whole cores for the domain.**

```
primary# ldm set-core number-of-CPU-cores domain
```

3. **(Optional) Set the `max-cores` property for the domain.**

```
primary# ldm set-domain max-cores=max-number-of-CPU-cores domain
```

4. **(Optional) Rebind and restart the domain.**

   This step is required only if you also set the max-cores constraint.

```
primary# ldm bind domain-name
primary# ldm start domain-name
```

Example 14–4 Configuring an Existing Domain With Four CPU Whole Cores

This example updates the configuration of an existing domain, `ldg1` by configuring it with four CPU whole cores.

```
primary# ldm set-core 4 ldg1
```

▼ How to Configure the Primary Domain With CPU Whole Cores

If the `primary` domain is configured to use CPU threads, you can change its configuration to use CPU whole cores.

1. **(Optional) Place the `primary` domain in delayed reconfiguration mode.**

   You need to initiate a delayed reconfiguration only if you want to modify the `max-cores` property.

   ```
   primary# ldm start-reconf primary
   ```

2. **Set the number of CPU whole cores for the `primary` domain.**

   ```
   primary# ldm set-core number-of-CPU-cores primary
   ```

3. **(Optional) Set the `max-cores` property for the `primary` domain.**

   ```
   primary# ldm set-domain max-cores=max-number-of-CPU-cores primary
   ```

4. **(Optional) Reboot the `primary` domain.**

   Use the appropriate procedure to reboot the `primary` domain depending on the system configuration. See "Rebooting the Root Domain With PCIe Endpoints Configured" on page 149.

   You need to reboot the domain only if you want to modify the `max-cores` property.

Example 14–5 Configuring the Control Domain With Two CPU Whole Cores

This example configures CPU whole cores on the `primary` domain. The first command initiates delayed reconfiguration mode on the `primary` domain. The second command configures the `primary` domain with two CPU whole cores. The third command sets the `max-cores` property to 2, and the fourth command reboots the `primary` domain.

```
primary# ldm start-reconf primary
primary# ldm set-core 2 primary
primary# ldm set-domain max-cores=2 primary
primary# shutdown -i 5
```

The optional Steps 1 and 4 are required only if you want to modify the `max-cores` property.

Interaction of Hard Partitioned Systems With Other Oracle VM Server for SPARC Features

This section describes how hard partitioned systems interact with other Oracle VM Server for SPARC features.

CPU Dynamic Reconfiguration

You can use CPU dynamic reconfiguration with domains that are configured with CPU whole cores. However, you can add or remove only entire CPU cores, not individual CPU threads. The hard partitioning state of the system is maintained by the CPU dynamic reconfiguration feature. In addition, if CPU cores are dynamically added to a domain, the maximum is enforced. Therefore, the CPU DR command would fail if it attempted to exceed the maximum number of CPUs.

Note – The `max-cores` property cannot be altered unless the domain is stopped and unbound. So, to increase the maximum number of cores from the value specified at the time the whole-core constraint was set, you must first stop and unbind the domain.

Use the following commands to dynamically add to or remove CPU whole cores from a bound or active domain and to dynamically set the number of CPU whole cores for a bound or active domain:

`ldm add-core` *number-of-CPU-cores domain*

`ldm rm-core` *number-of-CPU-cores domain*

`ldm set-core` *number-of-CPU-cores domain*

Note – If the domain is not active, these commands also adjust the maximum number of CPU cores for the domain. If the domain is bound or active, these commands do not affect the maximum number of CPU cores for the domain.

EXAMPLE 14–6 Dynamically Adding Two CPU Whole Cores to a Domain

This example shows how to dynamically add two CPU whole cores to the `ldg1` domain. The `ldg1` domain is an active domain that has been configured with CPU whole cores. The first command shows that the `ldg1` domain is active. The second command shows that the `ldg1` domain is configured with CPU whole cores and a maximum of four CPU cores. The third and fifth commands show the CPU cores that are assigned to the domain before and after the addition of two CPU whole cores. The fourth command dynamically adds two CPU whole cores to the `ldg1` domain.

EXAMPLE 14–6 Dynamically Adding Two CPU Whole Cores to a Domain *(Continued)*

```
primary# ldm list ldg1
NAME    STATE   FLAGS   CONS  VCPU  MEMORY UTIL  UPTIME
ldg1    active  -n----  5000  16    2G     0.4%  5d 17h 49m
primary# ldm list -o resmgmt ldg1
NAME
ldg1

CONSTRAINT
   whole-core
        max-cores=4
primary# ldm list -o core ldg1
NAME
ldg1

CORE
CID    PCPUSET
1    (8, 9, 10, 11, 12, 13, 14, 15)
2    (16, 17, 18, 19, 20, 21, 22, 23)
primary# ldm add-core 2 ldg1
primary# ldm list -o core ldg1
NAME
ldg1

CORE
CID    PCPUSET
1    (8, 9, 10, 11, 12, 13, 14, 15)
2    (16, 17, 18, 19, 20, 21, 22, 23)
3    (24, 25, 26, 27, 28, 29, 30, 31)
4    (32, 33, 34, 35, 36, 37, 38, 39)
```

CPU Dynamic Resource Management

Dynamic resource management (DRM) can be used to automatically manage CPU resources on some domains.

Power Management

You can set a separate power management (PM) policy for each hard-partitioned domain.

Domain Reboot or Rebind

A domain that is configured with CPU whole cores remains configured with CPU whole cores when the domain is restarted, or if the entire system is restarted. A domain uses the same physical CPU cores for the entire time it remains bound. For example, if a domain is rebooted, it uses the same physical CPU cores both before and after the reboot. Or, if the entire system is powered off while a domain is bound, that domain will be configured with the same physical CPU cores when the system is powered on again. If you unbind a domain and then rebind it, or if the entire system is restarted with a new configuration, the domain might use different physical CPU cores.

Assigning Physical Resources to Domains

The Logical Domains Manager automatically selects the physical resources to be assigned to a domain. The Oracle VM Server for SPARC 3.3 software also enables expert administrators to explicitly choose the physical resources to assign to or remove from a domain.

Resources that you explicitly assign are called *named resources*. Resources that are automatically assigned are called *anonymous resources*.

Caution – Do *not* assign named resources unless you are an expert administrator.

You can explicitly assign physical resources to the control domain and to guest domains. Because the control domain remains active, the control domain might optionally be in a delayed reconfiguration before you make physical resource assignments. Or, a delayed reconfiguration is automatically triggered when you make physical assignments. See "Managing Physical Resources on the Control Domain" on page 321. For information about physical resource restrictions, see "Restrictions for Managing Physical Resources on Domains" on page 321.

You can explicitly assign the following physical resources to the control domain and to guest domains:

- **Physical CPUs.** Assign the physical core IDs to the domain by setting the `cid` property.

  The `cid` property should be used *only* by an administrator who is knowledgeable about the topology of the system to be configured. This advanced configuration feature enforces specific allocation rules and might affect the overall performance of the system.

  You can set this property by running any of the following commands:

  `ldm add-core cid=`*core-ID*`[,`*core-ID*`[,...]]` *domain-name*

  `ldm set-core cid=`*core-ID*`[,`*core-ID*`[,...]]` *domain-name*

  `ldm rm-core [-f] cid=`*core-ID*`[,`*core-ID*`[,...]]` *domain-name*

  If you specify a core ID as the value of the `cid` property, *core-ID* is explicitly assigned to or removed from the domain.

  Note – You cannot use the `ldm add-core` command to add named core resources to a domain that already uses anonymous core resources.

- **Physical memory.** Assign a set of contiguous physical memory regions to a domain by setting the `mblock` property. Each physical memory region is specified as a physical memory start address and a size.

The `mblock` property should be used *only* by an administrator who is knowledgeable about the topology of the system to be configured. This advanced configuration feature enforces specific allocation rules and might affect the overall performance of the system.

You can set this property by running any of the following commands:

`ldm add-mem mblock=`*PA-start*`:`*size*`[,`*PA-start*`:`*size*`[,...]]` *domain-name*

`ldm set-mem mblock=`*PA-start*`:`*size*`[,`*PA-start*`:`*size*`[,...]]` *domain-name*

`ldm rm-mem mblock=`*PA-start*`:`*size*`[,`*PA-start*`:`*size*`[,...]]` *domain-name*

To assign a memory block to or remove it from a domain, set the `mblock` property. A valid value includes a physical memory starting address (*PA-start*) and a memory block size (*size*), separated by a colon (`:`).

Note – You cannot use dynamic reconfiguration (DR) to move memory or core resources between running domains when you set the `mblock` or `cid` property. To move resources between domains, ensure that the domains are in a bound or inactive state. For information about managing physical resources on the control domain, see "Managing Physical Resources on the Control Domain" on page 321.

Note – If you migrate a domain, any named resources that you assigned by using the `cid` and `mblock` properties are dropped. Instead, the domain uses anonymous resources on the target system.

You can use the `ldm list-constraints` command to view the resource constraints for domains. The `physical-bindings` constraint specifies which resource types have been physically assigned to a domain. When a domain is created, the `physical-bindings` constraint is unset until a physical resource is assigned to that domain.

The `physical-bindings` constraint is set to particular values in the following cases:

- `memory` when the `mblock` property is specified
- `core` when the `cid` property is specified
- `core,memory` when both the `cid` and `mblock` properties are specified

▼ How to Remove the physical-bindings Constraint

To remove the `physical-bindings` constraint for a guest domain, you must first remove all physically bound resources.

1 Unbind the domain.

```
primary# ldm unbind domain-name
```

2 Remove the named resources.

- To remove named cores:

  ```
  primary# ldm set-core cid= domain-name
  ```
- To remove named memory:

  ```
  primary# ldm set-mem mblock= domain-name
  ```

3 Add CPU or memory resources.

- To add a CPU resource:

  ```
  primary# ldm add-vcpu number domain-name
  ```
- To add a memory resource:

  ```
  primary# ldm add-mem size[unit] domain-name
  ```

4 Rebind the domain.

```
primary# ldm bind domain-name
```

How to Remove All Non-Physically Bound Resources

To constrain guest domains that do not have the `physical-bindings` constraint, you must first remove all non-physically bound resources.

1 Unbind the domain.

```
primary# ldm unbind domain-name
```

2 Set the number of resources to `0`.

- To set the CPU resource:

  ```
  primary# ldm set-core 0 domain-name
  ```
- To set the memory resource:

  ```
  primary# ldm set-mem 0 domain-name
  ```

3 Add CPU or memory resources that are physically bound.

- To add a CPU resource:

  ```
  primary# ldm add-core cid=core-ID domain-name
  ```
- To add a memory resource:

  ```
  primary# ldm add-mem mblock=PA-start:size domain-name
  ```

4 Rebind the domain.

```
primary# ldm bind domain-name
```

Managing Physical Resources on the Control Domain

To constrain or remove the `physical-bindings` constraint from the control domain, follow the appropriate steps in the previous section. However, instead of unbinding the domain, place the control domain in a delayed reconfiguration.

A change of constraint between anonymous resources and physically bound named resources auto-triggers a delayed reconfiguration. You can still explicitly enter a delayed reconfiguration by using the `ldm start-reconf primary` command.

As with any delayed reconfiguration change, you must perform a reboot of the domain, in this case the control domain, to complete the process.

Note – When the control domain is in delayed reconfiguration mode, you can perform unlimited memory assignments by using the `ldm add-mem` and `ldm rm-mem` commands on the control domain. However, you can perform only *one* core assignment to the control domain by using the `ldm set-core` command.

Restrictions for Managing Physical Resources on Domains

The following restrictions apply to the assignment of physical resources:

- You cannot make physical and non-physical memory bindings, or physical and non-physical core bindings, in the same domain.
- You can have non-physical memory and physical core bindings, or non-physical core and physical memory bindings, in the same domain.
- When you add a physical resource to a domain, the corresponding resource type becomes constrained as a physical binding.
- Attempts to add anonymous CPUs to or remove them from a domain where `physical-bindings=core` will fail.
- For unbound resources, the allocation and checking of the resources can occur *only* when you run the `ldm bind` command.
- When removing physical memory from a domain, you must remove the *exact* physical memory block that was previously added.
- Physical memory ranges must *not* overlap.
- You can use only the `ldm add-core cid=` or `ldm set-core cid=` command to assign a physical resource to a domain.
- If you use the `ldm add-mem mblock=` or `ldm set-mem mblock=` command to assign multiple physical memory blocks, the addresses and sizes are checked immediately for collisions with other bindings.

- A domain that has partial cores assigned to it can use the whole-core semantics if the remaining CPUs of those cores are free and available.

Using Memory Dynamic Reconfiguration

Memory dynamic reconfiguration (DR) is capacity-based and enables you to add an arbitrary amount of memory to or remove it from an active logical domain.

The requirements and restrictions for using the memory DR feature are as follows:

- You can perform memory DR operations on any domain. However, only a single memory DR operation can be in progress on a domain at a given time.
- The memory DR feature enforces 256-Mbyte alignment on the address and size of the memory involved in a given operation. See "Memory Alignment" on page 324.
- 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 325.

If the memory of a domain cannot be reconfigured by using a memory DR operation, the domain must be stopped before the memory can be reconfigured. If the domain is the control domain, you must first initiate a delayed reconfiguration.

Under certain circumstances, the Logical Domains Manager rounds up the requested memory allocation to either the next largest 8-Kbyte or 4-Mbyte multiple. The following example shows sample output of the `ldm list-domain -l` command, where the constraint value is smaller than the actual allocated size:

```
Memory:
         Constraints: 1965 M
         raddr          paddr5          size
         0x1000000      0x291000000     1968M
```

Adding Memory

If a domain is active, you can use the `ldm add-memory` command to dynamically add memory to the domain. The `ldm set-memory` command can also dynamically add memory if the specified memory size is greater than the current memory size of the domain.

Removing Memory

If a domain is active, you can use the `ldm remove-memory` command to dynamically remove memory from the domain. The `ldm set-memory` command can also dynamically remove memory if the specified memory size is smaller than the current memory size of the domain.

Memory removal can be a long-running operation. You can track the progress of an `ldm remove-memory` command by running the `ldm list -l` command for the specified domain.

You can cancel a removal request that is in progress by interrupting the `ldm remove-memory` command (by pressing Control-C) or by issuing the `ldm cancel-operation memdr` command. If you cancel a memory removal request, only the outstanding portion of the removal request is affected; namely, the amount of memory still to be removed from the domain.

Partial Memory DR Requests

A request to dynamically add memory to or remove memory from a domain might only be partially fulfilled. This result depends on the availability of suitable memory to add or remove, respectively.

Note – Memory is cleared after it is removed from a domain and before it is added to another domain.

Memory Reconfiguration of the Control Domain

You can use the memory DR feature to reconfigure the memory of the control domain. If a memory DR request cannot be performed on the control domain, you must first initiate a delayed reconfiguration.

Using memory DR might not be appropriate for removing large amounts of memory from an active domain because memory DR operations might be long running. In particular, during the initial configuration of the system, you should use delayed reconfiguration to decrease the memory in the control domain.

Decrease the Control Domain's Memory

Use a delayed reconfiguration instead of a memory DR to decrease the control domain's memory from an initial factory default configuration. In such a case, the control domain owns all of the host system's memory. The memory DR feature is not well suited for this purpose because an active domain is not guaranteed to add, or more typically give up, all of the requested memory. Rather, the OS running in that domain makes a best effort to fulfill the request. In addition, memory removal can be a long-running operation. These issues are amplified when large memory operations are involved, as is the case for the initial decrease of the control domain's memory.

For these reasons, use a delayed reconfiguration by following these steps:

1. Use the `ldm start-reconf primary` command to put the control domain in delayed reconfiguration mode.
2. Partition the host system's resources that are owned by the control domain, as necessary.
3. Use the `ldm cancel-reconf` command to undo the operations in Step 2, if necessary, and start over.

4. Reboot the control domain to make the reconfiguration changes take effect.

Dynamic and Delayed Reconfiguration

If a delayed reconfiguration is pending in the control domain, a memory reconfiguration request is rejected for any other domain. If a delayed reconfiguration is not pending in the control domain, a memory reconfiguration request is rejected for any domain that does not support memory DR. For those domains, the request is converted to a delayed reconfiguration request.

Memory Alignment

Memory reconfiguration requests have different alignment requirements that depend on the state of the domain to which the request is applied.

Memory Alignment for Active Domains

- **Dynamic addition and removal.** The address and size of a memory block are 256-Mbyte-aligned for dynamic addition and dynamic removal. The minimum operation size is 256 Mbytes.

  A nonaligned request or a removal request that is larger than the bound size is rejected.

  Use the following commands to adjust memory allocations:

  - `ldm add-memory`**.** If you specify the `--auto-adj` option with this command, the amount of memory to be added is 256-Mbyte-aligned, which might increase the amount of memory actually added to the domain.
  - `ldm remove-memory`**.** If you specify the `--auto-adj` option with this command, the amount of memory to be removed is 256-Mbyte-aligned, which might decrease the amount of memory actually removed from the domain.
  - `ldm set-memory`**.** This command is treated as an addition or a removal operation. If you specify the `--auto-adj` option, the amount of memory to be added or removed is 256-Mbyte-aligned as previously described. Note that this alignment might increase the resulting memory size of the domain.
- **Delayed reconfiguration.** The address and size of a memory block are 4-Mbyte-aligned. If you make a nonaligned request, the request is rounded up to be 4-Mbyte-aligned.

Memory Alignment for Bound Domains

The address and size of a memory block are 4-Mbyte-aligned for bound domains. If you make a nonaligned request, the request is rounded up to be 4-Mbyte-aligned. Therefore, the resulting memory size of the domain might be more than you specified.

For the `ldm add-memory`, `ldm set-memory`, and `ldm remove-memory` commands, the `--auto-adj` option rounds up the size of the resulting memory to be 256-Mbyte-aligned. Therefore, the resulting memory might be more than you specified.

Memory Alignment for Inactive Domains

For the `ldm add-memory`, `ldm set-memory`, and `ldm remove-memory` commands, the `--auto-adj` option rounds up the size of the resulting memory to be 256-Mbyte-aligned. There is no alignment requirement for an inactive domain. The restrictions described in "Memory Alignment for Bound Domains" on page 324 take effect after such a domain is bound.

Adding Unaligned Memory

The memory DR feature enforces 256-Mbyte memory alignment on the address and size of the memory that is dynamically added to or removed from an active domain. Therefore, any unaligned memory in an active domain cannot be removed by using memory DR.

Also, any unaligned memory in the free memory pool cannot be added to an active domain by using memory DR.

After all the aligned memory has been allocated, you can use the `ldm add-memory` command to add the remaining unaligned memory to a bound or inactive domain. You can also use this command to add the remaining unaligned memory to the control domain by means of a delayed reconfiguration operation.

The following example shows how to add the two remaining 128-Mbyte memory blocks to the `primary` and `ldom1` domains. The `ldom1` domain is in the bound state.

The following command initiates a delayed reconfiguration operation on the control domain.

```
primary# ldm start-reconf primary
Initiating a delayed reconfiguration operation on the primary domain.
All configuration changes for other domains are disabled until the
primary domain reboots, at which time the new configuration for the
primary domain also takes effect.
```

The following command adds one of the 128-Mbyte memory blocks to the control domain.

```
primary# ldm add-memory 128M primary
------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
------------------------------------------------------------------------------

primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -ndcv-  SP      8     2688M    0.1%  23d 8h 8m

primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
```

```
primary          active     -n-cv-  SP      8     2560M    0.5%  23d 8h 9m
ldom1            bound      ------  5000    1     524M
```

The following command adds the other 128-Mbyte memory block to the `ldom1` domain.

```
primary# ldm add-mem 128M ldom1
primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      8     2560M    0.1%  23d 8h 9m
ldom1            bound      ------  5000    1     652M
```

Memory DR Examples

The following examples show how to perform memory DR operations. For information about the related CLI commands, see the `ldm(1M)` man page.

EXAMPLE 14–7 Memory DR Operations on Active Domains

This example shows how to dynamically add memory to and remove it from an active domain, `ldom1`.

The `ldm list` output shows the memory for each domain in the Memory field.

```
primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.4%  1d 22h 53m
ldom1            active     -n----  5000    2     2G       0.4%  1d 1h 23m
ldom2            bound      ------  5001    2     200M
```

The following `ldm add-mem` command exits with an error because you must specify memory in multiples of 256 Mbytes. The next `ldm add-mem` command uses the `--auto-adj` option so that even though you specify `200M` as the amount of memory to add, the amount is rounded up to 256 Mbytes.

```
primary# ldm add-mem 200M ldom1
The size of memory must be a multiple of 256MB.

primary# ldm add-mem --auto-adj 200M ldom1
Adjusting request size to 256M.
The ldom1 domain has been allocated 56M more memory
than requested because of memory alignment constraints.

primary# ldm list
NAME            STATE   FLAGS   CONS    VCPU    MEMORY  UTIL    UPTIME
primary         active  -n-cv-  SP      4       27392M  5.0%    8m
ldom1           active  -n----  5000    2       2304M   0.5%    1m
ldom2           bound   ------  5001    2       200M
```

The `ldm rm-mem` command exits with an error because you must specify memory in multiples of 256 Mbytes. When you add the `--auto-adj` option to the same command, the memory removal succeeds because the amount of memory is rounded down to the next 256-Mbyte boundary.

EXAMPLE 14–7 Memory DR Operations on Active Domains *(Continued)*

```
primary# ldm rm-mem --auto-adj 300M ldom1
Adjusting requested size to 256M.
The ldom1 domain has been allocated 44M more memory
than requested because of memory alignment constraints.

primary# ldm list
NAME             STATE   FLAGS   CONS    VCPU    MEMORY  UTIL    UPTIME
primary          active  -n-cv-  SP      4       27392M  0.3%    8m
ldom1            active  -n----  5000    2       2G      0.2%    2m
ldom2            bound   ------  5001    2       200M
```

EXAMPLE 14–8 Memory DR Operations on Bound Domains

This example shows how to add memory to and remove it from a bound domain, `ldom2`.

The `ldm list` output shows the memory for each domain in the Memory field. The first `ldm add-mem` command adds 100 Mbytes of memory to the `ldom2` domain. The next `ldm add-mem` command specifies the `--auto-adj` option, which causes an additional 112 Mbytes of memory to be dynamically added to `ldom2`.

The `ldm rm-mem` command dynamically removes 100 Mbytes from the `ldom2` domain. If you specify the `--auto-adj` option to the same command to remove 300 Mbytes of memory, the amount of memory is rounded down to the next 256-Mbyte boundary.

```
primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.4%  1d 22h 53m
ldom1            active     -n----  5000    2     2G       0.4%  1d 1h 23m
ldom2            bound      ------  5001    2     200M

primary# ldm add-mem 100M ldom2

primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.5%  1d 22h 54m
ldom1            active     -n----  5000    2     2G       0.2%  1d 1h 25m
ldom2            bound      ------  5001    2     300M

primary# ldm add-mem --auto-adj 100M ldom2
Adjusting request size to 256M.
The ldom2 domain has been allocated 112M more memory
than requested because of memory alignment constraints.

primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.4%  1d 22h 55m
ldom1            active     -n----  5000    2     2G       0.5%  1d 1h 25m
ldom2            bound      ------  5001    2     512M

primary# ldm rm-mem 100M ldom2
primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   3.3%  1d 22h 55m
```

EXAMPLE 14–8 Memory DR Operations on Bound Domains *(Continued)*

```
ldom1            active     -n----  5000    2     2G       0.2%  1d 1h 25m
ldom2            bound      ------  5001    2     412M

primary# ldm rm-mem --auto-adj 300M ldom2
Adjusting request size to 256M.
The ldom2 domain has been allocated 144M more memory
than requested because of memory alignment constraints.

primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.5%  1d 22h 55m
ldom1            active     -n----  5000    2     2G       0.2%  1d 1h 26m
ldom2            bound      ------  5001    2     256M
```

EXAMPLE 14–9 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.

```
primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.5%  1d 22h 55m
ldom1            active     -n----  5000    2     2G       0.2%  1d 1h 26m
ldom2            bound      ------  5001    2     256M
```

The following `ldm set-mem` command attempts to set the `primary` domain's size to 3400 Mbytes. The resulting error states that the specified value is not on a 256-Mbyte boundary. Adding the `--auto-adj` option to the same command enables you to successfully remove some memory and stay on the 256-Mbyte boundary. This command also issues a warning to state that not all of the requested memory could be removed as the domain is using that memory.

```
primary# ldm set-mem 3400M primary
An ldm set-mem 3400M command would remove 23992MB, which is not a multiple
of 256MB. Instead, run ldm rm-mem 23808MB to ensure a 256MB alignment.

primary# ldm set-mem --auto-adj 3400M primary
Adjusting request size to 3.4G.
The primary domain has been allocated 184M more memory
than requested because of memory alignment constraints.
Only 9472M of memory could be removed from the primary domain
because the rest of the memory is in use.
```

The next `ldm set-mem` command sets the memory size of the `ldom2` domain, which is in the bound state, to 690 Mbytes. If you add the `--auto-adj` option to the same command, an additional 78 Mbytes of memory is dynamically added to `ldom2` to stay on a 256-Mbyte boundary.

```
primary# ldm set-mem 690M ldom2
primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
```

EXAMPLE 14–9 Setting Domain Memory Sizes *(Continued)*

```
primary          active     -n-cv-  SP      4     17920M   0.5%  1d 22h 56m
ldom1            active     -n----  5000    2     2G       0.6%  1d 1h 27m
ldom2            bound      ------  5001    2     690M

primary# ldm set-mem --auto-adj 690M ldom2
Adjusting request size to 256M.
The ldom2 domain has been allocated 78M more memory
than requested because of memory alignment constraints.

primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     17920M   2.1%  1d 22h 57m
ldom1            active     -n----  5000    2     2G       0.2%  1d 1h 27m
ldom2            bound      ------  5001    2     768M
```

Using Resource Groups

A *resource group* provides an alternate way to view the resources in a system. Resources are grouped based on the underlying physical relationships between processor cores, memory, and I/O buses. Different platforms, and even different platform configurations within the same server family, such as SPARC T5-2 and SPARC T5-8, can have different resource groups that reflect the differences in the hardware. Use the `ldm list-resrc-group` command to view resource group information.

The membership of resource groups is statically defined by the hardware configuration. You can use the `ldm remove-core` and `ldm remove-memory` commands to operate on resources from a particular resource group.

- The `remove-core` subcommand specifies the number of CPU cores to remove from a domain. When you specify a resource group by using the `-g` option, the cores that are selected for removal all come from that resource group.
- The `remove-memory` subcommand removes the specified amount of memory from a logical domain. When you specify a resource group by using the `-g` option, the memory that is selected for removal all comes from that resource group.

For information about these commands, see the `ldm`(1M) man page.

For examples, see "Listing Resource Group Information" on page 338.

Resource Group Requirements and Restrictions

The resource group feature is available only on SPARC T5 servers, SPARC T7 series servers, SPARC M5 servers, SPARC M6 servers, SPARC M7 series servers, and Fujitsu M10 servers.

The resource group feature has the following restrictions:

- It is not available on UltraSPARC T2, UltraSPARC T2 Plus, SPARC T3, and SPARC T4 platforms.
- The `ldm list-rsrc-group` command does not show any information about those unsupported platforms and the `-g` variants of the `ldm remove-core` and `ldm remove-memory` commands are not functional.
- On supported platforms, specifying `_sys_` in place of *domain-name* moves all system memory to free memory in a different resource group. This command is a no-op on unsupported platforms.

Using Power Management

To enable power management (PM), you first need to set the PM policy in at least version 3.0 of the ILOM firmware. This section summarizes the information that you need to be able to use PM with the Oracle VM Server for SPARC software.

For more information about PM features and ILOM features, see the following:

- Chapter 20, "Using Power Management"
- "Monitoring Power Consumption" in the *Oracle Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide*
- *Oracle Integrated Lights Out Manager (ILOM) 3.0 Feature Updates and Release Notes*

Using Dynamic Resource Management

You can use policies to determine how to automatically perform DR activities. At this time, you can create policies *only* to govern the dynamic resource management of virtual CPUs.

Caution – The following restrictions affect CPU dynamic resource management (DRM):

- On UltraSPARC T2 and UltraSPARC T2 Plus platforms, DRM cannot be enabled when the PM elastic policy is set.
- On UltraSPARC T2 and UltraSPARC T2 Plus platforms, any change from the performance policy to the elastic policy is delayed while DRM is enabled.
- Ensure that you disable CPU DRM prior to performing a domain migration operation, or you will see an error message.
- When the PM elastic policy is set, you can use DRM only when the firmware supports normalized utilization (8.2.0).

A *resource management policy* specifies the conditions under which virtual CPUs can be automatically added to and removed from a logical domain. A policy is managed by using the `ldm add-policy`, `ldm set-policy`, and `ldm remove-policy` commands:

```
ldm add-policy [enable=yes|no] [priority=value] [attack=value] [decay=value]
  [elastic-margin=value] [sample-rate=value] [tod-begin=hh:mm[:ss]]
  [tod-end=hh:mm[:ss]] [util-lower=percent] [util-upper=percent] [vcpu-min=value]
  [vcpu-max=value] name=policy-name  domain-name...
ldm set-policy [enable=[yes|no]] [priority=[value]] [attack=[value]] [decay=[value]]
  [elastic-margin=[value]] [sample-rate=[value]] [tod-begin=[hh:mm:ss]]
  [tod-end=[hh:mm:ss]] [util-lower=[percent]] [util-upper=[percent]] [vcpu-min=[value]]
  [vcpu-max=[value]] name=policy-name  domain-name...
ldm remove-policy [name=]policy-name...  domain-name
```

For information about these commands and about creating resource management policies, see the `ldm(1M)` man page.

A policy is in effect during the times specified by the `tod-begin` and `tod-end` properties. The time specified by `tod-begin` must be earlier than the time specified by `tod-end` in a 24-hour period. By default, values for the `tod-begin` and `tod-end` properties are 00:00:00 and 23:59:59, respectively. When the default values are used, the policy is always in effect.

The policy uses the value of the `priority` property to specify a priority for a dynamic resource management (DRM) policy. Priority values are used to determine the relationship between DRM policies on a single domain and between DRM-enabled domains on a single system. Lower numerical values represent higher (better) priorities. Valid values are between 1 and 9999. The default value is 99.

The behavior of the `priority` property depends on the availability of a pool of free CPU resources, as follows:

- **Free CPU resources are available in the pool.** In this case, the `priority` property determines which DRM policy will be in effect when more than one overlapping policy is defined for a single domain.
- **No free CPU resources are available in the pool.** In this case, the `priority` property specifies whether a resource can be dynamically moved from a lower-priority domain to a higher-priority domain on the same system. The priority of a domain is the priority specified by the DRM policy that is in effect for that domain.

  For example, a higher-priority domain can acquire CPU resources from another domain that has a DRM policy with a lower priority. This resource-acquisition capability pertains only to domains that have DRM policies enabled. Domains that have equal `priority` values are unaffected by this capability. So, if the default priority is used for all policies, domains cannot obtain resources from lower-priority domains. To take advantage of this capability, adjust the `priority` property values so that they have unequal values.

For example, the `ldg1` and `ldg2` domains both have DRM policies in effect. The `priority` property for the `ldg1` domain is `1`, which is more favorable than the `priority` property value of the `ldg2` domain (`2`). The `ldg1` domain can dynamically remove a CPU resource from the `ldg2` domain and assign it to itself in the following circumstances:

- The `ldg1` domain requires another CPU resource.
- The pool of free CPU resources has been exhausted.

The policy uses the `util-high` and `util-low` property values to specify the high and low thresholds for CPU utilization. If the utilization exceeds the value of `util-high`, virtual CPUs are added to the domain until the number is between the `vcpu-min` and `vcpu-max` values. If the utilization drops below the `util-low` value, virtual CPUs are removed from the domain until the number is between the `vcpu-min` and `vcpu-max` values. If `vcpu-min` is reached, no more virtual CPUs can be dynamically removed. If the `vcpu-max` is reached, no more virtual CPUs can be dynamically added.

EXAMPLE 14–10 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.

EXAMPLE 14–10 Adding Resource Management Policies *(Continued)*

- Uses the default values for those properties that are not specified, such as `enable` and `sample-rate`. See the `ldm(1M)` man page.

```
primary# ldm add-policy tod-begin=09:00 tod-end=18:00 util-lower=25 util-upper=75 \
vcpu-min=2 vcpu-max=16 attack=1 decay=1 priority=1 name=high-usage ldom1
```

The following `ldm add-policy` command creates the `med-usage` policy to be used during the low utilization period on the `ldom1` domain.

The following `med-usage` policy does the following:

- Specifies that the beginning and ending times are 6:00 p.m. and 9:00 a.m. by setting the `tod-begin` and `tod-end` properties, respectively.
- Specifies that the lower and upper limits at which to perform policy analysis are 10 percent and 50 percent by setting the `util-lower` and `util-upper` properties, respectively.
- Specifies that the minimum and maximum number of virtual CPUs is 2 and 16 by setting the `vcpu-min` and `vcpu-max` properties, respectively.
- Specifies that the maximum number of virtual CPUs to be added during any one resource control cycle is 1 by setting the `attack` property.
- Specifies that the maximum number of virtual CPUs to be removed during any one resource control cycle is 1 by setting the `decay` property.
- Specifies that the priority of this policy is 1 by setting the `priority` property. A priority of 1 means that this policy will be enforced even if another policy can take effect.
- Specifies that the name of the policy file is `high-usage` by setting the `name` property.
- Uses the default values for those properties that are not specified, such as `enable` and `sample-rate`. See the `ldm(1M)` man page.

```
primary# ldm add-policy tod-begin=18:00 tod-end=09:00 util-lower=10 util-upper=50 \
 vcpu-min=2 vcpu-max=16 attack=1 decay=1 priority=1 name=med-usage ldom1
```

Listing Domain Resources

This section shows the syntax usage for the `ldm` subcommands, defines some output terms, such as flags and utilization statistics, and provides examples that are similar to what appears as output.

Machine-Readable Output

If you are creating scripts that use `ldm list` command output, always use the `-p` option to produce the machine-readable form of the output.

To view syntax usage for all `ldm` subcommands, use the following command:

```
primary# ldm --help
```

For more information about the `ldm` subcommands, see the `ldm(1M)` man page.

Flag Definitions

The following flags can be shown in the output for a domain (`ldm list`). If you use the long, parseable options (`-l -p`) for the command, the flags are spelled out for example, `flags=normal,control,vio-service`. If not, you see the letter abbreviation, for example `-n-cv-`. The list flag values are position dependent. The following values can appear in each of the six columns from left to right.

Column 1 – Starting or stopping domains

- `s` – Starting or stopping

Column 2 – Domain status

- `n` – Normal
- `t` – Transition
- `d` – Degraded domain that cannot be started due to missing resources

Column 3 – Reconfiguration status

- `d` – Delayed reconfiguration
- `r` – Memory dynamic reconfiguration

Column 4 – Control domain

- `c` – Control domain

Column 5 – Service domain

- `v` – Virtual I/O service domain

Column 6 – Migration status

- `s` – Source domain in a migration
- `t` – Target domain in a migration
- `e` – Error occurred during a migration

Utilization Statistic Definition

The per virtual CPU utilization statistic (`UTIL`) is shown through the long (`-l`) option of the `ldm list` command. The statistic is the percentage of time that the virtual CPU spent executing on behalf of the guest operating system. A virtual CPU is considered to be executing on behalf of

the guest operating system except when it has been yielded to the hypervisor. If the guest operating system does not yield virtual CPUs to the hypervisor, the utilization of CPUs in the guest operating system will always show as `100%`.

The utilization statistic reported for a logical domain is the average of the virtual CPU utilizations for the virtual CPUs in the domain. The normalized utilization statistic (`NORM`) is the percentage of time the virtual CPU spends executing on behalf of the guest OS. This value takes into account such operations as cycle skip. Normalized virtualization is only available when your system runs at least version 8.2.0 of the system firmware.

When PM does not perform cycle skip operations, 100% utilization equals 100% normalized utilization. When PM adjusts the cycle skip to four eighths, 100% utilization equals 50% utilization, which means that the CPU effectively has only half the possible number of cycles available. So, a fully utilized CPU has a 50% normalized utilization. Use the `ldm list` or `ldm list -l` command to show normalized utilization for both virtual CPUs and the guest OS.

Viewing Various Lists

- To view the current software versions installed:

  ```
  primary# ldm -V
  ```
- To generate a short list for all domains:

  ```
  primary# ldm list
  ```
- To generate a long list for all domains:

  ```
  primary# ldm list -l
  ```
- To generate an extended list of all domains:

  ```
  primary# ldm list -e
  ```
- To generate a parseable, machine-readable list of all domains:

  ```
  primary# ldm list -p
  ```
- You can generate output as a subset of resources by entering one or more of the following *format* options. If you specify more than one format, delimit the items by a comma with no spaces.

  ```
  primary# ldm list -o resource[,resource...] domain-name
  ```
  - `console` – Output contains virtual console (`vcons`) and virtual console concentrator (`vcc`) service
  - `core` – Output contains information about domains that have whole cores allocated
  - `cpu` – Output contains information about the virtual CPU (`vcpu`), physical CPU (`pcpu`), and core ID
  - `crypto` – Cryptographic unit output contains Modular Arithmetic Unit (`mau`) and any other supported cryptographic unit, such as the Control Word Queue (CWQ)

- `disk` – Output contains virtual disk (`vdisk`) and virtual disk server (`vds`)
- `domain-name` – Output contains variables (`var`), host ID (`hostid`), domain state, flags, UUID, and software state
- `memory` – Output contains `memory`
- `network` – Output contains media access control (`mac`) address , virtual network switch (`vsw`), and virtual network (`vnet`) device
- `physio` – Physical input/output contains peripheral component interconnect (`pci`) and network interface unit (`niu`)
- `resmgmt` – Output contains dynamic resource management (DRM) policy information, indicates which policy is currently running, and lists constraints related to whole-core configuration
- `serial` – Output contains virtual logical domain channel (`vldc`) service, virtual logical domain channel client (`vldcc`), virtual data plane channel client (`vdpcc`), and virtual data plane channel service (`vdpcs`)
- `stats` – Output contains statistics that are related to resource management policies
- `status` – Output contains status about a domain migration in progress

The following examples show various subsets of output that you can specify.

- To list CPU information for the control domain:

```
primary# ldm list -o cpu primary
```

- To list domain information for a guest domain:

```
primary# ldm list -o domain ldm2
```

- To list memory and network information for a guest domain:

```
primary# ldm list -o network,memory ldm1
```

- To list DRM policy information for a guest domain:

```
primary# ldm list -o resmgmt,stats ldm1
```

- To show a variable and its value for a domain:

```
primary# ldm list-variable variable-name domain-name
```

For example, the following command shows the value for the `boot-device` variable on the `ldg1` domain:

```
primary# ldm list-variable boot-device ldg1
boot-device=/virtual-devices@100/channel-devices@200/disk@0:a
```

- To list the resources that are bound to a domain:

```
primary# ldm list-bindings domain-name
```

- To list logical domain configurations that have been stored on the SP:

The `ldm list-config` command lists the logical domain configurations that are stored on the service processor. When used with the `-r` option, this command lists those configurations for which autosave files exist on the control domain.

For more information about configurations, see "Managing Domain Configurations" on page 343. For more examples, see the `ldm(1M)` man page.

```
primary# ldm list-config
factory-default
3guests
foo [next poweron]
primary
reconfig-primary
```

The labels to the right of the configuration name mean the following:

- `[current]` – Last booted configuration, only as long as it matches the currently running configuration; that is, until you initiate a reconfiguration. After the reconfiguration, the annotation changes to `[next poweron]`.
- `[next poweron]` – Configuration to be used at the next power cycle.
- `[degraded]` – Configuration is a degraded version of the previously booted configuration.

- To list all server resources, bound and unbound:

```
primary# ldm list-devices -a
```

- To list the amount of memory available to be allocated:

```
primary# ldm list-devices mem
MEMORY
    PA                   SIZE
    0x14e000000          2848M
```

- To determine which portions of memory are unavailable for logical domains:

```
primary# ldm list-devices -a mem
MEMORY
    PA                   SIZE            BOUND
    0x0                  57M             _sys_
    0x3900000            32M             _sys_
    0x5900000            94M             _sys_
    0xb700000            393M            _sys_
    0x24000000           192M            _sys_
    0x30000000           255G            primary
    0x3ff0000000         64M             _sys_
    0x3ff4000000         64M             _sys_
    0x3ff8000000         128M            _sys_
    0x80000000000        2G              ldg1
    0x80080000000        2G              ldg2
    0x80100000000        2G              ldg3
    0x80180000000        2G              ldg4
    0x80200000000        103G
    0x81bc0000000        145G            primary
```

- To list the services that are available:

```
primary# ldm list-services
```

Listing Constraints

To the Logical Domains Manager, constraints are one or more resources you want to have assigned to a particular domain. You either receive all the resources you ask to be added to a domain or you get none of them, depending upon the available resources. The `list-constraints` subcommand lists those resources you requested assigned to the domain.

- To list constraints for one domain:

  ```
  # ldm list-constraints domain-name
  ```

- To list constraints in XML format for a particular domain:

  ```
  # ldm list-constraints -x domain-name
  ```

- To list constraints for all domains in a parseable format:

  ```
  # ldm list-constraints -p
  ```

Listing Resource Group Information

You can use the `ldm list-rsrc-group` command to show information about resource groups.

The following command shows information about all the resource groups:

```
primary# ldm list-rsrc-group
NAME                        CORE MEMORY IO
/SYS/CMU4                   12   256G   4
/SYS/CMU5                   12   256G   4
/SYS/CMU6                   12   128G   4
/SYS/CMU7                   12   128G   4
```

Like the other `ldm list-*` commands, you can specify options to show parseable output, detailed output, and information about particular resource groups and domains. For more information, see the `ldm(1M)` man page.

The following example uses the `-l` option to show detailed information about the `/SYS/CMU5` resource group.

```
primary# ldm list-rsrc-group -l /SYS/CMU5
NAME                                    CORE   MEMORY    IO
/SYS/CMU5                               12     256G      4

CORE
    CID                                                 BOUND
    192, 194, 196, 198, 200, 202, 208, 210          primary
    212, 214, 216, 218                              primary

MEMORY

    PA              SIZE              BOUND
    0xc0000000000   228M              ldg1
    0xc0030000000   127G              primary
```

```
    0xc1ffc000000   64M             _sys_
    0xd0000000000   130816M         primary
    0xd1ffc000000   64M             _sys_

IO
    DEVICE          PSEUDONYM       BOUND
    pci@900         pci_24          primary
    pci@940         pci_25          primary
    pci@980         pci_26          primary
    pci@9c0         pci_27          primary
```

Using Perf-Counter Properties

The performance register access control feature enables you to get, set and unset a domain's access rights to certain groups of performance registers.

Use the `ldm add-domain` and `ldm set-domain` commands to specify a value for the `perf-counters` property. The new `perf-counters` property value will be recognized by the guest domain on the next reboot. If no `perf-counters` value is specified, the value is `htstrand`. See the `ldm(1M)` man page.

You can specify the following values for the `perf-counters` property:

`global` Grants the domain access to the global performance counters that its allocated resources can access. Only one domain at a time can have access to the global performance counters. You can specify this value alone or with either the `strand` or `htstrand` value.

`strand` Grants the domain access to the strand performance counters that exist on the CPUs that are allocated to the domain. You cannot specify this value and the `htstrand` value together.

`htstrand` Behaves the same as the `strand` value and enables instrumentation of hyperprivilege mode events on the CPUs that are allocated to the domain. You cannot specify this value and the `strand` value together.

To disable all access to any of the performance counters, specify `perf-counters=`.

If the hypervisor does not have the performance access capability, attempting to set the `perf-counters` property fails.

The `ldm list -o domain` and `ldm list -e` commands show the value of the `perf-counters` property. If the performance access capability is not supported, the `perf-counters` value is not shown in the output.

EXAMPLE 14–11 Creating a Domain and Specifying Its Performance Register Access

Create the new `ldg0` domain with access to the `global` register set:

```
primary# ldm add-domain perf-counters=global ldg0
```

EXAMPLE 14–12 Specifying the Performance Register Access for a Domain

Specify that the `ldg0` domain has access to the `global` and `strand` register sets:

```
primary# ldm set-domain perf-counters=global,strand ldg0
```

EXAMPLE 14–13 Specifying that a Domain Does Not Have Access to Any Register Sets

Specify that the `ldg0` domain does not have access to any of the register sets:

```
primary# ldm set-domain perf-counters= ldg0
```

EXAMPLE 14–14 Viewing Performance Access Information

The following examples show how to view performance access information by using the `ldm list -o domain` command.

- The following `ldm list -o domain` command shows that the `global` and `htstrand` performance values are specified on the `ldg0` domain:

```
primary# ldm list -o domain ldg0
NAME     STATE     FLAGS     UTIL
NORM
ldg0     active    -n----    0.0% 0.0%

SOFTSTATE
Solaris running

UUID
    062200af-2de2-e05f-b271-f6200fd3eee3

HOSTID
    0x84fb315d

CONTROL
    failure-policy=ignore
    extended-mapin-space=on
    cpu-arch=native
    rc-add-policy=
    shutdown-group=15
    perf-counters=global,htstrand

DEPENDENCY
    master=

PPRIORITY    4000

VARIABLES
```

EXAMPLE 14–14 Viewing Performance Access Information *(Continued)*

```
    auto-boot?=false
    boot-device=/virtual-devices@100/channel-devices@200/disk@0:a
      /virtualdevices@100/channel@200/disk@0
    network-boot-arguments=dhcp,hostname=solaris,
      file=http://10.129.241.238:5555/cgibin/wanboot-cgi
    pm_boot_policy=disabled=0;ttfc=2000;ttmr=0;
```

- The following `ldm list -p -o domain` command shows the same information as in the previous example but in the parseable form:

```
primary# ldm list -p -o domain ldg0
VERSION 1.12
DOMAIN|name=ldg0|state=active|flags=normal|util=|norm_util=
UUID|uuid=4e8749b9-281b-e2b1-d0e2-ef4dc2ce5ce6
HOSTID|hostid=0x84f97452
CONTROL|failure-policy=reset|extended-mapin-space=on|cpu-arch=native|rc-add-policy=|
shutdown-group=15|perf-counters=global,htstrand
DEPENDENCY|master=
VARIABLES
|auto-boot?=false
|boot-device=/virtual-devices@100/channel-devices@200/disk@0
|pm_boot_policy=disabled=0;ttfc=2500000;ttmr=0;
```

◆ ◆ ◆ CHAPTER 15

Managing Domain Configurations

This chapter contains information about managing domain configurations.

This chapter covers the following topics:

Managing Domain Configurations

A domain *configuration* is a complete description of all the domains and their resource allocations within a single system. You can save and store configurations on the service processor (SP) for later use.

Saving a configuration on the SP makes it persist across system power cycles. You can save several configurations and specify which configuration to boot on the next power-on attempt.

When you power up a system, the SP boots the selected configuration. The system runs the same set of domains and uses the same virtualization and partitioning resource allocations that are specified in the configuration. The default configuration is the one that is most recently saved. You can also explicitly request another configuration by using the `ldm set-spconfig` command or the appropriate ILOM command.

Caution – Always save your stable configuration to the SP and save it as XML. Saving the configuration in these ways enable you to recover your system configuration after a power failure and save it for later use. See "Saving Domain Configurations" on page 347.

A local copy of the SP configuration and the Logical Domains constraint database is saved on the control domain whenever you save a configuration to the SP. This local copy is called a *bootset*. The bootset is used to load the corresponding Logical Domains constraints database when the system undergoes a power cycle.

On the SPARC T5 server, SPARC T7 series server, SPARC M5 server, SPARC M6 server, SPARC M7 series server, and Fujitsu M10 server, the bootsets on the control domain are the master copies of the configurations. On startup, the Logical Domains Manager automatically synchronizes all configurations with the SP, which ensures that the configurations on the SP are always identical to the bootsets that are stored on the control domain.

Note – Because the bootsets contain critical system data, ensure that the control domain's file system uses technology such as disk mirroring or RAID to reduce the impact of disk failures.

A physical domain is the scope of resources that are managed by a single Oracle VM Server for SPARC instance. A physical domain might be a complete physical system as is the case of supported SPARC T-Series platforms. Or, it might be either the entire system or a subset of the system as is the case of supported SPARC M-Series platforms.

Available Configuration Recovery Methods

Oracle VM Server for SPARC supports the following configuration recovery methods:

- The autosave method, used when the configuration is not available on the SP.

  This situation might occur in one of the following circumstances:

  - The hardware that holds the saved configurations has been replaced
  - The configuration is not up to date because you neglected to save the latest configuration changes to the SP or an unexpected power cycle occurred
- The `ldm add-domain` method, used if a subset of the domains need to have their configurations restored
- The `ldm init-system` method, which should be used only as a last resort. Use this method only when both the configuration on the SP and the autosave information from the control domain are lost.

Restoring Configurations By Using Autosave

A copy of the current configuration is automatically saved on the control domain whenever the domain configuration is changed. This autosave operation does not explicitly save the configuration to the SP.

The autosave operation occurs immediately, even in the following situations:

- When the new configuration is not explicitly saved on the SP
- When the configuration change is not made until after the affected domain reboots

This autosave operation enables you to recover a configuration when the configurations that are saved on the SP are lost. This operation also enables you to recover a configuration when the current configuration was not explicitly saved to the SP after a system power cycle. In these circumstances, the Logical Domains Manager can restore that configuration on restart if it is newer than the configuration marked for the next boot.

Note – Power management, FMA, and ASR events do not cause an update to the autosave files.

You can automatically or manually restore autosave files to new or existing configurations. By default, when an autosave configuration is newer than the corresponding running configuration, a message is written to the Logical Domains log. Thus, you must use the `ldm add-spconfig -r` command to manually update an existing configuration or create a new one based on the autosave data. Note that you must perform a power cycle after using this command to complete the manual recovery.

Note – When a delayed reconfiguration is pending, the configuration changes are immediately autosaved. As a result, if you run the `ldm list-config -r` command, the autosave configuration is shown as being newer than the current configuration.

For information about how to use the `ldm *-spconfig` commands to manage configurations and to manually recover autosave files, see the `ldm(1M)` man page.

For information about how to select a configuration to boot, see "Using Oracle VM Server for SPARC With the Service Processor" on page 366. You can also use the `ldm set-spconfig` command, which is described on the `ldm(1M)` man page.

Autorecovery Policy

The autorecovery policy specifies how to handle the recovery of a configuration when one configuration that is automatically saved on the control domain is newer than the corresponding running configuration. The autorecovery policy is specified by setting the `autorecovery_policy` property of the `ldmd` SMF service. This property can have the following values:

- `autorecovery_policy=1` – Logs warning messages when an autosave configuration is newer than the corresponding running configuration. These messages are logged in the `ldmd` SMF log file. You must manually perform any configuration recovery. This is the default policy.
- `autorecovery_policy=2` – Displays a notification message if an autosave configuration is newer than the corresponding running configuration. This notification message is printed in the output of any `ldm` command the first time an `ldm` command is issued after each restart of the Logical Domains Manager. You must manually perform any configuration recovery.
- `autorecovery_policy=3` – Automatically updates the configuration if an autosave configuration is newer than the corresponding running configuration. This action overwrites the SP configuration that will be used during the next power cycle. To make this configuration available, you must perform another power cycle. This configuration is updated with the newer configuration that is saved on the control domain. This action does not affect the currently running configuration. It affects only the configuration that will be used during the next power cycle. A message is also logged that states that a newer configuration has been saved on the SP and that it will be booted at the next system power cycle. These messages are logged in the `ldmd` SMF log file.

▼ How to Modify the Autorecovery Policy

1 Log in to the control domain.

2 Become an administrator.

For Oracle Solaris 11.3, see Chapter 1, "About Using Rights to Control Users and Processes," in *Securing Users and Processes in Oracle Solaris 11.3* .

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 15–1 Modifying the Autorecovery Policy From Log to Autorecovery

The following example shows how to view the current value of the `autorecovery_policy` property and change it to a new value. The original value of this property is `1`, which means that autosave changes are logged. The `svcadm` command is used to stop and restart the `ldmd` service, and the `svccfg` command is used to view and set the property value.

```
# svccfg -s ldmd listprop ldmd/autorecovery_policy
ldmd/autorecovery_policy integer 1
# svcadm disable ldmd
# svccfg -s ldmd setprop ldmd/autorecovery_policy=3
# svcadm refresh ldmd
# svcadm enable ldmd
```

Saving Domain Configurations

You can save a domain configuration for a single domain or for all the domains on a system.

With the exception of the named physical resources, the following method does not preserve actual bindings. However, the method does preserve the constraints used to create those bindings. After saving and restoring the configuration, the domains have the same virtual resources but are not necessarily bound to the same physical resources. Named physical resources are bound as specified by the administrator.

- To save the configuration for a single domain, create an XML file containing the domain's constraints.

  ```
  # ldm list-constraints -x domain-name >domain-name.xml
  ```

  The following example shows how to create an XML file, `ldg1.xml`, which contains the `ldg1` domain's constraints:

  ```
  # ldm list-constraints -x ldg1 >ldg1.xml
  ```

- To save the configurations for all the domains on a system, create an XML file containing the constraints for all domains.

  ```
  # ldm list-constraints -x >file.xml
  ```

  The following example shows how to create an XML file, `config.xml`, which contains the constraints for all the domains on a system:

```
# ldm list-constraints -x >config.xml
```

Restoring Domain Configurations

This section describes how to restore a domain configuration from an XML file for guest domains and for the control (`primary`) domain.

- To restore a domain configuration for guest domains, you use the `ldm add-domain -i` command, as described in "How to Restore a Domain Configuration From an XML File (`ldm add-domain`)" on page 348. Although you can save the `primary` domain's constraints to an XML file, you cannot use the file as input to this command.
- To restore a domain configuration for the `primary` domain, you use the `ldm init-system` command and the resource constraints from the XML file to reconfigure your `primary` domain. You can also use the `ldm init-system` command to reconfigure other domains that are described in the XML file, but those domains might be left inactive when the configuration is complete. See "How to Restore a Domain Configuration From an XML File (`ldm init-system`)" on page 349.

How to Restore a Domain Configuration From an XML File (`ldm add-domain`)

This procedure works for guest domains but not for the control (`primary`) domain. If you want to restore the configuration for the `primary` domain, or for other domains that are described in the XML file, see "How to Restore a Domain Configuration From an XML File (`ldm init-system`)" on page 349.

1 Create the domain by using the XML file that you created as input.

```
# ldm add-domain -i domain-name.xml
```

2 Bind the domain.

```
# ldm bind-domain [-fq] domain-name
```

The `-f` option forces the binding of the domain even if invalid back-end devices are detected. The `-q` option disables the validation of back-end devices so that the command runs more quickly.

3 Start the domain.

```
# ldm start-domain domain-name
```

Example 15–2 Restoring a Single Domain From an XML File

The following example shows how to restore a single domain. First, you restore the `ldg1` domain from the XML file. Then, you bind and restart the `ldg1` domain that you restored.

```
# ldm add-domain -i ldg1.xml
# ldm bind ldg1
# ldm start ldg1
```

How to Restore a Domain Configuration From an XML File (`ldm init-system`)

This procedure explains how to use the `ldm init-system` command with an XML file to re-create a previously saved configuration.

Caution – The `ldm init-system` command might not correctly restore a configuration in which physical I/O commands have been used. Such commands are `ldm add-io`, `ldm set-io`, `ldm remove-io`, `ldm create-vf`, and `ldm destroy-vf`. For more information, see "ldm init-system Command Might Not Correctly Restore a Domain Configuration on Which Physical I/O Changes Have Been Made" in *Oracle VM Server for SPARC 3.3 Release Notes*.

Before You Begin You should have created an XML configuration file by running the `ldm list-constraints -x` command. The file should describe one or more domain configurations.

1. **Log in to the `primary` domain.**

2. **Verify that the system is in the `factory-default` configuration.**

   ```
   primary# ldm list-config | grep "factory-default"
   factory-default [current]
   ```

   If the system is not in the `factory-default` configuration, see "How to Restore the Factory Default Configuration" on page 387.

3. **Become an administrator.**

   For Oracle Solaris 11.3, see Chapter 1, "About Using Rights to Control Users and Processes," in *Securing Users and Processes in Oracle Solaris 11.3* .

4. **Restore the domain configuration or configurations from the XML file.**

   ```
   # ldm init-system [-frs] -i filename.xml
   ```

   The `primary` domain must be rebooted for the configuration to take effect. The `-r` option reboots the `primary` domain after the configuration. If you do not specify the `-r` option, you must perform the reboot manually.

   The `-s` option restores only the virtual services configuration (`vds`, `vcc`, and `vsw`) and might be able to be performed without having to reboot.

   The `-f` option skips the factory-default configuration check and continues regardless of what was already configured on the system. Use the `-f` option with caution. The `ldm init-system` command assumes that the system is in the factory-default configuration and so directly applies the changes that are specified by the XML file. Using `-f` when the system is in a configuration

other than the factory default will likely result in a system that is not configured as specified by the XML file. One or more changes might fail to be applied to the system, depending on the combination of changes in the XML file and the initial configuration.

The `primary` domain is reconfigured as specified in the file. Any non-`primary` domains that have configurations in the XML file are reconfigured but left inactive.

Example 15–3 Restoring Domains From XML Configuration Files

The following examples show how to use the `ldm init-system` command to restore the `primary` domain and all the domains on a system from the `factory-default` configuration.

- **Restore the `primary` domain.** The `-r` option is used to reboot the `primary` domain after the configuration completes. The `primary.xml` file contains the XML domain configuration that you saved at an earlier time.

  ```
  primary# ldm init-system -r -i primary.xml
  ```

- **Restore all the domains on a system.** Restore the domains on the system to the configurations in the `config.xml` XML file. The `config.xml` file contains the XML domain configurations that you saved at an earlier time. The `primary` domain is restarted automatically by the `ldm init-system` command. Any other domains are restored but not bound and restarted.

  ```
  # ldm init-system -r -i config.xml
  ```

  After the system reboots, the following commands bind and restart the `ldg1` and `ldg2` domains:

  ```
  # ldm bind ldg1
  # ldm start ldg1
  # ldm bind ldg2
  # ldm start ldg2
  ```

Addressing Service Processor Connection Problems

On a SPARC T7 series server and SPARC M7 series server, the ILOM interconnect is used to communicate between the `ldmd` service and the SP.

If an attempt to use the `ldm` command to manage domain configurations fails because of an error communicating with the SP, perform the following recovery step that applies to your platform:

- **SPARC T7 Series Server and SPARC M7 Series Server:** Check the ILOM interconnect state and re-enable the `ilomconfig-interconnect` service. See "How to Verify the ILOM Interconnect Configuration" on page 54 and "How to Re-Enable the ILOM Interconnect Service" on page 55.
- **SPARC T3, SPARC T4, SPARC T5, SPARC M5, and SPARC M6 Servers:** Restart the `ldmd` service.

  ```
  primary# svcadm enable ldmd
  ```

If these steps fail to restore communications, restart the SP.

◆ ◆ ◆ CHAPTER 16

Handling Hardware Errors

This chapter contains information about how Oracle VM Server for SPARC handles hardware errors.

This chapter covers the following topics:

Hardware Error-Handling Overview

The Oracle VM Server for SPARC software adds the following RAS capabilities for the SPARC enterprise-class platforms starting with SPARC T5 and SPARC M5:

- **Fault Management Architecture (FMA) blacklisting.** When FMA detects faulty CPU or memory resources, Oracle VM Server for SPARC places them on a blacklist. A faulty resource that is on the blacklist cannot be reassigned to any domains until FMA marks it as being repaired.
- **Recovery mode.** Automatically recover domain configurations that cannot be booted because of faulty or missing resources.

The Fujitsu M10 platform also supports recovery mode. While the blacklisting of faulty resources is not supported, the Fujitsu M10 platform auto-replacement feature provides similar functionality.

Using FMA to Blacklist or Unconfigure Faulty Resources

FMA contacts the Logical Domains Manager when it detects a faulty resource. Then, the Logical Domains Manager attempts to stop using that resource in all running domains. To ensure that a faulty resource cannot be assigned to a domain in the future, FMA adds the resource to a blacklist.

The Logical Domains Manager supports blacklisting only for CPU and memory resources, not for I/O resources.

If a faulty resource is not in use, the Logical Domains Manager removes it from the available resource list, which you can see in the `ldm list-devices` output. At this time, this resource is internally marked as "blacklisted" so that it cannot be re-assigned to a domain in the future.

If the faulty resource is in use, the Logical Domains Manager attempts to evacuate the resource. To avoid a service interruption on the running domains, the Logical Domains Manager first attempts to use CPU or memory dynamic reconfiguration to evacuate the faulty resource. The Logical Domains Manager remaps a faulted core if a core is free to use as a target. If this "live evacuation" succeeds, the faulty resource is internally marked as blacklisted and is not shown in the `ldm list-devices` output so that it will not be assigned to a domain in the future.

If the live evacuation fails, the Logical Domains Manager internally marks the faulty resource as "evacuation pending." The resource is shown as normal in the `ldm list-devices` output because the resource is still in use on the running domains until the affected guest domains are rebooted or stopped.

When the affected guest domain is stopped or rebooted, the Logical Domains Manager attempts to evacuate the faulty resources and internally mark them as blacklisted so that the resource cannot be assigned in the future. Such a device is not shown in the `ldm` output. After the pending evacuation completes, the Logical Domains Manager attempts to start the guest domain. However, if the guest domain cannot be started because sufficient resources are not available, the guest domain is marked as "degraded" and the following warning message is logged for the user intervention to perform the manual recovery.

```
primary# ldm ls
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
primary          active     -n-cv-  UART    368   2079488M 0.1%  0.0%  16h 57m
gd0              bound      -d----  5000    8

warning: Could not restart domain gd0 after completing pending evacuation.
The domain has been marked degraded and should be examined to see
if manual recovery is possible.
```

When the system is power-cycled, FMA repeats the evacuation requests for resources that are still faulty and the Logical Domains Manager handles those requests by evacuating the faulty resources and internally marking them as blacklisted.

Prior to support for FMA blacklisting, a guest domain that panicked because of a faulty resource might result in a never-ending panic-reboot loop. By using resource evacuation and blacklisting when the guest domain is rebooted, you can avoid this panic-reboot loop and prevent future attempts to use a faulty resource.

Recovering Domains After Detecting Faulty or Missing Resources

If a SPARC T5 server, SPARC T7 series server, SPARC M5 server, SPARC M6 server, SPARC M7 series server, or Fujitsu M10 server detects a faulty or missing resource at power on, the Logical Domains Manager attempts to recover the configured domains by using the remaining available resources. While the recovery takes place, the system (or physical domain on SPARC M5, SPARC M6, and SPARC M7 series servers) is said to be in *recovery mode*. Recovery mode is enabled by default. See "Controlling Recovery Mode" on page 358.

At power on, the system firmware reverts to the factory-default configuration if the last selected power-on configuration cannot be booted in any of the following circumstances:

- The I/O topology within each PCIe switch in the configuration does not match the I/O topology of the last selected power-on configuration
- CPU resources or memory resources of the last selected power-on configuration are no longer present in the system

When recovery mode is enabled, the Logical Domains Manager recovers all active and bound domains from the last selected power-on configuration. The resulting running configuration is called the *degraded configuration*. The degraded configuration is saved to the SP and remains the active configuration until either a new SP configuration is saved or the physical domain is power-cycled.

Note – The physical domain does not require a power cycle to activate the degraded configuration after recovery as it is already the running configuration.

If the physical domain is power-cycled, the system firmware first attempts to boot the last original power-on configuration. That way, if the missing or faulty hardware was replaced in the meantime, the system can boot the original normal configuration. If the last selected power-on configuration is not bootable, the firmware next attempts to boot the associated degraded configuration if it exists. If the degraded configuration is not bootable or does not exist, the factory-default configuration is booted and recovery mode is invoked.

The recovery operation works in the following order:

- **Control domain.** The Logical Domains Manager recovers the control domain by restoring its CPU, memory, and I/O configuration as well as its virtual I/O services.

  If the amount of CPU or memory required for all recoverable domains is larger than the remaining available amounts, the number of CPUs or cores or memory is reduced in proportion to the size of the other domains. For example, in a four-domain system where each domain has 25% of the CPUs and memory assigned, the resulting degraded configuration still assigns 25% of the CPUs and memory to each domain. If the `primary` domain originally had up to two cores (16 virtual CPUs) and eight Gbytes of memory, the control domain size is not reduced.

  Root complexes and PCIe devices that are assigned to other domains are removed from the control domain. The virtual functions on root complexes that are owned by the control domain are re-created. Any missing root complex, PCIe device, physical function, or virtual function that is assigned to the control domain is marked as evacuated. The Logical Domains Manager then reboots the control domain to make the changes active.
- **Root domains.** After the control domain has been rebooted, the Logical Domains Manager recovers the root domains. The amount of CPU and memory is reduced in proportion to the other recoverable domains, if needed. If a root complex is no longer physically present in the system, it is marked as evacuated. This root complex is not configured into the domain during the recovery operation. A root domain is recovered as long as at least one of the root complexes that is assigned to the root domain is available. If none of its root complexes are available, the root domain is not recovered. The Logical Domains Manager boots the root domain and re-creates the virtual functions on the physical functions that are owned by the root domain. It also removes the PCIe slots that are loaned out by the root domain. Any missing PCIe slots, physical functions, and virtual functions are marked as evacuated. Any virtual I/O services that are provided by the domain are re-created, if possible.
- **I/O domains.** Logical Domains Manager recovers any I/O domains. Any PCIe slots and virtual functions that are missing from the system are marked as evacuated. If none of the required I/O devices are present, the domain is not recovered and its CPU and memory resources are available for use by other domains. Any virtual I/O services that are provided by the domain are re-created, if possible.
- **Guest domains.** A guest domain is recovered *only* if at least one of the service domains that serves the domain has been recovered. If the guest domain cannot be recovered, its CPU and memory resources are available for use by other guest domains.

When possible, the same number of CPUs and amount of memory are allocated to a domain as specified by the original configuration. If that number of CPUs or amount of memory are not available, these resources are reduced proportionally to consume the remaining available resources.

Note – When a system is in recovery mode, you can only perform `ldm list-*` commands. All other `ldm` commands are disabled until the recovery operation completes.

The Logical Domains Manager only attempts to recover bound and active domains. The existing resource configuration of any unbound domain is copied to the new configuration as-is.

During a recovery operation, fewer resources might be available than in the previously booted configuration. As a result, the Logical Domains Manager might only be able to recover some of the previously configured domains. Also, a recovered domain might not include all of the resources from its original configuration. For example, a recovered bound domain might have fewer I/O resources than it had in its previous configuration. A domain might not be recovered if its I/O devices are no longer present or if its parent service domain could not be recovered.

Recovery mode records its steps to the Logical Domains Manager SMF log, `/var/svc/log/ldoms-ldmd:default.log`. A message is written to the system console when Logical Domains Manager starts a recovery, reboots the control domain, and when the recovery completes.

Caution – A recovered domain is not guaranteed to be completely operable. The domain might not include a resource that is essential to run the OS instance or an application. For example, a recovered domain might only have a network resource and no disk resource. Or, a recovered domain might be missing a file system that is required to run an application. Using multipathed I/O for a domain reduces the impact of missing I/O resources.

Recovery Mode Hardware and Software Requirements

- **Hardware Requirements –** The recovery mode feature is supported on the SPARC T5 server, SPARC T7 series server, SPARC M5 server, SPARC M6 server, SPARC M7 series server, and Fujitsu M10 server.
- **Firmware Requirements –** At least version 9.1.0.a of the system firmware for the SPARC T5 server, SPARC M5 server, and SPARC M6 server. At least version 9.4.3 of the system firmware for the SPARC T7 series server and SPARC M7 series server. At least version XCP2230 of the system firmware for the Fujitsu M10 server.
- **Software Requirements –** Non-`primary` root domains that loan out PCIe slots must be running at least the Oracle Solaris 10 1/13 OS or the Oracle Solaris 11.2 OS.

Degraded Configuration

Each physical domain can have only one degraded configuration saved to the SP. If a degraded configuration already exists, it is replaced by the newly created degraded configuration.

You cannot interact directly with degraded configurations. The system firmware transparently boots the degraded version of the next power-on configuration, if necessary. This transparency enables the system to boot the original configuration after a power cycle when the missing resources reappear. When the active configuration is a degraded configuration, it is marked as `[degraded]` in the `ldm list-spconfig` output.

The autosave functionality is disabled while the active configuration is a degraded configuration. If you save a new configuration to the SP while a degraded configuration is active, the new configuration is a normal non-degraded configuration.

Note – A previously missing resource that reappears on a subsequent power cycle has no effect on the contents of a normal configuration. However, if you subsequently select the configuration that triggered recovery mode, the SP boots the original, non-degraded configuration now that all its hardware is available.

Controlling Recovery Mode

The `ldmd/recovery_mode` SMF property controls recovery mode behavior. Recovery mode is enabled by default.

When the `ldmd/recovery_mode` property is not present or is set to `auto`, recovery mode is enabled.

When the `ldmd/recovery_mode` property is set to `never`, the Logical Domains Manager exits recovery mode without taking any action and the physical domain runs the factory-default configuration.

Note – If the system firmware requests recovery mode while it is not enabled, issue the following commands to enable recovery mode after the request is made:

```
primary# svccfg -s ldmd setprop ldmd/recovery_mode = astring: auto
primary# svcadm refresh ldmd
primary# svcadm restart ldmd
```

Recovery mode is initiated immediately in this scenario only if no changes were made to the system, that is, if it is still in the factory-default configuration.

In addition to enabling recovery mode, you can specify a timeout value for a root domain boot during recovery. By default, the `ldmd/recovery_mode_boot_timeout` property value is 30 minutes. Valid values start at 5 minutes.

Marking Domains as Degraded

A domain is marked as degraded if the FMA blacklisting of a resource leaves a domain with insufficient resources to start. The domain then remains in the bound state, which prevents the remaining resources that are assigned to the domain from being reallocated to other domains.

Marking I/O Resources as Evacuated

An I/O resource that is detected as missing by recovery mode is marked as evacuated by showing an asterisk (*) in `ldm` list output.

◆ ◆ ◆ CHAPTER 17

Performing Other Administration Tasks

This chapter contains information about using the Oracle VM Server for SPARC software and tasks that are not described in the preceding chapters.

This chapter covers the following topics:

Entering Names in the CLI

The following sections describe the restrictions on entering names in the Logical Domains Manager CLI.

- File names (*file*) and variable names (*var-name*)
  - First character must be a letter, a number, or a forward slash (/).
  - Subsequent letters must be letters, numbers, or punctuation.
- Virtual disk server *backend* and virtual switch device names

  The names must contain letters, numbers, or punctuation.
- Configuration Name (*config-name*)

  The logical domain configuration name (*config-name*) that you assign to a configuration stored on the service processor (SP) must have no more than 64 characters.
- All other names

  The remainder of the names, such as the logical domain name (*domain-name*), service names (*vswitch-name*, *service-name*, *vdpcs-service-name*, and *vcc-name*), virtual network name (*if-name*), and virtual disk name (*disk-name*), must be in the following format:
  - First character must be a letter or number.
  - Subsequent characters must be letters, numbers, or any of the following characters `-_+#.:;~()`.

Updating Property Values in the `/etc/system` File

Do not make manual changes to the `/etc/system` file. This file is automatically generated upon reboot when files in the `/etc/system.d` directory specify tuning property values.

▼ How to Add or Modify a Tuning Property Value

1 Search for the tuning property name in the existing `/etc/system` file and in the `/etc/system.d` files.

For example, to specify a value for the `vds:vd_volume_force_slice` property, determine whether the property is already set.

```
# grep 'vds:vd_volume_force_slice' /etc/system /etc/system.d/*
```

2 Update the property value:

- **If the property is found in one of the `/etc/system.d` files, update the property value in the existing file.**

- **If the property is found in the `/etc/system` file or it is not found, create a file in the `/etc/system.d` directory with a name such as the following example:**

  `/etc/system.d/com.`*company-name*`:ldoms-config`

Connecting to a Guest Console Over the Network

You can connect to a guest console over a network if the `listen_addr` property is set to the IP address of the control domain in the `vntsd`(1M) SMF manifest. For example:

```
$ telnet hostname 5001
```

Note – Enabling network access to a console has security implications. Any user can connect to a console and for this reason it is disabled by default.

A Service Management Facility manifest is an XML file that describes a service. For more information about creating an SMF manifest, refer to the Oracle Solaris 10 System Administrator Documentation (`http://download.oracle.com/docs/cd/E18752_01/index.html`).

Note – To access a non-English OS in a guest domain through the console, the terminal for the console must be in the locale required by the OS.

Using Console Groups

The virtual network terminal server daemon, `vntsd`, enables you to provide access for multiple domain consoles using a single TCP port. At the time of domain creation, the Logical Domains Manager assigns a unique TCP port to each console by creating a new default group for that domain's console. The TCP port is then assigned to the console group as opposed to the console itself. The console can be bound to an existing group using the `set-vcons` subcommand.

▼ How to Combine Multiple Consoles Into One Group

1. **Bind the consoles for the domains into one group.**

   The following example shows binding the console for three different domains (`ldg1`, `ldg2`, and `ldg3`) to the same console group (`group1`).

   ```
   primary# ldm set-vcons group=group1 service=primary-vcc0 ldg1
   primary# ldm set-vcons group=group1 service=primary-vcc0 ldg2
   primary# ldm set-vcons group=group1 service=primary-vcc0 ldg3
   ```

2 Connect to the associated TCP port (`localhost` at port `5000` in this example).

```
# telnet localhost 5000
primary-vnts-group1: h, l, c{id}, n{name}, q:
```

You are prompted to select one of the domain consoles.

3 List the domains within the group by selecting `l` (list).

```
primary-vnts-group1: h, l, c{id}, n{name}, q: l
DOMAIN ID           DOMAIN NAME                   DOMAIN STATE
0                   ldg1                          online
1                   ldg2                          online
2                   ldg3                          online
```

Note – To reassign the console to a different group or `vcc` instance, the domain must be unbound; that is, it has to be in the inactive state. Refer to the `vntsd`(1M) man page for more information about configuring and using SMF to manage `vntsd` and using console groups.

Stopping a Heavily Loaded Domain Can Time Out

An `ldm stop-domain` command can time out before the domain completes shutting down. When this happens, an error similar to the following is returned by the Logical Domains Manager.

```
LDom ldg8 stop notification failed
```

However, the domain could still be processing the shutdown request. Use the `ldm list-domain` command to verify the status of the domain. For example:

```
# ldm list-domain ldg8
NAME         STATE   FLAGS  CONS   VCPU MEMORY  UTIL UPTIME
ldg8         active  s----  5000   22   3328M   0.3% 1d 14h 31m
```

The preceding list shows the domain as active, but the `s` flag indicates that the domain is in the process of stopping. This should be a transitory state.

The following example shows the domain has now stopped.

```
# ldm list-domain ldg8
NAME         STATE   FLAGS  CONS   VCPU MEMORY  UTIL UPTIME
ldg8         bound   -----  5000   22   3328M
```

The `ldm stop` command uses the `shutdown` command to stop a domain. The execution of the shutdown sequence usually takes much longer than a quick stop, which can be performed by running the `ldm stop -q` command. See the `ldm`(1M) man page.

A long shutdown sequence might generate the following timeout message:

```
domain-name stop timed out. The domain might still be in the process of shutting down.
Either let it continue, or specify -f to force it to stop.
```

While this shutdown sequence runs, the `s` flag is also shown for the domain.

Operating the Oracle Solaris OS With Oracle VM Server for SPARC

This section describes the changes in behavior in using the Oracle Solaris OS that occur once a configuration created by the Logical Domains Manager is instantiated.

OpenBoot Firmware Not Available After the Oracle Solaris OS Has Started

The OpenBoot firmware is not available after the Oracle Solaris OS has started because it is removed from memory.

To reach the `ok` prompt from the Oracle Solaris OS, you must halt the domain by using the Oracle Solaris OS `halt` command.

Performing a Power Cycle of a Server

Whenever performing any maintenance on a system running Oracle VM Server for SPARC software that requires you to perform a power cycle of the server, you must save your current logical domain configurations to the SP first.

To save your current domain configurations to the SP, use the following command:

```
# ldm add-config config-name
```

Result of Oracle Solaris OS Breaks

You can initiate Oracle Solaris OS breaks as follows:

1. Press the L1-A key sequence when the input device is set to `keyboard`.
2. Enter the `send break` command when the virtual console is at the `telnet` prompt.

When you initiate such a break, the Oracle Solaris OS issues the following prompt:

```
c)ontinue, s)ync, r)eset, h)alt?
```

Type the letter that represents what you want the system to do after these types of breaks.

Results From Rebooting the Control Domain

You can use the `reboot` and `shutdown -i 5` commands to reboot the control (`primary`) domain.

- `reboot`:
  - **No other domains configured.** Reboots the control domain without graceful shutdown, no power off.
  - **Other domains configured.** Reboots the control domain without graceful shutdown, no power off.
- `shutdown -i 5`:
  - **No other domains configured.** Host powered off after graceful shutdown, stays off until powered on at the SP.
  - **Other domains configured.** Reboots with graceful shutdown, no power off.

For information about the consequences of rebooting a domain that has the root domain role, see "Rebooting the Root Domain With PCIe Endpoints Configured" on page 149.

Using Oracle VM Server for SPARC With the Service Processor

The section describes information related to using the Integrated Lights Out Manager (ILOM) service processor (SP) with the Logical Domains Manager. For more information about using the ILOM software, see the documents for your specific platform at `http://www.oracle.com/technetwork/documentation/sparc-tseries-servers-252697.html`.

An additional `config` option is available to the existing ILOM command:

```
-> set /HOST/bootmode config=config-name
```

This option enables you to set the configuration on the next power on to another configuration, including the `factory-default` shipping configuration.

You can invoke the command regardless of whether the host is powered on or off. It takes effect on the next host reset or power on.

To reset the logical domain configuration, you set the option to `factory-default`.

```
-> set /HOST/bootmode config=factory-default
```

You also can select other configurations that have been created with the Logical Domains Manager using the `ldm add-config` command and stored on the service processor (SP). The name you specify in the Logical Domains Manager `ldm add-config` command can be used to select that configuration with the ILOM `bootmode` command. For example, assume you stored the configuration with the name `ldm-config1`.

```
-> set /HOST/bootmode config=ldm-config1
```

Now, you must perform a power cycle of the system to load the new configuration.

See the `ldm(1M)` man page for more information about the `ldm add-config` command.

Configuring Domain Dependencies

You can use the Logical Domains Manager to establish dependency relationships between domains. A domain that has one or more domains that depend on it is called a *master domain*. A domain that depends on another domain is called a *slave domain*.

Each slave domain can specify up to four master domains by setting the `master` property. For example, the `pine` slave domain specifies its four master domains in the following comma-separated list:

```
# ldm add-domain master=alpha,beta,gamma,delta pine
```

The `alpha`, `beta`, `gamma`, and `delta` master domains all specify a failure policy of `stop`.

Each master domain can specify what happens to its slave domains in the event that the master domain fails. For instance, if a master domain fails, it might require its slave domains to panic. If a slave domain has more than one master domain, each master domain must have the same failure policy. So, the first master domain to fail triggers its defined failure policy on all of its slave domains.

The master domain's failure policy is controlled by setting one of the following values to the `failure-policy` property:

- `ignore` ignores any slave domains
- `panic` panics any slave domains (similar to running the `ldm panic` command)
- `reset` immediately stops and then restarts any slave domains (similar to running the `ldm stop -f` command and then the `ldm start` command)
- `stop` stops any slave domains (similar to running the `ldm stop -f` command)

In this example, the master domains specify their failure policy as follows:

```
primary# ldm set-domain failure-policy=ignore apple
primary# ldm set-domain failure-policy=panic lemon
primary# ldm set-domain failure-policy=reset orange
primary# ldm set-domain failure-policy=stop peach
primary# ldm set-domain failure-policy=stop alpha
primary# ldm set-domain failure-policy=stop beta
primary# ldm set-domain failure-policy=stop gamma
primary# ldm set-domain failure-policy=stop delta
```

You can use this mechanism to create explicit dependencies between domains. For example, a guest domain implicitly depends on the service domain to provide its virtual devices. A guest domain's I/O is blocked when the service domain on which it depends is not up and running. By

defining a guest domain as a slave of its service domain, you can specify the behavior of the guest domain when its service domain goes down. When no such dependency is established, a guest domain just waits for its service domain to return to service.

Note – The Logical Domains Manager does not permit you to create domain relationships that create a dependency cycle. For more information, see "Dependency Cycles" on page 369.

For domain dependency XML examples, see Example 23–6.

Domain Dependency Examples

The following examples show how to configure domain dependencies.

EXAMPLE 17–1 Configuring a Failure Policy by Using Domain Dependencies

The first command creates a master domain called `twizzle`. This command uses `failure-policy=reset` to specify that slave domains reset if the `twizzle` domain fails. The second command modifies a master domain called `primary`. This command uses `failure-policy=reset` to specify that slave domains reset if the `primary` domain fails. The third command creates a slave domain called `chocktaw` that depends on two master domains, `twizzle` and `primary`. The slave domain uses `master=twizzle,primary` to specify its master domains. In the event either the `twizzle` or `primary` domain fails, the `chocktaw` domain will reset.

```
primary# ldm add-domain failure-policy=reset twizzle
primary# ldm set-domain failure-policy=reset primary
primary# ldm add-domain master=twizzle,primary chocktaw
```

EXAMPLE 17–2 Modifying a Domain to Assign a Master Domain

This example shows how to use the `ldm set-domain` command to modify the `orange` domain to assign `primary` as the master domain. The second command uses the `ldm set-domain` command to assign `orange` and `primary` as master domains for the `tangerine` domain. The third command lists information about all of these domains.

```
primary# ldm set-domain master=primary orange
primary# ldm set-domain master=orange,primary tangerine
primary# ldm list -o domain
NAME             STATE      FLAGS   UTIL
primary          active     -n-cv-  0.2%

SOFTSTATE
Solaris running

HOSTID
    0x83d8b31c
```

EXAMPLE 17–2 Modifying a Domain to Assign a Master Domain *(Continued)*

```
CONTROL
    failure-policy=ignore

DEPENDENCY
    master=

------------------------------------------------------------------------------
NAME             STATE      FLAGS   UTIL
orange           bound      ------

HOSTID
    0x84fb28ef

CONTROL
    failure-policy=ignore

DEPENDENCY
    master=primary

------------------------------------------------------------------------------
NAME             STATE      FLAGS   UTIL
tangerine        bound      ------

HOSTID
    0x84f948e9

CONTROL
    failure-policy=ignore

DEPENDENCY
    master=orange,primary
```

EXAMPLE 17–3 Showing a Parseable Domain Listing

The following shows an example listing with parseable output:

```
primary# ldm list -o domain -p
```

Dependency Cycles

The Logical Domains Manager does not permit you to create domain relationships that create a dependency cycle. A *dependency cycle* is a relationship between two or more domains that lead to a situation where a slave domain depends on itself or a master domain depends on one of its slave domains.

The Logical Domains Manager determines whether a dependency cycle exists before adding a dependency. The Logical Domains Manager starts at the slave domain and searches along all paths that are specified by the master array until the end of the path is reached. Any dependency cycles found along the way are reported as errors.

The following example shows how a dependency cycle might be created. The first command creates a slave domain called `mohawk` that specifies its master domain as `primary`. So, `mohawk` depends on `primary` in the dependency chain shown in the following diagram.

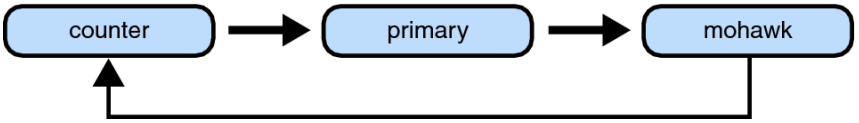
FIGURE 17–1 Single Domain Dependency

The second command creates a slave domain called `primary` that specifies its master domain as `counter`. So, `mohawk` depends on `primary`, which depends on `counter` in the dependency chain shown in the following diagram.

FIGURE 17–2 Multiple Domain Dependency

The third command attempts to create a dependency between the `counter` and `mohawk` domains, which would produce the dependency cycle shown in the following diagram.

FIGURE 17–3 Domain Dependency Cycle



The `ldm set-domain` command will fail with the following error message:

```
# ldm add-domain master=primary mohawk
# ldm set-domain master=counter primary
# ldm set-domain master=mohawk counter
Dependency cycle detected: LDom "counter" indicates "primary" as its master
```

Determining Where Errors Occur by Mapping CPU and Memory Addresses

This section describes how you can correlate the information that is reported by the Oracle Solaris Fault Management Architecture (FMA) with the logical domain resources that are marked as being faulty.

FMA reports CPU errors in terms of physical CPU numbers and memory errors in terms of physical memory addresses.

If you want to determine within which logical domain an error occurred and the corresponding virtual CPU number or real memory address within the domain, then you must perform a mapping.

CPU Mapping

You can find the domain and the virtual CPU number within the domain that correspond to a given physical CPU number.

First, generate a long parseable list for all domains by using the following command:

```
primary# ldm list -l -p
```

Look for the entry in the list's `VCPU` sections that has a `pid` field equal to the physical CPU number.

- If you find such an entry, the CPU is in the domain the entry is listed under, and the virtual CPU number within the domain is given by the entry's `vid` field.
- If you do not find such an entry, the CPU is not in any domain.

Memory Mapping

You can find the domain and the real memory address within the domain that correspond to a given physical memory address (PA).

First, generate a long parseable list for all domains.

```
primary# ldm list -l -p
```

Look for the line in the list's `MEMORY` sections where the PA falls within the inclusive range *pa* to (*pa* + *size* - 1); that is, *pa* ≤ PA ≤ (*pa* + *size* - 1). *pa* and *size* refer to the values in the corresponding fields of the line.

- If you find such an entry, the PA is in the domain the entry is listed under and the corresponding real address within the domain is given by *ra* + (PA - *pa*).
- If you do not find such an entry, the PA is not in any domain.

Example of CPU and Memory Mapping

EXAMPLE 17–4 Determining the Configuration of Domains

The following command produces a long parseable list of logical domains configurations.

EXAMPLE 17–4 Determining the Configuration of Domains *(Continued)*

```
primary# ldm list -l -p
VERSION 1.6
DOMAIN|name=primary|state=active|flags=normal,control,vio-service|
cons=SP|ncpu=4|mem=1073741824|util=0.6|uptime=64801|
softstate=Solaris running
VCPU
|vid=0|pid=0|util=0.9|strand=100
|vid=1|pid=1|util=0.5|strand=100
|vid=2|pid=2|util=0.6|strand=100
|vid=3|pid=3|util=0.6|strand=100
MEMORY
|ra=0x8000000|pa=0x8000000|size=1073741824
IO
|dev=pci@780|alias=bus_a
|dev=pci@7c0|alias=bus_b
...
DOMAIN|name=ldg1|state=active|flags=normal|cons=5000|
ncpu=2|mem=805306368|util=29|uptime=903|
softstate=Solaris running
VCPU
|vid=0|pid=4|util=29|strand=100
|vid=1|pid=5|util=29|strand=100
MEMORY
|ra=0x8000000|pa=0x48000000|size=805306368
...
DOMAIN|name=ldg2|state=active|flags=normal|cons=5001|
ncpu=3|mem=1073741824|util=35|uptime=775|
softstate=Solaris running
VCPU
|vid=0|pid=6|util=35|strand=100
|vid=1|pid=7|util=34|strand=100
|vid=2|pid=8|util=35|strand=100
MEMORY
|ra=0x8000000|pa=0x78000000|size=1073741824
...
```

EXAMPLE 17–5 Determining the Virtual CPU That Corresponds to a Physical CPU Number

The logical domain configuration is shown in Example 17–4. This example describes how to determine the domain and the virtual CPU corresponding to physical CPU number `5`, and the domain and the real address corresponding to physical address `0x7e816000`.

Looking through the `VCPU` entries in the list for the one with the `pid` field equal to `5`, you can find the following entry under logical domain `ldg1`.

```
|vid=1|pid=5|util=29|strand=100
```

Hence, the physical CPU number `5` is in domain `ldg1` and within the domain it has virtual CPU number `1`.

Looking through the `MEMORY` entries in the list, you can find the following entry under domain `ldg2`.

EXAMPLE 17–5 Determining the Virtual CPU That Corresponds to a Physical CPU Number *(Continued)*

```
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.

Using Universally Unique Identifiers

Each domain is assigned a universally unique identifier (UUID). The UUID is assigned when a domain is created. For legacy domains, the UUID is assigned when the `ldmd` daemon initializes.

Note – The UUID is lost if you use the `ldm migrate-domain -f` command to migrate a domain to a target machine that runs an older version of the Logical Domains Manager. When you migrate a domain from a source machine that runs an older version of the Logical Domains Manager, the domain is assigned a new UUID as part of the migration. Otherwise, the UUID is migrated.

You can obtain the UUID for a domain by running the `ldm list -l`, `ldm list-bindings`, or `ldm list -o domain` command. The following examples show the UUID for the `ldg1` domain:

```
primary# ldm create ldg1
primary# ldm ls -l ldg1
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg1             inactive   ------

UUID
    6c908858-12ef-e520-9eb3-f1cd3dbc3a59

primary# ldm ls -l -p ldg1
VERSION 1.6
DOMAIN|name=ldg1|state=inactive|flags=|cons=|ncpu=|mem=|util=|uptime=
UUID|uuid=6c908858-12ef-e520-9eb3-f1cd3dbc3a59
```

Virtual Domain Information Command and API

The `virtinfo` command enables you to gather information about a running virtual domain. You can also use the Virtual Domain Information API to create programs to gather information related to virtual domains.

The following list shows some of the information that you can gather about a virtual domain by using the command or API:

- Domain type (implementation, control, guest, I/O, service, root)
- Domain name determined by the Virtual Domain Manager
- Universally unique identifier (UUID) of the domain
- Network node name of the domain's control domain
- Chassis serial number on which the domain is running

For information about the `virtinfo` command, see the `virtinfo`(1M) man page. For information about the API, see the `libv12n`(3LIB) and `v12n`(3EXT) man pages.

Using Logical Domain Channels

Oracle VM Server for SPARC uses logical domain channels (LDCs) to implement all communications such as console, virtual I/O, and control traffic. An LDC is the method used to enable communications between two endpoints. Although typically each endpoint is in a different domain, the endpoints can be in the same domain to enable loopback communications.

This software and system firmware provide a large pool of LDC endpoints that you can use for the control domain and guest domains. This LDC endpoint pool is available only for the SPARC T4 servers, SPARC T5 servers, SPARC T7 series servers, SPARC M5 servers, SPARC M6 servers, SPARC M7 series servers, and Fujitsu M10 servers. The number of LDCs in the pool is based on the platform type as follows:

- **SPARC T4 Server –** 1984 LDC endpoints per guest domain, 98304 LDC endpoints total
- **SPARC T5 Server –** 1984 LDC endpoints per guest domain, 98304 LDC endpoints total
- **SPARC T7 Series Server –** 1984 LDC endpoints per guest domain, 98304 LDC endpoints total
- **SPARC M5 Server –** 1984 LDC endpoints per guest domain, 98304 LDC endpoints per physical domain
- **SPARC M6 Server –** 1984 LDC endpoints per guest domain, 98304 LDC endpoints per physical domain
- **SPARC M7 Series Server –** 1984 LDC endpoints per guest domain, 98304 LDC endpoints per physical domain
- **Fujitsu M10 server –** 1984 LDC endpoints per guest domain, 196608 LDC endpoints total

The required system firmware to support the LDC endpoint pool is 8.5.2 for SPARC T4 servers, 9.2.1 for SPARC T5 servers, SPARC M5 servers, and SPARC M6 servers, 9.4.3 for SPARC T7 series servers and SPARC M7 series servers, and XCP2240 for Fujitsu M10 servers.

The following LDC endpoint limits still apply if you run an older version of the system firmware on a supported platform or on an UltraSPARC T2, UltraSPARC T2 Plus, or SPARC T3 platform:

- **UltraSPARC T2 Server –** 512 LDC endpoints
- **UltraSPARC T2 Plus Server –** 768 LDC endpoints
- **SPARC T3 Server –** 768 LDC endpoints
- **SPARC T4 Server –** 768 LDC endpoints
- **SPARC T5 Server –** 768 LDC endpoints
- **SPARC T7 Series Server –** 768 LDC endpoints
- **SPARC M5 Server –** 768 LDC endpoints
- **SPARC M6 Server –** 768 LDC endpoints
- **SPARC M7 Series Servers –** 768 LDC endpoints
- **Fujitsu M10 servers –** 768 LDC endpoints

This limitation might be an issue on the control domain because of the potentially large number of LDC endpoints that are used for both virtual I/O data communications and the Logical Domains Manager control of the other domains.

If you attempt to add a service or bind a domain so that the number of LDC endpoints exceeds the limit on any single domain, the operation fails with an error message similar to the following:

```
13 additional LDCs are required on guest primary to meet this request,
but only 9 LDCs are available
```

The following guidelines enable you to plan properly for using LDC endpoints and explain why you might experience an overflow of the LDC capabilities of the control domain:

- The control domain uses approximately 15 LDC endpoints for various communication purposes with the hypervisor, Fault Management Architecture (FMA), and the system processor (SP), independent of the number of other domains configured. The number of LDC endpoints used by the control domain depends on the platform and on the version of the software that is used.
- The Logical Domains Manager allocates an LDC endpoint to the control domain for every domain, including itself, for control traffic.
- Each virtual I/O service on the control domain uses one LDC endpoint for every connected client of that service. Each domain needs at least a virtual network, a virtual disk, and a virtual console.

The following equation incorporates these guidelines to determine the number of LDC endpoints that are required by the control domain:

15 + *number-of-domains* + (*number-of-domains* x *number-of-virtual-services*) = *total-LDC-endpoints*

number-of-domains is the total number of domains including the control domain and *number-of-virtual-services* is the total number of virtual I/O devices that are serviced by this domain.

The following example shows how to use the equation to determine the number of LDC endpoints when there is a control domain and eight additional domains:

15 + 9 + (8 x 3) = 48 LDC endpoints

The following example has 45 guest domains and each domain includes five virtual disks, two virtual networks, and a virtual console. The calculation yields the following result:

15 + 46 + 45 x 8 = 421 LDC endpoints

Depending upon the number of supported LDC endpoints of your platform, the Logical Domains Manager will either accept or reject the configuration.

If you run out of LDC endpoints on the control domain, consider creating service domains or I/O domains to provide virtual I/O services to the guest domains. This action enables the LDC endpoints to be created on the I/O domains and the service domains instead of on the control domain.

A guest domain can also run out of LDC endpoints. This situation might be caused by the `inter-vnet-link` property being set to `on`, which assigns additional LDC endpoints to guest domains to connect directly to each other.

The following equation determines the number of LDC endpoints that are required by a guest domain when `inter-vnet-link=off`:

2 + *number-of-vnets* + *number-of-vdisks* = *total-LDC-endpoints*

2 represents the virtual console and control traffic, *number-of-vnets* is the total number of virtual network devices assigned to the guest domain, and *number-of-vdisks* is the total number of virtual disks assigned to the guest domain.

The following example shows how to use the equation to determine the number of LDC endpoints per guest domain when `inter-vnet-link=off` and you have two virtual disks and two virtual networks:

2 + 2 + 2 = 6 LDC endpoints

The following equation determines the number of LDC endpoints that are required by a guest domain when `inter-vnet-link=on`:

2 + [[(*number-of-vnets-from-vswX* x *number-of-vnets-in-vswX*)] ...] + *number-of-vdisks* = *total-LDC-endpoints*

2 represents the virtual console and control traffic, *number-of-vnets-from-vswX* is the total number of virtual network devices assigned to the guest domain from the *vswX* virtual switch,

number-of-vnets-in-vswX is the total number of virtual network devices on the *vswX* virtual switch, and *number-of-virtual-disks* is the total number of virtual disks assigned to the guest domain.

The following example shows how to use the equation to determine the number of LDC endpoints per guest domain when `inter-vnet-link=on` and you have two virtual disks and two virtual switches. The first virtual switch has eight virtual networks and assigns four of them to the domain. The second virtual switch assigns all eight of its virtual networks to the domain:

2 + (4 x 8) + (8 x 8) + 2 = 100 LDC endpoints

To resolve the issue of running out of LDC endpoints on a guest domain, consider using the `ldm add-vsw` or `ldm set-vsw` command to set the `inter-vnet-link` property to `off`. This action reduces the number of LDC endpoints in the domain or domains that have the virtual network devices. However, the `off` property value does not affect the service domain that has the virtual switch because the service domain still requires an LDC connection to each virtual network device. When this property is set to `off`, LDC channels are not used for inter-vnet communications. Instead, an LDC channel is assigned only for communication between virtual network devices and virtual switch devices. See the `ldm(1M)` man page.

Note – Although disabling the assignment of inter-vnet links reduces the number of LDC endpoints, it might negatively affect guest-to-guest network performance. This degradation would occur because all guest-to-guest communications traffic goes through the virtual switch rather than directly from one guest domain to another guest domain.

Booting a Large Number of Domains

You can boot the following number of domains depending on your server:

- Up to 256 on Fujitsu M10 servers per physical partition
- Up to 128 on SPARC M7 series servers per physical domain
- Up to 128 on SPARC M6 servers per physical domain
- Up to 128 on SPARC M5 servers per physical domain
- Up to 128 on SPARC T7 series servers
- Up to 128 on SPARC T5 servers
- Up to 128 on SPARC T4 servers
- Up to 128 on SPARC T3 servers
- Up to 128 on UltraSPARC T2 Plus servers
- Up to 64 on UltraSPARC T2 servers

If unallocated virtual CPUs are available, assign them to the service domain to help process the virtual I/O requests. Allocate 4 to 8 virtual CPUs to the service domain when creating more than 32 domains. In cases where maximum domain configurations have only a single CPU in the service domain, do not put unnecessary stress on the single CPU when configuring and

using the domain. The virtual switch (`vsw`) services should be spread across all the network adapters available in the machine. For example, if booting 128 domains on a Sun SPARC Enterprise T5240 server, create 4 `vsw` services, each serving 32 virtual net (`vnet`) instances. Assigning more than 32 `vnet` instances per `vsw` service could cause hard hangs in the service domain.

To run the maximum configurations, a machine needs an adequate amount of memory to support the guest domains. The amount of memory is dependent on your platform and your OS. See the documentation for your platform, *Oracle Solaris 10 8/11 Installation Guide: Planning for Installation and Upgrade* and *Installing Oracle Solaris 11.3 Systems* .

Memory and swap space usage increases in a guest domain when the `vsw` services used by the domain provide services to many virtual networks in multiple domains. This increase is due to the peer-to-peer links between all the `vnet` instances connected to the `vsw`. The service domain benefits from having extra memory. The recommended minimum is four Gbytes when running more than 64 domains. Start domains in groups of 10 or fewer and wait for them to boot before starting the next batch. The same advice applies to installing operating systems on domains. You can reduce the number of links by disabling inter-vnet links. See "Inter-Vnet LDC Channels" on page 230.

Cleanly Shutting Down and Power Cycling an Oracle VM Server for SPARC System

If you have made any configuration changes since last saving a configuration to the SC, before you attempt to power off or power cycle an Oracle VM Server for SPARC system, make sure that you save the latest configuration that you want to keep.

▼ How to Power Off a System With Multiple Active Domains

1. **Shut down, stop, and unbind all the non-I/O domains.**
2. **Shut down, stop, and unbind any active I/O domains.**
3. **Change the domain into init state 5.**

   ```
   primary# shutdown -i 5
   ```

   Instead of using the `shutdown` command, you can also use the `init 5` command.

▼ How to Power Cycle the System

1. **Power off the system.**

   See "How to Power Off a System With Multiple Active Domains" on page 378.

2. **Use the SP to power on the system.**

Logical Domains Variable Persistence

Variable updates persist across a reboot but not across a power cycle unless the variable updates are either initiated from OpenBoot firmware on the control domain or followed by saving the configuration to the SC.

Note the following conditions:

- When the control domain reboots, if there are no bound guest domains and no delayed reconfiguration in progress, the SC performs a power cycle of the system.
- When the control domain reboots, if guest domains are bound or active (or the control domain is in the middle of a delayed reconfiguration), the SC does not perform a power cycle of the system.

Logical Domains variables for a domain can be specified using any of the following methods:

- At the OpenBoot prompt.
- Using the Oracle Solaris OS `eeprom(1M)` command.
- Using the Logical Domains Manager CLI (`ldm`).
- In a limited fashion, from the system controller (SC) using the `bootmode` command. This method can be used for only certain variables, and only when in the `factory-default` configuration.

Variable updates that are made by using any of these methods should always persist across reboots of the domain. The variable updates also always apply to any subsequent domain configurations that were saved to the SC.

In Oracle VM Server for SPARC 3.3 software, variable updates do not persist as expected in a few cases:

- All methods of updating a variable persist across reboots of that domain. However, they do not persist across a power cycle of the system unless a subsequent logical domain configuration is saved to the SC.

  However in the control domain, updates made using either OpenBoot firmware commands or the `eeprom` command *do* persist across a power cycle of the system even without subsequently saving a new logical domain configuration to the SC. The `eeprom` command

supports this behavior on SPARC T5 servers, SPARC T7 series servers, SPARC M5 servers, SPARC M6 servers, and SPARC M7 series servers, and on SPARC T3 servers and SPARC T4 servers that run at least version 8.2.1 of the system firmware.

If you are concerned about Logical Domains variable changes, do one of the following:

- Bring the system to the `ok` prompt and update the variables.
- Update the variables while the Logical Domains Manager is disabled:

```
# svcadm disable ldmd
update variables
# svcadm enable ldmd
```

- When running Live Upgrade, perform the following steps:

```
# svcadm disable -t ldmd
# luactivate be3
# init 6
```

If you modify the time or date on a logical domain, for example, using the `ntpdate` command, the change persists across reboots of the domain but not across a power cycle of the host. To ensure that time changes persist, save the configuration with the time change to the SP and boot from that configuration.

The following Bug IDs have been filed to resolve these issues: 15375997, 15387338, 15387606, and 15415199.

Adjusting the Interrupt Limit

Hardware provides a finite number of interrupts, so Oracle Solaris limits the number of interrupts that each device can use. The default limit should match the needs of a typical system configuration but you might need to adjust this value for certain system configurations.

Note – These limitations do not apply to the SPARC M7 series servers and SPARC T7 series servers.

When you enable I/O virtualization on a PCIe bus, interrupt hardware resources are assigned to each I/O domain. Each domain is allotted a finite number of those resources, which might lead to some interrupt allocation issues. This situation affects only the UltraSPARC T2, UltraSPARC T2 Plus, SPARC T3, SPARC T4, SPARC T5, SPARC M5, and SPARC M6 platforms.

The following warning on the Oracle Solaris console means the interrupt supply was exhausted while attaching I/O device drivers:

```
WARNING: ddi_intr_alloc: cannot fit into interrupt pool
```

Specifically, the limit might need adjustment if the system is partitioned into multiple logical domains and if too many I/O devices are assigned to any guest domain. Oracle VM Server for SPARC divides the total number of interrupts into smaller sets and assigns them to guest domains. If too many I/O devices are assigned to a guest domain, its interrupt supply might be too small to provide each device with the default limit of interrupts. Thus, the guest domain exhausts its interrupt supply before it completely attaches all the drivers.

Some drivers provide an optional callback routine that permits the Oracle Solaris OS to automatically adjust their interrupts. The default limit does not apply to these drivers.

Use the `::irmpools` and `::irmreqs` MDB macros to determine how interrupts are used. The `::irmpools` macro shows the overall interrupt supply divided into pools. The `::irmreqs` macro shows which devices are mapped to each pool. For each device, `::irmreqs` shows whether the default limit is enforced by an optional callback routine, how many interrupts each driver requested, and how many interrupts each driver has.

Although these macros do not show information about drivers that failed to attach, you can use the information to calculate the extent to which you can adjust the default limit. You can force any device that uses more than one interrupt without providing a callback routine to use fewer interrupts by adjusting the default limit. For such devices, reduce the default limit to free interrupts that can be used by other devices.

To adjust the default limit, set the `ddi_msix_alloc_limit` property to a value from `1` to `8` in the `/etc/system` file. Then, reboot the system for the change to take effect.

For information about correctly creating or updating `/etc/system` property values, see "Updating Property Values in the `/etc/system` File" on page 362.

To maximize performance, start by assigning larger values and decrease the values in small increments until the system boots successfully without any warnings. Use the `::irmpools` and `::irmreqs` macros to measure the adjustment's impact on all attached drivers.

For example, suppose the following warnings are issued while booting the Oracle Solaris OS in a guest domain:

```
WARNING: emlxs3: interrupt pool too full.
WARNING: ddi_intr_alloc: cannot fit into interrupt pool
```

The `::irmpools` and `::irmreqs` macros show the following information:

```
# echo "::irmpools" | mdb -k
ADDR             OWNER   TYPE   SIZE  REQUESTED  RESERVED
00000400016be970 px#0    MSI/X  36    36         36

# echo "00000400016be970::irmreqs" | mdb -k
ADDR             OWNER   TYPE   CALLBACK NINTRS NREQ NAVAIL
00001000143acaa8 emlxs#0 MSI-X  No       32     8    8
000010001701997f8 emlxs#1 MSI-X  No       32     8    8
000010001400ca28 emlxs#2 MSI-X  No       32     8    8
```

```
0000100016151328 igb#3   MSI-X  No       10     3    3
0000100019549d30 igb#2   MSI-X  No       10     3    3
0000040000e0f878 igb#1   MSI-X  No       10     3    3
000010001955a5c8 igb#0   MSI-X  No       10     3    3
```

The default limit in this example is 8 interrupts per device, which is not enough interrupts to accommodate attaching the final `emlxs3` device to the system. Assuming that all `emlxs` instances behave in the same way, `emlxs3` probably requested 8 interrupts.

By subtracting the 12 interrupts used by all of the `igb` devices from the total pool size of 36 interrupts, 24 interrupts are available for the `emlxs` devices. Dividing the 24 interrupts by 4 suggests that 6 interrupts per device would enable all `emlxs` devices to attach with equal performance. So, the following adjustment is added to the `/etc/system` file:

```
set ddi_msix_alloc_limit = 6
```

For information about correctly creating or updating `/etc/system` property values, see "Updating Property Values in the `/etc/system` File" on page 362.

When the system successfully boots without warnings, the `::irmpools` and `::irmreqs` macros show the following updated information:

```
primary# echo "::irmpools" | mdb -k
ADDR             OWNER   TYPE   SIZE  REQUESTED  RESERVED
00000400018ca868 px#0    MSI/X  36    36         36

# echo "00000400018ca868::irmreqs" | mdb -k
ADDR             OWNER   TYPE   CALLBACK NINTRS NREQ NAVAIL
0000100016143218 emlxs#0 MSI-X  No       32     8    6
0000100014269920 emlxs#1 MSI-X  No       32     8    6
000010001540be30 emlxs#2 MSI-X  No       32     8    6
00001000140cbe10 emlxs#3 MSI-X  No       32     8    6
00001000141210c0 igb#3   MSI-X  No       10     3    3
0000100017549d38 igb#2   MSI-X  No       10     3    3
0000040001ceac40 igb#1   MSI-X  No       10     3    3
000010001acc3480 igb#0   MSI-X  No       10     3    3
```

Listing Domain I/O Dependencies

I/O operations for a domain are often provided by another domain such as a service domain or an I/O domain. For example, a service domain can export a virtual device or a root domain can provide direct access to a physical device.

Be aware of these implicit I/O dependencies, as an outage in a service domain or a root domain will result in a service interruption of the dependent domain, as well.

You can use the `ldm list-dependencies` command to view the I/O dependencies between domains. In addition to listing the dependencies of a domain, you can invert the output to show the dependents of a particular domain.

The following list shows the types of I/O dependencies that you can view by using the `ldm list-dependencies` command:

`VDISK` Dependency created when a virtual disk is connected to a virtual disk backend that has been exported by a virtual disk server

`VNET` Dependency created when a virtual network device is connected to a virtual switch

`IOV` Dependency created when an SR-IOV virtual function is associated with an SR-IOV physical function

The following `ldm list-dependencies` commands show some of the ways in which you can view domain dependency information:

- To show detailed domain dependency information, use the `-l` option.

```
primary# ldm list-dependencies -l
DOMAIN         DEPENDENCY      TYPE      DEVICE
primary
svcdom
ldg0           primary         VDISK     primary-vds0/vdisk0
                               VNET      primary-vsw0/vnet0
               svcdom          VDISK     svcdom-vds0/vdisk1
                               VNET      svcdom-vsw0/vnet1
ldg1           primary         VDISK     primary-vds0/vdisk0
                               VNET      primary-vsw0/vnet0
                               IOV       /SYS/MB/NET0/IOVNET.PF0.VF0
               svcdom          VDISK     svcdom-vds0/vdisk1
                               VNET      svcdom-vsw0/vnet1
                               IOV       /SYS/MB/NET2/IOVNET.PF0.VF0
```

- To show detailed information about dependents grouped by their dependencies, use both the `-l` and `-r` options.

```
primary# ldm list-dependencies -r -l
DOMAIN         DEPENDENT       TYPE      DEVICE
primary        ldg0            VDISK     primary-vds0/vdisk0
                               VNET      primary-vsw0/vnet0
               ldg1            VDISK     primary-vds0/vdisk0
                               VNET      primary-vsw0/vnet0
                               IOV       /SYS/MB/NET0/IOVNET.PF0.VF0
svcdom         ldg0            VDISK     svcdom-vds0/vdisk1
                               VNET      svcdom-vsw0/vnet1
               ldg1            VDISK     svcdom-vds0/vdisk1
                               VNET      svcdom-vsw0/vnet1
                               IOV       /SYS/MB/NET2/IOVNET.PF0.VF0
```

Enabling the Logical Domains Manager Daemon

The Logical Domains Manager daemon, `ldmd`, is automatically enabled when the Oracle VM Server for SPARC software package is installed. Once the daemon is enabled, you can create, modify, and control the logical domains.

On SPARC T7 series servers and SPARC M7 series servers, the ILOM interconnect service enables communication between the `ldmd` daemon and the service processor (SP). The `ilomconfig-interconnect` service is enabled by default. To verify that the ILOM interconnect service is enabled, see "How to Verify the ILOM Interconnect Configuration" on page 54.

Caution – Do not disable the `ilomconfig-interconnect` service. Disabling this service might prevent the correct operation of logical domains and the OS.

How to 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`.**

```
# svcadm enable ldmd
```

For more information about the `svcadm` command, see the `svcadm`(1M) man page.

2 **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      64    3264M    0.3%  19d 9m
```

Saving and Restoring Autosave Configuration Data

The following sections describe how to save and restore autosave configuration directories and the constraints database file.

Saving and Restoring Autosave Configuration Directories

You can save and restore autosave configuration directories prior to reinstalling the operating system on the control domain. Whenever you reinstall the operating system on the control domain, you must save and restore the domain autosave configuration data, which is found in the `/var/share/ldomsmanager/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 Domain Configurations" on page 343.

▼ How to Save and Restore Autosave Directories

1 **Save the autosave directories.**

```
# cd /
# tar -cvpf autosave.tar var/share/ldomsmanager/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/share/ldomsmanager/autosave-*
```

3 **Restore the autosave directories.**

These commands restore the files and directories in the `/var/share/ldomsmanager` 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 `/var/share/ldomsmanager/ldom-db.xml` Logical Domains constraints database file.

Also save and restore the `/var/share/ldomsmanager/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.

Note – The `/var/share/ldomsmanager` directory is shared between all boot environments. `/var/opt/SUNWldm` is a symbolic link to `/var/share/ldomsmanager` for backward-compatibility purposes.

Factory Default Configuration and Disabling Domains

The initial configuration, in which 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) that might have been assigned to other domains.

This section describes how to remove all guest domains, remove all domain configurations, and revert the configuration to the factory default.

▼ How to Remove All Guest Domains

1. **Stop all domains.**

   ```
   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
   ```

How to Remove All Domain Configurations

1. **List all the 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.

How to Restore the Factory Default Configuration

1. **Select the factory default configuration.**

   ```
   primary# ldm set-config factory-default
   ```

2. **Stop the control domain.**

   ```
   primary# shutdown -i5 -g0 -y
   ```

3. **Perform a power cycle of the system to load the factory default configuration.**

   ```
   -> stop /SYS
   -> start /SYS
   ```

How to Disable the Logical Domains Manager

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 action disables some services, such as error reporting and 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, you cannot re-enable power management. In addition, some system management or monitoring tools rely on the Logical Domains Manager.

- **Disable the Logical Domains Manager from the control domain.**

  ```
  primary# svcadm disable ldmd
  ```

▼ How to Restore the Factory Default Configuration From the Service Processor

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. **Perform a power cycle of the system to load the factory default configuration.**

   ```
   -> reset /SYS
   ```

PART II

Optional Oracle VM Server for SPARC Software

This part introduces optional software and features that you can use with the Oracle VM Server for SPARC 3.3 software.

◆ ◆ ◆ CHAPTER 18

Using Oracle VM Server for SPARC Templates

This chapter covers the following topics:

Oracle VM Server for SPARC Templates Overview

The Oracle VM Server for SPARC Templates commands enable you to create, deploy, and configure Oracle VM Server for SPARC templates for SPARC systems. These commands are based on the OVF template specification that includes disk image files and an XML descriptor of properties that is contained in an archive (`.ova`).

You run these commands in the control domain of an Oracle VM Server for SPARC system. In addition to running the commands on the command line, you can run them in a fully automated fashion or from other programs as part of a larger workflow.

The Oracle VM Server for SPARC template commands are:

- `ovmtconfig` – Performs configuration operations directly on a domain's file system. The command backmounts the domain disks and associated ZFS file systems and then runs commands on the file systems. These commands can be further controlled by properties files and by specifying property name-value pairs. See the `ovmtconfig(1M)` man page.

  Note that the `ovmtconfig` command requires that the guest domain run a version of the Oracle Solaris OS that is not newer than the version running on the control domain.
- `ovmtcreate` – Creates a template from an existing Oracle VM Server for SPARC domain. See the `ovmtcreate(1M)` man page.
- `ovmtdeploy` – Creates a domain from an Oracle VM Server for SPARC template. See the `ovmtdeploy(1M)` man page.

- `ovmtlibrary` – Manages a database and file-system-based repository for Oracle VM Server for SPARC templates by organizing files, and storing, retrieving, and editing information in the database. See the `ovmtlibrary(1M)` man page.
- `ovmtprop` – Enables you to view and set Oracle Solaris OS properties. The property is specified as a name-value pair. See the `ovmtprop(1M)` man page.

Installing the Oracle VM Server for SPARC Template Utilities

To install the Oracle VM Server for SPARC template utilities software package on the control domain, you must first download and install the Oracle VM Server for SPARC 3.3 software on the control domain. See Chapter 2, "Installing Software," in *Oracle VM Server for SPARC 3.3 Installation Guide*.

Then, install the Oracle VM Server for SPARC template utilities software package, `ovmtutils`.

```
# pkg install -v -g IPS-package-directory/ldoms.repo ovmtutils
```

Oracle VM Server for SPARC Template Lifecycle

This section describes each stage of the template creation process, the actions that are taken, and how to use the Oracle VM Server for SPARC template utilities to assist in the process:

Note – The creation and development of templates with applications and first-boot scripts is an iterative process. Take care to synchronize all aspects of the configuration by using a source code management system to manage the scripts and properties.

The following describes the stages of the template-creation process, the actions that are taken, and how to use the Oracle VM Server for SPARC template utilities to assist in the process:

1. **Authoring a template.** While pre-built, generic templates are available, you can create a custom template from an existing domain. This domain must have all of the operating system components, application software, and other utilities that you want fully installed.

   Typically, the environment is configured as fully as possible, with only a small number of actions required to finalize the environment. Any domain settings such as memory, virtual CPU, virtual networking, and disks should reflect the deployment that you want.

   At this stage, you create one or more "first-boot" scripts. Include these scripts in the environment that performs the final configuration based on the properties that you supply. Be sure to record and describe these properties in a README file for each template.

Note – If any first-boot scripts access domain variables, ensure that the `ovmtprop` utility is installed in the guest domain.

2. **Creating a template.** Before you create a template, ensure that the source domain environment is not configured so that it can be configured later by prescribed actions that are often part of the first-boot scripts.

   For example, perform the following steps:

   - Remove any application-specific configurations to be re-created later.
   - Use default values for configuration files.
   - Ensure that you reset any Oracle Solaris OS configuration information such as system name, network configuration, and passwords. This configuration information is later supplied by property values and configuration scripts.
   - Export any zpools other than the root file system so that they can be recognized by new domains.

   After you take these steps, you can shut down the domain and run the `ovmtcreate` utility to create a template from the domain.

3. **Specifying the name of the template.** Use the following format:

   technology`.`*OS*`.`*application*`.`*architecture*`.`*build*`.ova`

   For example, the following template name is for a domain that runs build 2 of the Oracle Solaris 11.2 OS on a SPARC platform and that runs version 12.1.2 of the WebLogic server: `OVM_S11.2_WLS12.1.2_SPARC_B2.ova`

4. **Distributing the template.** The template is a single file with a `.ova` extension. The file contains the compressed disk images and the metadata that are required for deploymment. The template also contains a manifest file of payload file checksums, which might be combined with an overall archive checksum to validate that the content has not been changed since distribution.

   You can distribute the template by using web-based services or maintain a central repository rather than duplicating templates.

5. **Deploying the template.** Because the template captures only those aspects of a system that are seen by the source domain, you must understand which services must be present to support the deployment of the template.

   The required services include the following:

   - One or more virtual switches to appropriate interfaces to which virtual networks from the template might be attached
   - Virtual disk services
   - Console services

- Sufficient virtual CPU and memory to accommodate the template requirements

While the `ovmtdeploy` utility can override many of these settings, the minimum values that are supplied with a template represent the baseline requirements.

You can use the `ovmtdeploy` utility to automatically extract, decompress, and copy the virtual disks to deployment directories, and to build the various virtual devices that the template describes.

At this point, you could start the domain but you might need to perform some manual configuration steps by using the domain console before the domain is fully functional.

6. **Automatically configuring the domain.** The configuration of a domain that is created by a template consists of several types of actions. For example, you might specify property name-value pairs to provide first-boot scripts with the information to configure. You might also back-mount virtual disks to the control domain to perform actions on the domain file systems such as copying configuration files.

   The `ovmtconfig` utility automates these domain-configuration activities and enables you to specify the actions to take and the properties to use to configure a domain by specifying one or more command scripts and property files.

   To configure the Oracle Solaris OS, the `ovmtconfig` utility back-mounts the domain's root file system and creates an `sc_profile.xml` file from the supplied configuration scripts and properties. This profile enables the Oracle Solaris OS to configure itself on first boot.

7. **First configuration.** Following the successful configuration of the Oracle Solaris OS and first boot, you must configure any installed applications. During the configuration phase, the `ovmtconfig` utility passes configuration information to the deployed domain by using one of the following methods:

   - **Direct action –** The `ovmtconfig` utility back-mounts the guest domain file systems to the control domain and takes direct action on the files and file systems. Actions might include the creation of configuration files or the copying of system binaries. These actions are described in the scripts that you supply to the `ovmtconfig` utility.

     These actions do not typically include processes that are designed to run in the guest domain because such actions might affect the control domain. Use the `ovmtconfig -c` command to specify the commands to run.

   - **Domain variables –** In addition to a local properties file, you can set domain variables by running the `ovmtconfig` utility in the control domain that can then be used by the `ovmtprop` utility in the guest domain. This method enables first-boot scripts to access the properties directly and provides configuration information directly to the guest domain after the configuration completes.

Caution – Do not use unencrypted properties to pass sensitive information to the domain such as passwords. Properties other than those that are used to configure the Oracle Solaris OS are passed to the domain as `ldm` variables in clear text. These property values are visible to a user on the control domain who is authorized to execute `ldm` commands and to a user who is logged in to the deployed domain.

To work around this issue, manually remove the domain variable after you have deployed the guest domain and the application is fully configured.

A domain variable is prefixed with the `OVMTVAR_`*ID* string, where *ID* is four digits. This example shows a domain variable declaration for the system netmask: `OVMTVAR_1006=ovmt.prop.key.com.oracle.solaris.network.netmask.0=24`.

From the control domain, use the `ldm list-variable | grep OVMTVAR_` command to list the domain variables. Then, remove any variables with sensitive information by using the `ldm remove-variable OVMTVAR_`*ID* command.

For example, you might automate a change of a configuration aspect that does not have network access by using a supervisor script that runs `ovmtprop` in the guest and by running `ovmtconfig -v` from the control domain.

At this point, the domain should be fully configured and operational.

Oracle VM Server for SPARC Template Examples

This section shows examples of the following Oracle VM Server for SPARC template tasks:

- Example 18–1
- Example 18–2
- Example 18–3
- Example 18–4

EXAMPLE 18–1 Creating an Oracle VM Server for SPARC Template

The following `ovmtcreate` command creates a template based on the `ldg1` domain called `ovmtcreate_example`. Note that the resulting template name has the `.ova` suffix.

```
# ovmtcreate -d ldg1 -o ovmtcreate_example
Oracle Virtual Machine for SPARC Template Creation Utility
ovmtcreate Version: 3.3.0.0.12
Copyright (c) 2014, 2015, Oracle and/or its affiliates. All rights reserved.

STAGE 1 - EXAMINING SYSTEM AND ENVIRONMENT
------------------------------------------
Performing platform & prerequisite checks
```

EXAMPLE 18–1 Creating an Oracle VM Server for SPARC Template *(Continued)*

```
Checking user permissions
Checking for required packages
Checking for required services
Checking directory permissions

STAGE 2 - ANALYZING DOMAIN
--------------------------
Retrieving and processing attributes
Checking domain state
Getting domain resource settings
Discovering network topology
Discovering disk topology

STAGE 3 - ARCHIVE CREATION
--------------------------
Checking destination and current directory capacity
Compressing disk image
Creating XML configuration
Calculating manifest checksums
Creating archive file
Checking archive

PROCESS COMPLETED
-----------------
Started: Tue Aug 18 15:29:14 PDT 2015
Completed: Tue Aug 18 15:41:25 PDT 2015
Elapsed time: 0:12:11
```

EXAMPLE 18–2 Configuring Oracle VM Server for SPARC Template Properties

You can use the `ovmtconfig` and `ovmtprop` utilities to specify Oracle VM Server for SPARC template property values and Oracle Solaris OS property values, respectively.

- The following `ovmtconfig` command performs configuration operations directly on the `ldg1` domain's file system.

  The `-c` option specifies the `/opt/ovmtutils/share/scripts/ovmt_s11_scprofile.sh` script to set property values. The `-p` option specifies particular values for the `com.oracle.solaris.network.ipaddr` and `com.oracle.solaris.system.computer-name` properties.

```
# ovmtconfig -v -d ldg1 -f -s \
-c /opt/ovmtutils/share/scripts/ovmt_s11_scprofile.sh \
-p com.oracle.solaris.network.ipaddr.0=10.153.118.211,\
com.oracle.solaris.system.computer-name=system1

Oracle Virtual Machine for SPARC Configuration Utility
ovmtconfig Version: 3.3.0.0.12
Copyright (c) 2014, 2015, Oracle and/or its affiliates. All rights reserved.

STAGE 1/7 - EXAMINING SYSTEM AND ENVIRONMENT
--------------------------------
Checking operating system
Checking platform type
```

EXAMPLE 18–2 Configuring Oracle VM Server for SPARC Template Properties *(Continued)*

```
Checking user permissions
Checking packages
Checking host domain name
Checking host domain type
Checking services

STAGE 2/7 - PROCESSING COMMAND LINE OPTIONS
-------------------------------
Parsing individual properties
Creating consolidated properties list

STAGE 3/7 - ANALYZING TARGET DOMAIN
-----------------------
Stopping domain ldg1
Analyzing domain disk topology for domain ldg1
Discovering 1 volumes for vDisks
Examining 1 backend devices
unbinding domain ldg1
Creating 1 virtual disks for back mount
Created virtual disk 0

STAGE 4/7 - PERFORMING BACKMOUNT
--------------------
Finding Solaris device for vdisks
Importing zpools for 1 Solaris devices
Detected conflicting zpool name, attempting rename
Getting boot file system for properties in 1 zpool
Setting properties in 1 zpools
Mounting ZFS file systems
Mounting ZFS found in zpool rpool_1

STAGE 5/7 - PERFORMING ACTIONS ON TARGET DOMAIN
-----------------------------------

STAGE 6/7 - UNMOUNTING AND RESTORING DOMAIN STATE
-----------------------------------
Rolling back commands DEBUG [20150819-07:02:42]: Rolling back 8 /usr/sbin/zfs unmount -f rpool_1/ROOT/solaris/var
completed

STAGE 7/7 - SETTING TARGET DOMAIN ENVIRONMENT
--------------------------------
Checking 2 properties to set as domain variables
/opt/ovmtutils/bin/agent/lib/ldoms/ldmxml.py:1692: FutureWarning: The behavior of this method will change in futu
'len(elem)' or 'elem is not None' test instead.
if outgoing:
.. Starting domain
Process completed
```

- The following `ovmtprop` command specifies Oracle Solaris OS property values.

```
# ovmtprop set-prop com.oracle.solaris.system.computer-name=test ldg1

Oracle Virtual Machine for SPARC Properties Utility
ovmtprop Version: 3.3.0.0.12
Copyright (c) 2014, 2015, Oracle and/or its affiliates. All rights reserved.

/opt/ovmtutils/bin/agent/lib/ldoms/ldmxml.py:1692: FutureWarning: The behavior of this method will change in futu
```

EXAMPLE 18–2 Configuring Oracle VM Server for SPARC Template Properties *(Continued)*

```
specific 'len(elem)' or 'elem is not None' test instead.
if outgoing:
solaris property set via VM-API
```

Use the `ldm list -l` command to verify that the value for the `com.oracle.solaris.system.computer-name` property is `test`.

```
# ldm ls -l ldg1
NAME STATE  FLAGS  CONS VCPU MEMORY UTIL NORM UPTIME
ldg1 active -n---- 5000 16   16G    0.0% 0.0% 2h 7m
..
CONSTRAINT

VARIABLES
auto-boot?=true

VMAPI TO GUEST
com.oracle.solaris.fmri.count=0
com.oracle.solaris.system.computer-name=test
```

EXAMPLE 18–3 Deploying Oracle VM Server for SPARC Templates

The following `ovmtdeploy` command creates a domain called `ldg1` by using the `ovmtcreate_example.ova` Oracle VM Server for SPARC template in the `/export/ovmtdeploy` directory.

```
# ovmtdeploy -d ldg1 -o /export/ovmtdeploy ovmtcreate_example.ova
Oracle Virtual Machine for SPARC Deployment Utility
ovmtdeploy Version 3.3.0.0.12
Copyright (c) 2014, 2015, Oracle and/or its affiliates. All rights reserved.

STAGE 1 - EXAMINING SYSTEM AND ENVIRONMENT
------------------------------------------
Checking user privilege
Performing platform & prerequisite checks
Checking for required services
Named resourced available

STAGE 2 - ANALYZING ARCHIVE & RESOURCE REQUIREMENTS
---------------------------------------------------
Checking .ova format and contents
Validating archive configuration
Checking sufficient resources present
WARNING: Virtual switch primary-vsw0 already exists

STAGE 3 - EXTRACTING ARCHIVE
----------------------------
Extracting archive
Validating checksums
Decompressing disk image(s)

STAGE 4 - DEPLOYING DOMAIN
--------------------------
Creating domain and adding resources
```

EXAMPLE 18–3 Deploying Oracle VM Server for SPARC Templates *(Continued)*

```
Validating deployment
Domain created:
```

The `ldm list` output shows that you have created a new domain called `ldg1`.

```
# ldm list
NAME    STATE  FLAGS  CONS VCPU MEMORY UTIL NORM UPTIME
primary active -n-cv- UART 8    40G    1.4% 1.1% 6d 2h 18m
ldg1    active -n---- 5000 8    8G     41%  38%  28s
```

EXAMPLE 18–4 Managing the Oracle VM Server for SPARC Template Library

The `ovmtlibrary` command manages a database and file-system-based repository for Oracle VM Server for SPARC templates by organizing files and by storing, retrieving, and editing information in the database.

- The following command creates a template library in `export/user1/ovmtlibrary_example`:

  ```
  # ovmtlibrary -c init -l /export/user1/ovmtlibrary_example

  Oracle Virtual Machine for SPARC Template Library Utility
  ovmtlibrary Version: 3.3.0.0.12
  Copyright (c) 2015 Oracle and/or its affiliates. All rights reserved.
  Init complete
  ```

- The following command stores the `sol-11_2-ovm-2-sparc.ova` template in the `export/user1/ovmtlibrary_example` library:

```
# ovmtlibrary -c store -d "ovmtlibrary example" -o http://system1.example.com/s11.2/templates/sol-11_2-ovm-2-spar

Oracle Virtual Machine for SPARC Template Library Utility
ovmtlibrary Version: 3.3.0.0.12
Copyright (c) 2015 Oracle and/or its affiliates. All rights reserved.

Templates present in path "/export/user1/ovmtlibrary_example"

event id is 2

********************************************************************************
converted 'http://system1.example.com/s11.2/templates/sol-11_2-ovm-2-sparc.ova' (646) ->
'http://system1.example.com/s11.2/templates/sol-11_2-ovm-2-sparc.ova' (UTF-8)
--2015-08-18 16:37:17-- http://system1.example.com/s11.2/templates/sol-11_2-ovm-2-sparc.ova
Resolving system1.example.com (system1.example.com)... 10.134.127.18
Connecting to system1.example.com (system1.example.com)|10.134.127.18|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1018341888 (971M) [text/plain]
Saving to: '/export/user1/ovmtlibrary_example/repository/templates/1/1/sol-11_2-ovm-2-sparc.ova'

/export/user1/ovmtlibrary_example/repo 100%
[================================================================================>] 971.17M 6.05MB/s in 5m 37s
2015-08-18 16:42:55 (2.88 MB/s) - '/export/user1/ovmtlibrary_example/repository/templates/1/1/sol-11_2-ovm-2-spar
[1018341888/1018341888]
********************************************************************************
```

EXAMPLE 18–4 Managing the Oracle VM Server for SPARC Template Library *(Continued)*

```
Download complete
Extracting the ova file...
Extract complete
Decompress file System.img.gz
Store complete
```

- The following command lists the contents of the `export/user1/ovmtlibrary_example` library:

```
# ovmtlibrary -c list -l /export/user1/ovmtlibrary_example
Oracle Virtual Machine for SPARC Template Library Utility
ovmtlibrary Version: 3.3.0.0.12
Copyright (c) 2015 Oracle and/or its affiliates. All rights reserved.

Templates present in path "/export/user1/ovmtlibrary_example"

ID Name                         Version Description         Date
--------------------------------------------------------------------------------
1  sol-11_2-ovm-2-sparc         1       ovmtlibrary example 2015-08-18
```

- The following command shows a detailed listing of the `export/user1/ovmtlibrary_example` library:

```
# ovmtlibrary -c list -i 1 -o -l /export/user1/ovmtlibrary_example
Oracle Virtual Machine for SPARC Template Library Utility
ovmtlibrary Version: 3.3.0.0.12
Copyright (c) 2015 Oracle and/or its affiliates. All rights reserved.

Templates present in path "/export/user1/ovmtlibrary_example"

ID Name                           Type Path                          Size(bytes)
--------------------------------------------------------------------------------
1  sol-11_2-ovm-2-sparc.ova       ova /export/user1/ovmtlibrary_example/repository/templates/1/1/sol-11_2-ovm-2-spar
2  sol-11_2-ovm-sparc.ovf         ovf /export/user1/ovmtlibrary_example/repository/templates/1/1/sol-11_2-ovm-sparc.
3  sol-11_2-ovm-sparc.mf          mf /export/user1/ovmtlibrary_example/repository/templates/1/1/sol-11_2-ovm-sparc.m
4  System.img                     img /export/user1/ovmtlibrary_example/repository/templates/1/1/System.img 21474836
```

◆ ◆ ◆ CHAPTER 19

Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool

This chapter covers the following topics:

Note – The `ldmp2v` utility will no longer be updated to address bug fix or enhancement requests. This utility will no longer be supported, but will continue to be included and documented as part of the Oracle VM Server for SPARC software.

Oracle VM Server for SPARC P2V Tool Overview

The Oracle VM Server for SPARC Physical-to-Virtual (P2V) Conversion tool automatically converts an existing physical system to a virtual system that runs the Oracle Solaris 10 OS in a logical domain on a chip multithreading (CMT) system. You can run the `ldmp2v` command from a control domain that runs the Oracle Solaris 11 OS to convert one of the following source systems to a logical domain:

- Any `sun4u` SPARC based system that runs at least the Solaris 8, Solaris 9, or Oracle Solaris 10 OS
- Any `sun4v` system that runs the Oracle Solaris 10 OS but does not run in a logical domain

Note – The `ldmp2v` command does not support any SPARC based system that runs the Oracle Solaris 10 OS with a ZFS root or the Oracle Solaris 11 OS.

The conversion from a physical system to a virtual system is performed in the following phases:

- **Collection phase.** Runs on the physical source system. In the `collect` phase, a file system image of the source system is created based on the configuration information that it collects about the source system.
- **Preparation phase.** Runs on the control domain of the target system. In the `prepare` phase, a logical domain is created on the target system based on the configuration information collected in the `collect` phase. The file system image is restored to one or more virtual disks. You can use the P2V tool to create virtual disks on plain files or ZFS volumes. You can also create virtual disks on physical disks or LUNs, or on volume manager volumes that you created. The image is modified to enable it to run as a logical domain.
- **Conversion phase.** Runs on the control domain of the target system. In the `convert` phase, the created logical domain is converted into a logical domain that runs the Oracle Solaris 10 OS by using the standard Oracle Solaris upgrade process.

For information about the P2V tool, see the `ldmp2v(1M)` man page.

The following sections describe how the conversion from a physical system to a virtual system is performed.

Collection Phase

The Collection phase runs on the system to be converted. To create a consistent file system image, ensure that the system is as quiet as possible and that all applications are stopped. The `ldmp2v` command creates a backup of all mounted UFS file systems, so ensure that any file systems to be moved to a logical domain are mounted. You can exclude mounted file systems that you do not want to move, such as file systems on SAN storage or file systems that will be moved by other means. Use the `-x` option to exclude such file systems. File systems that are excluded by the `-x` option are not re-created on the guest domain. You can use the `-O` option to exclude files and directories.

No changes are required on the source system. The only requirement is the `ldmp2v` script that was installed on the control domain. Ensure that the `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. You can then upgrade the system to the Oracle Solaris 10 OS by using the normal Oracle Solaris upgrade process. Modifications include updating the `/etc/vfstab` file to account for new disk names. Any Oracle Solaris Volume Manager or Veritas Volume Manager (VxVM) encapsulated boot disks are automatically unencapsulated during this process. When a disk is unencapsulated, it is converted into plain disk slices. If VxVM is installed on the source system, the P2V process disables VxVM on the created guest domain.
- **Non-automatic mode.** You must create the virtual disks and restore the file system data manually. This mode enables you to change the size and number of disks, the partitioning, and the file system layout. The preparation phase in this mode runs only the logical domain creation and the OS image modification steps on the file system.
- **Cleanup mode.** Removes a logical domain and all of the underlying back-end devices that are created by `ldmp2v`.

Conversion Phase

In the conversion phase, the logical domain uses the Oracle Solaris upgrade process to upgrade to the Oracle Solaris 10 OS. The upgrade operation removes all existing packages and installs the Oracle Solaris 10 `sun4v` packages, which automatically performs a `sun4u-to-sun4v` conversion. The `convert` phase can use an Oracle Solaris DVD ISO image or a network installation image. On Oracle Solaris 10 systems, you can also use the Oracle Solaris JumpStart feature to perform a fully automated upgrade operation.

Back-End Devices

You can create virtual disks for a guest domain on a number of back-end types: files (`file`), ZFS volumes (`zvol`), physical disks or LUNs (`disk`), or volume manager volumes (`disk`). The `ldmp2v` command automatically creates files or ZFS volumes of the appropriate size if you specify `file` or `zvol` as the back-end type in one of the following ways:

- By using the `-b` option
- By specifying the value of the `BACKEND_TYPE` parameter in the `/etc/ldmp2v.conf` file

The `disk` back-end type enables you to use a physical disk, LUN, or volume manager volume (Oracle Solaris Volume Manager and Veritas Volume Manager (VxVM)) as a back-end device for virtual disks. You must create the disk or volume with an appropriate size prior to beginning the `prepare` phase. For a physical disk or LUN, specify the back-end device as slice 2 of the block or character device of the disk, such as `/dev/dsk/c0t3d0s2`. For a volume manager volume, specify the block or character device for the volume, such as `/dev/md/dsk/d100` for Oracle Solaris Volume Manager or `/dev/vx/dsk/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 and Configuring the Oracle VM Server for SPARC P2V Tool

The Oracle VM Server for SPARC P2V Tool package must be installed and configured *only* on the control domain of the target system. You do not need to install the package on the source system. Instead, you can simply copy the `/usr/sbin/ldmp2v` script from the target system to the source system.

Note – The `ldmp2v` is installed by default from the `ldomsmanager` package.

Prerequisites for using the SPARC P2V Tool

Before you can run the Oracle VM Server for SPARC P2V tool, ensure that the following conditions are met:

- The following Flash Archive patches are installed on the source system:
  - **For the Solaris 8 OS:** At least patch ID 109318-34
  - **For the Solaris 9 OS:** At least patch ID 113434-06
- The control domain of the target system runs at least the Oracle Solaris 11 OS
- Guest domains run at least the Oracle Solaris 10 5/08 OS
- The source system runs at least the Solaris 8 OS

In addition to these prerequisites, configure an NFS file system to be shared by both the source and target systems. This file system should be writable by `root`. However, if a shared file system is not available, use a local file system that is large enough to hold a file system dump of the source system on both the source and target systems.

Limitations of Using the SPARC P2V Tool

The Oracle VM Server for SPARC P2V tool has the following limitations:

- Only UFS file systems are supported.
- Only plain disks (`/dev/dsk/c0t0d0s0`), Oracle Solaris Volume Manager metadevices (`/dev/md/dsk/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 Oracle Solaris 10 OS. However, you can also use the P2V tool to convert Solaris 8 and Solaris 9 operating systems that use VxVM.
- Oracle Solaris 10 systems that have zones can be converted if the zones are detached by using the `zoneadm detach` command prior to running the `ldmp2v collect` operation. After the P2V conversion completes, use the `zoneadm attach` command to reattach the zones that have been created on the guest domain. For information about performing these steps on a guest domain, see Chapter 23, "Moving and Migrating Non-Global Zones (Tasks)," in *System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones* .

Note – The P2V tool does not update any zone configuration, such as the zone path or network interface, nor does the tool move or configure the storage for the zone path. You must manually update the zone configuration and move the zone path on the guest domain. See Chapter 23, "Moving and Migrating Non-Global Zones (Tasks)," in *System Administration Guide: Oracle Solaris Containers-Resource Management and Oracle Solaris Zones* .

How to Configure the Oracle VM Server for SPARC P2V Tool

The `ldmp2v` command is installed by default when you install the `ldomsmanager` package.

1. **Become an administrator.**

   For Oracle Solaris 11.3, see Chapter 1, "About Using Rights to Control Users and Processes," in *Securing Users and Processes in Oracle Solaris 11.3* .

2. **Create the `/etc/ldmp2v.conf` file and configure the following default properties:**

   - `VDS` – Name of the virtual disk service, such as `VDS="primary-vds0"`
   - `VSW` – Name of the virtual switch, such as `VSW="primary-vsw0"`
   - `VCC` – Name of the virtual console concentrator, such as `VCC="primary-vcc0"`
   - `BACKEND_TYPE` – Back-end type of `zvol`, `file`, or `disk`
   - `BACKEND_SPARSE` – Determines whether to create back-end devices as sparse volumes or files (`BACKEND_SPARSE="yes"`) or non-sparse volumes or files (`BACKEND_SPARSE="no"`)
   - `BACKEND_PREFIX` – Location to create virtual disk back-end devices

When BACKEND_TYPE="zvol", specify the BACKEND_PREFIX value as a ZFS dataset name. When BACKEND_TYPE="files", the BACKEND_PREFIX value is interpreted as a path name of a directory that is relative to /.

For example, BACKEND_PREFIX="tank/ldoms" would result in having ZVOLs created in the tank/ldoms/*domain-name* dataset, and files created in the /tank/ldoms/*domain-name* subdirectory.

The BACKEND_PREFIX property is not applicable to the disk back end.

- BOOT_TIMEOUT – Timeout for Oracle Solaris OS boot in seconds

For more information, see the ldmp2v.conf.sample configuration file that is part of the downloadable bundle.

Using the ldmp2v Command

This section includes examples for the three phases of conversion.

EXAMPLE 19–1 Collection Phase Examples

The following examples show how you might use the ldmp2v collect command.

- **Sharing an NFS-mounted file system.** The following example shows the simplest way to perform the collect step where the source and target systems share an NFS-mounted file system.

  As superuser, ensure that all required UFS file systems are mounted.

```
volumia# df -k
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c1t1d0s0    16516485  463289 15888032     3%    /
/proc                      0       0       0     0%    /proc
fd                         0       0       0     0%    /dev/fd
mnttab                     0       0       0     0%    /etc/mnttab
/dev/dsk/c1t1d0s3    8258597    4304 8171708     1%    /var
swap                 4487448      16 4487432     1%    /var/run
swap                 4487448      16 4487432     1%    /tmp
/dev/dsk/c1t0d0s0    1016122       9  955146     1%    /u01
vandikhout:/u1/home/dana
                     6230996752 1051158977 5179837775    17%    /home/dana
```

  The following example shows how to run the collection tool when the source and target systems share an NFS-mounted file system:

```
volumia# ldmp2v collect -d /home/dana/volumia
Collecting system configuration ...
Archiving file systems ...
Determining which filesystems will be included in the archive...
Creating the archive...
895080 blocks
Archive creation complete.
```

EXAMPLE 19–1 Collection Phase Examples *(Continued)*

- **Not sharing an NFS-mounted file system.** When the source and target systems do not share an NFS-mounted file system, the file system image can be written to local storage and later copied to the control domain. The Flash Archive utility automatically excludes the archive that it creates.

```
volumia# ldmp2v collect -d /var/tmp/volumia
Collecting system configuration ...
Archiving file systems ...
Determining which filesystems will be included in the archive...
Creating the archive...
895080 blocks
Archive creation complete.
```

  Copy the flash archive and the `manifest` file from the `/var/tmp/volumia` directory to the target system.

Tip – In some cases, `ldmp2v` might show `cpio` command errors. Most commonly, these errors generate messages such as `File size of etc/mnttab has increased by 435`. You can ignore messages that pertain to log files or to files that reflect the system state. Be sure to review all error messages thoroughly.

- **Skip file-system backup step.** If you already create backups of the system using a third-party backup tool such as NetBackup, you can skip the file system backup step by using the `none` archiving method. When you use this option, only the system configuration manifest is created.

```
volumia# ldmp2v collect -d home/dana/p2v/volumia -a none
Collecting system configuration ...
The following file system(s) must be archived manually: / /u01 /var
```

  Note that if the directory specified by `-d` is not shared by the source and target systems, you must copy the contents of that directory to the control domain. The directory contents must be copied to the control domain prior to the preparation phase.

EXAMPLE 19–2 Preparation Phase Examples

The following examples show how you might use the `ldmp2v prepare` command.

- The following example creates a logical domain called `volumia` by using the defaults configured in `/etc/ldmp2v.conf` while keeping the MAC addresses of the physical system:

```
# ldmp2v prepare -d /home/dana/p2v/volumia -o keep-mac volumia
Creating vdisks ...
Creating file systems ...
Populating file systems ...
Modifying guest domain OS image ...
Removing SVM configuration ...
Unmounting guest file systems ...
```

EXAMPLE 19–2 Preparation Phase Examples *(Continued)*

```
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 example shows how to completely remove a domain and its back-end devices by using the `-C` option.

```
# ldmp2v prepare -C volumia
Cleaning up domain volumia ...
Removing vdisk disk0 ...
Removing vdisk disk1 ...
Removing domain volumia ...
Removing volume volumia-vol0@primary-vds0 ...
Removing ZFS volume tank/ldoms/volumia/disk0 ...
Removing volume volumia-vol1@primary-vds0 ...
Removing ZFS volume tank/ldoms/volumia/disk1 ...
```

- The following example shows how to resize one or more file systems during P2V by specifying the mount point and the new size with the `-m` option.

```
# ldmp2v prepare -d /home/dana/p2v/volumia -m /:8g volumia
Resizing file systems ...
Creating vdisks ...
Creating file systems ...
Populating file systems ...
Modifying guest domain OS image ...
Removing SVM configuration ...
Modifying file systems on SVM devices ...
Unmounting guest file systems ...
Creating domain volumia ...
Attaching vdisks to domain volumia ...
```

EXAMPLE 19–3 Conversion Phase Examples

The following examples show how you might use the `ldmp2v convert` command.

- **Using a network installation server.** The `ldmp2v convert` command boots the domain over the network by using the specified virtual network interface. You must run the `setup_install_server` and `add_install_client` scripts on the installation server.

  On Oracle Solaris 10 systems, you can use the Oracle Solaris JumpStart feature to perform a fully automated conversion. This feature requires that you create and configure the appropriate `sysidcfg` and profile files for the client on the JumpStart server. The profile should consist of the following lines:

  ```
  install_type    upgrade
  root_device     c0d0s0
  ```

  The `sysidcfg` file is only used for the upgrade operation, so a configuration such as the following should be sufficient:

  ```
  name_service=NONE
  root_password=uQkoXlMLCsZhI
  system_locale=C
  timeserver=localhost
  timezone=Europe/Amsterdam
  terminal=vt100
  security_policy=NONE
  nfs4_domain=dynamic
  auto_reg=disable
  network_interface=PRIMARY {netmask=255.255.255.192
          default_route=none protocol_ipv6=no}
  ```

  For more information about using JumpStart, see *Oracle Solaris 10 1/13 Installation Guide: JumpStart Installations* .

Note – The example `sysidcfg` file includes the `auto_reg` keyword, which was introduced in the Oracle Solaris 10 9/10 release. This keyword is required only if you are running at least the Oracle Solaris 10 9/10 release.

```
# ldmp2v convert -j -n vnet0 -d /p2v/volumia volumia
LDom volumia started
Waiting for Solaris to come up ...
Using Custom JumpStart
Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.

Connecting to console "volumia" in group "volumia" ....
Press ~? for control options ..
SunOS Release 5.10 Version Generic_137137-09 64-bit
Copyright (c) 1983-2010, Oracle and/or its affiliates.  All rights reserved.
Configuring devices.
Using RPC Bootparams for network configuration information.
Attempting to configure interface vnet0...
Configured interface vnet0
```

EXAMPLE 19–3 Conversion Phase Examples *(Continued)*

```
Reading ZFS config: done.
Setting up Java. Please wait...
Serial console, reverting to text install
Beginning system identification...
Searching for configuration file(s)...
Using sysid configuration file
  129.159.206.54:/opt/SUNWjet/Clients/volumia/sysidcfg
Search complete.
Discovering additional network configuration...
Completing system identification...
Starting remote procedure call (RPC) services: done.
System identification complete.
Starting Solaris installation program...
Searching for JumpStart directory...
Using rules.ok from 129.159.206.54:/opt/SUNWjet.
Checking rules.ok file...
Using begin script: Clients/volumia/begin
Using profile: Clients/volumia/profile
Using finish script: Clients/volumia/finish
Executing JumpStart preinstall phase...
Executing begin script "Clients/volumia/begin"...
Begin script Clients/volumia/begin execution completed.
Searching for SolStart directory...
Checking rules.ok file...
Using begin script: install_begin
Using finish script: patch_finish
Executing SolStart preinstall phase...
Executing begin script "install_begin"...
Begin script install_begin execution completed.
WARNING: Backup media not specified.  A backup media (backup_media)
  keyword must be specified if an upgrade with disk space reallocation
  is required

Processing profile

Loading local environment and services

Generating upgrade actions
Checking file system space: 100% completed
Space check complete.

Building upgrade script

Preparing system for Solaris upgrade

Upgrading Solaris: 10% completed
[...]
```

- **Using an ISO image.** The `ldmp2v convert` command attaches the Oracle Solaris DVD ISO image to the logical domain and boots from it. To upgrade, answer all `sysid` prompts and select Upgrade.

EXAMPLE 19–3 Conversion Phase Examples *(Continued)*

Caution – A safety check is performed prior to converting the guest domain. This check ensures that none of the original system's IP addresses are active so as to prevent duplicate active IP addresses on the network. You can use the `-x skip-ping-test` option to skip this safety check. Skipping this check speeds up the conversion process. Use this option *only* if you are certain that no duplicate IP addresses exist, such as when the original host is not active.

The answers to the `sysid` questions are used only for the duration of the upgrade process. This data is not applied to the existing OS image on disk. The fastest and simplest way to run the conversion is to select Non-networked. The `root` password that you specify does not need to match the `root` password of the source system. The system's original identity is preserved by the upgrade and takes effect after the post-upgrade reboot. The time required to perform the upgrade depends on the Oracle Solaris Cluster that is installed on the original system.

```
# ldmp2v convert -i /tank/iso/s10s_u5.iso -d /home/dana/p2v/volumia volumia
Testing original system status ...
LDom volumia started
Waiting for Solaris to come up ...

        Select 'Upgrade' (F2) when prompted for the installation type.
        Disconnect from the console after the Upgrade has finished.

Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.

Connecting to console "volumia" in group "volumia" ....
Press ~? for control options ..
Configuring devices.
Using RPC Bootparams for network configuration information.
Attempting to configure interface vnet0...
Extracting windowing system. Please wait...
Beginning system identification...
Searching for configuration file(s)...
Search complete.
Discovering additional network configuration...
Configured interface vnet0
Setting up Java. Please wait...

Select a Language

   0. English
   1. French
   2. German
   3. Italian
   4. Japanese
   5. Korean
   6. Simplified Chinese
```

EXAMPLE 19–3 Conversion Phase Examples *(Continued)*

```
      7. Spanish
      8. Swedish
      9. Traditional Chinese

   Please make a choice (0 - 9), or press h or ? for help:
   [...]
   - Solaris Interactive Installation --------------------------------------------

     This system is upgradable, so there are two ways to install the Solaris
     software.

     The Upgrade option updates the Solaris software to the new release, saving
     as many modifications to the previous version of Solaris software as
     possible.  Back up the system before using the Upgrade option.

     The Initial option overwrites the system disks with the new version of
     Solaris software.  This option allows you to preserve any existing file
     systems.  Back up any modifications made to the previous version of Solaris
     software before starting the Initial option.

     After you select an option and complete the tasks that follow, a summary of
     your actions will be displayed.

   ------------------------------------------------------------------------------
        F2_Upgrade    F3_Go Back    F4_Initial    F5_Exit    F6_Help
```

Known Issues With the Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool

`ldmp2v convert` Command: VxVM Warning Messages During Boot

Running Veritas Volume Manager (VxVM) 5.*x* on the Oracle Solaris 10 OS is the only supported (tested) version for the Oracle VM Server for SPARC P2V tool. Older versions of VxVM, such as 3.*x* and 4.*x* running on the Solaris 8 and Solaris 9 operating systems, might also work. In those cases, the first boot after running the `ldmp2v convert` command might show warning messages from the VxVM drivers. You can ignore these messages. You can remove the old `VRTS*` packages after the guest domain has booted.

```
Boot device: disk0:a File and args:
SunOS Release 5.10 Version Generic_139555-08 64-bit
Copyright 1983-2009 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
Hostname: normaal
Configuring devices.
/kernel/drv/sparcv9/vxdmp: undefined symbol 'romp'
```

```
WARNING: mod_load: cannot load module 'vxdmp'
WARNING: vxdmp: unable to resolve dependency, module 'misc/ted' not found
/kernel/drv/sparcv9/vxdmp: undefined symbol 'romp'
WARNING: mod_load: cannot load module 'vxdmp'
WARNING: vxdmp: unable to resolve dependency, module 'misc/ted' not found
/kernel/drv/sparcv9/vxio: undefined symbol 'romp'
WARNING: mod_load: cannot load module 'vxio'
WARNING: vxio: unable to resolve dependency, module 'drv/vxdmp' not found
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
WARNING: VxVM vxspec V-5-0-0 vxspec: vxio not loaded. Aborting vxspec load
WARNING: vxspec : CANNOT INITIALIZE vxio DRIVER
NOTICE: VxVM not started
```

Upgrade Option Not Presented When Using `ldmp2v prepare -R`

The Oracle Solaris Installer does not present the Upgrade option when the partition tag of the slice that holds the root (`/`) file system is not set to `root`. This situation occurs if the tag is not explicitly set when labeling the guest's boot disk. You can use the `format` command to set the partition tag as follows:

```
AVAILABLE DISK SELECTIONS:
0. c0d0 <SUN-DiskImage-10GB cyl 282 alt 2 hd 96 sec 768>
  /virtual-devices@100/channel-devices@200/disk@0
1. c4t2d0 <SUN146G cyl 14087 alt 2 hd 24 sec 848>
  /pci@400/pci@0/pci@1/scsi@0/sd@2,0
2. c4t3d0 <SUN146G cyl 14087 alt 2 hd 24 sec 848>
  /pci@400/pci@0/pci@1/scsi@0/sd@3,0
Specify disk (enter its number)[0]: 0
selecting c0d0
[disk formatted, no defect list found]
format> p


PARTITION MENU:
0      - change '0' partition
1      - change '1' partition
2      - change '2' partition
3      - change '3' partition
4      - change '4' partition
5      - change '5' partition
6      - change '6' partition
7      - change '7' partition
select - select a predefined table
```

```
modify - modify a predefined partition table
name   - name the current table
print  - display the current table
label  - write partition map and label to the disk
!<cmd> - execute <cmd>, then return
quit

partition> 0
Part      Tag    Flag     Cylinders       Size            Blocks
0 unassigned    wm       0              0         (0/0/0)          0

Enter partition id tag[unassigned]: root
Enter partition permission flags[wm]:
Enter new starting cyl[0]: 0
Enter partition size[0b, 0c, 0e, 0.00mb, 0.00gb]: 8g
partition> label
Ready to label disk, continue? y

partition>
```

`ldmp2v` Command: ufsdump Archiving Method Is No Longer Used

Restoring `ufsdump` archives on a virtual disk that is backed by a file on a UFS file system might cause the system to hang. In such a case, the `ldmp2v prepare` command will exit. You might encounter this problem when you manually restore `ufsdump` archives in preparation for the `ldmp2v prepare -R /altroot` command when the virtual disk is a file on a UFS file system. For compatibility with previously created `ufsdump` archives, you can still use the `ldmp2v prepare` command to restore `ufsdump` archives on virtual disks that are not backed by a file on a UFS file system. However, the use of `ufsdump` archives is not recommended.

◆ ◆ ◆ CHAPTER 20

Using Power Management

This chapter contains information about using power management on Oracle VM Server for SPARC systems.

Using Power Management

To enable power management (PM), you first need to set the PM policy in the Oracle Integrated Lights Out Manager (ILOM) 3.0 firmware. This section summarizes the information that you need to be able to use PM with the Oracle VM Server for SPARC software.

For more information about ILOM, see the following:

- "Monitoring Power Consumption" in the *Oracle Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide*
- *Oracle Integrated Lights Out Manager (ILOM) 3.0 Feature Updates and Release Notes*

The power policy governs system power usage at any point in time. The following power policies are supported, assuming that the underlying platform has implemented PM features:

- **Disabled.** Permits the system to use all the power that is available.
- **Performance.** Enables one or more of the following PM features that have a negligible affect on performance:
  - CPU core auto-disabling
  - CPU clock cycle skip
  - CPU dynamic voltage and frequency scaling (DVFS)
  - Coherency link scaling
  - Oracle Solaris Power Aware Dispatcher (PAD)
- **Elastic.** Adapts the system power usage to the current utilization level by using the PM features described in the performance section. For example, the power state of resources is reduced as utilization decreases.

Power Management Features

The PM features are as follows:

- **CPU core auto-disabling.** When the elastic or performance policy is in effect, the Logical Domains Manager automatically disables a CPU core when all the hardware threads (strands) on that core are not bound to a domain. This feature is available only for the UltraSPARC T2, UltraSPARC T2 Plus, SPARC T3, and SPARC T4 platforms.
- **CPU clock cycle skip.**When the elastic policy is in effect, the Logical Domains Manager automatically adjusts the number of clock cycles that execute instructions on the following CPU resources that are bound to domains:
  - Processors (SPARC T3 or SPARC T4 on domains that run the Oracle Solaris 10 or Oracle Solaris 11 OS)
  - Cores (SPARC M5 only on domains that run the Oracle Solaris 10 OS)
  - Core-pairs (SPARC T5 or SPARC M6 only on domains that run the Oracle Solaris 10 OS)
  - SPARC Cache Cluster (SCC) (SPARC T7 series servers and SPARC M7 servers only on domains that run the Oracle Solaris 10 OS)

  The Logical Domains Manager also applies cycle skipping if the processor, core, core-pair, or SCC has no bound strands.
- **CPU dynamic voltage and frequency scaling (DVFS).** When the elastic policy is in effect, the Logical Domains Manager automatically adjusts the clock frequency of processors or SCCs that are bound to domains running the Oracle Solaris 10 OS. The Logical Domains Manager also reduces the clock frequency on SPARC T5, SPARC M5, and SPARC M6 processors that have no bound strands. On SPARC T7 series servers, the clock frequency is reduced on SCCs. This feature is available only on SPARC T5 servers, SPARC T7 series servers, SPARC M5 servers, SPARC M6 servers, and SPARC M7 series servers.
- **Coherency link scaling.** When the elastic policy is in effect, the Logical Domains Manager causes the hypervisor to automatically adjust the number of coherency links that are in use. This feature is only available on SPARC T5-2 systems.
- **Power limit.** You can set a *power limit* on SPARC T3 servers, SPARC T4 servers, SPARC T5 servers, SPARC T7 series servers, SPARC M5 servers, SPARC M6 servers, and SPARC M7 series servers to restrict the power draw of a system. If the power draw is greater than the power limit, PM uses techniques to reduce power. You can use the ILOM service processor (SP) to set the power limit.

  See the following documents:
  - *Oracle Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide*
  - *Oracle Integrated Lights Out Manager (ILOM) 3.0 Feature Updates and Release Notes*

You can use the ILOM interface to set a power limit, grace period, and violation action. If the power limit is exceeded for more than the grace period, the violation action is performed.

If the current power draw exceeds the power limit, an attempt is made to reduce the power state of CPUs. If the power draw drops below the power limit, the power state of those resources is permitted to increase. If the system has the elastic policy in effect, an increase in the power state of resources is driven by the utilization level.

- **Solaris Power Aware Dispatcher (PAD).** A guest domain that runs the Oracle Solaris 11.1 OS uses the power-aware dispatcher (PAD) on SPARC T5 servers, SPARC T7 series servers, SPARC M5 servers, SPARC M6 servers, and SPARC M7 series servers to minimize power consumption from idle or under-utilized resources. PAD, instead of the Logical Domains Manager, adjusts the CPU or SCC clock cycle skip level and DVFS level.

For instructions on configuring the power policy by using the ILOM 3.0 firmware CLI, see "Monitoring Power Consumption" in the *Oracle Integrated Lights Out Manager (ILOM) 3.0 CLI Procedures Guide*.

Viewing Power-Consumption Data

The Power Management (PM) Observability Module and the `ldmpower` command enable you to view CPU thread power-consumption data for your domains.

The PM Observability Module is enabled by default because the `ldmd/pm_observability_enabled` Service Management Facility (SMF) property is set to `true`. See the `ldmd(1M)` man page.

The `ldmpower` command has the following options and operands with which you can customize the power-consumption reporting data:

```
ldmpower [-ehiprstvx | -o hours | -m minutes] | -c resource [-l domain-name[,domain-name[,...]]]
  [interval [count]]
```

For information about the options, see the `ldmpower(1M)` man page.

To run this command as a non-privileged user, you must be assigned the `LDoms Power Mgmt Observability` rights profile. If you already have been assigned the `LDoms Management` or `LDoms Review` rights profile, you automatically have permission to run the `ldmpower` command.

For information about how Oracle VM Server for SPARC uses rights, see "Logical Domains Manager Profile Contents" on page 36.

These rights profiles can be assigned directly to users or to a role that is then assigned to users. When one of these profiles is assigned directly to a user, you must use the `pfexec` command or a profile shell, such as `pfbash` or `pfksh`, to successfully use the `ldmpower` command to view CPU thread power-consumption data. See "Delegating the Management of Logical Domains by Using Rights" on page 33.

The following examples show how to enable the PM Observability Module and show ways in which to gather power-consumption data for the CPUs that are assigned to your domains.

EXAMPLE 20–1 Enabling the Power Management Observability Module

The following command enables the PM Observability Module by setting the `ldmd/pm_observability_enabled` property to `true` if the property is currently set to `false`.

```
# svccfg -s ldmd setprop ldmd/pm_observability_enabled=true
# svcadm refresh ldmd
# svcadm restart ldmd
```

EXAMPLE 20–2 Using a Profile Shell to Obtain CPU Thread Power-Consumption Data by Using Roles and Rights Profiles

- The following example shows how to create the `ldmpower` role with the `LDoms Power Mgmt Observability` rights profile, which permits you to run the `ldmpower` command.

  ```
  primary# roleadd -P "LDoms Power Mgmt Observability" ldmpower
  primary# passwd ldmpower
  New Password:
  Re-enter new Password:
  passwd: password successfully changed for ldmpower
  ```

  This command assigns the `ldmpower` role to the `sam` user.

  ```
  primary# usermod -R ldmpower sam
  ```

  User `sam` assumes the `ldmpower` role and can use the `ldmpower` command. For example:

  ```
  $ id
  uid=700299(sam) gid=1(other)
  $ su ldmpower
  Password:
  $ pfexec ldmpower
  Processor Power Consumption in Watts
  DOMAIN  15_SEC_AVG  30_SEC_AVG  60_SEC_AVG
  primary 75          84          86
  gdom1   47          24          19
  gdom2   10          24          26
  ```

- The following example shows how to use rights profiles to run the `ldmpower` command.

  Assign the rights profile to a user.

  ```
  primary# usermod -P +"LDoms Power Mgmt Observability" sam
  ```

  The following commands show how to verify that the user is `sam` and that the `All`, `Basic Solaris User`, and `LDoms Power Mgmt Observability` rights profiles are in effect.

  ```
  $ id
  uid=702048(sam) gid=1(other)
  $ profiles
  All
  Basic Solaris User
  LDoms Power Mgmt Observability
  $ pfexec ldmpower
  ```

EXAMPLE 20–2 Using a Profile Shell to Obtain CPU Thread Power-Consumption Data by Using Roles and Rights Profiles *(Continued)*

```
Processor Power Consumption in Watts
DOMAIN  15_SEC_AVG  30_SEC_AVG  60_SEC_AVG
primary 75          84          86
gdom1   47          24          19
gdom2   10          24          26
```

EXAMPLE 20–3 Viewing Processor Power-Consumption Data

The following examples show how to use the `ldmpower` to report processor power-consumption data for your domains.

- The following command shows the 15-second, 30-second, and 60-second rolling average processor power-consumption data for all domains:

```
primary# ldmpower
Processor Power Consumption in Watts
DOMAIN  15_SEC_AVG  30_SEC_AVG  60_SEC_AVG
primary 75          84          86
gdom1   47          24          19
gdom2   10          24          26
```

- The following command shows extrapolated power-consumption data for all the domains: `primary`, `gdom1`, and `gdom2`.

```
primary# ldmpower -x
System Power Consumption in Watts
DOMAIN  15_SEC_AVG  30_SEC_AVG  60_SEC_AVG
primary 585/57.47%  701/68.96%  712/70.22%
gdom1   132/12.97%  94/9.31%    94/9.30%
gdom2   298/29.27%  218/21.47%  205/20.22%
```

- The following command shows the instantaneous processor power-consumption data for the `gdom2` and `gdom5` domains. It reports the data every ten seconds for five times.

```
primary# ldmpower -itl gdom2,gdom5 10 5
Processor Power Consumption in Watts
DOMAIN          TIMESTAMP               INSTANT
gdom2           2013.05.17 11:14:45     13
gdom5           2013.05.17 11:14:45     24

gdom2           2013.05.17 11:14:55     18
gdom5           2013.05.17 11:14:55     26

gdom2           2013.05.17 11:15:05     9
gdom5           2013.05.17 11:15:05     16

gdom2           2013.05.17 11:15:15     15
gdom5           2013.05.17 11:15:15     19

gdom2           2013.05.17 11:15:25     12
gdom5           2013.05.17 11:15:25     18
```

- The following command shows the average power-consumption data for the last 12 hours for all domains. Data is shown at one-hour intervals starting from the last requested hourly calculation.

EXAMPLE 20–3 Viewing Processor Power-Consumption Data *(Continued)*

```
primary# ldmpower -eto 12
Per domain MINIMUM and MAXIMUM power consumption ever recorded:
primary         2013.05.17 08:53:06     3               Min Processors
primary         2013.05.17 08:40:44     273             Max Processors
gdom1           2013.05.17 09:56:35     2               Min Processors
gdom1           2013.05.17 08:53:06     134             Max Processors
gdom2           2013.05.17 10:31:55     2               Min Processors
gdom2           2013.05.17 08:56:35     139             Max Processors

primary         2013.05.17 08:53:06     99              Min Memory
primary         2013.05.17 08:40:44     182             Max Memory
gdom1           2013.05.17 09:56:35     13              Min Memory
gdom1           2013.05.17 08:53:06     20              Max Memory
gdom2           2013.05.17 10:31:55     65              Min Memory
gdom2           2013.05.17 08:56:35     66              Max Memory

Processor Power Consumption in Watts
12 hour's worth of data starting from 2013.05.16 23:17:02
DOMAIN          TIMESTAMP               1 HOUR AVG
primary         2013.05.17 09:37:35     112
gdom1           2013.05.17 09:37:35     15
gdom2           2013.05.17 09:37:35     26

primary         2013.05.17 10:37:35     96
gdom1           2013.05.17 10:37:35     12
gdom2           2013.05.17 10:37:35     21

primary         2013.05.17 11:37:35     85
gdom1           2013.05.17 11:37:35     11
gdom2           2013.05.17 11:37:35     23
...
```

◆ ◆ ◆ CHAPTER 21

Using the Oracle VM Server for SPARC Management Information Base Software

The Oracle VM Server for SPARC Management Information Base (MIB) enables third-party system management applications to perform remote monitoring of domains, and to start and stop logical domains (domains) by using the Simple Network Management Protocol (SNMP).

You can run only one instance of the Oracle VM Server for SPARC MIB software on the control domain. The control domain should run at least the Solaris 10 11/06 OS and at least the Oracle VM Server for SPARC 2.2 software.

This chapter covers the following topics:

Oracle VM Server for SPARC Management Information Base Overview

This section covers the following topics:

Related Products and Features

To successfully use the Oracle VM Server for SPARC MIB, you must understand how to use the following software products and features:

- Oracle Solaris OS
- Oracle VM Server for SPARC software
- Simple Network Management Protocol (SNMP)
- SNMP Management Information Base (MIB)
- Oracle Solaris SNMP Agent
- SNMP version 1 (SNMPv1), SNMP version 2 (SNMPv2c), and SNMP version 3 (SNMPv3) protocols
- Structure of Management Information (SMI) version 1 and version 2
- Management Information Base (MIB) structure
- Abstract Syntax Notation (ASN.1)

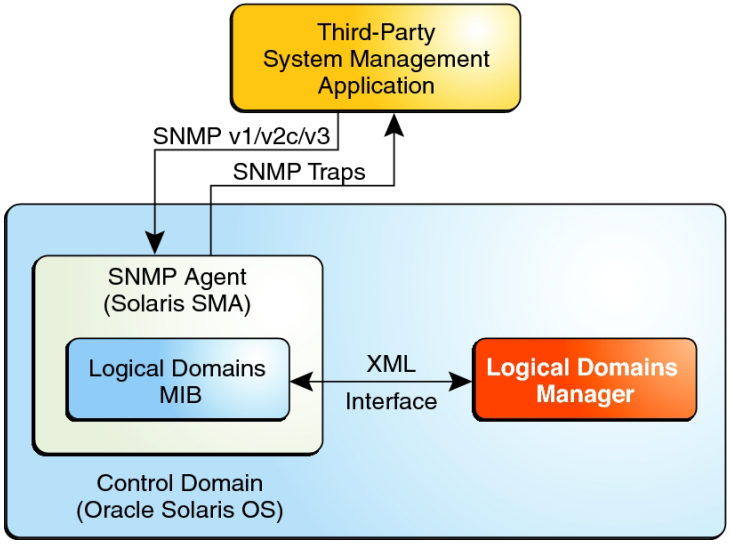
Software Components

The Oracle VM Server for SPARC MIB package, `SUNWldmib.v`, contains the following software components:

- `SUN-LDOM-MIB.mib` is an SNMP MIB in the form of a text file. This file defines the objects in the Oracle VM Server for SPARC MIB.
- `ldomMIB.so` is a System Management Agent extension module in the form of a shared library. This module enables the Oracle Solaris SNMP agent to respond to requests for information that are specified in the Oracle VM Server for SPARC MIB and to generate traps.

The following figure shows the interaction between the Oracle VM Server for SPARC MIB, the Oracle Solaris SNMP agent, the Logical Domains Manager, and a third-party system management application. The interaction shown in this figure is described in "System Management Agent" on page 425 and "Logical Domains Manager and the Oracle VM Server for SPARC MIB" on page 426.

FIGURE 21–1 Oracle VM Server for SPARC MIB Interaction With Oracle Solaris SNMP Agent, Logical Domains Manager, and a Third-Party System Management Application



System Management Agent

The Oracle Solaris SNMP agent performs the following functions:

- Listens for requests from a third-party system management application to get or set data offered by the Oracle VM Server for SPARC MIB. The agent listens on the standard SNMP port, `161`.
- Issues traps to the configured system management application by using the standard port for SNMP notifications, `162`.

The Oracle VM Server for SPARC MIB is exported by the Oracle Solaris OS default Oracle Solaris SNMP agent on the control domain.

The Oracle Solaris SNMP agent supports the get, set, and trap functions of SNMP versions v1, v2c, and v3. Most Oracle VM Server for SPARC MIB objects are read-only for monitoring purposes. However, to start or stop a domain, you must write a value to the `ldomAdminState` property of the `ldomTable` table. See Table 21–1.

Logical Domains Manager and the Oracle VM Server for SPARC MIB

A *domain* is a container that consists of a set of virtual resources for a guest operating system. The Logical Domains Manager provides the command-line interface (CLI) for creating, configuring, and managing the domains. The Logical Domains Manager and the Oracle VM Server for SPARC MIB support the following virtual resources:

- CPUs
- Memory
- Disk, network, and console I/O
- Cryptographic units

Parsing the XML-Based Control Interface

The Logical Domains Manager exports an XML-based control interface to the Oracle VM Server for SPARC MIB. The Oracle VM Server for SPARC MIB parses the XML interface and populates the MIB. The Oracle VM Server for SPARC MIB only provides support for the control domain.

Providing SNMP Traps

The Oracle VM Server for SPARC MIB polls the Logical Domains Manager periodically for updates or status changes, and then issues SNMP traps to the system management applications.

Providing Fault and Recovery Information

If the Oracle VM Server for SPARC MIB can no longer allocate a needed resource, the MIB returns a general error to the system management application through the SNMP agent. The SNMP trap-delivery mechanism does not confirm the error. No specific state or checkpointing is implemented in the Oracle VM Server for SPARC MIB. The Oracle Solaris SNMP agent with the Oracle VM Server for SPARC MIB is started and monitored by the `init` process and the Service Management Facility (SMF). If the Oracle Solaris SNMP agent fails and exits, SMF restarts the process automatically, and then the new process dynamically restarts the Oracle VM Server for SPARC MIB module.

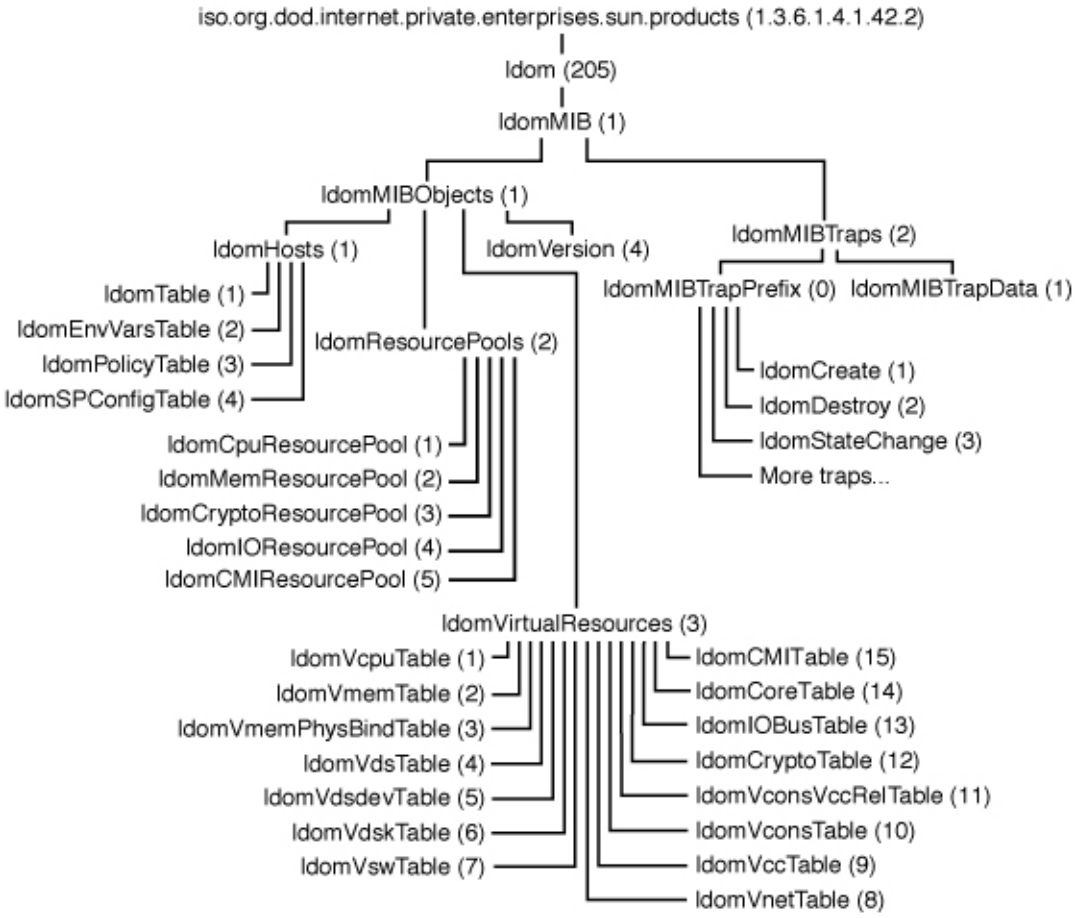
Oracle VM Server for SPARC MIB Object Tree

SNMP-managed objects are organized into a tree-like hierarchy. An object identifier (OID) consists of a series of integers based on the nodes in the tree, separated by dots. Each managed object has a numerical OID and an associated textual name. The Oracle VM Server for SPARC MIB is registered as the `ldom (205)` branch in this part of the object tree:

```
iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).sun(42).products(2)
```

The following figure shows the major subtrees under the Oracle VM Server for SPARC MIB.

FIGURE 21–2 Oracle VM Server for SPARC MIB Tree



Installing and Configuring the Oracle VM Server for SPARC MIB Software

This section covers the installation and configuration of the Oracle VM Server for SPARC MIB software. For information about administering SNMP, see the `snmpd.conf`(4) or `snmpd.conf`(5) man page.

Installing and Configuring the Oracle VM Server for SPARC MIB Software

The following table lists the tasks that you can use to install and configure the Oracle VM Server for SPARC MIB software.

Task	Description	For Instructions
Install the Oracle VM Server for SPARC MIB software package on the `primary` domain.	Use the `pkg install` command to install the `system/ldoms/mib` package.	"How to Install the Oracle VM Server for SPARC MIB Software Package" on page 428
Load the Oracle VM Server for SPARC MIB module into the Oracle Solaris SNMP agent to query the Oracle VM Server for SPARC MIB.	Modify the SNMP configuration file to load the `ldomMIB.so` module.	"How to Load the Oracle VM Server for SPARC MIB Module Into the Oracle Solaris SNMP Agent" on page 429
Remove the Oracle VM Server for SPARC MIB software package from the `primary` domain.	Use the `pkg remove` command to remove the `system/ldoms/mib` package.	"How to Remove the Oracle VM Server for SPARC MIB Software Package" on page 429

▼ How to Install the Oracle VM Server for SPARC MIB Software Package

This procedure describes how to install the Oracle VM Server for SPARC MIB software package. The Oracle VM Server for SPARC MIB software package is included as part of the Oracle VM Server for SPARC 3.3 software.

The Oracle VM Server for SPARC MIB package includes the following files:

- `/opt/SUNWldmib/lib/mibs/SUN-LDOM-MIB.mib`
- `/opt/SUNWldmib/lib/ldomMIB.so`

Before You Begin Download and install the Oracle VM Server for SPARC 3.3 software. See Chapter 2, "Installing Software," in *Oracle VM Server for SPARC 3.3 Installation Guide*.

- **Install the Oracle VM Server for SPARC MIB software package, `system/ldoms/mib`.**

```
primary# pkg install -v -g IPS-package-directory/ldoms.repo mib
```

Next Steps After you install this package, you can configure your system to dynamically load the Oracle VM Server for SPARC MIB module. See "How to Load the Oracle VM Server for SPARC MIB Module Into the Oracle Solaris SNMP Agent" on page 429.

How to Load the Oracle VM Server for SPARC MIB Module Into the Oracle Solaris SNMP Agent

The Oracle VM Server for SPARC MIB module, `ldomMIB.so`, must be loaded into the Oracle Solaris SNMP agent to query the Oracle VM Server for SPARC MIB. The Oracle VM Server for SPARC MIB module is dynamically loaded so that the module is included within the SNMP agent without requiring you to recompile and relink the agent binary.

This procedure describes how to configure your system to dynamically load the Oracle VM Server for SPARC MIB module. Instructions for dynamically loading the module without restarting the Oracle Solaris SNMP agent are provided in *Solaris System Management Agent Developer's Guide*. For more information about the Oracle Solaris SNMP agent, see *Solaris System Management Administration Guide*.

1. **Update the SNMP configuration file.**

   Append the following line to the `/etc/net-snmp/snmp/snmpd.conf` configuration file:

   ```
   dlmod ldomMIB /opt/SUNWldmib/lib/ldomMIB.so
   ```

2. **Restart the SMF service.**

   ```
   primary# svcadm restart svc:/application/management/net-snmp:default
   ```

How to Remove the Oracle VM Server for SPARC MIB Software Package

This procedure describes how to remove the Oracle VM Server for SPARC MIB software package and unload the Oracle VM Server for SPARC MIB module.

1. **Stop the SMF service.**

   ```
   primary# svcadm disable svc:/application/management/net-snmp:default
   ```

2. **Remove the Oracle VM Server for SPARC MIB software package from the `primary` domain.**

   ```
   primary# pkg uninstall system/ldoms/mib
   ```

3. **Update the SNMP configuration file.**

   Remove the line that you added to the `/etc/net-snmp/snmp/snmpd.conf` file during installation.

   ```
   dlmod ldomMIB /opt/SUNWldmib/lib/ldomMIB.so
   ```

4. **Restart the SMF service.**

   ```
   primary# svcadm restart svc:/application/management/net-snmp:default
   ```

Managing Security

This section describes how to create new Simple Network Management Protocol (SNMP) version 3 (v3) users to provide secure access to the Oracle Solaris SNMP agent. For SNMP version 1 (v1) and version 2 (v2c), the access control mechanism is the *community string*, which defines the relationship between an SNMP server and its clients. This string controls the client access to the server similar to a password controlling a user's access to a system. See *Solaris System Management Agent Administration Guide*.

Note – Creating `snmpv3` users enables you to use the Oracle Solaris SNMP agent in SNMP with the Oracle VM Server for SPARC MIB. This type of user does not interact with or conflict with users that you might have configured by using the rights feature of Oracle Solaris for the Logical Domains Manager.

▼ How to Create the Initial snmpv3 User

This procedure describes how to create the initial `snmpv3` user.

You can create additional users by cloning this initial user. Cloning enables subsequent users to inherit the initial user's authentication and security types. You can change these types later.

When you clone the initial user, you set secret key data for the new user. You must know the passwords for the initial user and for the subsequent users that you configure. You can only clone one user at a time from the initial user. See "To Create Additional SNMPv3 Users with Security" in *Solaris System Management Agent Administration Guide* for your version of the Oracle Solaris OS.

1 Stop the Oracle Solaris SNMP agent.

```
primary# svcadm disable svc:/application/management/net-snmp:default
```

2 Create the initial user.

This step creates user *initial-user* with a password that you choose, *my-password*, and adds an entry to the `/etc/sma/snmp/snmpd.conf` file. This entry gives the initial user read and write access to the agent.

Note – Passwords must contain at least eight characters.

```
primary# /usr/bin/net-snmp-config --create-snmpv3-user -a my-password initial-user
```

3 Start the Oracle Solaris SNMP agent.

```
primary# svcadm enable svc:/application/management/net-snmp:default
```

4 **Verify that the initial user has been created.**

```
primary# snmpget -v 3 -u initial-user -l authNoPriv -a MD5 \
-A my-password localhost sysUpTime.0
```

Monitoring Domains

This section describes how to monitor logical domains (domains) by querying the Oracle VM Server for SPARC MIB. This section also provides descriptions of the various types of MIB output.

This section covers the following topics:

Setting Environment Variables

Before you can query the Oracle VM Server for SPARC MIB, you must set the `PATH`, `MIBDIRS`, and `MIBS` environment variables.

```
$ PATH=/usr/bin:$PATH; export PATH
$ MIBDIRS=/opt/SUNWldmib/lib/mibs:/etc/net-snmp/snmp/mibs; export MIBDIRS
$ MIBS=+SUN-LDOM-MIB; export MIBS
```

Querying the Oracle VM Server for SPARC MIB

When a system has large number of domains, the SNMP agent might time out before being able to respond to an SNMP request. To increase the timeout value, use the `-t` option to specify a longer timeout value. For example, the following `snmpwalk` command sets the timeout value to 20 seconds:

```
# snmpwalk -t 20 -v1 -c public localhost SUN-LDOM-MIB::ldomTable
```

You can also use the `-t` option to specify the timeout value for the `snmpget` and `snmptable` commands.

- To retrieve a single MIB object:

  ```
  # snmpget -v version -c community-string host MIB-object
  ```

- To retrieve an array of MIB objects:

  Use the `snmpwalk` or `snmptable` command.

  ```
  # snmpwalk -v version -c community-string host MIB-object
  # snmptable -v version -c community-string host MIB-object
  ```

Note – You receive empty SNMP tables if you query the Oracle VM Server for SPARC MIB 2.1 software using the `snmptable` command with the `-v2c` or `-v3` option. The `snmptable` command with the `-v1` option works as expected.

To work around this issue, use the `-CB` option to use only `GETNEXT`, not `GETBULK`, requests to retrieve data. See "Querying the Oracle VM Server for SPARC MIB" on page 431.

EXAMPLE 21–1 Retrieving a Single Oracle VM Server for SPARC MIB Object (`snmpget`)

The following `snmpget` command queries the value of the `ldomVersionMajor` object. The command specifies `snmpv1` (`-v1`) and a community string (`-c public`) for the `localhost` host.

```
# snmpget -v1 -c public localhost SUN-LDOM-MIB::ldomVersionMajor.0
SUN-LDOM-MIB::ldomVersionMajor.0 = INTEGER: 1
```

EXAMPLE 21–2 Retrieving Object Values From ldomTable (`snmpwalk`)

The following examples show how to use the `snmpwalk` command to retrieve object values from `ldomTable`.

- The following `snmpwalk -v1` command returns the values for all objects in the `ldomTable` table.

```
# snmpwalk -v1 -c public localhost SUN-LDOM-MIB::ldomTable
SUN-LDOM-MIB::ldomName.1 = STRING: primary
SUN-LDOM-MIB::ldomName.2 = STRING: LdomMibTest_1
SUN-LDOM-MIB::ldomAdminState.1 = INTEGER: 0
SUN-LDOM-MIB::ldomAdminState.2 = INTEGER: 0
SUN-LDOM-MIB::ldomOperState.1 = INTEGER: active(1)
SUN-LDOM-MIB::ldomOperState.2 = INTEGER: bound(6)
SUN-LDOM-MIB::ldomNumVCpu.1 = INTEGER: 8
SUN-LDOM-MIB::ldomNumVCpu.2 = INTEGER: 4
SUN-LDOM-MIB::ldomMemSize.1 = INTEGER: 3360
SUN-LDOM-MIB::ldomMemSize.2 = INTEGER: 256
SUN-LDOM-MIB::ldomMemUnit.1 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomMemUnit.2 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomNumCrypto.1 = INTEGER: 1
SUN-LDOM-MIB::ldomNumCrypto.2 = INTEGER: 0
SUN-LDOM-MIB::ldomNumIOBus.1 = INTEGER: 2
SUN-LDOM-MIB::ldomNumIOBus.2 = INTEGER: 0
SUN-LDOM-MIB::ldomUUID.1 = STRING: 5f8817d4-5d2e-6f7d-c4af-91b5b34b5723
SUN-LDOM-MIB::ldomUUID.2 = STRING: 11284146-87ca-4877-8d80-cd0f60d5ec26
SUN-LDOM-MIB::ldomMacAddress.1 = STRING: 00:14:4f:46:47:d6
SUN-LDOM-MIB::ldomMacAddress.2 = STRING: 00:14:4f:f8:d5:6c
SUN-LDOM-MIB::ldomHostID.1 = STRING: 0x844647d6
SUN-LDOM-MIB::ldomHostID.2 = STRING: 0x84f8d56c
SUN-LDOM-MIB::ldomFailurePolicy.1 = STRING: ignore
SUN-LDOM-MIB::ldomFailurePolicy.2 = STRING: ignore
SUN-LDOM-MIB::ldomMaster.1 = STRING:
SUN-LDOM-MIB::ldomMaster.2 = STRING:
SUN-LDOM-MIB::ldomExtMapinSpace.1 = STRING: off
SUN-LDOM-MIB::ldomExtMapinSpace.2 = STRING: off
SUN-LDOM-MIB::ldomWholeCore.1 = INTEGER: 0
```

EXAMPLE 21–2 Retrieving Object Values From ldomTable (`snmpwalk`) *(Continued)*

```
SUN-LDOM-MIB::ldomWholeCore.2 = INTEGER: 0
SUN-LDOM-MIB::ldomCpuArch.1 = STRING: native
SUN-LDOM-MIB::ldomCpuArch.2 = STRING: native
SUN-LDOM-MIB::ldomShutdownGroup.1 = INTEGER: 0
SUN-LDOM-MIB::ldomShutdownGroup.2 = INTEGER: 15
SUN-LDOM-MIB::ldomPerfCounters.1 = STRING: htstrand
SUN-LDOM-MIB::ldomPerfCounters.2 = STRING: global,htstrand
SUN-LDOM-MIB::ldomNumCMI.1 = INTEGER: 0
SUN-LDOM-MIB::ldomNumCMI.2 = INTEGER: 0
```

- The following `snmpwalk` commands use `snmpv2c` and `snmpv3` to retrieve the contents of `ldomTable`:

```
# snmpwalk -v2c -c public localhost SUN-LDOM-MIB::ldomTable
# snmpwalk -v 3 -u test -l authNoPriv -a MD5 -A testpassword localhost \
SUN-LDOMMIB::ldomTable
```

EXAMPLE 21–3 Retrieving Object Values From ldomTable in Tabular Form (`snmptable`)

The following examples show how to use the `snmptable` command to retrieve object values from `ldomTable` in tabular form.

- The following `snmptable -v1` command shows the contents of `ldomTable` in tabular form.

```
# snmptable -v1 -c public localhost SUN-LDOM-MIB::ldomTable
```

- The following `snmptable` command shows the contents of `ldomTable` in tabular form by using `snmpv2c`.

  Note that for the `v2c` or `v3` `snmptable` command, use the `-CB` option to specify only `GETNEXT`, not `GETBULK`, requests to retrieve data.

```
# snmptable -v2c -CB -c public localhost SUN-LDOM-MIB::ldomTable
```

Retrieving Oracle VM Server for SPARC MIB Information

This section describes the information that you can retrieve from the Oracle VM Server for SPARC MIB in the form of tables or scalar objects.

Domain Table (ldomTable)

`ldomTable` is used to represent each domain in the system. Information includes resource constraints for virtual CPUs, memory, cryptographic units, and I/O buses. The table also includes other domain information, such as the universally unique identifier (UUID), MAC address, host ID, failure policy, and master domain.

TABLE 21–1 Domain Table (ldomTable)

Name	Data Type	Access	Description
`ldomIndex`	Integer	Not accessible	Integer that is used as an index of this table
`ldomName`	Display string	Read-only	Name of the domain
`ldomAdminState`	Integer	Read/Write	Starts or stops the domain for active management: ■ Value of 1 starts the domain ■ Value of 2 stops the domain
`ldomOperState`	Integer	Read-only	Current state of the domain, which can be one of the following values: ■ 1 is the Active state ■ 2 is the Stopping state ■ 3 is the Inactive state ■ 4 is the Binding state ■ 5 is the Unbinding state ■ 6 is the Bound state ■ 7 is the Starting state
`ldomNumVCPU`	Integer	Read-only	Number of virtual CPUs used. If the domain is in an inactive state, this value is the requested number of virtual CPUs.
`ldomMemSize`	Integer	Read-only	Amount of virtual memory used. If the domain is in an inactive state, this value is the requested memory size.
`ldomMemUnit`	Integer	Read-only	One of the following memory units: ■ 1 is KB ■ 2 is MB ■ 3 is GB ■ 4 is bytes If not specified, the unit value is bytes.
`ldomNumCrypto`	Integer	Read-only	Number of cryptographic units used. If the domain is in an inactive state, this value is the requested number of cryptographic units.
`ldomNumIOBus`	Integer	Read-only	Number of physical I/O devices used
`ldomUUID`	Display string	Read-only	UUID of the domain
`ldomMacAddress`	Display string	Read-only	MAC address of the domain
`ldomHostID`	Display string	Read-only	Host ID of the domain

TABLE 21–1 Domain Table (ldomTable) *(Continued)*

Name	Data Type	Access	Description
`ldomFailurePolicy`	Display string	Read-only	Master domain's failure policy, which can be one of `ignore`, `panic`, `reset`, or `stop`
`ldomMaster`	Display string	Read-only	Name of up to four master domains for a slave domain
`ldomExtMapinSpace`	Display string	Read-only	Extended mapin space for a domain. The extended mapin space refers to the additional LDC shared memory space. This memory space is required to support a large number of virtual I/O devices that use direct-mapped shared memory. This space is also used by virtual network devices to improve performance and scalability. The default value is `off`.
`ldomWholeCore`	Integer	Read-only	Constrains the domain to use whole-cores only. If the whole-core constraint is not enabled, the value is `0`. Otherwise, the value shows the number of max-cores.
`ldomCpuArch`	Display string	Read-only	CPU architecture for a domain. The CPU architecture specifies whether the domain can be migrated to another `sun4v` CPU architecture. Valid values are: ■ `native`, which means that the domain is permitted to be migrated only to platforms of the same `sun4v` CPU architecture (default value) ■ `generic`, which means that the domain is permitted to be migrated to all compatible `sun4v` CPU architectures
`ldomShutdownGroup`	Integer	Read-only	Shutdown-group number for a guest domain. On a Fujitsu M10 server, an SP-initiated ordered shutdown will shut down domains in descending order of their shutdown-group numbers, from `15` to `0`. The default value is `15`.
`ldomPerfCounters`	String	Read-only	Performance register access information for a guest domain. Values can be `global` (on one domain at a time only) and optionally one of the following: `htstrand` or `strand`. The default value is `htstrand`.

TABLE 21–1 Domain Table (ldomTable) *(Continued)*

Name	Data Type	Access	Description
`ldomNumCMI`	Integer	Read-only	Number of CMI resources used. If the domain is in an inactive state, this value is the requested number of CMI resources.

Environment Variables Table (ldomEnvVarsTable)

`ldomEnvVarsTable` describes the OpenBoot PROM environment variables that all domains use.

TABLE 21–2 Environment Variables Table (ldomEnvVarsTable)

Name	Data Type	Access	Description
`ldomEnvVarsLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the OpenBoot PROM environment variables
`ldomEnvVarsIndex`	Integer	Read-only	Integer that is used to index the OpenBoot PROM environment variables in this table
`ldomEnvVarsName`	Display string	Read-only	Name of the OpenBoot PROM variable
`ldomEnvVarsValue`	Display string	Read-only	Value of the OpenBoot PROM variable

Domain Policy Table (ldomPolicyTable)

`ldomPolicyTable` describes the dynamic resource management (DRM) policies that apply to all domains.

TABLE 21–3 Domain Policy Table (ldomPolicyTable)

Name	Data Type	Access	Description
`ldomPolicyLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the DRM policy
`ldomPolicyIndex`	Integer	Not accessible	Integer that is used to index the DRM policy in this table
`ldomPolicyName`	Display string	Read-only	Policy name
`ldomPolicyStatus`	Display string	Read-only	Policy status
`ldomPolicyPriority`	Integer	Read-only	Priority that is used to determine which DRM policy is selected when policies overlap

TABLE 21–3 Domain Policy Table (ldomPolicyTable) *(Continued)*

Name	Data Type	Access	Description
`ldomPolicyVcpuMin`	Integer	Read-only	Minimum number of virtual CPUs for a domain
`ldomPolicyVcpuMax`	Integer	Read-only	Maximum number of virtual CPUs for a domain. A value of `unlimited` uses the maximum integer value of 2147483647.
`ldomPolicyUtilLower`	Integer	Read-only	Lower utilization level at which policy analysis is triggered
`ldomPolicyUtilUpper`	Integer	Read-only	Upper utilization level at which policy analysis is triggered
`ldomPolicyTodBegin`	Display string	Read-only	Effective start time of a policy with a format of *hh*:*mm*:[*ss*]
`ldomPolicyTodEnd`	Display string	Read-only	Effective stop time of a policy with a format of *hh*:*mm*:[*ss*]
`ldomPolicySampleRate`	Integer	Read-only	Resource cycle time in seconds
`ldomPolicyElasticMargin`	Integer	Read-only	Amount of buffer between `util-lower` property (`ldomPolicyUtilLower`) and the number of free virtual CPUs to avoid oscillations at low virtual CPU counts
`ldomPolicyAttack`	Integer	Read-only	Maximum amount of a resource to be added during any one resource-control cycle. A value of `unlimited` uses the maximum integer value of 2147483647.
`ldomPolicyDecay`	Integer	Read-only	Maximum amount of a resource to be removed during any one resource-control cycle

Service Processor Configuration Table (ldomSPConfigTable)

`ldomSPConfigTable` describes the service processor (SP) configurations for all domains.

TABLE 21–4 Service Processor Configuration Table (ldomSPConfigTable)

Name	Data Type	Access	Description
`ldomSPConfigIndex`	Integer	Not accessible	Integer that is used to index an SP configuration in this table
`ldomSPConfigName`	Display string	Read-only	SP configuration name
`ldomSPConfigStatus`	Display string	Read-only	SP configuration status

Domain Resource Pool and Scalar Variables

The following resources can be assigned to domains:

- Virtual CPU (`vcpu`)
- Memory (`mem`)
- Cryptographic unit (`mau`)
- Virtual switch (`vsw`)
- Virtual network (`vnet`)
- Virtual disk server (`vds`)
- Virtual disk server device (`vdsdev`)
- Virtual disk (`vdisk`)
- Virtual console concentrator (`vcc`)
- Virtual console (`vcons`)
- Physical I/O device (`io`)
- CMI resources (`cmi`)

The following scalar MIB variables are used to represent resource pools and their properties.

TABLE 21–5 Scalar Variables for CPU Resource Pool

Name	Data Type	Access	Description
`ldomCpuRpCapacity`	Integer	Read-only	Maximum reservation allowed by the resource pool in `ldomCpuRpCapacityUnits`
`ldomCpuRpReserved`	Integer	Read-only	Accumulated processor clock speed of the CPU, in MHz, that is currently reserved from the resource pool
`ldomCpuRpCapacityUnit` and `ldomCpuRpReservedUnit`	Integer	Read-only	One of the following CPU allocation units: ■ 1 is MHz ■ 2 is GHz The default value is MHz.

TABLE 21–6 Scalar Variables for Memory Resource Pool

Name	Data Type	Access	Description
`ldomMemRpCapacity`	Integer	Read-only	Maximum reservation allowed by the resource pool in `MemRpCapacityUnits`
`ldomMemRpReserved`	Integer	Read-only	Amount of memory, in `MemRpReservedUnits`, that is currently reserved from the resource pool

TABLE 21–6 Scalar Variables for Memory Resource Pool *(Continued)*

Name	Data Type	Access	Description
`ldomMemRpCapacityUnit` and `ldomMemRpReservedUnit`	Integer	Read-only	One of the following memory allocation units: ■ 1 is KB ■ 2 is MB ■ 3 is GB ■ 4 is bytes If not specified, the unit value is bytes.

TABLE 21–7 Scalar Variables for Cryptographic Resource Pool

Name	Data Type	Access	Description
`ldomCryptoRpCapacity`	Integer	Read-only	Maximum reservation allowed by the resource pool
`ldomCryptoRpReserved`	Integer	Read-only	Number of cryptographic units that is currently reserved from the resource pool

TABLE 21–8 Scalar Variables for I/O Bus Resource Pool

Name	Data Type	Access	Description
`ldomIOBusRpCapacity`	Integer	Read-only	Maximum reservation allowed by the pool
`ldomIOBusRpReserved`	Integer	Read-only	Number of I/O buses that is currently reserved from the resource pool

TABLE 21–9 Scalar Variables for CMI Resource Pool

Name	Data Type	Access	Description
`ldomCMIRpCapacity`	Integer	Read-only	Maximum reservation allowed by the pool
`ldomCMIRpReserved`	Integer	Read-only	Number of CMI resources that are currently reserved from the resource pool

Virtual CPU Table (ldomVcpuTable)

`ldomVcpuTable` describes the virtual CPUs that all domains use.

TABLE 21–10 Virtual CPU Table (ldomVcpuTable)

Name	Data Type	Access	Description
`ldomVcpuLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the virtual CPU
`ldomVcpuIndex`	Integer	Not accessible	Integer that is used to index the virtual CPU in this table
`ldomVcpuDeviceID`	Display string	Read-only	Identifier of the virtual CPU (VID)
`ldomVcpuOperationalStatus`	Integer	Read-only	One of the following CPU statuses: 1=Unknown 2=Other 3=OK 4=Degraded 5=Stressed 6=Predictive failure 7=Error 8=Nonrecoverable error 9=Starting 10=Stopping 11=Stopped 12=In service 13=No contact 14=Lost communication 15=Aborted 16=Dormant 17=Supporting entity in error 18=Completed 19=Power mode The default value is 1 (Unknown) because the Logical Domains Manager does not provide the CPU state.

TABLE 21–10 Virtual CPU Table (ldomVcpuTable) *(Continued)*

Name	Data Type	Access	Description
`ldomVcpuPhysBind`	Display string	Read-only	Physical binding (PID). Contains the identifier of a hardware thread (strand) that is assigned to this virtual CPU. This identifier uniquely identifies the core and the chip.
`ldomVcpuPhysBindUsage`	Integer	Read-only	Indicates how much (in MHz) of the total capacity of the thread is used by this virtual CPU. For example, assume a thread can run at a maximum of one GHz. If only half of that capacity is allocated to this virtual CPU (50% of the thread), the property value is 500.
`ldomVcpuCoreID`	Display string	Read-only	Identifier of the core (core ID).
`ldomVcpuUtilPercent`	Display string	Read-only	Indicates the utilization percentage of the virtual CPU.

Virtual Memory Tables

A domain's memory space is referred to as *real memory*, that is, *virtual memory*. Host platform memory space that is detected by the hypervisor is referred to as *physical memory*. The hypervisor maps blocks of physical memory to form a block of real memory that is used by a domain.

The following example shows that the requested memory size can be split between two memory blocks instead of being assigned to a single large memory block. Assume that a domain requests 521 Mbytes of real memory. The memory can be assigned two 256-Mbyte blocks on the host system as physical memory by using the {*physical-address*, *real-address*, *size*} format.

```
{0x1000000, 0x1000000, 256}, {0x2000000, 0x2000000,256}
```

A domain can have up to 64 physical memory segments assigned to a guest domain. Therefore, an auxiliary table is used to hold each memory segment instead of a display string. A display string has a 255-character limit.

Virtual Memory Table (ldomVmemTable)

`ldomVmemTable` describes the properties of virtual memory that domains use.

TABLE 21–11 Virtual Memory Table (ldomVmemTable)

Name	Data Type	Access	Description
`ldomVmemLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the virtual memory
`ldomVmemIndex`	Integer	Not accessible	Integer that is used to index the virtual memory in this table
`ldomVmemNumberofBlocks`	Integer	Read-only	Number of blocks of virtual memory

Virtual Memory Physical Binding Table (ldomVmemPhysBindTable)

`ldomVmemPhysBindTable` is an auxiliary table that contains physical memory segments for all domains.

TABLE 21–12 Virtual Memory Physical Binding Table (ldomVmemPhysBindTable)

Name	Data Type	Access	Description
`ldomVmemPhysBindLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the physical memory segments
`ldomVmemPhysBind`	Display string	Read-only	List of physical memory that is mapped to this virtual memory block in the following format: {*physical-address*, *real-address*, *size*}

Virtual Disk Tables

A virtual disk service (`vds`) and the physical device to which it maps (`vdsdev`) provide the virtual disk capability to the Oracle VM Server for SPARC technology. A virtual disk service exports a number of local volumes (physical disks or file systems). When a virtual disk service is specified, the following are included:

- Complete `/dev` path of the backing device (`vdsdev`)
- Unique name (volume name) for the device being added to the service

One or more disks, disk slices, and file systems can be bound to a single disk service. Each disk has a unique name and volume name. The volume name is used when the disk is bound to the service. The Logical Domains Manager creates virtual disk clients (`vdisk`) from the virtual disk service and its logical volumes.

Virtual Disk Service Table (ldomVdsTable)

`ldomVdsTable` describes the virtual disk services for all domains.

TABLE 21–13 Virtual Disk Service Table (ldomVdsTable)

Name	Data Type	Access	Description
`ldomVdsLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the virtual disk service
`ldomVdsIndex`	Integer	Not accessible	Integer that is used to index the virtual disk service in this table
`ldomVdsServiceName`	Display string	Read-only	Service name for the virtual disk service. The property value is the *service-name* specified by the `ldm add-vds` command.
`ldomVdsNumofAvailVolume`	Integer	Read-only	Number of logical volumes exported by this virtual disk service
`ldomVdsNumofUsedVolume`	Integer	Read-only	Number of logical volumes used (bound) to this virtual disk service

Virtual Disk Service Device Table (ldomVdsdevTable)

`ldomVdsdevTable` describes the virtual disk service devices that all virtual disk services use.

TABLE 21–14 Virtual Disk Service Device Table (ldomVdsdevTable)

Name	Data Type	Access	Description
`ldomVdsdevVdsIndex`	Integer	Read-only	Integer that is used to index into `ldomVdsTable` that represents the virtual disk service that contains the virtual disk device
`ldomVdsdevIndex`	Integer	Not accessible	Integer that is used to index the virtual disk service device in this table
`ldomVdsdevVolumeName`	Display string	Read-only	Volume name for the virtual disk service device. This property specifies a unique name for the device that is being added to the virtual disk service. This name is exported by the virtual disk service to the clients for the purpose of adding this device. The property value is the *volume-name* specified by the `ldm add-vdsdev` command.
`ldomVdsdevDevPath`	Display string	Read-only	Path name of the physical disk device. The property value is the *backend* specified by the `ldm add-vdsdev` command.

TABLE 21–14 Virtual Disk Service Device Table (ldomVdsdevTable) *(Continued)*

Name	Data Type	Access	Description
`ldomVdsdevOptions`	Display string	Read-only	One or more of the options for the disk device, which are `ro`, `slice`, or `excl`
`ldomVdsdevMPGroup`	Display string	Read-only	Multipath group name for the disk device

Virtual Disk Table (ldomVdiskTable)

`ldomVdiskTable` describes the virtual disks for all domains.

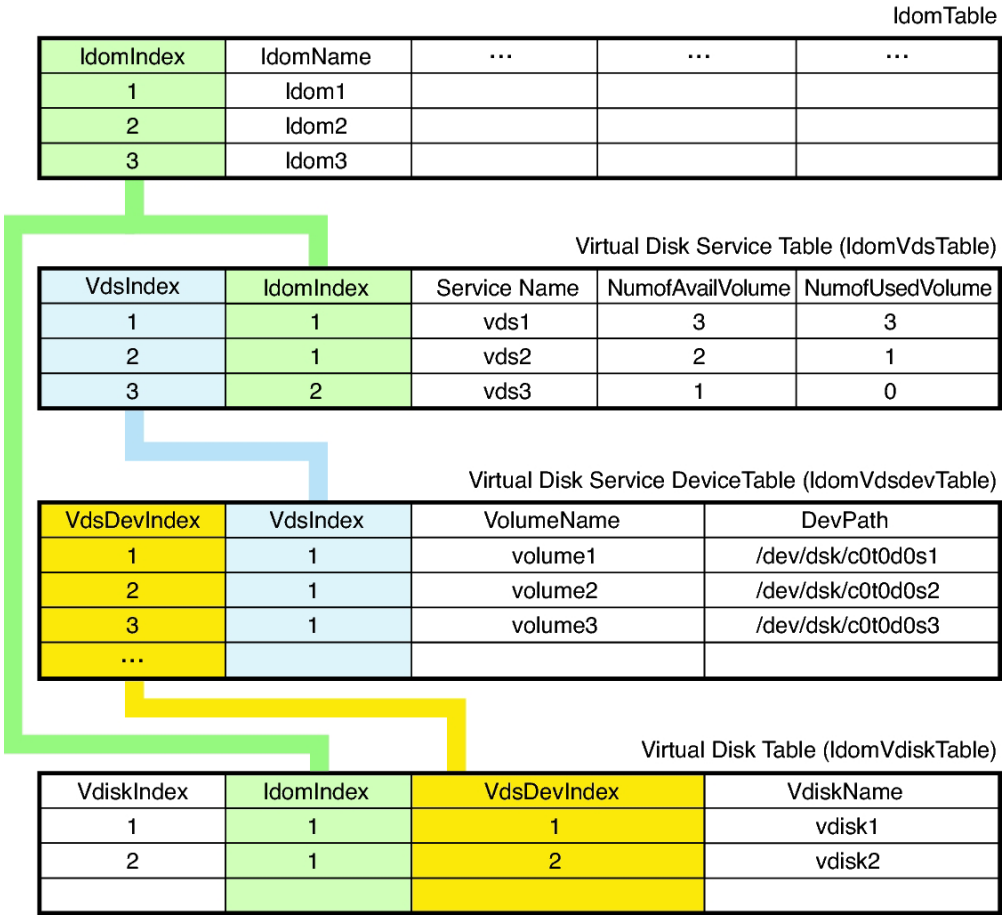
TABLE 21–15 Virtual Disk Table (ldomVdiskTable)

Name	Data Type	Access	Description
`ldomVdiskLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the virtual disk device
`ldomVdiskVdsDevIndex`	Integer	Read-only	Integer that is used to index into `ldomVdsdevTable` that represents the virtual disk service device
`ldomVdiskIndex`	Integer	Not accessible	Integer that is used to index the virtual disk in this table
`ldomVdiskName`	Display string	Read-only	Name of the virtual disk. The property value is the *disk-name* specified by the `ldm add-vdisk` command.
`ldomVdiskTimeout`	Integer	Read-only	Timeout, in seconds, for establishing a connection between a virtual disk client and a virtual disk server
`ldomVdiskID`	Display string	Read-only	Identifier of the virtual disk

The following figure shows how indexes are used to define relationships among the virtual disk tables and the domain table. The indexes are used as follows:

- `ldomIndex` in `ldomVdsTable` and `ldomVdiskTable` points to `ldomTable`.
- `VdsIndex` in `ldomVdsdevTable` points to `ldomVdsTable`.
- `VdsDevIndex` in `ldomVdiskTable` points to `ldomVdsdevTable`.

FIGURE 21–3 Relationship Among Virtual Disk Tables and the Domain Table



Virtual Network Tables

Oracle VM Server for SPARC virtual network support enables guest domains to communicate with each other and with external hosts through a physical Ethernet device. The virtual network contains the following main components:

- Virtual switch (`vsw`)
- Virtual network device (`vnet`)

After you create a virtual switch on a service domain, you can bind a physical network device to the virtual switch. After that, you can create a virtual network device for a domain that uses the virtual switch service for communication. The virtual switch service communicates with other domains by connecting to the same virtual switch. The virtual switch service communicates with external hosts if a physical device is bound to the virtual switch.

Virtual Switch Service Table (ldomVswTable)

`ldomVswTable` describes the virtual switch services for all domains.

TABLE 21–16 Virtual Switch Service Table (ldomVswTable)

Name	Data Type	Access	Description
`ldomVswLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the virtual switch service
`ldomVswIndex`	Integer	Not accessible	Integer that is used to index the virtual switch device in this table
`ldomVswServiceName`	Display string	Read-only	Virtual switch service name
`ldomVswMacAddress`	Display string	Read-only	MAC address used by the virtual switch
`ldomVswPhysDevPath`	Display string	Read-only	Physical device path for the virtual network switch. The property value is null when no physical device is bound to the virtual switch.
`ldomVswMode`	Display string	Read-only	Value is `mode=sc` for running cluster nodes
`ldomVswDefaultVlanID`	Display string	Read-only	Default VLAN ID for the virtual switch
`ldomVswPortVlanID`	Display string	Read-only	Port VLAN ID for the virtual switch
`ldomVswVlanID`	Display string	Read-only	VLAN ID for the virtual switch
`ldomVswLinkprop`	Display string	Read-only	Value is `linkprop=phys-state` to report the link status based on the physical network device
`ldomVswMtu`	Integer	Read-only	Maximum transmission unit (MTU) for a virtual switch device
`ldomVswID`	Display string	Read-only	Identifier of the virtual switch device
`ldomVswInterVnetLink`	Display string	Read-only	State of LDC channel assignment for inter-vnet communications. Value is either `on` or `off`.

Virtual Network Device Table (ldomVnetTable)

`ldomVnetTable` describes the virtual network devices for all domains.

TABLE 21–17 Virtual Network Device Table (ldomVnetTable)

Name	Data Type	Access	Description
`ldomVnetLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the virtual network device
`ldomVnetVswIndex`	Integer	Read-only	Integer that is used to index into the virtual switch service table
`ldomVnetIndex`	Integer	Not accessible	Integer that is used to index the virtual network device in this table
`ldomVnetDevName`	Display string	Read-only	Virtual network device name. The property value is the `net-dev` property specified by the `ldm add-vnet` command.
`ldomVnetDevMacAddress`	Display string	Read-only	MAC address for this network device. The property value is the `mac-addr` property specified by the `ldm add-vnet` command.
`ldomVnetMode`	Display string	Read-only	Value is `mode=hybrid` to use NIU hybrid I/O on the virtual network device
`ldomVnetPortVlanID`	Display string	Read-only	Port VLAN ID for the virtual network device
`ldomVnetVlanID`	Display string	Read-only	VLAN ID for the virtual network device
`ldomVnetLinkprop`	Display string	Read-only	Value is `linkprop=phys-state` to report the link status based on the physical network device
`ldomVnetMtu`	Integer	Read-only	MTU for a virtual network device
`ldomVnetID`	Display string	Read-only	Identifier of the virtual network device

Virtual Console Tables

The Oracle VM Server for SPARC service domain provides a virtual network terminal service (`vNTS`). `vNTS` provides a virtual console service, called a virtual console concentrator (`vcc`), with a range of port numbers. Each virtual console concentrator has multiple console groups (`vcons`), and each group is assigned a port number. Each group can contain multiple domains.

Virtual Console Concentrator Table (ldomVccTable)

`ldomVccTable` describes the virtual console concentrators for all domains.

TABLE 21–18 Virtual Console Concentrator Table (ldomVccTable)

Name	Data Type	Access	Description
`ldomVccLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the virtual console service
`ldomVccIndex`	Integer	Not accessible	Integer that is used to index the virtual console concentrator in this table
`ldomVccName`	Display string	Read-only	Virtual console concentrator name. The property value is the *vcc-name* specified by the `ldm add-vcc` command.
`ldomVccPortRangeLow`	Integer	Read-only	Low number for the range of TCP ports to be used by the virtual console concentrator. The property value is the *x* part of the `port-range` specified by the `ldm add-vcc` command.
`ldomVccPortRangeHigh`	Integer	Read-only	High number for the range of TCP ports to be used by the virtual console concentrator. The property value is the *y* part of the `port-range` specified by the `ldm add-vcc` command.

Virtual Console Group Table (ldomVconsTable)

`ldomVconsTable` describes the virtual console groups for all virtual console services. This table also shows whether console logging is enabled or disabled on each domain.

TABLE 21–19 Virtual Console Group Table (ldomVconsTable)

Name	Data Type	Access	Description
`ldomVconsIndex`	Integer	Not accessible	Integer that is used to index a virtual group in this table
`ldomVconsGroupName`	Display string	Read-only	Group name to which to attach the virtual console. The property value is the `group` specified by the `ldm set-vcons` command.
`ldomVconsLog`	Display string	Read-only	Console logging status. The property value is the string `on` or `off` as specified by the `ldm set-vcons` command. When a group contains more than one domain, this property shows the console logging status of the domain that has most recently been modified by the `ldm set-vcons` command.

TABLE 21–19 Virtual Console Group Table (ldomVconsTable) *(Continued)*

Name	Data Type	Access	Description
`ldomVconsPortNumber`	Integer	Read-only	Port number assigned to this group. The property value is the `port` specified by the `ldm set-vcons` command.

Virtual Console Relationship Table (ldomVconsVccRelTable)

`ldomVconsVccRelTable` contains index values to show the inter-table relationships among a domain, a virtual console concentrator, and console groups.

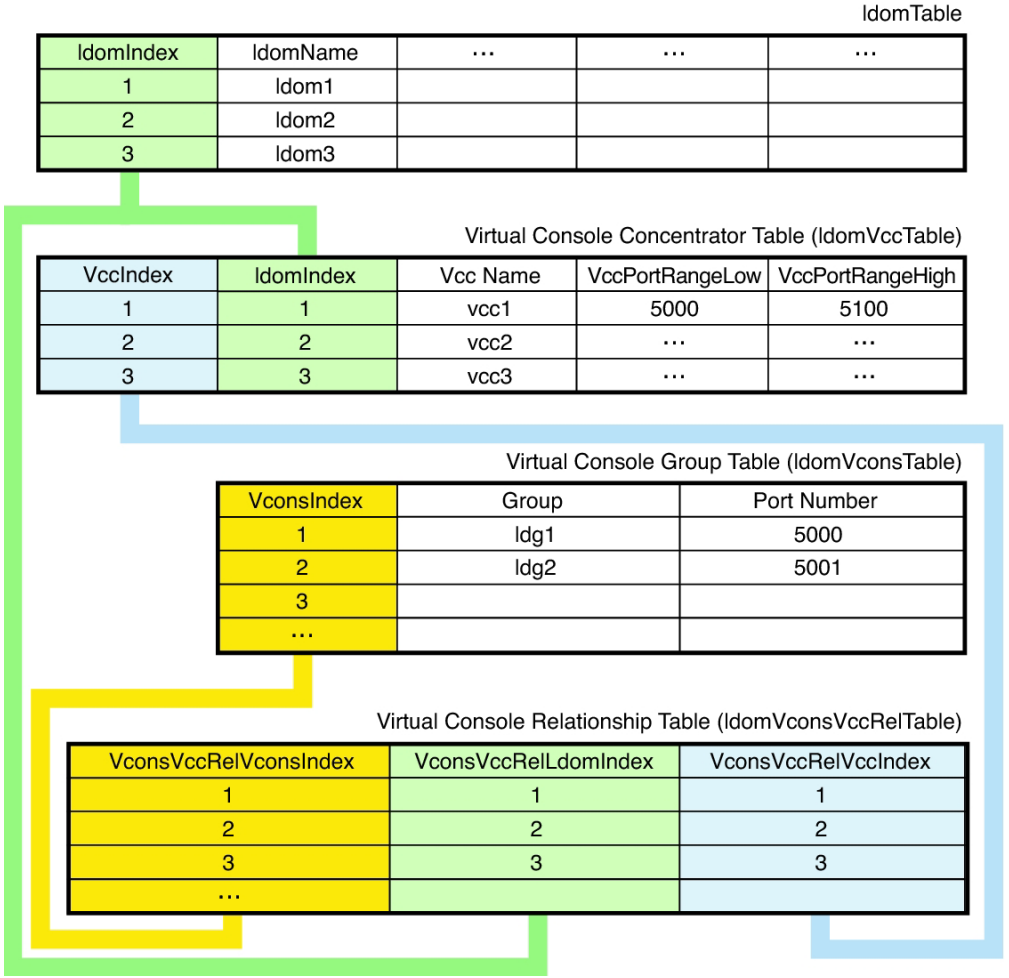
TABLE 21–20 Virtual Console Relationship Table (ldomVconsVccRelTable)

Name	Data Type	Access	Description
`ldomVconsVccRelVconsIndex`	Integer	Read-only	Value of `ldomVconsIndex` in `ldomVconsTable`
`ldomVconsVccRelLdomIndex`	Integer	Read-only	Value of `ldomIndex` in `ldomTable`
`ldomVconsVccRelVccIndex`	Integer	Read-only	Value of `ldomVccIndex` in `ldomVccTable`

The following figure shows how indexes are used to define relationships among the virtual console tables and the domain table. The indexes are used as follows:

- `ldomIndex` in `ldomVccTable` and `ldomVconsVccRelTable` points to `ldomTable`.
- `VccIndex` in `ldomVconsVccRelTable` points to `ldomVccTable`.
- `VconsIndex` in `ldomVconsVccRelTable` points to `ldomVconsTable`.

FIGURE 21–4 Relationship Among Virtual Console Tables and the Domain Table



Cryptographic Units Table (ldomCryptoTable)

`ldomCryptoTable` describes the cryptographic units that all domains use. A cryptographic unit is sometimes referred to as a modular arithmetic unit (MAU).

TABLE 21–21 Cryptographic Units Table (ldomCryptoTable)

Name	Data Type	Access	Description
`ldomCryptoLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the cryptographic unit

TABLE 21–21 Cryptographic Units Table (ldomCryptoTable) *(Continued)*

Name	Data Type	Access	Description
`ldomCryptoIndex`	Integer	Not accessible	Integer that is used to index the cryptographic unit in this table
`ldomCryptoCpuSet`	Display string	Read-only	List of CPUs that is mapped to `MAU-unit cpuset`. For example, `{0, 1, 2, 3}`.

I/O Bus Table (ldomIOBusTable)

`ldomIOBusTable` describes the physical I/O devices and PCI buses that all domains use.

TABLE 21–22 I/O Bus Table (ldomIOBusTable)

Name	Data Type	Access	Description
`ldomIOBusLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the I/O bus
`ldomIOBusIndex`	Integer	Not accessible	Integer that is used to index the I/O bus in this table
`ldomIOBusName`	Display string	Read-only	Physical l/O device name
`ldomIOBusPath`	Display string	Read-only	Physical I/O device path
`ldomIOBusOptions`	Display string	Read-only	Physical I/O device options

CMI Table (ldomCMITable)

`ldomCMITable` describes the CMI resource information for all domains.

TABLE 21–23 CMI Table (ldomCMITable)

Name	Data Type	Access	Description
`ldomCMILdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the CMI resource
`ldomCMIIndex`	Integer	Not accessible	Integer that is used to index the CMI resource in this table
`ldomCMIID`	Display string	Read-only	Identifier of a CMI resource (CMI ID)
`ldomCMICpuSet`	Display string	Read-only	List of CPUs that are mapped to the CMI resource
`ldomCMICores`	Display string	Read-only	List of cores that are mapped to the CMI resource

Core Table (ldomCoreTable)

`ldomCoreTable` describes the core information, such as `core-id` and `cpuset`, for all domains.

TABLE 21–24 Core Table (ldomCoreTable)

Name	Data Type	Access	Description
`ldomCoreLdomIndex`	Integer	Read-only	Integer that is used as an index into `ldomTable` that represents the domain that contains the core
`ldomCoreIndex`	Integer	Not accessible	Integer that is used to index a core in this table
`ldomCoreID`	Display string	Read-only	Identifier of a core (core ID)
`ldomCoreCpuSet`	Display string	Read-only	List of CPUs that is mapped to the core `cpuset`

Scalar Variables for Domain Version Information

The Logical Domains Manager protocol supports domain versions, which consists of a major number and a minor number. The Oracle VM Server for SPARC MIB has scalar variables to describe the domain version information.

TABLE 21–25 Scalar Variables for Domain Version Information

Name	Data Type	Access	Description
`ldomVersionMajor`	Integer	Read-only	Major version number
`ldomVersionMinor`	Integer	Read-only	Minor version number

The values for `ldomVersionMajor` and `ldomVersionMinor` are equivalent to the version shown by the `ldm list -p` command. For example:

```
$ ldm ls -p
VERSION 1.6
...

$ snmpget -v1 -c public localhost SUN-LDOM-MIB::ldomVersionMajor.0
SUN-LDOM-MIB::ldomVersionMajor.0 = INTEGER: 1

$ snmpget -v1 -c public localhost SUN-LDOM-MIB::ldomVersionMinor.0
SUN-LDOM-MIB::ldomVersionMinor.0 = INTEGER: 5
```

Using SNMP Traps

This section describes how to set up your system to send and receive traps. It also describes the traps that you can use to receive change notification for logical domains (domains), as well as other traps that you can use.

The `snmptrapd` daemon does not automatically accept all incoming traps. Instead, the daemon must be configured with authorized SNMP v1 and v2c community strings, with SNMPv3 users, or both. Unauthorized traps or notifications are dropped. See the `snmptrapd.conf`(4) or `snmptrapd.conf`(5) man page.

Using Oracle VM Server for SPARC MIB Module Traps

Access control checks are applied to incoming notifications. If `snmptrapd` runs without a suitable configuration file, or with equivalent access control settings, such traps are not processed. See the `snmptrapd.conf`(4) or `snmptrapd.conf`(5) man page.

▼ How to Send Traps

1 Configure the trap.

Edit the `/etc/net-snmp/snmp/snmpd.conf` SNMP configuration file to add the directives to define the trap, inform version, and destination.

You must use the `pfedit` command to edit the `/etc/net-snmp/snmp/snmpd.conf` file.

```
trapcommunity string --> define community string to be used when sending traps
trapsink host[community [port]] --> to send v1 traps
trap2sink host[community [port]] --> to send v2c traps
informsink host[community [port]] --> to send informs
```

For more information, see the `snmpd.conf`(4) or `snmpd.conf`(5) man page.

For example, the following directives use the `public` string as the community string when sending traps and indicate that the `v1` traps are sent to the `localhost` destination:

```
trapcommunity public
trapsink localhost
```

2 Configure access control settings by creating or editing the `/usr/etc/snmp/snmptrapd.conf` SNMP `trapd` configuration file.

You must use the `pfedit` command to edit the `/etc/net-snmp/snmp/snmpd.conf` file.

The following example shows who is authorized to send traps (`public`) and how incoming traps should be processed (`log,execute,net`). See the `snmptrapd.conf`(4) or `snmptrapd.conf`(5) man page.

```
authCommunity log,execute,net public
```

3 **To receive SNMP trap messages, start the SNMP trap daemon utility, `snmptrapd`.**

Example 21–4 Sending SNMP v1 and v2c Traps

This example sends send both `v1` and `v2c` traps to the SNMP trap daemon that runs on the same host. Update the `/etc/net-snmp/snmp/snmpd.conf` file with the following directives:

```
trapcommunity public
trapsink localhost
trap2sink localhost
```

▼ How to Receive Traps

- **Start the SNMP trap daemon utility.**

  For information about the output format options, see the `snmptrapd`(1M) man page.

  The `snmptrapd` utility is an SNMP application that receives and logs SNMP `TRAP` messages. For example, the following `snmptrapd` command shows that a new domain was created (`ldomTrapDesc` = `Ldom Created`) with a name of `ldg2` (`ldomName` = `ldg2`).

```
# /usr/sbin/snmptrapd -f -Le -F \
"TRAP from %B on %m/%l/%y at %h:%j:%k Enterprise=%N Type=%w SubType=%q\n
with Varbinds: %v\nSecurity info:%P\n\n" localhost:162
NET-SNMP version 5.4.1
TRAP from localhost on 6/27/2012 at 12:13:48
Enterprise=SUN-LDOM-MIB::ldomMIBTraps Type=6 SubType=SUN-LDOM-MIB::ldomCreate
with Varbinds: SUN-LDOM-MIB::ldomIndexNotif = INTEGER: 3
SUN-LDOM-MIB::ldomName = STRING: ldg2   SUN-LDOM-MIB::ldomTrapDesc = STRING:
Ldom Created
Security info:TRAP, SNMP v1, community public
```

  Note that the `-F` option argument string is broken on to two lines for readability purposes.

Oracle VM Server for SPARC MIB Trap Descriptions

This section describes the Oracle VM Server for SPARC MIB traps that you can use.

Domain Creation (ldomCreate)

This trap notifies you when any domains are created.

TABLE 21–26 Domain Creation Trap (ldomCreate)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain

TABLE 21–26 Domain Creation Trap (ldomCreate) *(Continued)*

Name	Data Type	Description
`ldomTrapDesc`	Display string	Description of the trap

Domain Destroy (ldomDestroy)

This trap notifies you when any domains are destroyed.

TABLE 21–27 Domain Destroy Trap (ldomDestroy)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain
`ldomTrapDesc`	Display string	Description of the trap

Domain State Change (ldomStateChange)

This trap notifies you of any domain operating state changes.

TABLE 21–28 Domain State Change Trap (ldomStateChange)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain
`ldomOperState`	Integer	New state of the domain
`ldomStatePrev`	Integer	Previous state of the domain
`ldomTrapDesc`	Display string	Description of the trap

Virtual CPU Change (ldomVCpuChange)

This trap notifies you when the number of virtual CPUs in a domain changes.

TABLE 21–29 Domain Virtual CPU Change Trap (ldomVCpuChange)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain that contains the virtual CPU
`ldomNumVCPU`	Integer	New number of virtual CPUs for the domain

TABLE 21–29 Domain Virtual CPU Change Trap (ldomVCpuChange) *(Continued)*

Name	Data Type	Description
`ldomNumVCPUPrev`	Integer	Previous number of virtual CPUs for the domain
`ldomTrapDesc`	Display string	Description of the trap

Virtual Memory Change (ldomVMemChange)

This trap notifies you when the amount of virtual memory in a domain changes.

TABLE 21–30 Domain Virtual Memory Change Trap (ldomVMemChange)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain that contains the virtual memory
`ldomMemSize`	Integer	Amount of virtual memory for the domain
`ldomMemSizePrev`	Integer	Previous amount of virtual memory for the domain
`ldomMemUnit`	Integer	Memory unit for virtual memory, which is one of the following: ■ 1 is KB ■ 2 is MB ■ 3 is GB ■ 4 is bytes If not specified, the unit value is bytes.
`ldomMemUnitPrev`	Integer	Memory unit for previous virtual memory, which is one of the following: ■ 1 is KB ■ 2 is MB ■ 3 is GB ■ 4 is bytes If not specified, the unit value is bytes.
`ldomTrapDesc`	Display string	Description of the trap

Virtual Disk Service Change (ldomVdsChange)

This trap notifies you when a domain's virtual disk service changes.

TABLE 21–31 Domain Virtual Disk Service Change Trap (ldomVdsChange)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain that contains the virtual disk service
`ldomVdsServiceName`	Display string	Name of the virtual disk service that has changed
`ldomChangeFlag`	Integer	Indicates one of the following changes that occurred to the virtual disk service: ■ 1 is Added ■ 2 is Modified ■ 3 is Removed
`ldomTrapDesc`	Display string	Description of the trap

Virtual Disk Change (ldomVdiskChange)

This trap notifies you when a domain's virtual disk changes.

TABLE 21–32 Virtual Disk Change Trap (ldomVdiskChange)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain that contains the virtual disk device
`ldomVdiskName`	Display string	Name of the virtual disk device that has changed
`ldomChangeFlag`	Integer	Indicates one of the following changes that occurred to the virtual disk service: ■ 1 is Added ■ 2 is Modified ■ 3 is Removed
`ldomTrapDesc`	Display string	Description of the trap

Virtual Switch Change (ldomVswChange)

This trap notifies you when a domain's virtual switch changes.

TABLE 21–33 Virtual Switch Change Trap (ldomVswChange)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain that contains the virtual switch service
`ldomVswServiceName`	Display string	Name of the virtual switch service that has changed
`ldomChangeFlag`	Integer	Indicates one of the following changes that occurred to the virtual switch service: ■ 1 is Added ■ 2 is Modified ■ 3 is Removed
`ldomTrapDesc`	Display string	Description of the trap

Virtual Network Change (ldomVnetChange)

This trap notifies you when a domain's virtual network changes.

TABLE 21–34 Virtual Network Change Trap (ldomVnetChange)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain that contains the virtual network device
`ldomVnetDevName`	Display string	Name of the virtual network device for the domain
`ldomChangeFlag`	Integer	Indicates one of the following changes that occurred to the virtual disk service: ■ 1 is Added ■ 2 is Modified ■ 3 is Removed
`ldomTrapDesc`	Display string	Description of the trap

Virtual Console Concentrator Change (ldomVccChange)

This trap notifies you when a domain's virtual console concentrator changes.

TABLE 21–35 Virtual Console Concentrator Change Trap (ldomVccChange)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain that contains the virtual console concentrator
`ldomVccName`	Display string	Name of the virtual console concentrator service that has changed
`ldomChangeFlag`	Integer	Indicates one of the following changes that occurred to the virtual console concentrator: ■ 1 is Added ■ 2 is Modified ■ 3 is Removed
`ldomTrapDesc`	Display string	Description of the trap

Virtual Console Group Change (ldomVconsChange)

This trap notifies you when a domain's virtual console group changes.

TABLE 21–36 Virtual Console Group Change Trap (ldomVconsChange)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain that contains the virtual console group
`ldomVconsGroupName`	Display string	Name of the virtual console group that has changed
`ldomChangeFlag`	Integer	Indicates one of the following changes that occurred to the virtual console group: ■ 1 is Added ■ 2 is Modified ■ 3 is Removed
`ldomTrapDesc`	Display string	Description of the trap

CMI Resource Change (ldomCMIChange)

This trap notifies you when the number of CMI resources in a domain changes.

TABLE 21–37 CMI Resource Change Trap (ldomCMIChange)

Name	Data Type	Description
`ldomIndexNotif`	Integer	Index into `ldomTable`
`ldomName`	Display string	Name of the domain that contains the CMI resource
`ldomNumCMI`	Integer	New number of CMI resources for the domain
`ldomNumCMIPrev`	Integer	Previous number of CMI resources for the domain
`ldomTrapDesc`	Display string	Description of the trap

Starting and Stopping Domains

This section describes the active management operations that you use to stop and start domains. You can control these active management operations by setting a value for the `ldomAdminState` property of the Domain Table, `ldomTable`. See Table 21–1.

▼ How to Start a Domain

This procedure describes how to start an existing bound domain. If a domain with the specified domain name does not exist or is not already bound, this operation fails.

1 Verify that the *domain-name* domain exists and is bound.

```
# ldm list domain-name
```

2 Identify *domain-name* in `ldomTable`.

```
# snmpwalk -v1 -c public localhost SUN-LDOM-MIB::ldomTable
SUN-LDOM-MIB::ldomName.1 = STRING: primary
SUN-LDOM-MIB::ldomName.2 = STRING: LdomMibTest_1
SUN-LDOM-MIB::ldomAdminState.1 = INTEGER: 0
SUN-LDOM-MIB::ldomAdminState.2 = INTEGER: 0
SUN-LDOM-MIB::ldomOperState.1 = INTEGER: active(1)
SUN-LDOM-MIB::ldomOperState.2 = INTEGER: bound(6)
SUN-LDOM-MIB::ldomNumVCpu.1 = INTEGER: 8
SUN-LDOM-MIB::ldomNumVCpu.2 = INTEGER: 4
SUN-LDOM-MIB::ldomMemSize.1 = INTEGER: 3360
SUN-LDOM-MIB::ldomMemSize.2 = INTEGER: 256
SUN-LDOM-MIB::ldomMemUnit.1 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomMemUnit.2 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomNumCrypto.1 = INTEGER: 1
SUN-LDOM-MIB::ldomNumCrypto.2 = INTEGER: 0
SUN-LDOM-MIB::ldomNumIOBus.1 = INTEGER: 2
SUN-LDOM-MIB::ldomNumIOBus.2 = INTEGER: 0
```

```
SUN-LDOM-MIB::ldomUUID.1 = STRING: 5f8817d4-5d2e-6f7d-c4af-91b5b34b5723
SUN-LDOM-MIB::ldomUUID.2 = STRING: 11284146-87ca-4877-8d80-cd0f60d5ec26
SUN-LDOM-MIB::ldomMacAddress.1 = STRING: 00:14:4f:46:47:d6
SUN-LDOM-MIB::ldomMacAddress.2 = STRING: 00:14:4f:f8:d5:6c
SUN-LDOM-MIB::ldomHostID.1 = STRING: 0x844647d6
SUN-LDOM-MIB::ldomHostID.2 = STRING: 0x84f8d56c
SUN-LDOM-MIB::ldomFailurePolicy.1 = STRING: ignore
SUN-LDOM-MIB::ldomFailurePolicy.2 = STRING: ignore
SUN-LDOM-MIB::ldomMaster.1 = STRING:
SUN-LDOM-MIB::ldomMaster.2 = STRING:
SUN-LDOM-MIB::ldomExtMapinSpace.1 = STRING: off
SUN-LDOM-MIB::ldomExtMapinSpace.2 = STRING: off
SUN-LDOM-MIB::ldomWholeCore.1 = INTEGER: 0
SUN-LDOM-MIB::ldomWholeCore.2 = INTEGER: 0
SUN-LDOM-MIB::ldomCpuArch.1 = STRING: native
SUN-LDOM-MIB::ldomCpuArch.2 = STRING: native
SUN-LDOM-MIB::ldomShutdownGroup.1 = INTEGER: 0
SUN-LDOM-MIB::ldomShutdownGroup.2 = INTEGER: 15
SUN-LDOM-MIB::ldomPerfCounters.1 = STRING: htstrand
SUN-LDOM-MIB::ldomPerfCounters.2 = STRING: global,htstrand
SUN-LDOM-MIB::ldomNumCMI.1 = INTEGER: 0
SUN-LDOM-MIB::ldomNumCMI.2 = INTEGER: 0
```

3 Start the *domain-name* domain.

Use the `snmpset` command to start the domain by setting a value of `1` to the `ldomAdminState` property. *n* specifies the domain to start.

```
# snmpset -v version -c community-string hostname \
SUN-LDOM-MIB::ldomTable.1.ldomAdminState.n = 1
```

4 Verify that the *domain-name* domain is active by using one of the following commands:

-
  ```
  # ldm list domain-name
  ```
-
  ```
  # snmpget -v version -c community-string hostname SUN-LDOM-MIB::ldomOperState.n
  ```

Example 21–5 Starting a Guest Domain

This example verifies that the `LdomMibTest_1` domain exists and is bound before setting the `ldomAdminState` property to `1`. Finally, the `ldm list LdomMibTest_1` command verifies that the `LdomMibTest_1` domain is active.

```
# ldm list LdomMibTest_1
# snmpset -v1 -c private localhost SUN-LDOM-MIB::ldomTable.1.ldomAdminState.2 = 1
# ldm list LdomMibTest_1
```

You can also use the `snmpget` command to retrieve the `LdomMibTest_1` domain's state instead of using the `ldm list` command.

```
# snmpget -v1 -c public localhost SUN-LDOM-MIB::ldomOperState.2
```

Note that if the domain is inactive when you use `snmpset` to start the domain, the domain is first bound and then started.

How to Stop a Domain

This procedure describes how to stop a started domain. Any operating system instances that are hosted by the domain are stopped.

1 Identify *domain-name* in `ldomTable`.

```
# snmpwalk -v1 -c public localhost SUN-LDOM-MIB::ldomTable
SUN-LDOM-MIB::ldomName.1 = STRING: primary
SUN-LDOM-MIB::ldomName.2 = STRING: LdomMibTest_1
SUN-LDOM-MIB::ldomAdminState.1 = INTEGER: 0
SUN-LDOM-MIB::ldomAdminState.2 = INTEGER: 0
SUN-LDOM-MIB::ldomOperState.1 = INTEGER: active(1)
SUN-LDOM-MIB::ldomOperState.2 = INTEGER: bound(6)
SUN-LDOM-MIB::ldomNumVCpu.1 = INTEGER: 8
SUN-LDOM-MIB::ldomNumVCpu.2 = INTEGER: 4
SUN-LDOM-MIB::ldomMemSize.1 = INTEGER: 3360
SUN-LDOM-MIB::ldomMemSize.2 = INTEGER: 256
SUN-LDOM-MIB::ldomMemUnit.1 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomMemUnit.2 = INTEGER: megabytes(2)
SUN-LDOM-MIB::ldomNumCrypto.1 = INTEGER: 1
SUN-LDOM-MIB::ldomNumCrypto.2 = INTEGER: 0
SUN-LDOM-MIB::ldomNumIOBus.1 = INTEGER: 2
SUN-LDOM-MIB::ldomNumIOBus.2 = INTEGER: 0
SUN-LDOM-MIB::ldomUUID.1 = STRING: 5f8817d4-5d2e-6f7d-c4af-91b5b34b5723
SUN-LDOM-MIB::ldomUUID.2 = STRING: 11284146-87ca-4877-8d80-cd0f60d5ec26
SUN-LDOM-MIB::ldomMacAddress.1 = STRING: 00:14:4f:46:47:d6
SUN-LDOM-MIB::ldomMacAddress.2 = STRING: 00:14:4f:f8:d5:6c
SUN-LDOM-MIB::ldomHostID.1 = STRING: 0x844647d6
SUN-LDOM-MIB::ldomHostID.2 = STRING: 0x84f8d56c
SUN-LDOM-MIB::ldomFailurePolicy.1 = STRING: ignore
SUN-LDOM-MIB::ldomFailurePolicy.2 = STRING: ignore
SUN-LDOM-MIB::ldomMaster.1 = STRING:
SUN-LDOM-MIB::ldomMaster.2 = STRING:
SUN-LDOM-MIB::ldomExtMapinSpace.1 = STRING: off
SUN-LDOM-MIB::ldomExtMapinSpace.2 = STRING: off
SUN-LDOM-MIB::ldomWholeCore.1 = INTEGER: 0
SUN-LDOM-MIB::ldomWholeCore.2 = INTEGER: 0
SUN-LDOM-MIB::ldomCpuArch.1 = STRING: native
SUN-LDOM-MIB::ldomCpuArch.2 = STRING: native
SUN-LDOM-MIB::ldomShutdownGroup.1 = INTEGER: 0
SUN-LDOM-MIB::ldomShutdownGroup.2 = INTEGER: 15
SUN-LDOM-MIB::ldomPerfCounters.1 = STRING: htstrand
SUN-LDOM-MIB::ldomPerfCounters.2 = STRING: global,htstrand
SUN-LDOM-MIB::ldomNumCMI.1 = INTEGER: 0
SUN-LDOM-MIB::ldomNumCMI.2 = INTEGER: 0
```

2 Stop the *domain-name* domain.

Use the `snmpset` command to stop the domain by setting a value of `2` to the `ldomAdminState` property. *n* specifies the domain to stop.

```
# snmpset -v version -c community-string hostname \
SUN-LDOM-MIB::ldomTable.1.ldomAdminState.n = 2
```

3 Verify that the *domain-name* domain is bound by using one of the following commands:

-
  ```
  # ldm list domain-name
  ```
-
  ```
  # snmpget -v version -c community-string hostname SUN-LDOM-MIB::ldomOperState.n
  ```

Example 21–6 Stopping a Guest Domain

This example sets the `ldomAdminState` property to `2` to stop the guest domain and then uses the `ldm list LdomMibTest_1` command to verify that the `LdomMibTest_1` domain is bound.

```
# snmpset -v1 -c private localhost SUN-LDOM-MIB::ldomTable.1.ldomAdminState.2 = 2
# ldm list LdomMibTest_1
```

◆ ◆ ◆ CHAPTER 22

Logical Domains Manager Discovery

This chapter provides information about discovering the Logical Domains Manager running on systems on a subnet.

Discovering Systems Running the Logical Domains Manager

Logical Domains Managers can be discovered on a subnet by using multicast messages. The `ldmd` daemon is able to listen on a network for a specific multicast packet. If that multicast message is of a certain type, `ldmd` replies to the caller. This enables `ldmd` to be discovered on systems that are running Oracle VM Server for SPARC.

Multicast Communication

This discovery mechanism uses the same multicast network that is used by the `ldmd` daemon to detect collisions when automatically assigning MAC addresses. To configure the multicast socket, you must supply the following information:

```
#define    MAC_MULTI_PORT        64535
#define    MAC_MULTI_GROUP        "239.129.9.27"
```

By default, *only* multicast packets can be sent on the subnet to which the machine is attached. You can change the behavior by setting the `ldmd/hops` SMF property for the `ldmd` daemon.

Message Format

The discovery messages must be clearly marked so as not to be confused with other messages. The following multicast message format ensures that discovery messages can be distinguished by the discovery listening process:

```
#include <netdb.h> /* Used for MAXHOSTNAMELEN definition */
#define    MAC_MULTI_MAGIC_NO    92792004
#define    MAC_MULTI_VERSION    1

enum {
    SEND_MSG = 0,
    RESPONSE_MSG,
    LDMD_DISC_SEND,
    LDMD_DISC_RESP,
};

typedef struct {
    uint32_t    version_no;
    uint32_t    magic_no;
    uint32_t    msg_type;
    uint32_t    resv;
    union {
        mac_lookup_t        Mac_lookup;
        ldmd_discovery_t    Ldmd_discovery;
    } payload;
#define    lookup        payload.Mac_lookup
#define    discovery    payload.Ldmd_discovery
} multicast_msg_t;

#define    LDMD_VERSION_LEN    32

typedef struct {
    uint64_t mac_addr;
    char        source_ip[INET_ADDRSTRLEN];
} mac_lookup_t;

typedef struct {
    char        ldmd_version[LDMD_VERSION_LEN];
    char        hostname[MAXHOSTNAMELEN];
    struct in_addr    ip_address;
    int        port_no;
} ldmd_discovery_t;
```

▼ How to Discover Logical Domains Managers Running on Your Subnet

1 Open a multicast socket.

Ensure that you use the port and group information specified in "Multicast Communication" on page 465.

2 Send a `multicast_msg_t` message over the socket.

The message should include the following:

- Valid value for `version_no`, which is `1` as defined by `MAC_MULTI_VERSION`
- Valid value for `magic_no`, which is `92792004` as defined by `MAC_MULTI_MAGIC_NO`
- `msg_type` of `LDMD_DISC_SEND`

3 Listen on the multicast socket for responses from Logical Domains Managers.

The responses must be a `multicast_msg_t` message with the following information:

- Valid value for `version_no`
- Valid value for `magic_no`
- `msg_type` set to `LDMD_DISC_RESP`
- Payload consisting of a `ldmd_discovery_t` structure, which contains the following information:
  - `ldmd_version` – Version of the Logical Domains Manager running on the system
  - `hostname` – Host name of the system
  - `ip_address` – IP address of the system
  - `port_no` – Port number being used by the Logical Domains Manager for communications, which should be XMPP port `6482`

When listening for a response from Logical Domains Managers, ensure that any auto-allocation MAC collision-detection packets are discarded.

◆ ◆ ◆ CHAPTER 23

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

External programs can use the Extensible Messaging and Presence Protocol (XMPP – RFC 3920) to communicate with the Logical Domains Manager. XMPP is supported for both local and remote connections and is on by default. To disable a remote connection, set the `ldmd/xmpp_enabled` SMF property to `false` and restart the Logical Domains Manager.

```
# svccfg -s ldom/ldmd setprop ldmd/xmpp_enabled=false
# svcadm refresh ldmd
# svcadm restart ldmd
```

Note – Disabling the XMPP server also prevents domain migration and the dynamic reconfiguration of memory.

XMPP Server

The Logical Domains Manager implements an XMPP server which can communicate with numerous available XMPP client applications and libraries. The Logical Domains Manager uses the following security mechanisms:

- Transport Layer Security (TLS) to secure the communication channel between the client and itself.
- Simple Authentication and Security Layer (SASL) for authentication. `PLAIN` is the only SASL mechanism supported. You must send in a user name and password to the server so it can authorize you before allowing monitoring or management operations.

Local Connections

The Logical Domains Manager detects whether user clients are running on the same domain as itself and, if so, does a minimal XMPP handshake with that client. Specifically, the SASL authentication step after the setup of a secure channel through TLS is skipped. Authentication and authorization are done based on the credentials of the process implementing the client interface.

Clients can choose to implement a full XMPP client or to simply run a streaming XML parser, such as the `libxml2` Simple API for XML (SAX) parser. Either way, the client has to handle an XMPP handshake to the point of TLS negotiation. Refer to the XMPP specification for the sequence needed.

XML Protocol

After communication initialization is complete, Oracle VM Server for SPARC-defined XML messages are sent next. The two general types of XML messages are:

- Request and response messages, which use the `<LDM_interface>` tag. This type of XML message is used for communicating commands and getting results back from the Logical Domains Manager, analogous to executing commands using the command-line interface (CLI). This tag is also used for event registration and unregistration.
- Event messages, which use the `<LDM_event>` tag. This type of XML message is used to asynchronously report events posted by the Logical Domains Manager.

Request and Response Messages

The XML interface into Oracle VM Server for SPARC has two different formats:

- A format for sending commands into the Logical Domains Manager
- A format for Logical Domains Manager to respond on the status of the incoming message and the actions requested within that message.

The two formats share many common XML structures, but they are separated in this discussion for a better understanding of the differences between them.

Request Messages

An incoming XML request to the Logical Domains Manager at its most basic level includes a description of a single command operating on a single object. More complicated requests can handle multiple commands and multiple objects per command. The following example shows the structure of a basic XML command.

EXAMPLE 23–1 Format of a Single Command Operating on a Single Object

```
<LDM_interface version="1.3">
  <cmd>
    <action>Place command here</action>
    <options>Place options for certain commands here</options>
    <arguments>Place arguments for certain commands here</arguments>
    <data version="3.0">
      <Envelope>
        <References/>
        <!-- Note a <Section> section can be here instead of <Content>
        <Content xsi:type="ovf:VirtualSystem_Type" id="Domain name">
          <Section xsi:type="ovf:ResourceAllocationSection_type">
            <Item>
              <rasd:OtherResourceType>LDom Resource Type</rasd:OtherResourceType>
              <gprop:GenericProperty
              key="Property name">Property Value</gprop:GenericProperty>
            </Item>
          </Section>

        </Content>
      </Envelope>
    </data>

  </cmd>

</LDM_interface>
```

<LDM_interface> Tag

All commands sent to the Logical Domains Manager must start with the `<LDM_interface>` tag. Any document sent into the Logical Domains Manager must have only one `<LDM_interface>` tag contained within it. The `<LDM_interface>` tag must include a version attribute as shown in Example 23–1.

The <cmd> Tag

Within the <LDM_interface> tag, the document must include at least one <cmd> tag. Each <cmd> section must have only one <action> tag. Use the <action> tag to describe the command to run. Each <cmd> tag must include at least one <data> tag to describe the objects on which the command is to operate.

The <cmd> tag can also have an <options> tag, which is used for options and flags that are associated with some commands. The following commands use options:

- The `ldm remove-domain` command can use the `-a` option.
- The `ldm bind-domain` command can use the `-f` option.
- The `ldm add-vdsdev` command can use the `-f` option.
- The `ldm cancel-operation` command can use the `migration` or `reconf` option.
- The `ldm add-spconfig` command can use the `-r` *autosave-name* option.
- The `ldm remove-spconfig` command can use the `-r` option.
- The `ldm list-spconfig` command can use the `-r` [*autosave-name*] option.
- The `ldm stop-domain` command can use the following tags to set the command arguments:
  - <force> represents the `-f` option.
  - <halt> represents the `-h` option.
  - <message> represents the `-m` option.
  - <quick> represents the `-q` option.
  - <reboot> represents the `-r` option.
  - <timeout> represents the `-t` option.

  Note that the tags must not have any content value. However, the `-t` and `-m` options must have a non-null value, for example, <timeout>10</timeout> or <message>Shutting down now</message>.

The following XML example fragment shows how to pass a reboot request with a reboot message to the `ldm stop-domain` command:

```
<action>stop-domain</action>
<arguments>
    <reboot/>
    <message>my reboot message</message>
</arguments>
```

The <data> Tag

Each <data> section contains a description of an object pertinent to the command specified. The format of the <data> section is based on the XML schema portion of the Open Virtualization Format (OVF) draft specification. That schema defines an <Envelope> section which contains a <References> tag (unused by Oracle VM Server for SPARC) and <Content> and <Section> sections.

For Oracle VM Server for SPARC, the <Content> section is used to identify and describe a particular domain. The domain name in the id= attribute of the <Content> node identifies the domain. Within the <Content> section are one or more <Section> sections describing resources of the domain as needed by the particular command.

If you need to identify only a domain name, then you do not need to use any <Section> tags. Conversely, if no domain identifier is needed for the command, then you need to provide a <Section> section describing the resources needed for the command, placed outside a <Content> section but still within the <Envelope> section.

A <data> section does not need to contain an <Envelope> tag in cases where the object information can be inferred. This situation mainly applies to requests for monitoring all objects applicable to an action, and event registration and unregistration requests.

Two additional OVF types enable the use of the OVF specification's schema to properly define all types of objects:

- <gprop:GenericProperty> tag
- <Binding> tag

The <gprop:GenericProperty> tag handles any object's property for which the OVF specification does not have a definition. The property name is defined in the key= attribute of the node and the value of the property is the contents of the node. The <binding> tag is used in the ldm list-bindings command output to define resources that are bound to other resources.

Response Messages

An outgoing XML response closely matches the structure of the incoming request in terms of the commands and objects included, with the addition of a <Response> section for each object and command specified, as well as an overall <Response> section for the request. The <Response> sections provide status and message information. The following example shows the structure of a response to a basic XML request.

EXAMPLE 23–2 Format of a Response to a Single Command Operating on a Single Object

```
<LDM_interface version="1.3">
  <cmd>
    <action>Place command here</action>
    <data version="3.0">
      <Envelope>
        <References/>
        <!-- Note a <Section> section can be here instead of <Content>
        <Content xsi:type="ovf:VirtualSystem_Type" id="Domain name">
          <Section xsi:type="ovf:ResourceAllocationSection_type">
            <Item>
              <rasd:OtherResourceType>
                LDom Resource Type
              </rasd:OtherResourceType>
              <gprop:GenericProperty
              key="Property name">
```

EXAMPLE 23–2 Format of a Response to a Single Command Operating on a Single Object *(Continued)*

```
                Property Value
              </gprop:GenericProperty>
              </Item>
            </Section>
            <!-- Note: More <Section>
          </Content>
        </Envelope>
        <response>
          <status>success or failure</status>
          <resp_msg>Reason for failure</resp_msg>
        </response>
      </data>

      <response>
        <status>success or failure</status>
        <resp_msg>Reason for failure</resp_msg>
      </response>
    </cmd>

    <response>
      <status>success or failure</status>
      <resp_msg>Reason for failure</resp_msg>
    </response>
  </LDM_interface>
```

Overall Response

This <response> section, which is the direct child of the <LDM_interface> section, indicates overall success or failure of the entire request. Unless the incoming XML document is malformed, the <response> section includes only a <status> tag. If this response status indicates success, all commands on all objects have succeeded. If this response status is a failure and there is no <resp_msg> tag, then one of the commands included in the original request failed. The <resp_msg> tag is used only to describe some problem with the XML document itself.

Command Response

The <response> section under the <cmd> section alerts the user to success or failure of that particular command. The <status> tag shows whether that command succeeds or fails. As with the overall response, if the command fails, the <response> section includes only a <resp_msg> tag if the contents of the <cmd> section of the request is malformed. Otherwise, the failed status means one of the objects that the command ran against caused a failure.

Object Response

Finally, each <data> section in a <cmd> section also has a <response> section. This section shows whether the command being run on this particular object passes or fails. If the status of the response is SUCCESS, there is no <resp_msg> tag in the <response> section. If the status is

FAILURE, there are one or more <resp_msg> tags in the <response> field depending on the errors encountered when running the command against that object. Object errors can result from problems found when running the command, or a malformed or unknown object.

In addition to the <response> section, the <data> section can contain other information. This information is in the same format as an incoming <data> field, describing the object that caused a failure. See "The <data> Tag" on page 472. 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 476 for complete details.

Registration and Unregistration

Use an <LDM_interface> message to register for events. See "<LDM_interface> Tag" on page 471. The <action> tag details the type of event for which to register or unregister and the <data> section is left empty.

EXAMPLE 23–3 Example Event Registration Request Message

```
<LDM_interface version="1.3">
  <cmd>
    <action>reg-domain-events</action>
    <data version="3.0"/>
  </cmd>
</LDM_interface>
```

The Logical Domains Manager responds with an <LDM_interface> response message stating whether the registration or unregistration was successful.

EXAMPLE 23–4 Example Event Registration Response Message

```
<LDM_interface version="1.3">
  <cmd>
    <action>reg-domain-events</action>
    <data version="3.0"/>
      <response>
        <status>success</status>
```

EXAMPLE 23–4 Example Event Registration Response Message *(Continued)*

```
      </response>
    </data>
    <response>
      <status>success</status>
    </response>
  </cmd>
  <response>
    <status>success</status>
  </response>
</LDM_interface>
```

The action string for each type of event is listed in the events subsection.

<LDM_event> Messages

Event messages have the same format as an incoming <LDM_interface> message with the exception that the start tag for the message is <LDM_event>. The <action> tag of the message is the action that was performed to trigger the event. The <data> section of the message describes the object associated with the event; the details depend on the type of event that occurred.

EXAMPLE 23–5 Example <LDM_event> Notification

```
<LDM_event version='1.1'>
  <cmd>
    <action>Event command here</action>
    <data version='3.0'>
      <Envelope
        <References/>
        <Content xsi:type='ovf:VirtualSystem_Type' ovf:id='ldg1'/>
          <Section xsi:type="ovf:ResourceAllocationSection_type">
            <Item>
              <rasd:OtherResourceType>LDom Resource Type</rasd:OtherResourceType>
              <gprop:GenericProperty
              key="Property name">Property Value</gprop:GenericProperty>
            </Item>
          </Section>
      </Envelope>
    </data>
  </cmd>
</LDM_event>
```

Event Types

You can subscribe to the following event types:

- Domain events
- Hardware events
- Progress events

- Resource events

All the events correspond to `ldm` subcommands.

Domain Events

Domain events describe which actions can be performed directly to a domain. The following domain events can be specified in the `<action>` tag in the `<LDM_event>` message:

- `add-domain`
- `bind-domain`
- `domain-reset`
- `migrate-domain`
- `panic-domain`
- `remove-domain`
- `start-domain`
- `stop-domain`
- `unbind-domain`

These events always contain *only* a `<Content>` tag in the OVF `<data>` section that describes the domain to which the event happened. To register for the domain events, send an `<LDM_interface>` message with the `<action>` tag set to `reg-domain-events`. To unregister for these events, send an `<LDM_interface>` message with the `<action>` tag set to `unreg-domain-events`.

Hardware Events

Hardware events pertain to changing the physical system hardware. In the case of Oracle VM Server for SPARC software, the only hardware changes are those to the service processor (SP) when you add, remove, or set an SP configuration. Currently, the only three events of this type are:

- `add-spconfig`
- `set-spconfig`
- `remove-spconfig`

The hardware events always contain *only* a `<Section>` tag in the OVF `<data>` section which describes which SP configuration to which the event is happening. To register for these events, send an `<LDM_interface>` message with the `<action>` tag set to `reg-hardware-events`. To unregister for these events, send an `<LDM_interface>` message with the `<action>` tag set to `unreg-hardware-events`.

Progress Events

Progress events are issued for long-running commands, such as a domain migration. These events report the amount of progress that has been made during the life of the command. At this time, only the `migration-process` event is reported.

Progress events always contain only a <Section> tag in the OVF <data> section that describes the SP configuration affected by the event. To register for these events, send an <LDM_interface> message with the <action> tag set to reg-hardware-events. To unregister for these events, send an <LDM_interface> message with the <action> tag set to unreg-hardware-events.

The <data> section of a progress event consists of a <content> section that describes the affected domain. This <content> section uses an ldom_info <Section> tag to update progress. The following generic properties are shown in the ldom_info section:

- --progress – Percentage of the progress made by the command
- --status – Command status, which can be one of ongoing, failed, or done
- --source – Machine that is reporting the progress

Resource Events

Resource events occur when resources are added, removed, or changed in any domain. The <data> section for some of these events contains the <Content> tag with a <Section> tag providing a service name in the OVF <data> section.

The following events can be specified in the <action> tag in the <LDM_event> message:

- add-vdiskserverdevice
- remove-vdiskserverdevice
- set-vdiskserverdevice
- remove-vdiskserver
- set-vconscon
- remove-vconscon
- set-vswitch
- remove-vswitch
- remove-vdpcs

The following resource events always contain *only* the <Content> tag in the OVF <data> section that describes the domain to which the event happened:

- add-vcpu
- add-crypto
- add-memory
- add-io
- add-variable
- add-vconscon
- add-vdisk
- add-vdiskserver
- add-vnet
- add-vswitch
- add-vdpcs
- add-vdpcc

- set-vcpu
- set-crypto
- set-memory
- set-variable
- set-vnet
- set-vconsole
- set-vdisk
- remove-vcpu
- remove-crypto
- remove-memory
- remove-io
- remove-variable
- remove-vdisk
- remove-vnet
- remove-vdpcc

To register for the resource events, send an <LDM_interface> message with the <action> tag set to reg-resource-events. Unregistering for these events requires an <LDM_interface> message with the <action> tag set to unreg-resource-events.

All Events

You can also register to listen for all three type of events without having to register for each one individually. To register for all three types of events simultaneously, send an <LDM_interface> message with the <action> tag set to reg-all-events. Unregistering for these events require an <LDM_interface> message with the <action> tag set to unreg-all-events.

Logical Domains Manager Actions

The commands specified in the <action> tag, with the exception of *-*-events commands, correspond to those of the ldm command-line interface. For details about ldm subcommands, see the ldm(1M) man page.

Note – The XML interface does not support the verb or command aliases supported by the Logical Domains Manager CLI.

The supported strings in the <action> tag are as follows:

- add-domain
- add-io
- add-mau
- add-memory
- add-spconfig

- add-variable
- add-vconscon
- add-vcpu
- add-vdisk
- add-vdiskserver
- add-vdiskserverdevice
- add-vdpcc
- add-vdpcs
- add-vnet
- add-vswitch
- bind-domain
- cancel-operation
- list-bindings
- list-constraints
- list-devices
- list-domain
- list-services
- list-spconfig
- list-variable
- migrate-domain
- reg-all-events
- reg-domain-events
- reg-hardware-events
- reg-resource-events
- remove-domain
- remove-io
- remove-mau
- remove-memory
- remove-reconf
- remove-spconfig
- remove-variable
- remove-vconscon
- remove-vcpu
- remove-vdisk
- remove-vdiskserver
- remove-vdiskserverdevice
- remove-vdpcc
- remove-vdpcs
- remove-vnet
- remove-vswitch
- set-domain
- set-mau
- set-memory
- set-spconfig

- set-variable
- set-vconscon
- set-vconsole
- set-vcpu
- set-vnet
- set-vswitch
- start-domain
- stop-domain
- unbind-domain
- unreg-all-events
- unreg-domain-events
- unreg-hardware-events
- unreg-resource-events

Logical Domains Manager Resources and Properties

This section provides examples of the Logical Domains Manager resources and the properties that can be defined for each of those resources. The resources and properties are shown in **bold** type in the XML examples. These examples show resources, not binding output. The constraint output can be used to create input for the Logical Domains Manager actions except domain migration output. See "Domain Migration" on page 496. Each resource is defined in a `<Section>` OVF section and is specified by a `<rasd:OtherResourceType>` tag.

Domain Information (ldom_info) Resource

The following example shows the optional properties of the `ldom_info` resource:

EXAMPLE 23–6 Example ldom_info XML Output

The following example shows values specified for several `ldom_info` properties such as `uuid`, `hostid`, and `Address`.

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="primary">
    <Section xsi:type="ovf:ResourceAllocationSection_type">
      <Item>
        <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
        <uuid>c2c3d93b-a3f9-60f6-a45e-f35d55c05fb6</uuid>
        <rasd:Address>00:03:ba:d8:ba:f6</rasd:Address>
        <gprop:GenericProperty key="hostid">83d8baf6</gprop:GenericProperty>
        <gprop:GenericProperty key="master">plum</gprop:GenericProperty>
        <gprop:GenericProperty key="failure-policy">reset</gprop:GenericProperty>
        <gprop:GenericProperty key="extended-mapin-space">on</gprop:GenericProperty>
        <gprop:GenericProperty key="progress">45%</gprop:GenericProperty>
        <gprop:GenericProperty key="status">ongoing</gprop:GenericProperty>
```

EXAMPLE 23–6 Example ldom_info XML Output *(Continued)*

```
        <gprop:GenericProperty key="source">system1</gprop:GenericProperty>
        <gprop:GenericProperty key="rc-add-policy"></gprop:GenericProperty>
        <gprop:GenericProperty key="perf-counters">global</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

The `ldom_info` resource is always contained within a `<Content>` section. The following properties within the `ldom_info` resource are optional properties:

- `<uuid>` tag, which specifies the UUID of the domain.
- `<rasd:Address>` tag, which specifies the MAC address to be assigned to a domain.
- `<gprop:GenericProperty key="extended-mapin-space">` tag, which specifies whether extended mapin space is enabled (`on`) or disabled (`off`) for the domain. The default value is `off`.
- `<gprop:GenericProperty key="failure-policy">` tag, which specifies how slave domains should behave should the master domain fail. The default value is `ignore`. Following are the valid property values:
  - `ignore` ignores failures of the master domain (slave domains are unaffected).
  - `panic` panics any slave domains when the master domain fails.
  - `reset` resets any slave domains when the master domain fails.
  - `stop` stops any slave domains when the master domain fails.
- `<gprop:GenericProperty key="hostid">` tag, which specifies the host ID to be assigned to the domain.
- `<gprop:GenericProperty key="master">` tag, which specifies up to four comma-separated master domain names.
- `<gprop:GenericProperty key="progress">` tag, which specifies the percentage of progress made by the command.
- `<gprop:GenericProperty key="source">` tag, which specifies the machine reporting on the progress of the command.
- `<gprop:GenericProperty key="status">` tag, which specifies the status of the command (`done`, `failed`, or `ongoing`).
- `<gprop:GenericProperty key="rc-add-policy">` tag, which specifies whether to enable or disable the direct I/O and SR-IOV I/O virtualization operations on any root complex that might be added to the specified domain. Valid values are `iov` and no value (`rc-add-policy=`).
- `<gprop:GenericProperty key="perf-counters">` tag, which specifies the performance register sets to access (`global`, `htstrand`, `strand`).

  If the platform does not have the performance access capability, the `perf-counters` property value is ignored.

CPU (cpu) Resource

Note that the allocation units property, <rasd:AllocationUnits>, for the cpu resource always specifies the number of virtual CPUs and not the number of cores.

A cpu resource is always contained within a <Content> section.

EXAMPLE 23–7 cpu XML Section Output from the ldm list-bindings Command

The following example shows the XML output for the <cpu> section by using the ldm list-bindings command.

```
<?xml version="1.0"?>
<LDM_interface
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ovf="./schemas/envelope"
xmlns:rasd="./schemas/CIM_ResourceAllocationSettingData"
xmlns:vssd="./schemas/CIM_VirtualSystemSettingData"
xmlns:gprop="./schemas/GenericProperty"
xmlns:bind="./schemas/Binding"
version="1.3"
xsi:noNamespaceSchemaLocation="./schemas/combined-v3.xsd">
  <cmd>
    <action>list-bindings</action>
    <data version="3.0">
      <Envelope>
        <References/>
        <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="primary">
          <Section xsi:type="ovf:ResourceAllocationSection_Type">
            <Item>
              <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
              <uuid>1e04cdbd-472a-e8b9-ba4c-d3eee86e7725</uuid>
              <rasd:Address>00:21:28:f5:11:6a</rasd:Address>
              <gprop:GenericProperty key="hostid">0x8486632a</gprop:GenericProperty>
              <failure-policy>fff</failure-policy>
              <wcore>0</wcore>
              <extended-mapin-space>0</extended-mapin-space>
              <cpu-arch>native</cpu-arch>
              <rc-add-policy/>
              <gprop:GenericProperty key="state">active</gprop:GenericProperty>
            </Item>
          </Section>
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>cpu</rasd:OtherResourceType>
              <rasd:AllocationUnits>8</rasd:AllocationUnits>
              <bind:Binding>
                <Item>
                  <rasd:OtherResourceType>cpu</rasd:OtherResourceType>
                  <gprop:GenericProperty key="vid">0</gprop:GenericProperty>
                  <gprop:GenericProperty key="pid">0</gprop:GenericProperty>
                  <gprop:GenericProperty key="cid">0</gprop:GenericProperty>
                  <gprop:GenericProperty key="strand_percent">100</gprop:GenericProperty>
                  <gprop:GenericProperty key="util_percent">1.1%</gprop:GenericProperty>
                  <gprop:GenericProperty key="normalized_utilization">0.1%</gprop:GenericProperty>
                </Item>
```

EXAMPLE 23–7 cpu XML Section Output from the `ldm list-bindings` Command *(Continued)*

```
        </Section>
      </Content>
    </Envelope>
  </data>
 </cmd>
</LDM_interface>
```

EXAMPLE 23–8 cpu XML Section Output from the `ldm list-domain` Command

The following example shows the XML output for the <cpu> section by using the `ldm list-domain` command.

```
<?xml version="1.0"?>
<LDM_interface
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ovf="./schemas/envelope"
xmlns:rasd="./schemas/CIM_ResourceAllocationSettingData"
xmlns:vssd="./schemas/CIM_VirtualSystemSettingData"
xmlns:gprop="./schemas/GenericProperty"
xmlns:bind="./schemas/Binding"
version="1.3"
xsi:noNamespaceSchemaLocation="./schemas/combined-v3.xsd">
  <cmd>
    <action>list-domain</action>
    <data version="3.0">
      <Envelope>
        <References/>
        <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="primary">
          <Section xsi:type="ovf:ResourceAllocationSection_Type">
            <Item>
              <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
              <gprop:GenericProperty key="state">active</gprop:GenericProperty>
              <gprop:GenericProperty key="flags">-n-cv-</gprop:GenericProperty>
              <gprop:GenericProperty key="utilization">0.7%</gprop:GenericProperty>
              <gprop:GenericProperty key="uptime">3h</gprop:GenericProperty>
              <gprop:GenericProperty key="normalized_utilization">0.1%</gprop:GenericProperty>
            </Item>
          </Section>
        </Content>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

MAU (mau) Resource

A `mau` resource is always contained within a <Content> section. The only property is the <rasd:AllocationUnits> tag, which signifies the number of MAUs or other cryptographic units.

Note – The `mau` resource is any supported cryptographic unit on a supported server. Currently, the two cryptographic units supported are the Modular Arithmetic Unit (MAU) and the Control Word Queue (CWQ).

EXAMPLE 23–9 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>
```

Memory (memory) Resource

A memory resource is always contained within a `<Content>` section. The only property is the `<rasd:AllocationUnits>` tag, which signifies the amount of memory.

EXAMPLE 23–10 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>
```

Virtual Disk Server (vds) Resource

A virtual disk server (`vds`) resource can be in a `<Content>` section as part of a domain description, or it can appear on its own in an `<Envelope>` section. The only property is the `<gprop:GenericProperty>` tag with a key of `service_name`, which contains the name of the `vds` resource being described.

EXAMPLE 23–11 Example vds XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
```

EXAMPLE 23–11 Example vds XML *(Continued)*

```
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vds</rasd:OtherResourceType>
        <gprop:GenericProperty
          key="service_name">vdstmp</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

Virtual Disk Server Volume (vds_volume) Resource

A `vds_volume` resource can be in a `<Content>` section as part of a domain description, or it can appear on its own in an `<Envelope>` section. It must have `<gprop:GenericProperty>` tags with the following keys:

- `vol_name` – Name of the volume
- `service_name` – Name of the virtual disk server to which this volume is to be bound
- `block_dev` – File or device name to be associated with this volume

Optionally, a `vds_volume` resource can also have the following properties:

- `vol_opts` – One or more of the following, comma-separated, within one string: {`ro,slice,excl`}
- `mpgroup` – Name of the multipath (failover) group

EXAMPLE 23–12 Example vds_volume XML

```
<Envelope>
  <References/>
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vds_volume</rasd:OtherResourceType>
        <gprop:GenericProperty key="vol_name">vdsdev0</gprop:GenericProperty>
        <gprop:GenericProperty key="service_name">primary-vds0</gprop:GenericProperty>
        <gprop:GenericProperty key="block_dev">
          opt/SUNWldm/domain_disks/testdisk1</gprop:GenericProperty>
        <gprop:GenericProperty key="vol_opts">ro</gprop:GenericProperty>
        <gprop:GenericProperty key="mpgroup">mpgroup-name</gprop:GenericProperty>
      </Item>
    </Section>
</Envelope>
```

Disk (disk) Resource

A `disk` resource is always contained within a `<Content>` section. It must have `<gprop:GenericProperty>` tags with the following keys:

- `vdisk_name` – Name of the virtual disk
- `service_name` – Name of the virtual disk server to which this virtual disk is to be bound
- `vol_name` – Virtual disk service device with which this virtual disk is to be associated

Optionally, the `disk` resource can also have the `timeout` property, which is the timeout value in seconds for establishing a connection between a virtual disk client (`vdc`) and a virtual disk server (`vds`). If there are multiple virtual disk (`vdisk`) paths, then the `vdc` can try to connect to a different `vds`. The timeout ensures that a connection to any `vds` is established within the specified amount of time.

EXAMPLE 23–13 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>
```

Virtual Switch (vsw) Resource

A `vsw` resource can be either in a `<Content>` section as part of a domain description, or it can appear on its own in an `<Envelope>` section. It must have a `<gprop:GenericProperty>` tag with the `service_name` key, which is the name to be assigned to the virtual switch.

Optionally, the `vsw` resource can also have the following properties:

- `<rasd:Address>` – Assigns a MAC address to the virtual switch
- `default-vlan-id` – Specifies the default virtual local area network (VLAN) to which a virtual network device or virtual switch needs to be a member, in tagged mode. The first VLAN ID (*vid1*) is reserved for the `default-vlan-id`.
- `dev_path` – Path of the network device to be associated with this virtual switch
- `id` – Specifies the ID of a new virtual switch device. By default, ID values are generated automatically, so set this property if you need to match an existing device name in the OS.

- `inter_vnet_link` – Specifies whether to assign LDC channels for inter-vnet communication. The default value is `on`.
- `linkprop` – Specifies whether the virtual device should get physical link state updates. When the value is `phys-state`, the virtual device gets physical link state updates. When the value is blank, the virtual device does not get physical link state updates (the default setting).
- `mode` – `sc` for Oracle Solaris Cluster heartbeat support.
- `pvid` – Port virtual local area network (VLAN) identifier (ID) indicates the VLAN of which the virtual network needs to be a member, in untagged mode.
- `mtu` – Specifies the maximum transmission unit (MTU) of a virtual switch, virtual network devices that are bound to the virtual switch, or both. Valid values are in the range of 1500-16000. The `ldm` command issues an error if an invalid value is specified.
- `vid` – Virtual local area network (VLAN) identifier (ID) indicates the VLAN of which a virtual network and virtual switch need to be a member, in tagged mode.

EXAMPLE 23–14 Example vsw XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg2">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vsw</rasd:OtherResourceType>
        <rasd:Address>00:14:4f:fb:ec:00</rasd:Address>
        <gprop:GenericProperty key="service_name">test-vsw1</gprop:GenericProperty>
        <gprop:GenericProperty key="inter_vnet_link">on</gprop:GenericProperty>
        <gprop:GenericProperty key="default-vlan-id">1</gprop:GenericProperty>
        <gprop:GenericProperty key="pvid">1</gprop:GenericProperty>
        <gprop:GenericProperty key="mtu">1500</gprop:GenericProperty>
        <gprop:GenericProperty key="dev_path">switch@0</gprop:GenericProperty>
        <gprop:GenericProperty key="id">0</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

Network (network) Resource

A `network` resource is always contained within a `<Content>` section. It must have `<gprop:GenericProperty>` tags with the following keys:

- `linkprop` – Specifies whether the virtual device should get physical link state updates. When the value is `phys-state`, the virtual device gets physical link state updates. When the value is blank, the virtual device does not get physical link state updates (the default setting).
- `vnet_name` – Name of the virtual network (`vnet`)
- `service_name` – Name of the virtual switch (`vswitch`) to which this virtual network is to be bound

Optionally, the `network` resource can also have the following properties:

- `<rasd:Address>` – Assigns a MAC address to the virtual switch
- `pvid` – Port virtual local area network (VLAN) identifier (ID) indicates the VLAN of which the virtual network needs to be a member, in untagged mode.
- `vid` – Virtual local area network (VLAN) identifier (ID) indicates the VLAN of which a virtual network and virtual switch need to be a member, in tagged mode.
- `mode` – `hybrid` to enable hybrid I/O for that virtual network.

Note – The NIU Hybrid I/O feature is deprecated in favor of SR-IOV. Oracle VM Server for SPARC 3.3 is the last software release to include this feature.

EXAMPLE 23–15 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>
```

Virtual Console Concentrator (vcc) Resource

A `vcc` resource can be either in a `<Content>` section as part of a domain description, or it can appear on its own in an `<Envelope>` section. It can have `<gprop:GenericProperty>` tags with the following keys:

- `service_name` – Name to be assigned to the virtual console concentrator service
- `min_port` – Minimum port number to be associated with this `vcc`
- `max_port` – Maximum port number to be associated with this `vcc`

EXAMPLE 23–16 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>
```

EXAMPLE 23–16 Example vcc XML *(Continued)*

```
        <gprop:GenericProperty key="min_port">6000</gprop:GenericProperty>
        <gprop:GenericProperty key="max_port">6100</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

Variable (var) Resource

A `var` resource is always contained within a `<Content>` section. It can have `<gprop:GenericProperty>` tags with the following keys:

- `name` – Name of the variable
- `value` – Value of the variable

EXAMPLE 23–17 Example var XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>var</rasd:OtherResourceType>
        <gprop:GenericProperty key="name">test_var</gprop:GenericProperty>
        <gprop:GenericProperty key="value">test1</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

Physical I/O Device (physio_device) Resource

A `physio_device` resource is always contained within a `<Content>` section. This resource can be modified by using the `add-io`, `set-io`, `remove-io`, `create-vf`, `destroy-vf`, and `set-domain` subcommands.

EXAMPLE 23–18 Example physio_device XML

The following examples show how to perform actions on virtual functions, physical functions, and root complexes.

- The following XML example fragment shows how to use the `ldm add-io` command to add the `/SYS/MB/NET0/IOVNET.PF0.VF0` virtual function to the `ldg1` domain.

```
<LDM_interface version="1.3">
  <cmd>
    <action>add-io</action>
    <data version="3.0">
```

EXAMPLE 23–18 Example physio_device XML *(Continued)*

```
        <Envelope>
          <References/>
          <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="ldg1">
            <Section xsi:type="ovf:VirtualHardwareSection_Type">
              <Item>
                <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
                <gprop:GenericProperty key="name">
                /SYS/MB/NET0/IOVNET.PF0.VF0</gprop:GenericProperty>
              </Item>
            </Section>
          </Content>
        </Envelope>
      </data>
    </cmd>
</LDM_interface>
```

- The following XML example fragment shows how to use the `ldm set-io` command to set the `iov_bus_enable_iov` property value to `on` for the `pci_1` root complex.

```
<LDM_interface version="1.3">
  <cmd>
    <action>set-io</action>
    <data version="3.0">
      <Envelope>
        <References/>
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
              <gprop:GenericProperty key="name">pci_1</gprop:GenericProperty>
              <gprop:GenericProperty key="iov_bus_enable_iov">
              on</gprop:GenericProperty>
            </Item>
        </Section>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

- The following XML example fragment shows how to use the `ldm set-io` command to set the `unicast-slots` property value to `6` for the `/SYS/MB/NET0/IOVNET.PF1` physical function.

```
<LDM_interface version="1.3">
  <cmd>
    <action>set-io</action>
    <data version="3.0">
      <Envelope>
        <References/>
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
              <gprop:GenericProperty key="name">
              /SYS/MB/NET0/IOVNET.PF1</gprop:GenericProperty>
              <gprop:GenericProperty key="unicast-slots">6</gprop:GenericProperty>
            </Item>
          </Section>
```

EXAMPLE 23–18 Example physio_device XML *(Continued)*

```
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

- The following XML example fragment shows how to use the `ldm create-vf` command to create the `/SYS/MB/NET0/IOVNET.PF1.VF0` virtual function with the following property values.
  - `unicast-slots=6`
  - `pvid=3`
  - `mtu=1600`

```
<LDM_interface version="1.3">
  <cmd>
    <action>create-vf</action>
    <data version="3.0">
      <Envelope>
        <References/>
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>vf_device</rasd:OtherResourceType>
              <gprop:GenericProperty key="iov_pf_name">
              /SYS/MB/NET0/IOVNET.PF1</gprop:GenericProperty>
              <gprop:GenericProperty key="unicast-slots">6</gprop:GenericProperty>
              <gprop:GenericProperty key="pvid">3</gprop:GenericProperty>
              <gprop:GenericProperty key="mtu">1600</gprop:GenericProperty>
            </Item>
          </Section>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

- The following XML example fragment shows how to use the `ldm create-vf` command to create the number of virtual functions specified by the `iov_pf_repeat_count_str` value (`3`) with the `/SYS/MB/NET0/IOVNET.PF1` physical function. You cannot specify other property values when you create multiple virtual functions with the `iov_pf_repeat_count_str` property.

```
<LDM_interface version="1.3">
  <cmd>
    <action>create-vf</action>
    <data version="3.0">
      <Envelope>
        <References/>
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>vf_device</rasd:OtherResourceType>
              <gprop:GenericProperty key="iov_pf_name">
              /SYS/MB/NET0/IOVNET.PF1</gprop:GenericProperty>
              <gprop:GenericProperty key="iov_pf_repeat_count_str">
              3</gprop:GenericProperty>
            </Item>
          </Section>
```

EXAMPLE 23–18 Example physio_device XML *(Continued)*

```
        </Envelope>
      </data>
    </cmd>
  </LDM_interface>
```

SP Configuration (spconfig) Resource

A service processor (SP) configuration (`spconfig`) resource always appears on its own in an `<Envelope>` section. It can have `<gprop:GenericProperty>` tags with the following keys:

- `spconfig_name` – Name of a configuration to be stored on the SP
- `spconfig_status` – The current status of a particular SP configuration. This property is used in the output of an `ldm list-spconfig` command.

EXAMPLE 23–19 Example spconfig XML

```
<Envelope>
    <Section xsi:type="ovf:ResourceAllocationSection_type">
      <Item>
        <rasd:OtherResourceType>spconfig</rasd:OtherResourceType>
        <gprop:GenericProperty
          key="spconfig_name">primary</gprop:GenericProperty>
        <gprop:GenericProperty
          key="spconfig_status">current</gprop:GenericProperty>
      </Item>
    </Section>
</Envelope>
```

DRM Policy Configuration (policy) Resource

A DRM policy (`policy`) resource appears in an `<Envelope>` section and can have `<gprop:GenericProperty>` tags with the following keys:

- `policy_name` – Name of the DRM policy
- `policy_enable` – Specifies whether the DRM policy is enabled or disabled
- `policy_priority` – Priority of the DRM policy
- `policy_vcpu_min` – Minimum number of virtual CPU resources for a domain
- `policy_vcpu_max` – Maximum number of virtual CPU resources for a domain
- `policy_util_lower` – Lower utilization level at which policy analysis is triggered
- `policy_util_upper` – Upper utilization level at which policy analysis is triggered
- `policy_tod_begin` – Effective start time of the DRM policy
- `policy_tod_end` – Effective stop time of the DRM policy

- `policy_sample_rate` – The sample rate, which is the cycle time in seconds
- `policy_elastic_margin` – Amount of buffer between the upper and lower CPU utilization bounds
- `policy_attack` – Maximum amount of a resource to be added during any one resource control cycle
- `policy_decay` – Maximum amount of a resource to be removed during any one resource control cycle

EXAMPLE 23–20 Example policy XML

```
<Envelope>
   <Section xsi:type="ovf:VirtualHardwareSection_Type">
     <Item>
       <rasd:OtherResourceType>policy</rasd:OtherResourceType>
       <gprop:GenericProperty key="policy_name">test-policy</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_enable">on</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_priority">1</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_vcpu_min">12</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_vcpu_max">13</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_util_lower">8</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_util_upper">9</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_tod_begin">07:08:09</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_tod_end">09:08:07</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_sample_rate">1</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_elastic_margin">8</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_attack">8</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_decay">9</gprop:GenericProperty>
     </Item>
   </Section>
</Envelope>
```

Virtual Data Plane Channel Service (vdpcs) Resource

This resource is of interest only in a Netra DPS environment. A `vdpcs` resource can be either in a `<Content>` section as part of a domain description, or it can appear on its own in an `<Envelope>` section. The only property is the `<gprop:GenericProperty>` tag with the `service_name` key property value, which is the name of the virtual data plane channel service (`vdpcs`) resource being described.

EXAMPLE 23–21 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>
```

EXAMPLE 23–21 Example vdpcs XML *(Continued)*

```
</Envelope>
```

Virtual Data Plane Channel Client (vdpcc) Resource

This resource is of interest only in a Netra DPS environment. A virtual data plane channel client resource is always contained within a `<Content>` section. It can have `<gprop:GenericProperty>` tags with the following keys:

- `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

EXAMPLE 23–22 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>
```

Console (console) Resource

A `console` resource is always contained within a `<Content>` section. It can have `<gprop:GenericProperty>` tags with the following keys:

- `port` – Port to which to change this virtual console (`console`)
- `service_name` – Virtual console concentrator (`vcc`) service to which to bind this `console`
- `group` – Name of the group to which to bind this `console`
- `enable-log` – Enable or disable virtual console logging for this console

EXAMPLE 23–23 Example console XML

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
```

EXAMPLE 23–23 Example console XML *(Continued)*

```
        <rasd:OtherResourceType>console</rasd:OtherResourceType>
        <gprop:GenericProperty key="port">6000</gprop:GenericProperty>
        <gprop:GenericProperty key="service_name">vcc2</gprop:GenericProperty>
        <gprop:GenericProperty key="group">group-name</gprop:GenericProperty>
        <gprop:GenericProperty key="enable-log">on</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

Domain Migration

This example shows what is contained in the `<data>` section for a `ldm migrate-domain` command.

- First `<Content>` node (without an `<ldom_info>` section) is the source domain to migrate.
- Second `<Content>` node (with an `<ldom_info>` section) is the target domain to which to migrate. The source and target domain names can be the same.
- The `<ldom_info>` section for the target domain describes the machine to which to migrate and the details needed to migrate to that machine:
  - `target-host` is the target machine to which to migrate.
  - `user-name` is the login user name for the target machine, which must be SASL 64-bit encoded.
  - `password` is the password to use for logging into the target machine, which must be SASL 64-bit encoded.

Note – The Logical Domains Manager uses `sasl_decode64()` to decode the target user name and password and uses `sasl_encode64()` to encode these values. SASL 64 encoding is equivalent to `base64` encoding.

EXAMPLE 23–24 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>
        <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>
```

EXAMPLE 23–24 Example `migrate-domain` <data> Section *(Continued)*

```
</Envelope>
```

XML Schemas

The XML schemas that are used by the Logical Domains Manager are located in the `/opt/SUNWldm/bin/schemas` directory. The file names are as follows:

- `cim-common.xsd` – `cim-common.xsd` schema
- `cim-rasd.xsd` – `cim-rasd.xsd` schema
- `cim-vssd.xsd` – `cim-vssd.xsd` schema
- `cli-list-constraint-v3.xsd` – `cli-list-constraint-v3.xsd` schema
- `combined-v3.xsd` – `LDM_interface` XML schema
- `event-v3.xsd` – `LDM_Event` XML schema
- `ldmd-binding.xsd` – `Binding_Type` XML schema
- `ldmd-property.xsd` – `GenericProperty` XML schema
- `ovf-core.xsd` – `ovf-core.xsd` schema
- `ovf-envelope.xsd` – `ovf-envelope.xsd` schema
- `ovf-section.xsd` – `ovf-section.xsd` schema
- `ovf-strings.xsd` – `ovf-strings.xsd` schema
- `ovfenv-core.xsd` – `ovfenv-core.xsd` schema
- `ovfenv-section.xsd` – `ovfenv-section.xsd` schema

Glossary

This glossary defines terminology, abbreviations, and acronyms in the Oracle VM Server for SPARC documentation.

A

API Application programming interface.

ASN Abstract Syntax Notation.

`auditreduce` Command to merge and select audit records from audit trail files (see the `auditreduce`(1M) man page).

auditing Tracking changes to the system and identifying the user who made the changes.

authorization A way in which to determine who has permission to perform tasks and access data by using Oracle Solaris OS rights.

B

`bge` Broadcom Gigabit Ethernet driver on Broadcom BCM57xx devices.

BSM Basic Security Module.

`bsmconv` Command to enable the BSM (see the `bsmconv`(1M) man page).

`bsmunconv` Command to disable the BSM (see the `bsmunconv`(1M) man page).

C

CMT Chip multithreading.

compliance Determining whether a system's configuration is in compliance with a predefined security profile.

configuration Name of the logical domain configuration that is saved on the service processor.

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 A privileged domain that creates and manages other logical domains and services by using the Logical Domains Manager.

CWQ Control Word Queue; cryptographic unit.

D

DHCP Dynamic Host Configuration Protocol.

DIO Direct I/O.

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).

domain See logical domain.

DPS Data plane software.

DR Dynamic reconfiguration.

drd Oracle Solaris OS dynamic reconfiguration daemon for Logical Domains Manager (see the `drd`(1M) man page).

DRM Dynamic resource management.

DS Domain Services module.

DVD Digital versatile disc.

E

EFI Extensible firmware interface.

ETM Encoding Table Management module.

F

FC_AL Fiber Channel Arbitrated Loop.

FMA Fault Management Architecture.

fmd Oracle Solaris OS fault manager daemon (see the `fmd`(1M) man page).

format Disk partitioning and maintenance utility (see the `format`(1M) man page).

fmthard Command to populate label on hard disks (see the `fmthard`(1M) man page).

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.

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, a dedicated system of hardware and supporting software that enables you to manage your server independently of the operating system.

I/O Input/output devices, such as internal disks and PCIe controllers and their attached adapters and devices.

ioctl Input/output control call.

IPMP Internet Protocol Network Multipathing.

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. Also called a *domain*.

Logical Domains Manager A CLI to create and manage logical domains and allocate resources to domains.

M

MAC Media access control address, which Logical Domains Manager 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 Command to create and delete replicas of the Solaris Volume Manager metadevice state database (see the `metadb`(1M) man page).

metaset Command to configure disk sets (see the `metaset`(1M) man page).

mhd Command to perform 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

ndpsldcc Netra DPS Logical Domain Channel Client. *See also* `vdpcc`.

ndpsldcs Netra DPS Logical Domain Channel Service. *See also* `vdpcs`.

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

OID Object identifier, which is a sequence of numbers that uniquely identifies each object in a MIB.

OVF Open Virtualization Format.

P

P2V Oracle VM Server for SPARC 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.

physical domain The scope of resources that are managed by a single Oracle VM Server for SPARC instance. A physical domain might be a complete physical system as is the case of supported SPARC T-Series platforms. Or, it might be either the entire system or a subset of the system as is the case of supported SPARC M-Series platforms.

physical function A PCI function that supports the SR-IOV capabilities as defined in the SR-IOV specification. A physical function contains the SR-IOV capability structure and is used to manage the SR-IOV functionality. Physical functions are fully featured PCIe functions that can be discovered, managed, and manipulated like any other PCIe device. Physical functions have full configuration resources, and can be used to configure or control the PCIe device.

`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` Command to 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, which enables you to combine independent disks into a logical unit.

RPC Remote Procedure Call.

S

SAN Storage Area Network.

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.

SCSA Sun Common SCSI Architecture.

SCSI HBA SCSI Host Bus Adapter.

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.

SMI Structure of Management Information, which defines and groups managed objects for use by a MIB.

SNMP Simple Network Management Protocol.

service processor (SP) The SP, also known as the system controller (SC), monitors and runs the physical machine.

SR-IOV Single root I/O virtualization.

SSH Secure Shell.

`ssh` Secure Shell command (see the `ssh`(1) man page).

`sshd` Secure Shell daemon (see the `sshd`(1M) man page).

SunVTS Sun Validation Test Suite.

`svcadm` Manipulates service instances (see the `svcadm`(1M) man page).

T

TLS Transport Layer Security.

U

UDP User Diagram Protocol.

unicast Network communication that takes place between a single sender and a single receiver.

`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.

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.

virtual function A PCI function that is associated with a physical function. A virtual function is a lightweight PCIe function that shares one or more physical resources with the physical function and with other virtual functions that are associated with the same physical function. Virtual functions are only permitted to have configuration resources for its own behavior.

VNIC Virtual network interface card, which is a virtual instantiation of a physical network device that can be created from the physical network device and assigned to a zone.

vldc Virtual logical domain channel service.

vldcc Virtual logical domain channel client.

vnet Virtual network device implements a virtual Ethernet device and communicates with other `vnet` devices in the system by using the virtual network switch (`vswitch`).

vNTS Virtual network terminal service.

vntsd Oracle Solaris OS virtual network terminal server daemon for domain 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.

X

XFP eXtreme Fast Path.

XML Extensible Markup Language.

XMPP Extensible Messaging and Presence Protocol.

Z

ZFS Zettabyte File System.

zpool ZFS storage pool (see the `zpool`(1M) man page).

ZVOL ZFS Volume Emulation Driver.

Index

C

D

E

F

G

H

I

M

N

O

P

Q

R

S

T

U

V

W

X

Z