Oracle VM Server for SPARC 3.6 Administration Guide

E93617-02
January 2023

Oracle VM Server for SPARC 3.6 Administration Guide,

E93617-02

Primary Author: Cathleen Reiher

Contents

6 Configuring I/O Domains

7 Creating a Root Domain by Assigning PCIe Buses

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

12 Using Virtual SCSI Host Bus Adapters

13 Using Virtual Networks

14 Migrating Domains

15 Managing Resources

16 Managing SP Configurations

17 Handling Hardware Errors

18 Performing Other Administration Tasks

Using This Documentation

- **Overview** – Provides Oracle Solaris OS system administrators with detailed information and procedures that describe the installation, configuration, and use of the Oracle VM Server for SPARC 3.6 software.
- **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.

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, SPARC M-series, SPARC S-series, Fujitsu SPARC M12 servers, and Fujitsu M10 servers. Using the Oracle VM Server for SPARC software, you can create up to 128 virtual servers, called logical domains, on a single system. This kind of configuration enables you to take advantage of the massive thread scale offered by SPARC T-series, SPARC M-series, SPARC S-series servers, Fujitsu SPARC M12 servers, and Fujitsu M10 servers and the Oracle Solaris OS.

This chapter covers the following topics:

- About Oracle VM Server for SPARC and Oracle Solaris OS Versions
- Hypervisor and Logical Domains
- Logical Domains Manager
- Oracle VM Server for SPARC Management Information Base
- Oracle VM Server for SPARC Troubleshooting

> **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 3.6 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 3.6 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).

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 Oracle Solaris OS Versions in *Oracle VM Server for SPARC 3.6 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:**
>
> The Oracle Solaris 10 OS is no longer supported in the `primary` domain. You can continue to run the Oracle Solaris 10 OS in guest domains. The minimum version of the Oracle Solaris 10 OS requires patch ID 150400-64. You must purchase Oracle Solaris 10 Extended Support to obtain patch ID 150400-64.

Hypervisor and Logical Domains

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

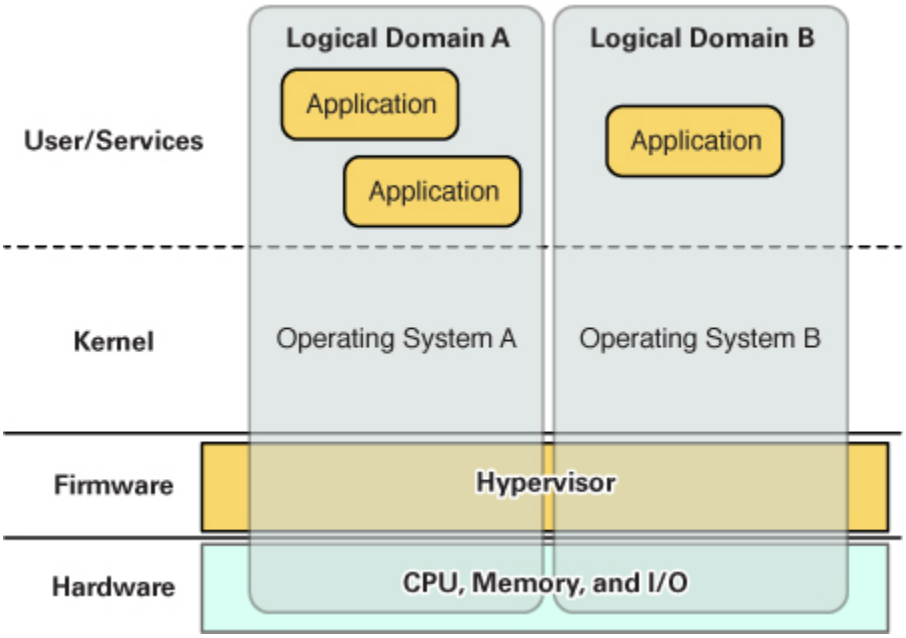
The SPARC *hypervisor* is a small firmware layer that provides a stable virtualized machine architecture to which an operating system can be written. SPARC servers that use the hypervisor provide hardware features to support the hypervisor's control over a logical operating system's activities.

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

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.
  - A PCIe SR-IOV virtual function. See 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`(8) 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 configure up to 128 domains on a system (up to 256 on a Fujitsu SPARC M12 server and 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 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. You can use the `ldmconsole` command to connect to consoles and to list the available consoles. See the Oracle Solaris OS `ldmconsole`(8) man page.

Resource Configuration

A system that runs the Oracle VM Server for SPARC software can configure resources such as virtual CPUs, virtual I/O devices, 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.

Persistent Configurations

You can use the `ldm` command to store the current configuration of a logical domain on the service processor. You can add an SP configuration, specify an SP configuration to be used, remove an SP configuration, and list the SP configurations. For details, see the `ldm`(8) man page. You can also specify an SP configuration to boot, as described in Using Oracle VM Server for SPARC With the Service Processor.

For information about managing SP configurations, see Managing SP Configurations.

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 *Oracle VM Server for SPARC 3.6 Management Information Base User's Guide*.

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 3.6 Release Notes*

- Information Center: Overview of Oracle VM Server for SPARC (LDoms) (Doc ID 1589473.2) (https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=227880986952919&id=1589473.2&_afrWindowMode=0&_adf.ctrl-state=wu098o5r6_96)

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:

- Delegating the Management of Logical Domains by Using Rights
- Using Verified Boot

> **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 the following 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.
- The `LDoms Consoles` profile permits a user to connect to all domain consoles.

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 11.4*.

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. 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 11.4*.

1. **Become an administrator.**

   For Oracle Solaris 11.4, see Chapter 1, About Using Rights to Control Users and Processes in *Securing Users and Processes in Oracle Solaris 11.4*.

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 11.4, see Chapter 1, About Using Rights to Control Users and Processes in *Securing Users and Processes in Oracle Solaris 11.4*.

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

Idm 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.

Using Verified Boot

The Logical Domains Manager uses the Oracle Solaris OS verified boot technology to verify the digital signature of kernel modules at boot time. Signature verification occurs silently unless the verified boot policies are enabled. Depending on the `boot-policy` value, a guest domain might not boot if the kernel module is not signed with Oracle Solaris release certificate files or is corrupted.

Use the `ldm add-domain` or `ldm set-domain` command to specify the values for the `boot-policy` property. See the `ldm`(8) man page.

To use this feature, your system must run at least the following versions of the system firmware and operating system:

- **System firmware –** Version 9.5.0 for Oracle SPARC servers except as follows:
  - Any released version for SPARC S7, SPARC T8, and SPARC M8 series servers
  - Any released version for Fujitsu SPARC M12 servers
  - XCP 2280 for Fujitsu M10 servers
- **Operating system –** Oracle Solaris 11.2 OS

> **Note:**
>
> By default, any domain created by using a version of Oracle VM Server for SPARC earlier than 3.4 sets `boot-policy=warning`. This setting results in warning messages being issued while the domain boots after an Oracle VM Server for SPARC update if the kernel module is unsigned or corrupted.

> **Note:**
>
> The `boot-policy` property of a guest domain is not preserved when when the guest is migrated to a system running an older version of Logical Domains Manager and migrated back to a system running Logical Domains Manager 3.4. Logical Domains Manager 3.4 introduced a new property named `boot-policy` for Verified Boot. Older versions of Logical Domains Manager do not know this property so the `boot-policy` property is dropped when a guest is migrated from a system running Logical Domains Manager 3.4 to a system running Logical Domains Manager older than 3.4. When the guest is migrated back to a system running Logical Domains Manager 3.4 the default `boot-policy` of `warning` will be applied to the incoming guest. You must manually set `boot-policy` to the desired value after migrating the guest back to a system running Logical Domains Manager 3.4 if the default value of `warning` is not appropriate.
>
> ```
> # ldm set-domain boot-policy=none ldg1
> ```
>
> Then reboot the guest to make the new boot policy take effect.

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:

> **Note:**
>
> Running the Oracle Solaris 10 OS in a service domain is no longer supported.

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

Before You Begin

If a network is not configured on your machine and a Network Information Services (NIS) client is running, the Logical Domains Manager will not start on your system. So, disable the NIS client on your non-networked machine:

```
# svcadm disable nis/client
```

Then, configure networking on the system:

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.

   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
   VCC
       NAME         LDOM         PORT-RANGE
   ```

```
    primary-vcc0 primary      5000-5255

VSW
    NAME         LDOM         MACADDRESS          NET-DEV   DVID|PVID|VIDs
    ----         ----         ----------          -------   --------------
    primary-vsw0 primary      00:14:4f:fb:87:30   net0      1|1|--

VDS
    NAME         LDOM    VOLUME   OPTIONS MPGROUP DEVICE
    primary-vds0 primary test-mig                /net/10.26.169.136/tmp/
                                                 Sol11.3/vdisk_s11sru3.img1
```

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. **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
   ```

3. **Save the SP configuration to the service processor (SP).**

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

   ```
   primary# ldm add-spconfig initial
   ```

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

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

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

5. **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. 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 11.4*.

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.

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 SP configuration saved to the service processor (SP), which is then reflected in the `ldm list-spconfig` output.

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`(8) 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 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 servers starting with the SPARC T7, SPARC M7, and SPARC S7 series server. Do not disable the ILOM interconnect on these servers. For more information, see the `ilomconfig`(8) man page.

> **Note:**
>
> Avoid disabling the ILOM interconnect on other SPARC T-series and M-series servers. However, if you do so, the `ldmd` daemon can still communicate with the SP.

On servers starting with the SPARC T7, SPARC M7, and SPARC S7 series server, an attempt to use the `ldm` command to manage SP configurations might fail. If the failure is 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 and How to Re-Enable the ILOM Interconnect Service.

> **Note:**
>
> If the ILOM interconnect is down, the `ldm list-spconfig` command fails as follows:
>
> ```
> primary# ldm list-spconfig
> The requested operation could not be performed because the
> communication
> channel between the LDoms Manager and the system controller is down.
> The ILOM interconnect may be disabled or down.
> ```
>
> The ILOM interconnect might go down if you add a resource that provides a communication channel between the Logical Domains Manager and the SP to the system but do not manually add the resource to the `primary` domain. By adding the resource to the `primary` domain, the communication channel is established.

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 ILOM interconnect 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 ILOM interconnect 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
   ```

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
- Installing the Oracle Solaris OS on a Guest Domain

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 3.6 features. See *Oracle VM Server for SPARC 3.6 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 or has a physical CPU ID greater than or equal to 1024 cannot run the Oracle Solaris 10 OS. You cannot use CPU DR to reduce the number of CPUs or CPU IDs below 1024 to run the Oracle Solaris 10 OS.

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 and Using ZFS With Virtual Disks.

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

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

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

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

     Where:

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

   - **File Example.** This example exports a file as a block device.

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

     Where:

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

6. **Add a virtual disk to the guest domain.**

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

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

   Where:

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

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

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

   > **Note:**
   >
   > When setting the `boot-device` property value, only use lowercase characters even if the name of the virtual disk contains uppercase characters.

   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.**

The value in the `CONS` column shows 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 the 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.

Memory Size Requirements

The Oracle VM Server for SPARC software allocates memory in multiples of 256 Mbytes aligned on 256-Mbyte address boundaries. Thus, the smallest domain that can be created by the Logical Domains Manager must have 256 Mbytes of memory. However, the actual 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*, *Oracle Solaris 11.3 Release Notes*, and *Oracle Solaris 11.4 Release Notes*.

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

1. **Insert the Oracle Solaris OS DVD into the DVD drive.**
2. **Stop the removable media service on the primary domain.**

```
primary# svcadm disable rmvolmgr
```

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-domain 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-domain 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.

  - **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
  ```

5
Using Domain Consoles

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

This chapter covers the following topics:

- Controlling Access to a Domain Console by Using Rights
- Using Domain Console Logging
- Connecting to a Guest Domain Console Over the Network
- Using Console Groups

> **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.

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. This authorization checking applies to accessing a console with either the `ldmconsole` or `telnet` command. 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 and How to Control Access to All Domain Consoles by Using Rights Profiles.
- To control access to a single domain console, see How to Control Access to a Single Console by Using Roles and How to Control Access to a Single Console by Using Rights Profiles.

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 5-1 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. **Assign the LDoms Consoles rights profile to a user.**

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

3. **Connect to the domain console as the user.**

   ```
   $ telnet localhost 5000
   ```

Example 5-2 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.

Assign the LDoms Consoles 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-name*. Use the `auths` command to add the authorization.

   ```
   # auths add -t "Access domain-name Console" solaris.vntsd.console-domain-name
   ```

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-name 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 5-3 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.

```
# auths add -t "Access ldg1 Console" solaris.vntsd.console-ldg1
```

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 command adds the `solaris.vntsd.console-`*`domain`*`:::Access `*`domain`*` Console::` authorization entry for a domain console:

   ```
   # auths add -t "Access domain-name Console" solaris.vntsd.console-domain-name
   ```

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
   ```

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`(8) and `logadm.conf`(5) 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.

Connecting to a Guest Domain Console Over the Network

You can use the `ldmconsole` command or the `telnet` command to connect to a Oracle VM Server for SPARC console on non-`primary` domains or to a console group. See the *man pages section 8: System Administration Commands* and the `vntsd`(8) man pages.

- The following `ldmconsole` command connects to the console of the `ldg1` domain:

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

- 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`(8) 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 (https://docs.oracle.com/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.

Also, use the `ldmconsole` command to list the available domain consoles and console groups.

The following command shows how to list the available console groups and domain consoles:

```
primary# ldmconsole
GROUP    DOMAINS
ldg1     ldg1
ldg2     ldg2
```

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`(8) man page for more information about configuring and using SMF to manage `vntsd` and using console groups.

6
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
- General Guidelines for Creating an I/O Domain

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:

- Creating a Root Domain by Assigning PCIe Buses
- Creating an I/O Domain by Using Direct I/O
- Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions
- 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 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.

7

Creating a Root Domain by Assigning PCIe Buses

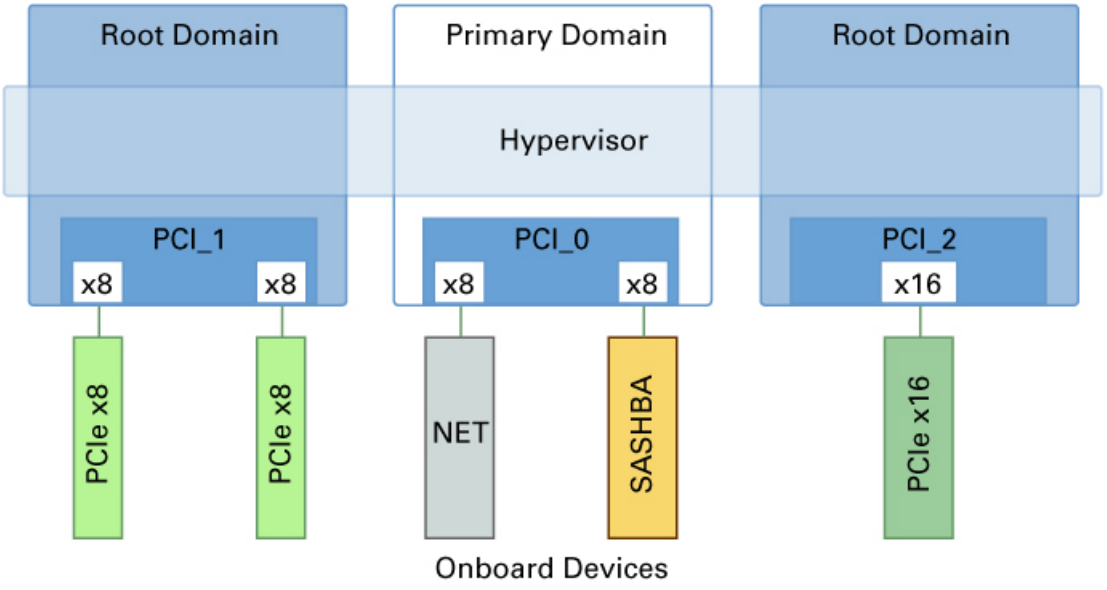
This chapter describes how to create a root domain by assigning PCIe buses.

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`.

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` SP 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 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. 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 servers starting with the SPARC M5, SPARC T5, and SPARC S7 series server and the Fujitsu M10 server that run the Oracle Solaris 11 OS in the root domain.

SPARC T5, SPARC M5 and SPARC M6 servers must run at least the 9.4.2 version of the system firmware. SPARC T7 and SPARC M7 series servers must run at least

9.4.3. SPARC S7, SPARC T8, and SPARC M8 series servers can run any released version of the system firmware. Fujitsu SPARC M12 servers can run any released version of the system firmware. Fujitsu M10 servers 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.

Note:

Do not add independent root domains to the following system types:

- **Single-bus systems.** Such systems, like a SPARC T4-1 server, can use only the `primary` domain as a root domain.
- **Some smaller multi-bus systems.** Systems such as SPARC S7-2 and SPARC S7-2L servers have built-in cards that communicate only with a single bus and cannot be split across multiple buses. To create a second root domain, you must install additional cards to ensure that each root domain has a network interface and a boot disk.

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
----                                      ----   ---      ------   ------
/SYS/PM0/CMP0/PEX                         BUS    pci_0    primary  IOV
/SYS/PM0/CMP1/PEX                         BUS    pci_1
/SYS/PM0/CMP2/PEX                         BUS    pci_2    primary
/SYS/PM0/CMP3/PEX                         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 to which the boot disk is connected by using the `mpathadm` command.

     Starting with the SPARC T4 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.

  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 SP configuration to the service processor.**

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

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

This SP configuration, `io-domain`, is also set as the next SP 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
----                                      ----   ---      ------   ------
/SYS/PM0/CMP0/PEX                         BUS    pci_0    primary  IOV
/SYS/PM0/CMP1/PEX                         BUS    pci_1    primary
/SYS/PM0/CMP2/PEX                         BUS    pci_2    ldg1
/SYS/PM0/CMP3/PEX                         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.

8

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.

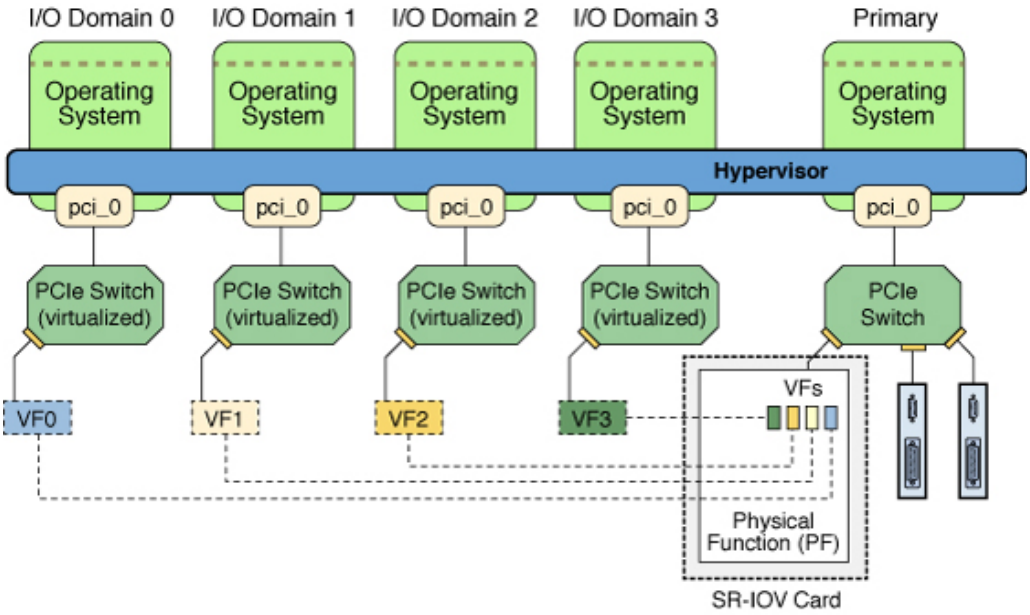
The Peripheral Component Interconnect Express (PCIe) single root I/O virtualization (SR-IOV) implementation is based on version 1.1 of the standard as defined by the PCI-SIG. The SR-IOV standard enables the efficient sharing of PCIe devices among virtual machines and is implemented in the hardware to 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.

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 and Dynamic SR-IOV.

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`(8) man page.

> **Note:**
>
> By default, PCIe buses are enabled for I/O virtualization on systems starting with the SPARC M7, SPARC T7, and SPARC S7 series server and the Fujitsu M10 server.

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.

SR-IOV Hardware and Software Requirements

The dynamic and static PCIe SR-IOV features are supported on servers starting with the SPARC T4, SPARC M5, and SPARC S7 series server and the Fujitsu M10 server.

- **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 servers starting with the SPARC T4, SPARC M5, and SPARC S7 series server and the Fujitsu M10 server.
- **Fibre Channel SR-IOV.** Fibre Channel devices are supported on servers starting with the SPARC T4, SPARC M5, and SPARC S7 series server and the Fujitsu M10 server.

  For an up-to-date list of supported devices on Fujitsu SPARC M12 platforms or Fujitsu M10 platforms. See *Fujitsu SPARC M12 Systems PCI Card Installation Guide* or *Fujitsu M10/SPARC M10 Systems PCI Card Installation Guide* 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, SPARC M5, and SPARC M6 servers must run at least version 9.1.0.a of the system firmware. SPARC T7 and SPARC M7 series servers must run at least version 9.4.3 of the system firmware. SPARC S7 series servers must run at least version 9.7.2 of the system firmware. SPARC T8 and SPARC M8 series servers can run at least version 9.8 of the system firmware. Fujitsu SPARC M12 servers can run any released version of the system firmware. Fujitsu M10 servers must run at least version XCP2210 of the system firmware.

    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 T8 Series Servers –** 9.8
  - **SPARC M5 and SPARC M6 Servers –** 9.1.0.*x*
  - **SPARC M7 Series Servers –** 9.4.3
  - **SPARC M8 Series Servers –** 9.8
  - **Fujitsu M10 Server –** XCP2210
  - **Fujitsu SPARC M12 Server –** XCP3021

  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`(8) 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 T8 Series Servers –** 9.8
  - **SPARC M5 Server –** 9.1.2.d
  - **SPARC M6 Server –** 9.1.2.d
  - **SPARC M7 Series Server –** 9.4.3
  - **SPARC M8 Series Servers –** 9.8
  - **SPARC S7 Series Server –** 9.7.2
  - **Fujitsu M10 Server –** XCP2210

* **Fujitsu SPARC M12 Server –** XCP3021

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.

> **Note:**
>
> If you plan to use the SR-IOV feature, you must update the firmware to meet the minimum required level.

- **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.

      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
- Dynamic SR-IOV Software Requirements

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

- Ethernet SR-IOV Hardware Requirements
- InfiniBand SR-IOV Hardware Requirements
- Fibre Channel SR-IOV Hardware Requirements

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 SR-IOV physical functions or SR-IOV 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, SPARC M-series, and SPARC S-series servers. On servers starting with the SPARC T7, SPARC M7, and SPARC S7 series server, 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 SPARC M12 server or 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 servers starting with the SPARC T7, SPARC M7, and SPARC S7 series server. For a description of the problems related to SR-IOV, see *Oracle VM Server for SPARC 3.6 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.

  If the root domain providing PCIe SR-IOV virtual functions to an I/O domain is rebooted while that I/O domain is running then unpredictable behavior can result in that I/O domain. For instance, I/O domains with PCIe endpoint devices assigned might panic during or after the reboot of the root domain if that I/O domain is not configured with I/O domain resiliency (IOR). To recover an I/O domain after such a panic, it must be manually stopped and started once the root domain has completed booting.

  However if the I/O domain is made resilient, it can continue to operate unhindered even if the root domain that is the owner of the PCIe bus for one set of SR-IOV virtual functions becomes unavailable. IOR allows the domain to be resilient to root domain downtime. See the I/O Domain Resiliency section for details.

  Also see I/O Domain Resiliency Limitations for best practices when using SR-IOV in an IOR 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.

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-domain 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

For information about SR-IOV hardware and software requirements, see https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&id=1325454.1.

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`(8) man page.

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

Dynamic SR-IOV

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

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

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

Dynamic SR-IOV Software Requirements

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

> **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.

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.

Destroying All Virtual Functions and Returning the Slots to the Root Domain Does Not Restore the Root Complex Resources

> **Note:**
>
> This section applies to servers up to and including SPARC M6 and SPARC T5 servers.

The resources on the root complex are not restored after you destroy all the virtual functions and return the slots to the root domain.

Recovery: Return all the virtual I/O resources that are associated with the root complex to their root domain.

First, put the control domain in delayed reconfiguration.

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

Return all child PCIe slots to the root domain that owns the `pci_0` bus. Then, remove all of the child virtual functions on the `pci_0` bus and destroy them.

Finally, set `iov=off` for the `pci_0` bus and reboot the root domain.

```
primary# ldm set-io iov=off pci_0
primary# shutdown -y -g 10
```

Workaround: Set the `iov` option to `off` for the specific PCIe bus.

```
primary# ldm start-reconf primary
primary# ldm set-io iov=off pci_0
```

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:**
>
> By default, PCIe buses are enabled for I/O virtualization on systems starting with the SPARC M7, SPARC T7, and SPARC S7 series server and the Fujitsu M10 server.

How to Enable I/O Virtualization for a PCIe Bus

This procedure must 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.

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.

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

   See How to Enable I/O Virtualization for a PCIe Bus.

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 [name=user-assigned-name] pf-name
   ```

   For more information, see How to Create an Ethernet SR-IOV Virtual Function, How to Create an InfiniBand Virtual Function, and How to Create a Fibre Channel SR-IOV Virtual Function.

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

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

Using Ethernet SR-IOV Virtual Functions

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

Caution:

When using some Intel network adapters that support SR-IOV, a virtual function might be the target of malicious behavior. Unexpected software-generated frames can throttle traffic between the host and the virtual switch, which might negatively affect performance. Configure all SR-IOV-enabled ports to use VLAN tagging to drop unexpected and potentially malicious frames.

- To configure VLAN tagging on a physical function and its associated virtual functions, use the following command:

  ```
  ldm create-vf [pvid=pvid] [vid=vid1,vid2,...>] net-pf-name
  ```

- To configure VLAN tagging on an existing virtual function, use the following command:

  ```
  ldm set-io [pvid=[pvid]] [vid=[vid1,vid2,...]] net-vf-name
  ```

For information about creating the VLAN interface in the I/O domain, see Using VLAN Tagging.

Ethernet SR-IOV Hardware Requirements

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

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.

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.

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.

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 [name=user-assigned-name] pf-name
       ```

       Use the `ldm create-vf -n max` command to create all the virtual functions for that physical function at one time. You can use the `name` property to optionally specify a name for the virtual function.

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

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

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

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

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

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

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

- Static method:

  a. Initiate a delayed reconfiguration.

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

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

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

  c. Reboot the root domain.

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

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

  – To reboot the `primary` root domain:

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

Example 8-1 Displaying Information About the Ethernet Physical Function

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

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

```
primary# ldm list-io
/SYS/PM0/CMP0/NIU_CORE                  NIU    niu_0    primary
/SYS/PM0/CMP1/NIU_CORE                  NIU    niu_1    primary
/SYS/PM0/CMP0/PEX                       BUS    pci_0    primary  IOV
/SYS/PM0/CMP1/PEX                       BUS    pci_1
/SYS/MB/PCIE0                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE2                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE4                           PCIE   pci_0    primary  OCC
/SYS/MB/PCIE6                           PCIE   pci_0    primary  EMP
/SYS/MB/PCIE8                           PCIE   pci_0    primary  EMP
/SYS/MB/SASHBA                          PCIE   pci_0    primary  OCC
/SYS/MB/NET0                            PCIE   pci_0    primary  OCC
/SYS/MB/PCIE1                           PCIE   pci_1    primary  OCC
/SYS/MB/PCIE3                           PCIE   pci_1    primary  OCC
/SYS/MB/PCIE5                           PCIE   pci_1    primary  OCC
/SYS/MB/PCIE7                           PCIE   pci_1    primary  EMP
/SYS/MB/PCIE9                           PCIE   pci_1    primary  EMP
/SYS/MB/NET2                            PCIE   pci_1    primary  OCC
/SYS/MB/NET0/IOVNET.PF0                 PF     pci_0    primary
/SYS/MB/NET0/IOVNET.PF1                 PF     pci_0    primary
/SYS/MB/PCIE5/IOVNET.PF0                PF     pci_1    primary
/SYS/MB/PCIE5/IOVNET.PF1                PF     pci_1    primary
/SYS/MB/NET2/IOVNET.PF0                 PF     pci_1    primary
/SYS/MB/NET2/IOVNET.PF1                 PF     pci_1    primary
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The following examples show ways in which you can create multiple SR-IOV Ethernet virtual functions:

- The following command shows how you can create 8 virtual functions from the `/SYS/MB/NET2/IOVNET.PF1` physical function and specify name of `new_vf` for the virtual function:

```
primary# ldm create-vf -n 8 name=new_vf /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.VF7
```

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.

The name you specify as the value of the `name` property is used as a base name for the generated virtual function names. The `ldm list-io` command shows the virtual function name you specified:

```
primary# ldm list-io /SYS/MB/NET2/IOVNET.PF1
NAME                                    TYPE   BUS      DOMAIN    STATUS
----                                    ----   ---      ------    ------
/SYS/MB/NET2/IOVNET.PF1                 PF     pci_1    ldg1
/SYS/MB/NET2/IOVNET.PF1.VF0             VF     pci_1
Assigned-Name: new_vf.0
/SYS/MB/NET2/IOVNET.PF1.VF1             VF     pci_1
Assigned-Name: new_vf.1
/SYS/MB/NET2/IOVNET.PF1.VF2             VF     pci_1
Assigned-Name: new_vf.2
...
/SYS/MB/NET2/IOVNET.PF1.VF7             VF     pci_1
Assigned-Name: new_vf.7
```

- The following command shows how you can create 8 virtual functions from the `/SYS/MB/NET2/IOVNET.PF1` physical function:

```
primary# ldm create-vf -n 8 /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.VF7
```

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.

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

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

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

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

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

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

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

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

     - To destroy a specified virtual function:

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

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

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

   - Static method:

     a. Initiate a delayed reconfiguration.

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

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

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

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

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

        - To destroy a specified virtual function:

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

     c. Reboot the root domain.

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

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

        - To reboot the `primary` root domain:

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

Example 8-7 Destroying an Ethernet Virtual Function

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

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

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

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

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

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

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

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

Modifying Ethernet SR-IOV Virtual Functions

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

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

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. For more information about device-specific properties, see Advanced SR-IOV Topics: Ethernet SR-IOV.

How to Modify Ethernet SR-IOV Virtual Function Properties

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

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

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

2. **Modify a virtual function property.**

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

Example 8-9 Modifying Ethernet Virtual Function Properties

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

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

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

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

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

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

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

> **Note:**
>
> Before you run this command, stop the domain that owns the virtual function.

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

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

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

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

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

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

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

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 domain must be in the inactive or bound state.

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

     b. Reboot the root domain.

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

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

– To reboot the `primary` root domain:

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

Example 8-10 Adding an Ethernet Virtual Function

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

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

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

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

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

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

> **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 domain must be in the inactive or bound state.

c. Start the I/O domain.

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

Example 8-11 Dynamically Removing an Ethernet Virtual Function

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

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

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

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

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

Advanced SR-IOV Topics: Ethernet SR-IOV

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

Advanced Network Configuration for Virtual Functions

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

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

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

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

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

SR-IOV Device-Specific Properties

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

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

P
Applies to a physical function

V
Applies to a virtual function

R
Read-only or informative parameter only

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

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

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 Virtual NICs on SR-IOV Virtual Functions

The creation of Oracle Solaris 11 virtual NICs (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-domain 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

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.

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

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 Creating Multiple SR-IOV Ethernet Virtual Functions and the `ldm`(8) 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.

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

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

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

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

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

This example assumes that the following circumstances are true:

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

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

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

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

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

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

```
primary# ldm list-io
NAME                           TYPE  BUS      DOMAIN    STATUS
----                           ----  ---      ------    ------
/SYS/PM0/CMP0/NIU_CORE         NIU   niu_0    primary
/SYS/PM0/CMP1/NIU_CORE         NIU   niu_1    primary
/SYS/PM0/CMP0/PEX              BUS   pci_0    primary   IOV
/SYS/PM0/CMP1/PEX              BUS   pci_1
/SYS/MB/PCIE0                  PCIE  pci_0    primary   OCC
/SYS/MB/PCIE2                  PCIE  pci_0    primary   OCC
/SYS/MB/PCIE4                  PCIE  pci_0    primary   OCC
/SYS/MB/PCIE6                  PCIE  pci_0    primary   EMP
/SYS/MB/PCIE8                  PCIE  pci_0    primary   EMP
/SYS/MB/SASHBA                 PCIE  pci_0    primary   OCC
/SYS/MB/NET0                   PCIE  pci_0    primary   OCC
/SYS/MB/PCIE1                  PCIE  pci_1    primary   OCC
/SYS/MB/PCIE3                  PCIE  pci_1    primary   OCC
/SYS/MB/PCIE5                  PCIE  pci_1    primary   OCC
/SYS/MB/PCIE7                  PCIE  pci_1    primary   EMP
/SYS/MB/PCIE9                  PCIE  pci_1    primary   EMP
/SYS/MB/NET2                   PCIE  pci_1    primary   OCC
/SYS/MB/NET0/IOVNET.PF0        PF    pci_0    primary
/SYS/MB/NET0/IOVNET.PF1        PF    pci_0    primary
/SYS/MB/PCIE5/IOVNET.PF0       PF    pci_1    primary
/SYS/MB/PCIE5/IOVNET.PF1       PF    pci_1    primary
/SYS/MB/NET2/IOVNET.PF0        PF    pci_1    primary
/SYS/MB/NET2/IOVNET.PF1        PF    pci_1    primary
/SYS/MB/NET0/IOVNET.PF0.VF0    VF    pci_0    ldg1
```

Example 8-13 Statically Creating an I/O Domain by Assigning an SR-IOV Virtual Function to It

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

This example assumes that the following circumstances are true:

- The OS on the `primary` domain does not support dynamic SR-IOV operations
- The `pci_0` bus is assigned to the `primary` domain and has not been initialized for I/O virtualization operations
- The `/SYS/MB/NET0/IOVNET.PF0` physical function belongs to the `pci_0` bus
- The `/SYS/MB/NET0/IOVNET.PF0` physical function does not have any existing virtual functions assigned to domains
- The `ldg1` domain is active and booted and its OS does not support dynamic SR-IOV operations
- The `ldg1` domain has the `auto-boot?` property set to `true` so that the domain boots automatically when the domain is started

First, initiate a delayed reconfiguration on the `primary` domain, enable I/O virtualization, and create the virtual function from the `/SYS/MB/NET0/IOVNET.PF0` physical function.

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

primary# ldm set-io iov=on pci_0
primary# ldm create-vf /SYS/MB/NET0/IOVNET.PF0

--------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
--------------------------------------------------------------------------------
Created new vf: /SYS/MB/NET0/IOVNET.PF0.VF0
```

Next, shut down the `primary` domain.

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

Stop the `ldg1` domain, add the virtual function, and start the domain.

```
primary# ldm stop ldg1
primary# ldm add-io /SYS/MB/NET0/IOVNET.PF0.VF0 ldg1
primary# ldm start-domain 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
----                          ----  ---      ------    ------
/SYS/PM0/CMP0/NIU_CORE        NIU   niu_0    primary
/SYS/PM0/CMP1/NIU_CORE        NIU   niu_1    primary
/SYS/PM0/CMP0/PEX             BUS   pci_0    primary   IOV
/SYS/PM0/CMP1/PEX             BUS   pci_1
/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

> **Note:**
>
> You can use only the static SR-IOV method 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.

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 Creating Multiple SR-IOV Ethernet Virtual Functions and the `ldm`(8) man page.

4. **Reboot the root domain.**

   Run one of the following commands:

   - Reboot the non-`primary` root domain.

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

   - Reboot the `primary` root domain.

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

Example 8-14 Creating an InfiniBand Virtual Function

The following example shows information about the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0` physical function:

- This physical function is in PCIE slot 4.
- The `IOVIB` string indicates that the physical function is an InfiniBand SR-IOV device.

```
primary# ldm list-io
NAME                                    TYPE   BUS      DOMAIN   STATUS
----                                    ----   ---      ------   ------
/SYS/PM0/CMP0/NIU_CORE                  NIU    niu_0    primary
/SYS/PM0/CMP0/PEX                       BUS    pci_0    primary  IOV
/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.

An attempt to use a dynamic SR-IOV remove operation on an InfiniBand device results in confusing and inappropriate error messages.

Dynamic SR-IOV remove operations are not supported for InfiniBand devices.

Remove InfiniBand virtual functions by performing one of the following procedures:

- How to Remove an InfiniBand Virtual Function From an I/O Domain
- How to Remove an InfiniBand Virtual Function From a Root Domain

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 Destroying Multiple Ethernet SR-IOV Virtual Functions and the `ldm`(8) man page.

3. **Reboot the root domain.**

   Run one of the following commands:

   - Reboot the non-`primary` root domain.

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

   - Reboot the `primary` root domain.

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

Example 8-15 Destroying an InfiniBand Virtual Function

The following example shows how to destroy a static InfiniBand virtual function, `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1`.

The `ldm list-io` command shows information about the buses, physical functions, and virtual functions.

```
primary# ldm list-io
NAME                                    TYPE   BUS      DOMAIN  STATUS
----                                    ----   ---      ------  ------
/SYS/PM0/CMP0/PEX                       BUS    pci_0    primary IOV
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0          PF     pci_0    primary
...
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0      VF     pci_0
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1      VF     pci_0
```

You can obtain more details about the physical function and related virtual functions by using the `ldm list-io -l` command.

```
primary# ldm list-io -l /SYS/MB/RISER1/PCIE4/IOVIB.PF0
NAME                                    TYPE   BUS      DOMAIN STATUS
----                                    ----   ---      ------ ------
/SYS/MB/RISER1/PCIE4/IOVIB.PF0          PF     pci_0    primary
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0]
    maxvfs = 64
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0      VF     pci_0
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,1]
/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1      VF     pci_0
[pci@400/pci@1/pci@0/pci@0/pciex15b3,673c@0,2]
```

A virtual function can be destroyed only if it is unassigned to a domain. The DOMAIN column of the `ldm list-io -l` output shows the name of any domain to which a virtual function is assigned. Also, virtual functions must be destroyed in the reverse order of their creation. Therefore, in this example, you must destroy the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1` virtual function before you can destroy the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0` virtual function.

After you identify the proper virtual function, you can destroy it. First, initiate a delayed reconfiguration.

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

primary# ldm destroy-vf /SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF1
------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
------------------------------------------------------------------------
```

You can issue more than one `ldm destroy-vf` command while in delayed reconfiguration. Thus, you could also destroy the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF0`.

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

```
primary# shutdown -i6 -g0 -y
Shutdown started.

Changing to init state 6 - please wait
...
```

Adding and Removing InfiniBand Virtual Functions on I/O Domains

How to Add an InfiniBand Virtual Function to an I/O Domain

This procedure describes how to add an InfiniBand SR-IOV virtual function to an I/O domain.

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

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

2. **Add one or more virtual functions to the I/O domain.**

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

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

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

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

Example 8-16 Adding an InfiniBand Virtual Function

The following example shows how to add the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2` virtual function to the `iodom1` I/O domain.

First, identify the virtual function that you want to assign.

```
primary# ldm list-io
NAME                                    TYPE   BUS      DOMAIN  STATUS
----                                    ----   ---      ------  ------
/SYS/PM0/CMP0/PEX                       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-domain iodom1
LDom iodom1 started
primary# ldm list-domain
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
primary          active     -n-cv-  UART    32    64G      1.0%  1.0%  1h 18m
iodom1           active     -n----  5000    8     8G        36%   36%  1m
```

How to Remove an InfiniBand Virtual Function From an I/O Domain

This procedure describes how to remove an InfiniBand SR-IOV virtual function from an I/O domain.

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

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

2. **Remove one or more virtual functions from the I/O domain.**

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

   > **Note:**
   >
   > Before removing the virtual function from the I/O domain, ensure that it is not critical for booting that domain.

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

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

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

Example 8-17 Removing an InfiniBand Virtual Function

The following example shows how to remove the `/SYS/MB/RISER1/PCIE4/IOVIB.PF0.VF2` virtual function from the `iodom1` I/O domain.

First, identify the virtual function that you want to remove.

```
primary# ldm list-io
NAME                                      TYPE   BUS      DOMAIN  STATUS
----                                      ----   ---      ------  ------
/SYS/PM0/CMP0/PEX                         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-domain 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
----                                    ----   ---      ------   ------
/SYS/PM0/CMP0/PEX                       BUS    pci_0    primary  IOV
/SYS/PM0/CMP0/NIU_CORE                  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 management includes zoning, LUN masking, and quality of service (QoS). You can configure the storage so that it is accessible exclusively to a specific logical domain without being visible to the physical function in the root domain.

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

Fibre Channel SR-IOV Hardware Requirements

For information about the required PCIe Fibre Channel SR-IOV hardware, see SR-IOV Hardware and Software Requirements.

- **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.
- To avoid a panic when switching a Fibre Channel SR-IOV configuration from Fibre Channel 16 (FC16) to Converged Network Adapter (CNA) mode, you must first remove all configurations from the SP.

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.

`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.

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.

> **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. Use this method to guarantee that the same WWNs are used when Fibre Channel virtual functions are destroyed and re-created. Automatically allocated WWNs are not saved for later recovery.

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.

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.

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 max [name=user-assigned-name] 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. You can use the `name` property to optionally specify a name for the virtual function.

> **Tip:**
>
> Create all virtual functions for the physical function at once. If you want to manually assign WWNs, first create all of the virtual functions and then use the `ldm set-io` command to manually assign your WWN values for each virtual function. This technique minimizes the number of state transitions when creating virtual functions from a physical function.

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

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

```
ldm create-vf [bw-percent=value] [port-wwn=value node-wwn=value] pf-name
```

You can also manually specify Fibre Channel class-specific property values.

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

- Static method:

  a. Initiate a delayed reconfiguration.

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

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

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

  c. Reboot the root domain.

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

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

  – To reboot the `primary` root domain:

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

Example 8-18 Displaying Information About the Fibre Channel Physical Function

This example shows information about the `/SYS/MB/PCIE7/IOVFC.PF0` physical function:

- This physical function is from a board in a PCIe slot, `PCIE7`.
- The `IOVFC` string indicates that the physical function is a Fibre Channel SR-IOV device.

```
primary# ldm list-io
NAME                                    TYPE   BUS      DOMAIN   STATUS
----                                    ----   ---      ------   ------
/SYS/PM0/CMP0/PEX                       BUS    pci_0    primary  IOV
/SYS/PM0/CMP1/PEX                       BUS    pci_1    rootdom1 IOV
/SYS/PM0/CMP0/NIU_CORE                  NIU    niu_0    primary
/SYS/PM0/CMP1/NIU_CORE                  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/IOVFC.PF0                 PF     pci_0    rootdom1
[pci@400/pci@1/pci@0/pci@6/SUNW,emlxs@0]
    maxvfs = 8
```

Example 8-19 Dynamically Creating a Fibre Channel Virtual Function Without Setting Optional Properties

This example dynamically creates a virtual function without setting any optional properties. In this case, the `ldm create-vf` command automatically allocates the default bandwidth percentage, port world-wide name (WWN), and node WWN values.

Ensure that I/O virtualization is enabled on the `pci_1` PCIe bus. See How to Enable I/O Virtualization for a PCIe Bus.

You can use the `ldm create-vf` command to create all the virtual functions from the `/SYS/MB/PCIE7/IOVFC.PF0` physical function.

```
primary# ldm create-vf -n max /SYS/MB/PCIE7/IOVFC.PF0
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF0
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF1
```

```
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF2
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF3
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF4
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF5
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF6
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF7
```

Example 8-20 Dynamically Creating a Fibre Channel Virtual Function and Setting Properties

This example dynamically creates a virtual function while setting the `bw-percent` property value to `25` and specifies the port and node WWNs.

```
primary# ldm create-vf port-wwn=10:00:00:14:4F:FC:00:01 \
node-wwn=20:00:00:14:4F:FC:00:01 bw-percent=25 /SYS/MB/PCIE7/IOVFC.PF0
```

Example 8-21 Statically Creating a Fibre Channel Virtual Function Without Setting Optional Properties

This example statically creates a virtual function without setting any optional properties. In this case, the `ldm create-vf` command automatically allocates the default bandwidth percentage, port world-wide name (WWN), and node WWN values.

First you initiate a delayed reconfiguration on the `rootdom1` domain. Then, enable I/O virtualization on the `pci_1` PCIe bus. Because the `pci_1` bus has already been assigned to the `rootdom1` root domain, use the `ldm set-io` command to enable I/O virtualization.

```
primary# ldm start-reconf rootdom1
Initiating a delayed reconfiguration operation on the rootdom1 domain.
All configuration changes for other domains are disabled until the rootdom1
domain reboots, at which time the new configuration for the rootdom1 domain
will also take effect.

primary# ldm set-io iov=on pci_1
```

Now, you can use the `ldm create-vf` command to create all the virtual functions from the `/SYS/MB/PCIE7/IOVFC.PF0` physical function.

```
primary# ldm create-vf -n max /SYS/MB/PCIE7/IOVFC.PF0

------------------------------------------------------------------------------
Notice: The rootdom1 domain is in the process of a delayed reconfiguration.
Any changes made to the rootdom1 domain will only take effect after it reboots.
------------------------------------------------------------------------------

Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF0
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF1
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF2
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF3
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF4
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF5
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF6
Created new vf: /SYS/MB/PCIE7/IOVFC.PF0.VF7
```

Finally, reboot the `rootdom1` root domain to make the changes take effect in one of the following ways:

- `rootdom1` is a non-`primary` root domain

  ```
  primary# ldm stop-domain -r rootdom1
  ```

- `rootdom1` is the `primary` domain

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

Destroying Fibre Channel SR-IOV Virtual Functions

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

How to Destroy a Fibre Channel SR-IOV Virtual Function

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

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

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

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

   - Dynamic method:

     – To destroy all of the virtual functions from a physical function at one time, use the following command:

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

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

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

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

     – To destroy a specified virtual function:

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

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

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

   - Static method:

     a. Initiate a delayed reconfiguration.

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

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

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

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

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

   – To destroy a specified virtual function:

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

c. Reboot the root domain.

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

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

   – To reboot the `primary` root domain:

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

Example 8-22 Dynamically Destroying Multiple Fibre Channel SR-IOV Virtual Functions

This example shows the results of destroying all the virtual functions from the `/SYS/MB/PCIE5/IOVFC.PF1` physical function. The `ldm list-io` output shows that the physical function has eight virtual functions. The `ldm destroy-vf -n max` command destroys all the virtual functions, and the final `ldm list-io` output shows that none of the virtual functions remain.

```
primary# ldm list-io
...
/SYS/MB/PCIE5/IOVFC.PF1                   PF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF0               VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF1               VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF2               VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF3               VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF4               VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF5               VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF6               VF     pci_1
/SYS/MB/PCIE5/IOVFC.PF1.VF7               VF     pci_1
primary# ldm destroy-vf -n max /SYS/MB/PCIE5/IOVFC.PF1
primary# ldm list-io
...
/SYS/MB/PCIE5/IOVFC.PF1                   PF     pci_1
```

Example 8-23 Destroying a Fibre Channel Virtual Function

This example shows how to statically destroy the virtual functions from the `/SYS/MB/PCIE7/IOVFC.PF0` physical function.

```
primary# ldm start-reconf rootdom1
Initiating a delayed reconfiguration operation on the rootdom1 domain.
All configuration changes for other domains are disabled until the rootdom1
domain reboots, at which time the new configuration for the rootdom1 domain
will also take effect.
```

```
primary# ldm destroy-vf -n max /SYS/MB/PCIE7/IOVFC.PF0
primary# ldm stop-domain -r rootdom1
```

Modifying Fibre Channel SR-IOV Virtual Functions

The `ldm set-io` command modifies the current configuration of a virtual function by changing the property values or by setting new properties.

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

You can use the `ldm set-io` command to modify the `bw-percent`, `port-wwn`, and `node-wwn` properties.

You can dynamically change only the `bw-percent` property while the virtual functions are assigned to a domain.

How to Modify Fibre Channel SR-IOV Virtual Function Properties

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

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

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

2. **Modify a virtual function property.**

   ```
   ldm set-io [bw-percent=value] [port-wwn=value node-wwn=value] pf-name
   ```

   Unlike the `bw-percent` property value, which you can dynamically change at any time, you can dynamically modify the `port-wwn` and `node-wwn` property values only when the virtual function is not assigned to a domain.

Example 8-24 Modifying Fibre Channel SR-IOV Virtual Function Properties

This example modifies the properties of the specified virtual function, `/SYS/MB/PCIE7/IOVFC.PF0.VF0`, to specify the bandwidth percentage and the port and node WWN values.

```
primary# ldm set-io port-wwn=10:00:00:14:4f:fc:f4:7c \
node-wwn=20:00:00:14:4f:fc:f4:7c bw-percent=25 /SYS/MB/PCIE7/IOVFC.PF0.VF0
```

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

How to Add a Fibre Channel SR-IOV Virtual Function to an I/O Domain

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

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 domain must be in the inactive or bound state.

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

  b. Restart the domain.

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

Example 8-25 Adding a Fibre Channel Virtual Function

This example shows how to dynamically add the `/SYS/MB/PCIE7/IOVFC.PF0.VF0` virtual function to the `ldg2` domain.

```
primary# ldm add-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
```

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

```
primary# ldm stop-domain ldg2
primary# ldm add-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
primary# ldm start-domain ldg2
```

How to Remove a Fibre Channel SR-IOV Virtual Function From an I/O Domain

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

> **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 domain must be in the inactive or bound state.

  c. Start the I/O domain.

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

Example 8-26 Dynamically Removing a Fibre Channel Virtual Function

This example shows how to dynamically remove the `/SYS/MB/PCIE7/IOVFC.PF0.VF0` virtual function from the `ldg2` domain.

```
primary# ldm remove-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
```

If the command succeeds, the virtual function is removed from the `ldg2` domain. When `ldg2` is restarted, the specified virtual function no longer appears in that domain.

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

```
primary# ldm stop-domain ldg2
primary# ldm remove-io /SYS/MB/PCIE7/IOVFC.PF0.VF0 ldg2
primary# ldm start-domain ldg2
```

Advanced SR-IOV Topics: Fibre Channel SR-IOV

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

Accessing a Fibre Channel Virtual Function in a Guest Domain

The `ldg2` console log shows the operations of the assigned Fibre Channel virtual function device. Use the `fcadm` command to view and access the Fibre Channel virtual function device.

```
ldg2# fcadm hba-port
HBA Port WWN: 100000144ffb8a99
       Port Mode: Initiator
       Port ID: 13d02
       OS Device Name: /dev/cfg/c3
       Manufacturer: Emulex
       Model: 7101684
       Firmware Version: 7101684 1.1.60.1
       FCode/BIOS Version: Boot:1.1.60.1 Fcode:4.03a4
       Serial Number: 4925382+133400002R
```

```
        Driver Name: 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.

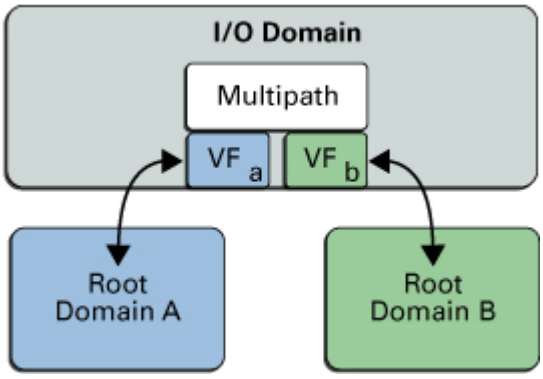
```
ldg2# format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
   0. c2d0 <Unknown-Unknown-0001-25.00GB>
      /virtual-devices@100/channel-devices@200/disk@0
   1. c3t21000024FF4C4BF8d0 <SUN-COMSTAR-1.0-10.00GB>
      /pci@340/pci@1/pci@0/pci@6/SUNW,emlxs@0,2/fp@0,0/ssd@w21000024ff4c4bf8,0
Specify disk (enter its number): ^D
ldg2#
```

I/O Domain Resiliency

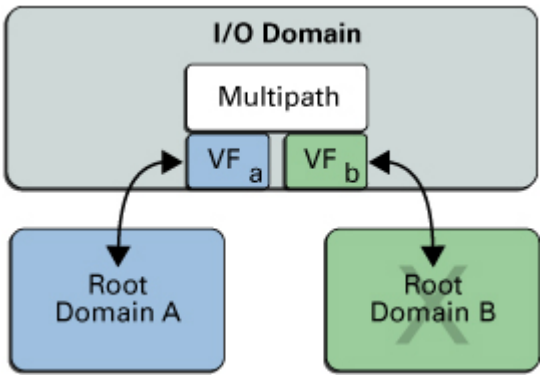
I/O domain resiliency improves the availability and performance of an I/O domain by enabling it to continue to run even when one of its associated root domains is interrupted. When a root domain is interrupted, the I/O domains that use its services continue to run by enabling its affected devices to 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.

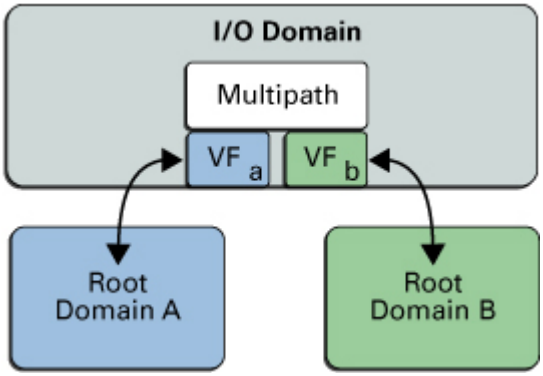
The first diagram shows a resilient I/O domain configuration that has two virtual functions, A and B. Root domain A and root domain B provide a virtual function to the I/O domain. The I/O domain uses virtual device multipathing such as IPMP for network devices and Oracle Solaris I/O multipathing for Fibre Channel devices.



The second diagram shows what occurs when the resilient I/O domain configuration loses its connection to root domain B. Such an interruption might occur when root domain B is interrupted by a panic or reboot. While root domain B is unavailable, virtual function B is suspended in the I/O domain and then multipathing engages to route all I/O through root domain A by using virtual function A.



The final diagram shows the resilient I/O domain configuration after root domain B returns to service. When root domain B is restored to service, virtual function B resumes operation in the I/O domain. The multipath group is restored to full redundancy.



In this configuration, the virtual function could be a 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.

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

- 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 the above cases, set the `failure-policy` from `ignore` to `reset` or `stop`.
- When you hotplug an SR-IOV card to the root domain and then assign virtual functions from it to an I/O domain, the I/O domain might fail to provide resiliency when the root domain fails. Therefore, it is best practice to add the SR-IOV card while the root domain is down. Then assign the virtual functions after the root domain boots.

Configuring Resilient I/O Domains

How to Configure a Resilient I/O Domain

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 are all running at least the Oracle Solaris 11.4 SRU 13 OS.

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.

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, IPMP, and IP Tunnels in Oracle Solaris 11.4*.

- **Fibre Channel.** Use Oracle Solaris I/O multipathing to configure multipathing across the paths.

  For information about using Oracle Solaris I/O multipathing to configure multipathing, see *Managing SAN Devices and I/O Multipathing in Oracle Solaris 11.4*.

Example 8-27 Using IPMP to Configure Multipathing With Ethernet SR-IOV Functions

This example shows how to use IPMP to configure network virtual-function devices for a resilient I/O domain. For more information, see *Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.4*.

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       1000     full      vnet0
net1         Ethernet     up       1000     full      igbvf0
net2         Ethernet     up       1000     full      igbvf1
```

   b. Create IP interfaces for the newly added network devices.

```
io-1# ipadm create-ip net1
io-1# ipadm create-ip net2
```

   c. Create the `ipmp0` IPMP group for the two network interfaces.

```
io-1# ipadm create-ipmp -i net1 -i net2 ipmp0
```

   d. Assign an IP address to the IPMP group.

   This example configures the DHCP option.

```
io-1# ipadm create-addr -T dhcp ipmp0/v4
```

e. Check the status of the IPMP group interface.

```
io-1# ipmpstat -g
```

Example 8-28 Using Oracle Solaris I/O Multipathing to Configure Multipathing With Fibre Channel SR-IOV Functions

This example shows how to use Oracle Solaris I/O multipathing to configure Fibre Channel virtual-function devices for a resilient I/O domain. For more information, see *Managing SAN Devices and I/O Multipathing in Oracle Solaris 11.4*.

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.

```
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 Oracle Solaris I/O multipathing 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, Oracle Solaris I/O multipathing 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 Oracle Solaris I/O multipathing 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
```

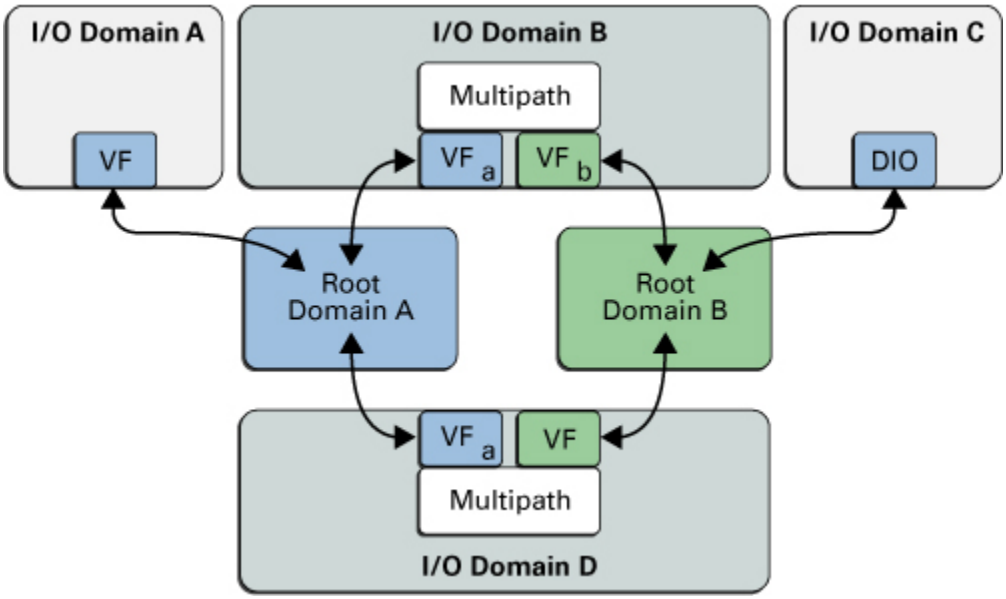
```
Current Load Balance:  round-robin
Logical Unit Group ID:  NA
Auto Failback:  on
Auto Probing:  NA

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.

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.

Replacing PCIe Hardware on a System With an IOR Configuration

This section outlines a streamlined procedure for replacing a PCIe device in a running system without rebooting any of the domains. This method will accomplish the same end result as performing a manual replacement but is less prone to errors and allows the existing SR-IOV and IOR configuration to be automatically recreated after the insertion of the new PCIe device. This method is especially useful for systems with large or complex configurations such as those often employed for IOR.

The commands used for the configuration save and device poweroff and subsequent restoral are `ldm evactuate-io` and `ldm restore-io` respectively.

System Requirements

Ensure that the I/O domain, root domain, service domain, and primary domain run at least the Oracle Solaris 11.4 SRU 13 OS. This also ensures that Oracle VM Server for SPARC version 3.6.1 or later is running, which is also a requirement.

The system hardware must be capable of using SR-IOV enabled PCIe cards in an IOR configuration. See Resilient I/O Domain Requirements.

Limitations

- This feature is only relevant to Dynamic I/O using SR-IOV, see the section Dynamic PCIe Bus Assignment Requirements.
- The system must fully support SR-IOV and IOR in both hardware and software versions as outlined above.
- Only an equivalent PCIe card may be used as the replacement device. This means it must be the same manufacturer and model supporting the same number of SR-IOV PFs and VFs.
- For Fujitsu M10 servers or Fujitsu SPARC M12 servers, Oracle Solaris 11.4 SRU 24 OS or later is required for the I/O domain, root domain, service domain, and primary domain.

Example 8-29 Example Faulty PCIe Card Replacement Procedure

In this example the PCIe device with path `/SYS/IOU1/PCIE13` will be replaced in a Non Primary Root Domain (NPRD) which is the owner of that PCIe slot. In effect the target slot and all it's children (PFs and VFs) are removed and restored during this procedure.

The target for the commands is the SR-IOV device itself, as represented in NAC name format. Thus, you can take the output of an `ldm ls-io` command and directly copy and paste it into an `ldm evacuate-io` or `restore-io` command. NAC name format is the standard for `ldm` commands and ILOM utilities.

The target device must be considered by the hotplug daemon to be a "connector". A connector is a device that is listed in the output of a `hotplug list -c` command run in the

root domain which owns the target PCIe device (be it the primary or an NPRD). See `hotplug`(8), and for information about Oracle Solaris OS hotplug capabilities, see Chapter 2, Dynamically Configuring Devices in Managing Devices in Oracle Solaris 11.4.

Steps

1. Identify the card to remove by reviewing fault logs on the primary.

   In this example the target is `/SYS/IOU1/PCIE13`.

   ```
   primary# fmadm faulty
   ```

2. Review and save a copy of the current I/O configuration on the machine (not strictly required, done to allow manual verification of the restored configuration).

   ```
   primary# ldm ls-io > io_config.txt
   ```

3. Perform evacuation command to automatically save current configuration, remove and destroy the VF children, and power down the device.

   ```
   nprd# ldm evacuate-io /SYS/IOU1/PCIE13
   ```

4. Wait for the `ldm` command to complete and the power LED on the target SR-IOV card/slot to be unlit.

5. Physically remove the device and replace it in the same slot with an equivalent card.

6. Restore the previous configuration by running the following command.

   ```
   nprd# ldm restore-io /SYS/IOU1/PCIE13
   ```

7. Wait for the `ldm` command to complete and the power LED on the target SR-IOV card/slot to be lit.

8. Check that the configuration matches the previously saved configuration (not strictly required).

   ```
   primary# ldm ls-io
   ```

   Note that if any portion of either of the above `ldm` commands fails, the remaining steps are not attempted. In the case of a command failure, no attempt is made to undo the effect of the completed actions. The error message printed to the console should indicate the cause of the failure. If the command is run again, an attempt will be made to complete all unfinished work.

   For background details see Making PCIe Hardware Changes. For general guidelines on hardware changes and manual instructions for PCIe device replacement in systems configured with Oracle VM Server, 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).

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.

As with PCIe slots in the I/O domain, the concerns that are described in Rebooting the Root Domain With PCIe Endpoints Configured 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.

9

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
- Direct I/O Hardware and Software Requirements
- Current Direct I/O Feature Limitations
- Planning PCIe Endpoint Device Configuration
- Rebooting the Root Domain With PCIe Endpoints Configured
- Making PCIe Hardware Changes
- Creating an I/O Domain by Assigning a PCIe Endpoint Device

> **Note:**
>
> The direct I/O feature is not supported on servers starting with the SPARC M7, SPARC T7, and SPARC S7 series server. Instead, use the PCIe bus assignment feature. See Creating a Root Domain by Assigning PCIe Buses.

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

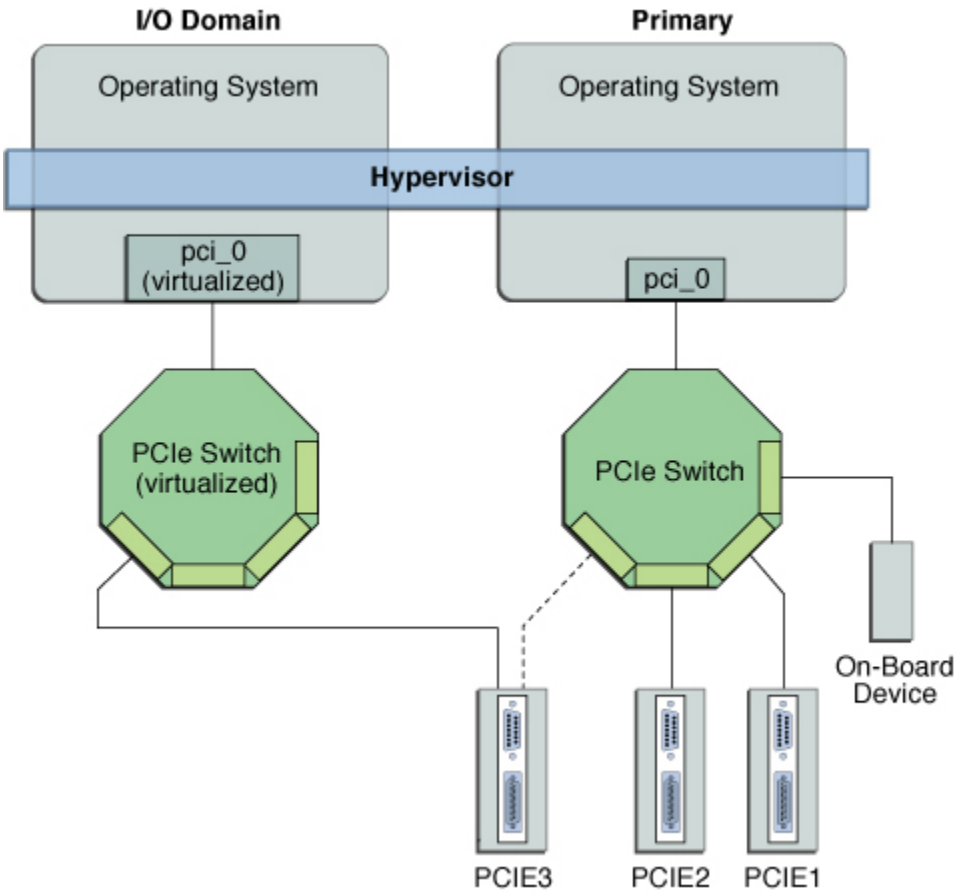
> **Note:**
>
> Because root domains cannot have dependencies on other root domains, a root domain that owns a PCIe bus cannot have its PCIe endpoint devices or SR-IOV virtual functions assigned to another root domain. However, you *can* assign a PCIe endpoint device or virtual function from a PCIe bus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

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.

> **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.
- 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. 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:**
  >
  > Servers starting with the SPARC T7, SPARC M7, and SPARC S7 series server have an I/O controller that provides several PCIe buses and you can assign PCIe buses to different domains. For information, see 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 Oracle Solaris OS Versions in *Oracle VM Server for SPARC 3.6 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`(8) 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.

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 SPARC M12 server and Fujitsu M10 server support 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 SPARC M12 Systems System Operation and Administration Guide* or *Fujitsu M10/SPARC M10 Systems System Operation and Administration Guide* for your model at http://www.fujitsu.com/global/services/computing/server/sparc/downloads/manual/.

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

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.

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 SP configuration to the service processor (SP).

   Use the `ldm add-spconfig` 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.

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.

> **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`(8) 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 9-1 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-domain 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-domain ldg1
  ```

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

  ```
  primary# ldm start-domain 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 11.4*.

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 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 T4-1 system to a DIO domain. The SPARC T4-1 system includes 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 T4-2 system has 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.

> **Note:**
>
> For Oracle Solaris 11 releases prior to Oracle Solaris 11.4, 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
----                                     ----   ---      ------   ------
/SYS/MB/CMP0/PEX                         BUS    pci_0    primary
[pci@400]
/SYS/MB/CMP0/NIU_CORE                    NIU    niu_0    primary
```

```
[niu@480]

/SYS/MB/CMP1/PEX                          BUS    pci_1    primary
[pci@500]
/SYS/MB/CMP1/NIU_CORE                     NIU    niu_1    primary
[niu@580]

/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.

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.

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.

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 SP configuration to the service processor (SP).

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

```
primary# ldm add-spconfig 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
----                                      ----   ---      ------   ------
/SYS/PM0/CMP0/NIU_CORE                    NIU    niu_0    primary
/SYS/PM0/CMP1/NIU_CORE                    NIU    niu_1    primary
/SYS/PM0/CMP0/PEX                         BUS    pci_0    primary  IOV
/SYS/PM0/CMP1/PEX                         BUS    pci_1    rootdom1 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-domain 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
```

10
Using Non-primary Root Domains

This chapter covers the following non-primary root domain topics:

- Non-primary Root Domains Overview
- Non-primary Root Domain Requirements
- Non-primary Root Domain Limitations
- Non-primary Root Domain Examples

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 starting with the SPARC T4, SPARC M5, and SPARC S7 series server and the Fujitsu M10 server.

> **Note:**
>
> The Oracle Solaris 10 OS and the Oracle Solaris 11.1 OS do not support the Direct I/O functionality or the SR-IOV functionality.

- **Hardware Requirements.**

  In addition to the PCIe cards for the direct I/O and SR-IOV described in https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&doctype=REFERENCE&id=1325454.1, other PCIe cards can be used, but not for DIO and SR-IOV. To determine which cards you can use on your platform, see your platform's hardware documentation.

- **Firmware Requirements.**

  SPARC T4 platforms must run at least version 8.4.0.a of the system firmware.

SPARC T5, SPARC M5, and SPARC M6 servers must run at least version 9.1.0.*x* of the system firmware.

SPARC T7 and SPARC M7 series servers must run at least version 9.4.3 of the system firmware.

SPARC S7, SPARC T8, and SPARC M8 series servers can run any released version of the system firmware.

Fujitsu M10 servers must run at least version XCP2210 of the system firmware. Fujitsu SPARC M12 servers must run XCP3021 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.
- 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
----                                    ----   ---      ------   ------
/SYS/PM0/CMP0/NIU_CORE                  NIU    niu_0    primary
/SYS/PM0/CMP1/NIU_CORE                  NIU    niu_1    primary
/SYS/PM0/CMP0/PXE                       BUS    pci_0    primary
/SYS/PM0/CMP1/PXE                       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             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-domain` command starts the `rootdom1` domain.

```
primary# ldm add-io iov=on pci_1 rootdom1
primary# ldm start-domain 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
----                                    ----   ---      ------   ------
/SYS/PM0/CMP0/PXE                       BUS    pci_0    primary
/SYS/PM0/CMP1/PXE                       BUS    pci_1    primary
/SYS/PM0/CMP0/NIU_CORE                  NIU    niu_0    primary
/SYS/PM0/CMP1/NIU_CORE                  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
----                                      ----   ---      ------   ------
/SYS/PM0/CMP0/PXE                         BUS    pci_0    primary
/SYS/PM0/CMP1/PXE                         BUS    pci_1    primary
/SYS/PM0/CMP0/NIU_CORE                    NIU    niu_0    primary
/SYS/PM0/CMP1/NIU_CORE                    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-domain` command starts the `ldg2` domain.

```
primary# ldm add-io /SYS/MB/PCIE7 ldg2
primary# ldm start-domain 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
----                                      ----   ---      ------   ------
/SYS/PM0/CMP0/PXE                         BUS    pci_0    primary  IOV
/SYS/PM0/CMP1/PXE                         BUS    pci_1    ldg1     IOV
/SYS/PM0/CMP0/NIU_CORE                    NIU    niu_0    primary
/SYS/PM0/CMP1/NIU_CORE                    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-domain 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.

```
primary# ldm list-io
NAME                                      TYPE   BUS      DOMAIN   STATUS
----                                      ----   ---      ------   ------
/SYS/PM0/CMP0/PXE                         BUS    pci_0    primary  IOV
/SYS/PM0/CMP1/PXE                         BUS    pci_1    ldg1     IOV
/SYS/PM0/CMP0/NIU_CORE                    NIU    niu_0    primary
/SYS/PM0/CMP1/NIU_CORE                    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
```

11

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.

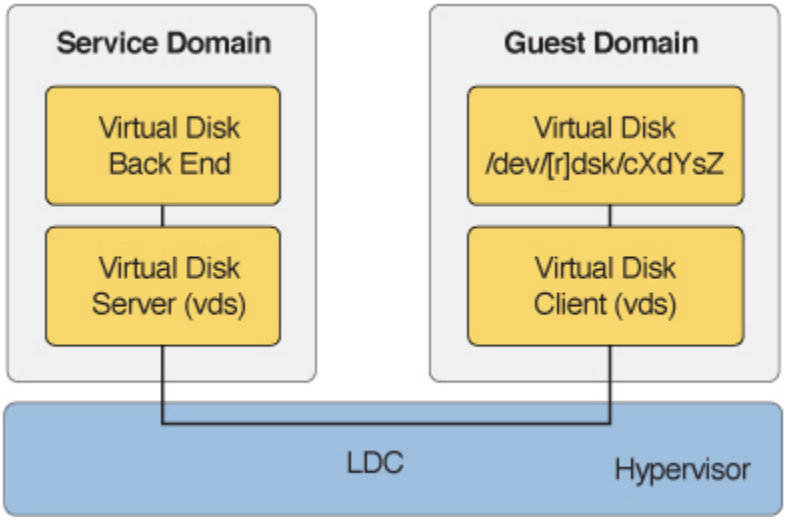
The virtual disk back end can be physical or logical. Physical devices can include the following:

- 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

Virtual Disks With Oracle VM Server for SPARC



To use the maximum number of virtual disks on the server, ensure that the `segkpsize` kernel tunable has a value of at least 524288. Note that an insufficient `segkpsize` value might result in a guest domain hanging during boot or during a dynamic addition of a virtual disk.

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 for a description of virtual disk options. See Virtual Disk Timeout 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.

How to Add a Virtual Disk

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

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

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

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

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

   > **Note:**
   >
   > A back end is actually exported from the service domain and assigned to the guest domain when the guest domain (*domain-name*) is bound.

How to Export a Virtual Disk Back End Multiple Times

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

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

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

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

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

2. **Assign the exported back end to each guest domain.**

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

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

How to Change Virtual Disk Options

For more information about virtual disk options, see Virtual Disk Back End Options.

- **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.

- **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 remove-vdisk disk-name domain-name
   ```

2. **Stop exporting the corresponding back end from the service domain.**

   ```
   primary# ldm remove-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.

> **Note:**
>
> Non-Volatile Memory Express (NVMe) storage is available starting with the SPARC T7, SPARC M7, and SPARC S7 series server. This storage can be a disk drive or a Flash Accelerator F160 PCIe card. This disk type can be used to build a virtual disk back end. Starting with the Oracle Solaris 11.3 SRU 2.4 OS, you can use the NVMe storage disk type as a full disk or as a single-slice disk. Prior to the Oracle Solaris 11.3 SRU 2.4 OS, you can use the NVMe storage disk type only as a single-slice disk.

> **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`(8) 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`(8) 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` 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 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 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 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.

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.

> **Note:**
>
> Non-Volatile Memory Express (NVMe) storage is available starting with the SPARC T7, SPARC M7, and SPARC S7 series server. This storage can be a disk drive or a Flash Accelerator F160 PCIe card. Starting with the Oracle Solaris 11.3 SRU 2.4 OS, you can use the NVMe storage disk type as a full disk or as a single-slice disk. Prior to the Oracle Solaris 11.3 SRU 2.4 OS, you can use the NVMe storage disk type only as a single-slice disk.

How to Export a Physical Disk Slice as a Virtual Disk

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

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

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

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

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

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

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

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

   ```
   ldg1# ls -1 /dev/dsk/c0d13s*
   /dev/dsk/c0d13s0
   /dev/dsk/c0d13s1
   /dev/dsk/c0d13s2
   /dev/dsk/c0d13s3
   /dev/dsk/c0d13s4
   /dev/dsk/c0d13s5
   /dev/dsk/c0d13s6
   /dev/dsk/c0d13s7
   ```

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

How to Export Slice 2

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

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

File and Volume Exporting

A file or volume (for example from ZFS or Solaris Volume Manager) is exported either as a full disk or as single-slice disk depending on whether the `slice` option is set.

File or Volume Exported as a Full Disk

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

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

> **Note:**
>
> You must run the `format` command in the guest domain to create partitions.

How to Export a File as a Full Disk

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

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

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

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

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

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

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

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

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

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

   A full disk is a regular disk with eight slices.

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

```
ldg1# ls -1 /dev/dsk/c0d5s*
/dev/dsk/c0d5s0
/dev/dsk/c0d5s1
/dev/dsk/c0d5s2
/dev/dsk/c0d5s3
/dev/dsk/c0d5s4
/dev/dsk/c0d5s5
/dev/dsk/c0d5s6
/dev/dsk/c0d5s7
```

How to Export a ZFS Volume as a Full Disk

1. **Create a ZFS volume to use as a full disk.**

   The following example shows how to create a ZFS volume, `zdisk0`, to use as a full disk:

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

   The size of the volume defines the size of the virtual disk. This example creates a 100-Mbyte volume to result in a 100-Mbyte virtual disk.

2. **From the control domain, export the corresponding device to that ZFS volume.**

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

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

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

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

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

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

   A full disk is a regular disk with eight slices.

   The following example shows how to list the disk, `c0d9`, and verify that it is accessible and is a full disk:

```
ldg1# ls -1 /dev/dsk/c0d9s*
/dev/dsk/c0d9s0
/dev/dsk/c0d9s1
/dev/dsk/c0d9s2
/dev/dsk/c0d9s3
/dev/dsk/c0d9s4
/dev/dsk/c0d9s5
/dev/dsk/c0d9s6
/dev/dsk/c0d9s7
```

File or Volume Exported as a Single-Slice Disk

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

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

How to Export a ZFS Volume as a Single-Slice Disk

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

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

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

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

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

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

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

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

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

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

   ```
   ldg1# ls -1 /dev/dsk/c0d9s*
   /dev/dsk/c0d9s0
   /dev/dsk/c0d9s1
   /dev/dsk/c0d9s2
   /dev/dsk/c0d9s3
   /dev/dsk/c0d9s4
   /dev/dsk/c0d9s5
   /dev/dsk/c0d9s6
   /dev/dsk/c0d9s7
   ```

Exporting Volumes and Backward Compatibility

If you have a configuration exporting volumes as virtual disks, volumes are now exported as full disks instead of single-slice disks. To preserve the old behavior and to have your volumes exported as single-slice disks, you need to do either of the following:

- Use the `ldm set-vdsdev` command in Oracle VM Server for SPARC 3.6 software, and set the `slice` option for all volumes you want to export as single-slice disks. See the `ldm`(8) 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.

> **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

Table 11-1 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.

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 multipathing 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.

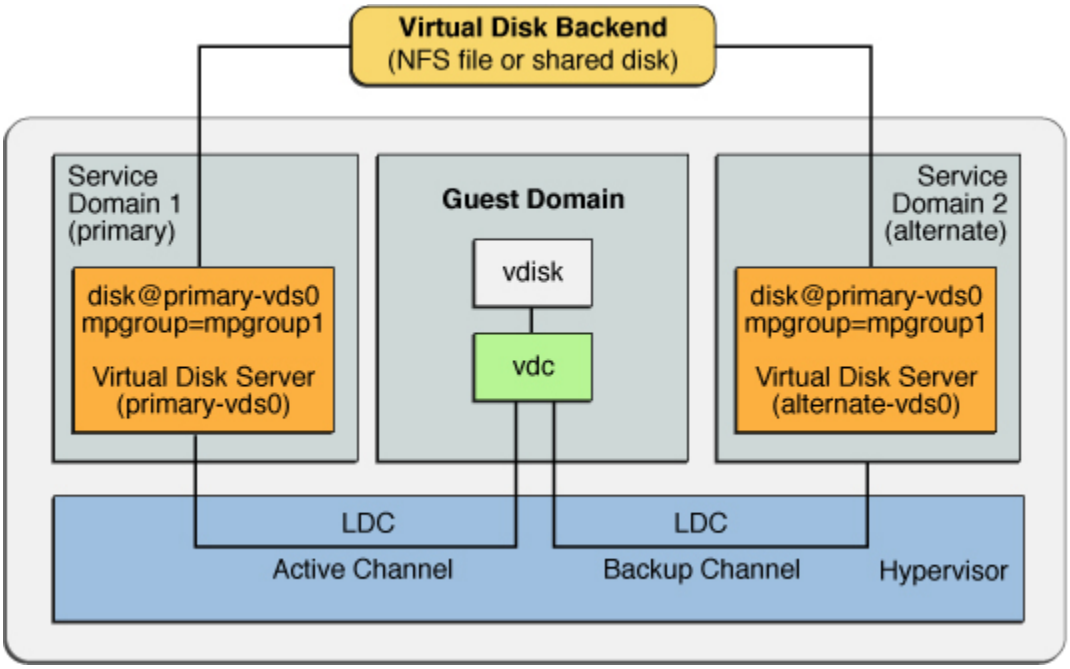
> **Note:**
>
> You cannot use mpgroups and SCSI reservation together.

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. 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`.

Configuring Virtual Disk Multipathing



Virtual Disk Multipathing and NFS

When setting up mpgroup failover with NFS file systems, ensure that you mount the NFS file system with the `soft` NFS mount option.

By using the `soft` option, I/O errors are reported to the VDS or VDC, and the VDC sends additional messages to determine whether this is an I/O error or if the entire back end is unavailable. Failover time is proportional to the NFS timeout and the possible retransmissions.

Do not change the `timeo` NFS mount option from the default value of 60 seconds (`timeo=600`). A short timeout value such as `timeo=40` means that you might encounter spurious I/O errors. For example, when an NFS server or network is unavailable for five seconds and then it returns to operation, I/O errors might be reported because the disruption was not long enough to cause a failover. A longer timeout, such as 60 seconds, masks disruptions that are several seconds long.

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.

How to Configure Virtual Disk Multipathing

See the figure titled Configuring Virtual Disk Multipathing.

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 11-1 Using an Mpgroup to Add a LUN to the Virtual Disk Service of Both Primary and Alternate Domains

The following shows how to create a LUN and add it to the virtual disk service for both primary and alternate domains by using the same mpgroup:

To determine which domain to use first when accessing the LUN, specify the associated path when adding the disk to the domain.

- Create the virtual disk devices:

  ```
  primary# ldm add-vdsdev mpgroup=ha lun1@primary-vds0
  primary# ldm add-vdsdev mpgroup=ha lun1@alternate-vds0
  ```

- To use the LUN from `primary-vds0` first, perform the following command:

  ```
  primary# ldm add-vdisk disk1 lun1@primary-vds0 gd0
  ```

- To use the LUN from `alternate-vds0` first, perform the following command:

  ```
  primary# ldm add-vdisk disk1 lun1@alternate-vds0 gd0
  ```

Result of Virtual Disk Multipathing

After you configure the virtual disk with multipathing and start the guest domain, the virtual disk accesses its back end through one of the service domains it has been associated with. If this service domain becomes unavailable, the virtual disk attempts to access its back end through another service domain that is part of the same multipathing group.

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

Dynamic Path Selection

You can dynamically select the path to be used for a virtual disk on guest domains and alternate domains that run at least the Oracle Solaris 11.2 SRU 1 OS. The control domain must run at least the Oracle Solaris 11.2 SRU 8 OS and at least the Oracle VM Server for SPARC 3.2 software.

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 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. **If the removable media service is running and online, perform the following steps:**
   - **Oracle Solaris 11 OS:** Determine whether the removable media service is running and online.

     ```
     service# svcs rmvolmgr
     service# svcs dbus
     service# svcs hal
     ```

   - **Oracle Solaris 10 OS:** Determine 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 removable media service is running and online, perform the following steps:**
   - **Oracle Solaris 11.3 OS:** Insert the CD or DVD in the CD or DVD drive.
   - **Oracle Solaris 10 OS, Oracle Solaris 11 OS, Oracle Solaris 11.1 OS, Oracle Solaris 11.2 OS:** If the volume management daemon is running and online, 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`(5) man page.
     b. Insert the CD or DVD in the CD or DVD drive.
     c. From the service domain, restart the volume management file system service.

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

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

```
service# cdrw -l
Looking for CD devices...
Node                    Connected Device                 Device type
----------------------+--------------------------------+-----------------
/dev/rdsk/c1t0d0s2    | MATSHITA CD-RW  CW-8124   DZ13 | CD Reader/Writer
```

4. **Export the CD or DVD disk device as a full disk.**

```
primary# ldm add-vdsdev /dev/dsk/c1t0d0s2 cdrom@primary-vds0
```

5. **Assign the exported CD or DVD to the guest domain.**

   The following command shows how to assign the exported CD or DVD to domain `ldg1`:

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

Exporting a CD or DVD Multiple Times

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

How to Export an ISO Image From the Control Domain to Install a Guest Domain

Before You Begin

This procedure assumes that both the `primary` domain and the guest domain are configured.

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

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

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

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

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

2. **Stop the guest domain.**

   In this example, the logical domain is `ldom1`.

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

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

   In this example, the logical domain is `ldom1`.

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

4. **Restart the guest domain.**

   In this example, the logical domain is `ldom1`.

```
primary# ldm start-domain ldom1
LDom ldom1 started
# ldm list
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active   -n-cv   SP      8     8G       0.4%  25m
ldom1            active   -t---   5000    4     1G       0.0%  0s
```

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

5. **Connect to the guest domain.**

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

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

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

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

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

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

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

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

Virtual Disk Timeout

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

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

Set this timeout by using one of the following methods:

- Using the `ldm add-vdisk` command.

```
ldm add-vdisk timeout=seconds
disk-name
volume-name@service-name
domain-name
```

- Using the `ldm set-vdisk` command.

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

- Adding the following line to the `/etc/system` file on the guest domain.

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

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

> **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 implemented by SCSI commands on 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`(8) subcommands. Virtual disks whose back ends are not SCSI disks do not support some `format`(8) subcommands, such as `repair` and `defect`. In that case, the behavior of `format`(8) 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 11-2 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 11-3 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.

```
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 11-4 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/c10t600C0FF000000000015153295A4B100d0s1  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.

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. The current system firmware versions of all supported servers support EFI labels.

The following servers cannot boot from a disk that has an EFI GPT disk label:

- SPARC T4 servers that run system firmware versions prior to 8.4.0
- SPARC T5, SPARC M5, and SPARC M6 servers that run system firmware versions prior to 9.1.0
- SPARC T7 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, SPARC T5, SPARC M5, SPARC M6, SPARC T7, 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, SPARC M5, or SPARC M6 server with at least system firmware version 9.1.0, and on a SPARC T7 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.

12

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).

This feature is available starting with the Oracle Solaris 11.3 OS. Ensure that the control domain, service domain, and guest domains that use the virtual SCSI HBA feature run at least the Oracle Solaris 11.3 SRU 23 OS.

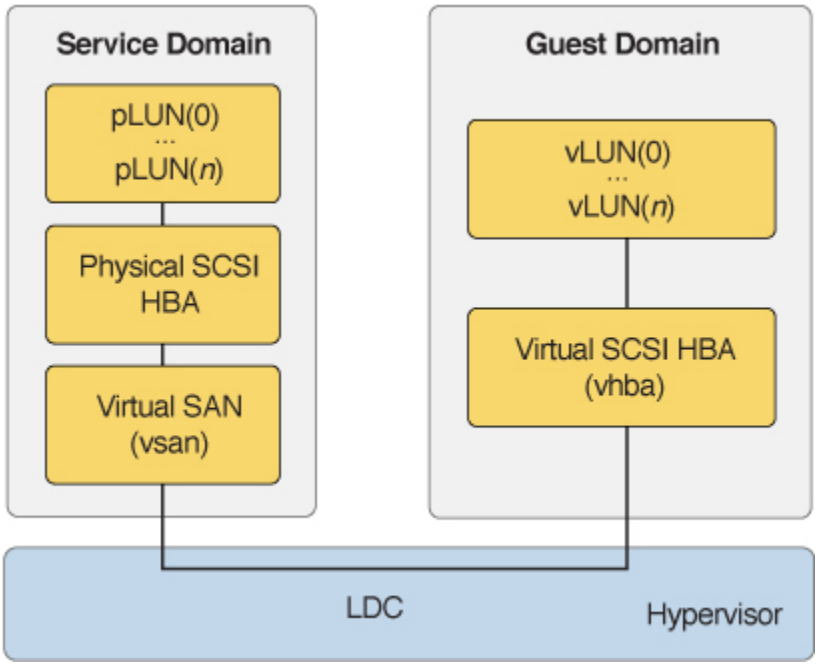
A vHBA instance provides access to all SCSI devices that are reachable by a specific vSAN instance. A vHBA can recognize any SCSI device type such as disk, CD, DVD, or tape. The set of reachable SCSI devices changes based on the set of physical SCSI devices that is

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 Oracle Solaris I/O multipathing solution with a vHBA instance and its vLUNs. The device path for a vLUN uses the full `cXtYdZsN` notation: `/dev/[r]dsk/cXtYdZsN` . The `tY` 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.

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.

The `vsan` module supports the I/O Domain Resiliency (IOR) feature. As a result, if the root domain that provides access to a virtual SAN's physical SCSI HBA initiator port is

interrupted, the `vsan` module ensures that all new and current I/O requests fail gracefully.

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`(8) 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 Oracle Solaris I/O multipathing is disabled in the guest domain. If Oracle Solaris I/O multipathing 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)
```

Discovering SCSI Devices

The `vhba` module cooperates with the `vsan` module to discover any SCSI devices that are reachable from the `vsan` module's SCSI Initiator Port.

Each time the shared LDC channel resets, the `vhba` and `vsan` modules determine the maximum supported vHBA Protocol version. This version determines which algorithm to use to discover devices.

If the `vhba` and `vsan` modules use vHBA protocol Version 1.0, any attempt to replace a device fails so as to prevent data corruption of the new device.

If the `vhba` and `vsan` modules both support at least vHBA protocol Version 1.2, the `vhba` module permits the insertion of a new device at a previously used path. The kernel and the `vhba` module ensure that I/O sent to the path's old device are gracefully terminated. I/O that is sent to the path's new device successfully complete.

The following example command shows the path and metadata of LUN3 underneath the `vhba@1` instance. The worldwide number (WWN) for LUN3 is presented in boldface:

```
# echo "::vhba -v" | mdb -k
...
vhba_t( 6400b4401900 ) vhba@1
  vhba_iport_t( 6400a6a0ce30 )
      vhba_tport_t( 6400b75cabb8 ) tport_phys(w200000110d211500) flags[COMMON]
...
      vhba_lun_t( 6400b75ca738 ) lun(3) vlun-id(6)
wwn(naa.600110D000211500010900009DEB2585) [COMMON]
      devinfo(6400c3c631a8) scsi_device(6400c3c2b0c8)
scsi_inquiry(6400c3a96640).inq_dtype(0=disk)
      /virtual-devices@100/channel-devices@200/scsi@1/iport@0/
disk@w200000110d211500,3
```

If you replace the device at LUN3's path with a new device, that new device has a different WWN as shown in the following example command output:

```
      vhba_lun_t( 6400b75cbdb8 ) lun(3) vlun-id(12)
wwn(naa.600110D000211500010900065106D8A) [NEW]
      devinfo(6400c3c631a8) scsi_device(6400cce2a8e8)
scsi_inquiry(6400d2cc3908).inq_dtype(0=disk)
      /virtual-devices@100/channel-devices@200/scsi@1/iport@0/
disk@w200000110d211500,3
```

Note that the `scsi@1/iport@0/disk@w200000110d211500,3` device path does not change when you replace the device. Only the WWN identity of the device changes.

Discovering SCSI Tape Devices

The SCSI target driver for tape devices, `st`, permits only one concurrent `open`. Thus, a service domain configuration that has two or more `vsan` instances that reference the

same SCSI HBA initiator port issues the following pair of messages for each `vsan` instance other than the one that successfully called `open`:

```
vsan: WARNING: vs@7: dip(6400a921b1f8): open(/pci@3c0/pci@1/pci@0/pci@7/SUNW,qlc@0/
fp@0,2/st@w200000110d211500,2) err(16)
vsan: WARNING: vs@7: device(6400a921b1f8) is of type TAPE; ensure all the domain's
vSANs are configured correctly.
```

At boot time, you cannot determine which `vsan` successfully calls `open` for a set of `vsan`s that indirectly reference the same tape device. This happens because the initialization of each `vsan` is independent and executes concurrently.

Use the `ldm add-vsan` and `ldm add-vhba` commands to specify which `vsan` is the first to open the tape devices that are reachable from that `vsan`'s initiator port. Also, set the `mask` attribute of the `vsan` to isolate a specific tape device within its own `vsan`.

For more information about using the `mask` attribute, see Managing the Physical Devices in a Virtual Storage Area Network.

To identify the device path of the tape device that opened successfully, perform the following commands in the guest domain. Use the `::vhba -v` command to find the entry that corresponds to the tape device's worldwide number. Note that the device type field shows `inq_dtype(1=tape)` for a tape device. The device path in the following example output is highlighted in boldface:

```
# mdb -k
> ::vhba -v
vhba_t( 640014816600 ) vhba@3
...
    vhba_iport_t( 6400222b1e50 )
        vhba_tport_t( 400040de0e0 ) tport_phys(w200000110d211500) flags[dirty,NEW]
...
            vhba_lun_t( 400040de200 ) lun(2) vlun-id(18)
wwn(naa.600110D0002115000109000001ABA60C) [NEW]
                devinfo(640022b0d840) scsi_device(40000176468)
scsi_inquiry(6400228bc560).inq_dtype(1=tape)
                /scsi_vhci/tape@g600110d0002115000109000001aba60c
```

The `/scsi_vhci/tape@g600110d0002115000109000001aba60c` device path shows that Solaris Multipathing is enabled in the `vhba.conf` file for the `vhba` instance that successfully opened the tape device. If Solaris Multipathing is disabled, the device path syntax would look similar to `/virtual-devices@100/channel-devices@200/scsi@3/iport@0/tape@w200000110d211500,2`.

The following example shows how to identify the `/dev` path that matches the device path shown by the `::vhba` command. In this example, Solaris Multipathing is enabled and the tape device is referenced by `/dev/rmt/1`:

```
# find /dev/rmt -ls | grep :$
... /dev/rmt/1 -> ../../devices/scsi_vhci/tape@g600110d0002115000109000001aba60c:
```

If you disable Solaris Multipathing in the `vhba.conf` file and reboot the domain, the tape device is referenced by `/dev/rmt/0`:

```
# find /dev/rmt -ls | grep :$
... /dev/rmt/0 -> ../../devices/virtual-devices@100/channel-devices@200/scsi@3/iport@0/
tape@w200000110d211500,2:
```

Protocol Version Combinations

The following statements describe the device discovery behavior based on the maximum vHBA protocol version that is compiled into each of the `vhba` and `vsan` modules:

Case 1:
Both modules use Version 1.0 of the vHBA protocol. The vHBA subsystem uses the original algorithm and does not support the replacement of a device at a previously used path.

> **Note:**
>
> If you install at least Oracle Solaris 11.4 SRU 39, any discovered LUNs disappear until you perform one of the corrective actions specified in Case 4.

Case 2:
Both modules use at least Version 1.2 of the vHBA protocol. The vHBA subsystem supports the replacement of a device at a previously used path.

Case 3:
`vhba` **uses Version 1.0 and** `vsan` **uses at least Version 1.2.** The `vsan` module adjusts to use Version 1.0.

Case 4:
`vhba` **uses at least Version 1.2 and** `vsan` **uses Version 1.0.** The vHBA subsystem cannot discover devices and writes the following message to the syslog each time the `vhba` module attempts to discover devices:

```
WARNING: vh@0 vhba(1.2) and vsan(1.0) VIO version mismatch; Device Discovery is disabled.
```

In this situation, ensure that your system is configured in one of the following ways:

- The `vsan` module supports at least Version 1.2 by upgrading your service domain to run at least Oracle Solaris 11.4 SRU 39. This configuration results in both the `vsan` and `vhba` modules using Version 1.2.
- The `vhba` module supports Version 1.0 at runtime by adding the following entry to the `/etc/system` file:

  ```
  set scsi:scsi_force_detach_enable = 0
  ```

  After updating the `/etc/system` file, reboot the `vhba` module's domain to ensure that both the `vsan` and `vhba` modules use Version 1.0.

  After updating the service domain, you can restore the `vhba` module's compiled-in support of Version 1.2 by removing the line from the `/etc/system` file and rebooting the `vhba` module's domain.

In addition to configuring the vHBA protocol between a `vhba` instance and its associated `vsan` instance, you must configure the vHBA protocol for all `vhba` instances in each guest domain.

- To support the new device discovery behavior provided by Version 1.2, ensure that all `vhba` instances in a domain support at least Version 1.2. Also, ensure that all associated `vsan` instances support at least Version 1.2.
- To support the original device discovery behavior provided by Version 1.0 in a domain that has multiple `vhba` instances, ensure that you set the `scsi_force_detach_enable` entry in the `/etc/system` file as follows:

  ```
  set scsi:scsi_force_detach_enable = 0
  ```

  After you update the `/etc/system` file, reboot the domain. Also, ensure that each of `vhba`'s associated `vsan` instances support Version 1.0.

The Hidden Device at LUN0

You might experience a *hidden device* at LUN0 if both of the following conditions occur:

1. The `vhba` binary can run at least vHBA Protocol Version 1.2, but is currently running Version 1.0
2. You replaced the device at LUN0 with another device.

> **Note:**
>
> A hidden device occurs only at LUN0 because a device at LUN0 is a special case for the vHBA subsystem, see Simulating a LUN0. If you insert another device at a non-LUN0 path, the new device is unusable because the WWN of the new device does not match the WWN of the original device. `vhba` issues the following message:
>
> ```
> WARNING: {vh@0,w200000110d211500,3} WWNs of (old, new) do not match:
> (naa.600110d0002115000109000008207288b,
> naa.600110d000211500010900009deb2582)
> ```

For a LUN0 device, the most basic SCSI commands must remain operational. `vhba` fails gracefully for all other SCSI commands to ensure that an unauthenticated device is not modified.

If `vhba` detects these conditions, the new device is *hidden* by the original device. `vhba` issues the following message:

```
WARNING: {vh@0,w200000110d211500,0} new(naa.600110d000211500010900025d42777 is hidden
by old(naa.600110d000211500010900025d4274a)
```

When you restore the original device to the LUN0 path, `vhba` no longer filters any I/O that is sent to that path and device. `vhba` issues the following message:

```
WARNING: {vh@0,w200000110d211500,0}: was restored to expected
wwn(naa.600110d000211500010900025d4274a)
```

Virtual SCSI HBA Subsystem Limitations

Virtual SCSI HBA Subsystem Does Not Support All SCSI Enclosure Services Devices

An SES device that is seen by the Oracle Solaris OS as a secondary function is an SES device type that cannot be supported by `vhba`. `vhba` can support an SES device whose device type has a value of `0xd` as specified in the `inq_dtype` field of the `INQUIRY` payload.

When the `vhba` binary in the guest domain attempts to initialize some SCSI enclosure services (SES) devices, `vhba` causes `scsi` to issue the following warning message:

```
... scsi: WARNING: scsi_enumeration_failed: vhba2 probe@w50080e51bfd32004,0,d
enumeration failed during tran_tgt_init
```

The `,d` substring represents the `0xd` hexadecimal digit, which is the SCSI industry standard code for an SES device. The `,d` string indicates that this warning message is a result of an unsupported type of SES device.

`vhba` can support an SES device that has a device type of `0xd` that is specified in the `inq_dtype` field of the `INQUIRY` payload:

```
# mdb -k
> ::vsan
vsan_t( 6400126e08c0 ) cfg-hdl(0) iport-path(/pci@300/pci@1/pci@0/pci@4/
SUNW,emlxs@0,11/fp@0,0)
    vsan_iport_t( 6400125b8710 )
        vsan_tport_t( 64001bf89718 ) tport_phys(w216000c0ff8089d5)
        vsan_lun_t( 640011aa65d0 ) lun(0) vlun-id(1127b) []

> 640011aa65d0::print vsan_lun_t vl_sd |::print struct scsi_device sd_inq
|::print struct scsi_inquiry inq_dtype
inq_dtype = d
```

Cannot Execute a Virtual SCSI HBA and a Virtual SAN in the Same Domain

You can use the `ldm add-vhba` to add a `vhba` instance that is associated with a `vsan` instance in the same domain only if Solaris multipathing is enabled for at most one of the following:

- Virtual SCSI HBA's initiator port
- Physical HBA initiator port that is encapsulated by the `vsan` instance

> **Caution:**
>
> Do not enable multipathing in both the virtual SCSI HBA's initiator port and in the virtual SAN's initiator port. This configuration causes the virtual SCSI HBA's discovery of LUNs to deadlock.

For information about how to manage the multipathing state for a physical HBA's initiator port, see *Managing SAN Devices and I/O Multipathing in Oracle Solaris 11.4*.

For information about how to manage the multipathing state of the initiator port that is represented by a `vhba` instance, see How to Manage Multipathing for Virtual SCSI HBAs in a Guest Domain.

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:

- Obtaining Physical SCSI HBA Information
- Creating a Virtual Storage Area Network
- Creating a Virtual SCSI Host Bus Adapter

- Verifying the Presence of a Virtual SCSI HBA
- Setting the Virtual SCSI HBA Timeout Option
- Removing a Virtual SCSI Host Bus Adapter
- Removing a Virtual Storage Area Network
- Adding or Removing a LUN

For more information about the commands that are shown in this section, see the `ldm`(8) 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 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. See the `ldm`(8) man page.

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/scsi@0/iport@1
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@1]
/SYS/MB/SASHBA0/scsi@0/iport@2
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@2]
/SYS/MB/SASHBA0/scsi@0/iport@4
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@4]
/SYS/MB/SASHBA0/scsi@0/iport@8
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@8]
/SYS/MB/PCIE1/SUNW,emlxs@0/fp@0,0
[/pci@300/pci@1/pci@0/pci@4/SUNW,emlxs@0/fp@0,0]
/SYS/MB/PCIE1/SUNW,emlxs@0,1/fp@0,0
[/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 I/O Multipathing in Oracle Solaris 11.4*.

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 system 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 create more than one virtual SAN from the same initiator port path. This action allows multiple guest domains to use the same initiator port.

> **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/SUNW,emlxs@0,1/fp@0,0` 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/SUNW,emlxs@0,1/fp@0,0 port0 svcdom
/SYS/MB/PCIE1/SUNW,emlxs@0,1/fp@0,0 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.

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 seconds 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
```

If you attempt to remove a virtual HBA that is actively managing I/O operations, the `ldm remove-vhba` command fails with the following error:

```
primary# ldm remove-vhba my-vhba my-domain
Dynamic reconfiguration of the virtual device on domain primary
```

```
failed with error code (-122).
The OS on domain primary did not report a reason for the failure.
Check the logs on that OS instance for any further information.
```

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 11.4*.

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. A virtual SCSI HBA and a virtual SAN can execute in the same domain. The more common configuration has the virtual SCSI HBA

and the virtual SAN execute in different domains where the virtual SAN executes in a service domain that has direct access to a physical HBA card.

Although a virtual SAN is associated conceptually with a physical SAN, it does not need to be. For example, you can create a virtual SAN that comprises the set of local storage devices that are reachable from a motherboard HBA.

The virtual HBA subsystem unconditionally creates a virtual LUN for each physical LUN that is discovered. So, as with virtual disks, you must not have conflicting workloads access the same virtual LUN.

For example, if an initiator port reaches ten physical SCSI devices, the virtual HBA subsystem creates ten virtual LUNs in the guest domain. If the guest operating system boots from one of those virtual LUNs, you must ensure that no other guest domain accesses the virtual LUN, and that the domain that owns the physical SCSI device does not access the physical LUN.

A similar warning holds for any virtual LUNs that might be in use by a guest domain. You must strictly control access to such virtual LUNs on other guest domains and access to the underlying physical LUN on the service domain to prevent conflicting access. Such conflicting access might result in data corruption.

Configuring Virtual SCSI HBA Multipathing

The virtual SCSI HBA subsystem supports multipathing in the guest domain and in the service domain by leveraging the Oracle Solaris I/O multipathing implementation. For more information, see *Managing SAN Devices and I/O Multipathing in Oracle Solaris 11.4*.

As in Oracle Solaris I/O 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 Oracle Solaris I/O 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`(4D) and `mpathadm`(8) man pages.

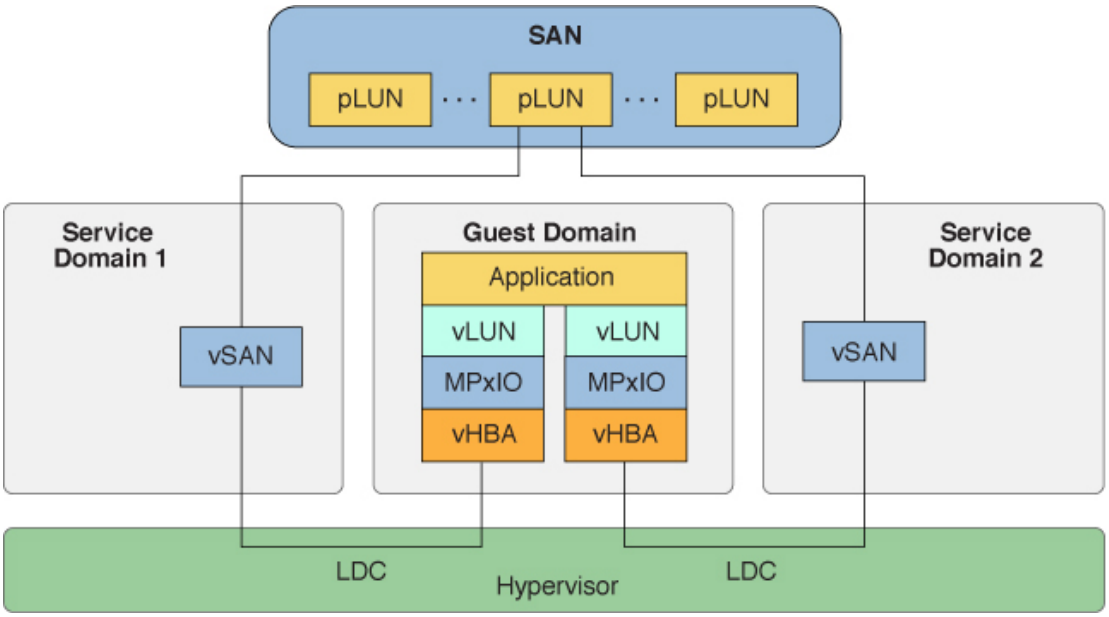
When multipathing is enabled in the service domain, as shown in the figure titled Configuring Virtual SCSI HBA Multipathing in a Service Domain, the `ldm add-vsan` command enables you to create a `vsan` instance that represents all of the paths that reference the SCSI devices that are reachable through a specified initiator port. However, when multipathing is disabled in the service domain, the `vsan` instance only represents those paths that originate at the specified initiator port and reference the SCSI devices.

To configure multipathing, you must configure two or more distinct paths from the guest domain or the service 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.

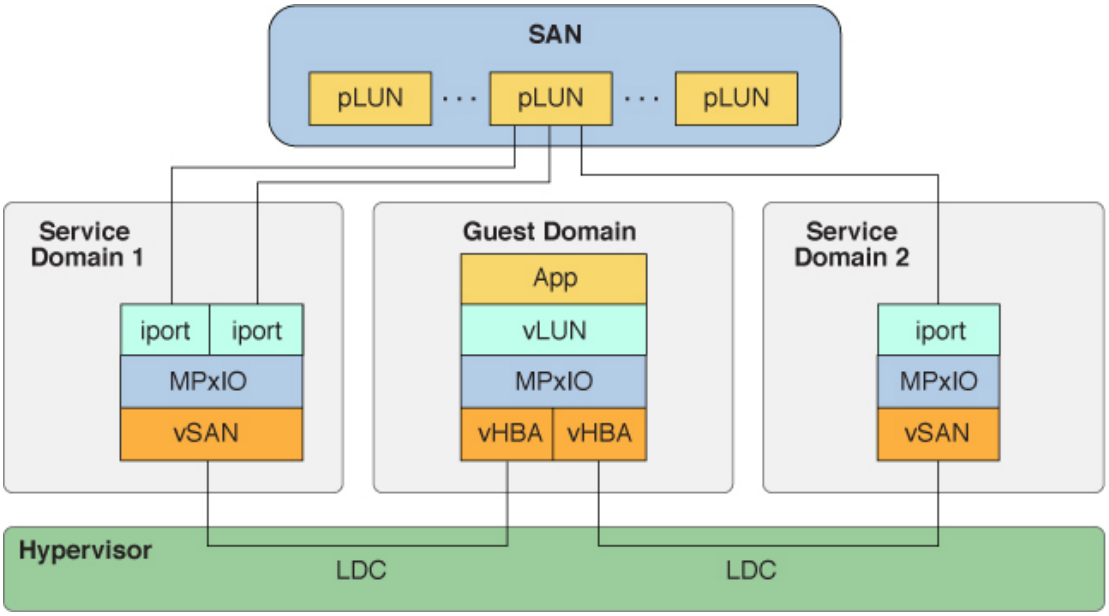
1. Execute a pair of `ldm add-vhba` and `ldm add-vsan` commands for each separate path to the back-end storage.
2. Enable Oracle Solaris I/O multipathing in the guest domain for the initiator ports that are managed by the `vhba` virtual HBA module.

The following figure is an example of a multipathing configuration in a guest domain. It shows one physical LUN of a SAN that is accessed by two paths that are managed by Oracle Solaris I/O multipathing. For a procedure that describes how to create the configuration shown in this figure, see How to Configure Virtual SCSI HBA Multipathing.

Configuring Virtual SCSI HBA Multipathing in a Guest Domain



Configuring Virtual SCSI HBA Multipathing in a Service Domain



How to Configure Virtual SCSI HBA Multipathing

This procedure describes how to create the virtual SCSI HBA multipathing configuration that is shown in the figure titled Configuring Virtual SCSI HBA Multipathing in a Guest Domain. 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 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/SUNW,emlxs@0/fp@0,0 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/SUNW,emlxs@0/fp@0,0 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.**

Example 12-1 Adding a Virtual SCSI HBA and a Virtual SAN

The following example shows how to create a virtual SAN for a specific SCSI HBA initiator port and how to associate that virtual SAN with a virtual SCSI HBA in the `ldg1` domain.

Use the `ldm list-hba` command to identify the physical SCSI HBA initiator ports in the `ldg1` domain:

```
primary# ldm list-hba -l ldg1
NAME                                                VSAN
----                                                ----
/SYS/MB/SASHBA0/scsi@0/iport@1
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@1]
/SYS/MB/SASHBA0/scsi@0/iport@2
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@2]
/SYS/MB/SASHBA0/scsi@0/iport@4
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@4]
/SYS/MB/SASHBA0/scsi@0/iport@8
[/pci@300/pci@1/pci@0/pci@2/scsi@0/iport@8]
/SYS/MB/PCIE1/SUNW,emlxs@0/fp@0,0
[/pci@300/pci@1/pci@0/pci@4/SUNW,emlxs@0/fp@0,0]
/SYS/MB/PCIE1/SUNW,emlxs@0,1/fp@0,0
[/pci@300/pci@1/pci@0/pci@4/SUNW,emlxs@0,1/fp@0,0]
```

Create a virtual SAN in the `ldg1` domain that you can use to manage all of the SCSI devices that are associated with the last initiator port in the previous output.

The following command exports the `vsan-mpxio1` virtual SAN from the `ldg1` domain:

```
primary# ldm add-vsan /SYS/MB/PCIE1/SUNW,emlxs@0,1/fp@0,0 vsan-mpxio1 ldg1
/SYS/MB/PCIE1/SUNW,emlxs@0,1/fp@0,0 resolved to device:
/pci@300/pci@1/pci@0/pci@4/SUNW,emlxs@0,1/fp@0,0
```

How to Manage Multipathing for Virtual SCSI HBAs in a 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. **Configure multipathing by modifying the `/etc/driver/drv/vhba.conf` file.**

   You can configure virtual HBA initiator port (port) multipathing on a per-port or global basis. Per-port settings have priority over the global setting.

   - Enable multipathing for all `vhba` instances in a guest domain.

     Add only the following line to the `vhba.conf` file:

     ```
     mpxio-disable="no"
     ```

   - Disable multipathing for all `vhba` instances in a guest domain.

     Add only the following line to the `vhba.conf` file:

     ```
     mpxio-disable="yes"
     ```

   - Enable all `vhba` instances except `vhba@1`.

     Add the following lines to the `vhba.conf` file:

```
mpxio-disable="yes"
compatible="/virtual-devices@100/channel-devices@200/scsi@1/iport@0"
mpxio-disable="no";
```

Note that `vhba@1` is what is shown in the `ldm list -o hba` command output, but `scsi@1` is how `vhba@1` is known to the operating system.

- Disable all `vhba` instances except `vhba@1`.

  Add the following lines to the `vhba.conf` file:

```
mpxio-disable="no"
compatible="/virtual-devices@100/channel-devices@200/scsi@1/iport@0"
mpxio-disable="yes";
```

3. **Reboot the guest domain.**

How to Enable Multipathing for Virtual SCSI HBAs in a Service Domain

1. **Enable Oracle Solaris I/O multipathing for all initiator ports in the service domain.**

```
svcdom# stmsboot -e
```

For more information, see Enabling and Disabling Multipathing in *Oracle Solaris SAN Configuration and Multipathing Guide*.

2. **List the SCSI devices that are reachable from each initiator port for a service domain.**

For example, the `ldm list-hba` command might show the following information about Service Domain 1 as shown in the figure titled Configuring Virtual SCSI HBA Multipathing in a Service Domain.

```
primary# ldm list-hba -d svcdom
DOMAIN
svcdom

IPORT                                                VSAN
-----                                                ----
/SYS/MB/PCIE0/SUNW,emlxs@0/fp@1
  c0t600110D000211501010900001061ADBF4d0
  c0t600110D000211501010900146489D34d0
/SYS/MB/PCIE1/SUNW,emlxs@0/fp@1
  c0t600110D000211501010900001061ADBF4d0
  c0t600110D000211501010900146489D34d0
```

3. **Create a virtual SAN instance that references a particular initiator port.**

In the following command, the initiator port references two SCSI devices that are also referenced by `PCIE0`. Either of the initiator ports that have physical paths to the same LUNs can be used as part of the `ldm add-vsan` command to configure the virtual SAN if multipathing is enabled.

```
primary# ldm add-vsan /SYS/MB/PCIE1/SUNW,emlxs@0/fp@1 my_mpxio_vsan svcdom
```

4. **Add the virtual SAN to the guest domain in a virtual SCSI HBA.**

```
primary# ldm add-vsan my_vhba my_mpxio_vsan gdom
```

5. **View the physical devices by running the `format` command in the service domain.**

   The following output shows two physical SCSI devices, each of which can have one or more paths to them.

```
svcdom# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c0t600110D000211501010900001061ADBF4d0 <SANBlaze-VLUN P0T1L7-V7.3-1.00GB>
          /scsi_vhci/ssd@g600110d000211501010900001061adbf4
       1. c0t600110D000211501010900146489D34d0 <SANBlaze-VLUN P0T1L6-V7.3-1.00GB>
          /scsi_vhci/ssd@g600110d000211501010900146489d34
       2. c1d0 <SUN-DiskImage-10GB cyl 282 alt 2 hd 96 sec 768>
          /virtual-devices@100/channel-devices@200/disk@0
Specify disk (enter its number):
```

   This command shows that the service domain configuration has two paths to each physical device.

```
svcdom# mpathadm list lu
        /dev/rdsk/c0t600110D000211501010900001061ADBF4d0s2
                Total Path Count: 2
                Operational Path Count: 2
        /dev/rdsk/c0t600110D000211501010900146489D34d0s2
                Total Path Count: 2
                Operational Path Count: 2
```

   Note that the `format` output in the guest domain is essentially identical, because the Oracle Solaris I/O multipathing implementation executes in both the guest domain and in the service domain. Oracle Solaris I/O multipathing also creates a device path that uses the worldwide number (WWN) of the logical unit such as `g600110d000211501010900146489d34` in the following output:

```
gdom# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c0t600110D000211501010900146489D34d0 <SANBlaze-VLUN P0T1L6-V7.3-1.00GB>
          /scsi_vhci/disk@g600110d000211501010900146489d34
       1. c0t600110D000211501010900001061ADBF4d0 <SANBlaze-VLUN P0T1L7-V7.3-1.00GB>
          /scsi_vhci/disk@g600110d000211501010900001061adbf4
       2. c1d0 <SUN-DiskImage-10GB cyl 282 alt 2 hd 96 sec 768>
          /virtual-devices@100/channel-devices@200/disk@0
Specify disk (enter its number):
```

How to Disable Multipathing for Virtual SCSI HBAs on Service Domains

- **Disable Oracle Solaris I/O multipathing for all initiator ports in the service domain.**

```
svcdom# stmsboot -d
```

  Also, you can enable or disable Oracle Solaris I/O multipathing on a per-initiator-port device in the service domain. For more information, see Enabling or Disabling Multipathing on a Per-Port Basis in *Oracle Solaris SAN Configuration and Multipathing Guide*.

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
```

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 Oracle Solaris I/O multipathing 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.

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 Oracle Solaris I/O 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.

Simulating a LUN0

The virtual SCSI HBA subsystem simulates the presence of a LUN0 for a SCSI target whose LUN0 or LUN0s are not visible in the service domain. A device is visible to the Oracle Solaris OS if it appears in `prtconf` output.

Because the `vhba` module uses a specific Oracle Solaris I/O framework, a LUN0 must be simulated if it is not visible in the service domain. The framework relies on the sending of a SCSI command (REPORT LUNS) to a SCSI target's LUN0 devices to discover the virtual SAN's LUNs. If a LUN0 is not visible, no LUNs within the virtual SAN can be discovered.

The `vsan` module creates metadata for a simulated LUN0 if the Oracle Solaris metadata for a physical LUN0 is not found during the physical LUN discovery process. The `vsan` module performs minimal simulation, which generates responses to the REPORT LUNS, INQUIRY, TEST UNIT READY, and REQUEST SENSE SCSI commands.

The requirement to simulate a LUN0 derives from an industry where it has become common for a storage vendor to implement a product that does not have a visible LUN0.

At runtime, the `vsan` module cannot determine whether a LUN0 is not present physically or is invisible, so the `vsan` module must simulate a LUN0 in both these instances. If a LUN0 is present but invisible, any functionality that is exported by the underlying SCSI device is not usable, and the `vsan` module returns `INVALID COMMAND` for any SCSI commands not mentioned previously.

The following shows how you can identify a simulated LUN0 before your system runs Oracle Solaris 11.4 SRU 39 and after your system runs Oracle Solaris 11.4 SRU 39:

- **Oracle Solaris 11.4 SRU 39.** The following commands show that the device type of the simulated LUN0 is DISK (`inq_dtype(0=disk)`) and that `sd` is bound to the device:

```
# echo "::vhba" | mdb -k
vhba_t( 6400c779d080 ) vhba@0
...
  vhba_lun_t( 6400c5282558 ) lun(0) vlun-id(0) [NEW]
      devinfo(6400cd323ba8) scsi_device(6400cd10c548)
scsi_inquiry(40012402060).inq_dtype(0=disk)
      /virtual-devices@100/channel-devices@200/scsi@0/iport@0/
disk@w200000110d211500,0
#
# prtconf -D /virtual-devices@100/channel-devices@200/scsi@0/iport@0/
disk@w200000110d211500,0
disk, instance #16 (driver name: sd)
```

  For more information about `sd`, see `sd`(4D) man page.

  Because a simulated LUN0 implements only the most basic SCSI commands, any Oracle Solaris software that assumes the simulated LUN0 is a real disk can see the warning messages. For example, because the simulated LUN0 is of type DISK, you might see the `drive not available` message:

```
# format
AVAILABLE DISK SELECTIONS:
...
  9. c16t200000110D211500d0 (drive not available)
     /virtual-devices@100/channel-devices@200/scsi@0/iport@0/
disk@w200000110d211500,0
```

  When LUN0 is simulated, the `vsan` module only issues the most basic SCSI commands. As a result, when applications such as the `format` command issue SCSI commands to read a disk label, they fail. So, the `drive not available` message appears, as expected.

- **Prior to Oracle Solaris 11.4 SRU 39.** You can identify a simulated LUN0 by executing the following commands in the domain in which the `vhba` module executes:

```
# echo "::vhba" | mdb -k
vhba_t( 400154422c0 ) vhba@1
...
  vhba_lun_t( 64002d478630 ) lun(0) vlun-id(0) [COMMON]
      devinfo(4001a1cd7f8) scsi_device(400152d7b58)
scsi_inquiry(40016a334f8).inq_dtype(0x1F=unknown)
      /virtual-devices@100/channel-devices@200/scsi@1/iport@0/
unknown@w200000110d211500,0
```

  Pass the `scsi_inquiry` value to the following command that outputs the vendor, product, and version number of the simulated device:

```
# echo "40016a334f8::print scsi_inquiry inq_vid inq_pid inq_revision" | mdb -
k
inq_vid = [ "ORCL    " ]
inq_pid = [ "vHBA:vsan       " ]
inq_revision = [ "1.0 " ]
```

A simulated LUN0 has a SCSI device type of `UNKNOWN`, as shown by the `inq_dtype` value of `0x1F` in the previous `mdb -k` example output.

Executing the following command and specifying LUN0's device path shows that the 'nulldriver' is bound to this device:

```
# prtconf -D /virtual-devices@100/channel-devices@200/scsi@0/iport@0/
unknown@w200000110d211500,0
unknown, instance #1 (driver name: nulldriver)
```

Managing the Physical Devices in a Virtual Storage Area Network

You can create a virtual SAN instance that represents all of or a subset of the physical devices that are reachable by an initiator port. By default, a virtual SAN instance represents the set of all physical devices that are reachable by the specified initiator port.

You can use the `ldm add-vsan` or `ldm set-vsan` command to specify the `mask` property, which controls how vSAN instances are created. You can create vSAN instances in the following ways:

- `mask=off` . Create a vSAN instance that represents the set of all physical devices that are reachable by the specified initiator port. This method is used by default to create vSAN instances.
- `mask=on` . Create a vSAN instance that represents a subset of the physical devices that are reachable by the specified initiator port.

When you use the `ldm set-vsan` command to specify the value of the `mask` property, the virtual SAN automatically notifies the virtual SCSI HBA instance that you changed the property value. If you change the `mask` property value to `off`, all devices that are reachable by the virtual SAN's initiator port become members of the virtual SAN. If you change the `mask` property value to `on`, the content of the virtual SAN's `mask` property value is reset to remove all physical device identification data. To populate the `mask` property with identification data, run the `ldm add-vsan-dev` command for each physical device you want to add to the virtual SAN.

> **Note:**
>
> When you issue the `ldm set-vsan` command, any running I/O commands are terminated gracefully. Subsequent I/O requests to a previously known vSAN member return an error stating that the device is no longer reachable.

When a virtual SAN instance has its `mask` property set to `on`, use the `ldm add-vsan-dev` command to add one or more physical devices with associated worldwide numbers (WWNs) to the virtual SAN instance. Use the `ldm remove-vsan-dev` command to remove a physical device with the specified WWN from the virtual SAN. Run the `ldm remove-vsan-dev` command for each physical device that you want to remove from the virtual SAN instance.

You can use the `ldm list-vsan` command to obtain information about a virtual SAN. The `ldm list-vsan` command lists the members of the specified virtual SAN. When `mask=on`,

the output shows the WWN of each virtual SAN member. When `mask=off`, the output states that the `mask` property value is `off`.

Obtaining Worldwide Numbers

Use the `ldm list-hba -u` command to view a device's WWN and all the paths that reference the device's WWN.

The following example shows information about the physical devices in the `primary` domain. The example output shows both the paths that reference the `naa.600c0ff0000000000089d513107ecb00` WWN of the physical device.

```
primary# ldm list-hba -u primary
DOMAIN
primary

naa.600c0ff0000000000089d513107ecb00
    /SYS/MB/RISER1/PCIE4/SUNW,qlc@0/fp@0,0/w216000c0ff8089d5,0
    /SYS/MB/RISER1/PCIE4/SUNW,qlc@0,1/fp@0,0/w216000c0ff8089d5,0
...
```

By including the `-l` option, the `ldm list-hba -u -l` command shows per-path and per-device metadata that you can use to identify a specific physical device within your system's topology.

```
primary# ldm list-hba -u -l primary
DOMAIN
primary

naa.600c0ff0000000000089d513107ecb00
    /SYS/MB/RISER1/PCIE4/SUNW,qlc@0/fp@0,0/w216000c0ff8089d5,0
    [/pci@0/pci@0/pci@8/pci@0/pci@2/SUNW,qlc@0/fp@0,0/
ssd@w216000c0ff8089d5,0]          (c0t600C0FF0000000000089D513107ECB00d0s0)
    /SYS/MB/RISER1/PCIE4/SUNW,qlc@0,1/fp@0,0/w216000c0ff8089d5,0
    [/pci@0/pci@0/pci@8/pci@0/pci@2/SUNW,qlc@0,1/fp@0,0/
ssd@w216000c0ff8089d5,0]        (c0t600C0FF0000000000089D513107ECB00d0s0)
...
```

By using the metadata in the previous example output, you can add the device associated with this WWN to the `my_vsan` virtual SAN.

In the following example, first create the `my_vsan` virtual SAN on the `my_domain` domain. Then, add the physical device associated with the `naa.600c0ff0000000000089d513107ecb00` WWN to the `my_vsan` virtual SAN:

```
primary# ldm add-vsan mask=on $iport_path my_vsan my_domain
primary# ldm add-vsan-dev my_vsan naa.600c0ff0000000000089d513107ecb00
```

13

Using Virtual Networks

This chapter describes how to use a virtual network with Oracle VM Server for SPARC software, and covers the following topics:

Introduction to a Virtual Network

A virtual network 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.

> **Note:**
>
> Oracle Solaris networking behaves the same whether the OS is running inside a zone, a kernel zone, a logical domain, or natively on the system. For more information about Oracle Solaris OS networking, see Oracle Solaris 10 Documentation and Oracle Solaris 11.4 Documentation.

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 11.4 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.
- 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 "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.

  The following command creates a virtual switch for the `primary` domain by specifying the generic link 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.

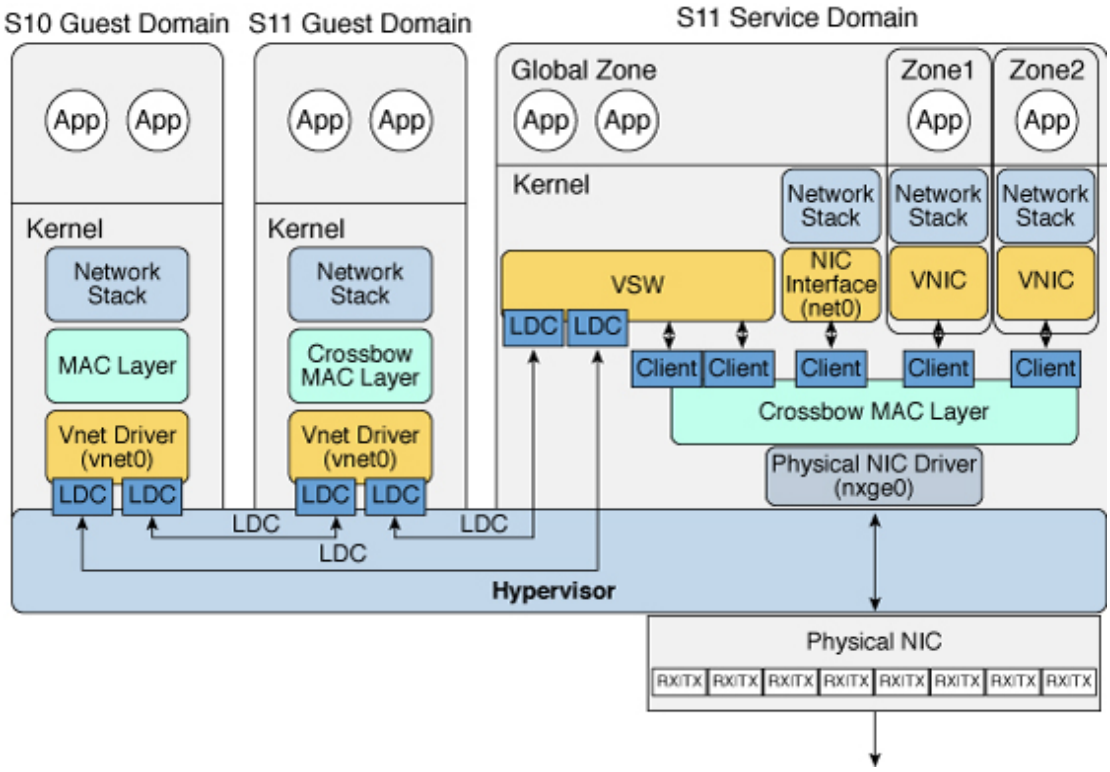
  A VNIC is a virtual NIC. When configured by using the `dladm create-vnic` command, a virtual NIC behaves like a physical NIC. You can configure a virtual NIC on an SR-IOV virtual function and on a virtual network device (VNET). See Creating Virtual NICs on SR-IOV Virtual Functions and Configuring Virtual NICs on Virtual Network Devices.
- In the Oracle Solaris 11.4 OS, use the `dladm`, `ipadm`, and `route` commands to perform network configuration of datalinks and IP interfaces. Network configuration profiles (NCP) are no longer supported in Oracle Solaris 11.4.

  However, when running versions of the Oracle Solaris 11 OS prior to Oracle Solaris 11.4, use the `DefaultFixed` NCP when configuring the Oracle VM Server for SPARC software in the control domain and in other domains. 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.

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.

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.

> **Note:**
>
> 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 even though the device might be visible.

- The virtual network device `vnet0` in an Oracle Solaris 10 guest domain can be configured as a network interface by using the `ifconfig` command.
- The virtual network device `vnet0` in an Oracle Solaris 11 guest domain might appear with a generic link name, such as `net0`. It can be configured as a network interface by using the `ipadm` command.

A virtual switch behaves like a regular physical network switch and switches network packets between the different systems to which it is connected. A system can be a guest domain, a service domain, or a physical network.

Maximizing Virtual Network Performance

You can achieve high transfer rates for guest and external networks and for guest-to-guest communications when you configure your platform and the domains as described in this section. The virtual network stack introduces support for large segment offload (LSO), which produces high TCP performance without requiring the use of jumbo frames.

Hardware and Software Requirements

Meet the following requirements to maximize the network performance for your domains:

- **Hardware requirements.** These performance improvements are available starting with the SPARC T4 server.
- **System firmware requirements.** These SPARC systems must run the latest system firmware. See System Firmware Versions in *Oracle VM Server for SPARC 3.6 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 Oracle Solaris OS Versions in *Oracle VM Server for SPARC 3.6 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 16 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.

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.

In an Oracle Solaris 11 service domain, do not use the virtual switch 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.

Although the virtual switch appears as a physical network device (`vsw`*N*) in `dladm show-phys` output, you cannot configure it as a network device in Oracle Solaris 11 because this capability has been deprecated in Oracle Solaris 11 and certain key features are inoperable.

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.

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 remove-vsw` commands, respectively. See the `ldm`(8) 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. Note that IPoIB and Ethernet-over-InfiniBand (EoIB) are unsupported back ends for virtual switches.

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).

> **Note:**
>
> A guest domain supports up to 999 virtual network devices.

A virtual network device can be used as a network interface with the name `net`*n*, which can be used like any regular network interface and configured with the Oracle Solaris 11 `ipadm` command.

You can add a virtual network device to a domain, set options for an existing virtual network device, and remove a virtual network device by using the `ldm add-vnet`, `ldm set-vnet`, and `ldm remove-vnet` commands, respectively. See the `ldm`(8) man page.

See the information about Oracle VM Server for SPARC networking for Oracle Solaris 11 in the figure titled Oracle VM Server for SPARC Network Overview for the Oracle Solaris 11 OS.

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`, `off`, or `auto` for the `inter-vnet-link` property.

By default, the `inter-vnet-link` property is set to `auto`, which means that inter-vnet LDC channels are allocated unless the number of virtual networks in a particular virtual switch grows beyond the default maximum limit specified by the `ldmd/auto_inter_vnet_link_limit` SMF property. The default `ldmd/auto_inter_vnet_link_limit` value is 8. If more than the maximum number of virtual networks are present for a virtual switch, the inter-vnet LDCs are disabled. See Determining What Networks Are Present in Logical Domains.

If binding a guest domain or adding virtual networks to a bound domain results in the number of virtual networks in the virtual switch exceeding the limit, the inter-vnet LDCs are disabled automatically. The reverse is true. If unbinding a guest domain or removing virtual networks from a bound domain results in the number of virtual networks in a virtual switch being less than the limit, the inter-vnet LDCs are enabled automatically.

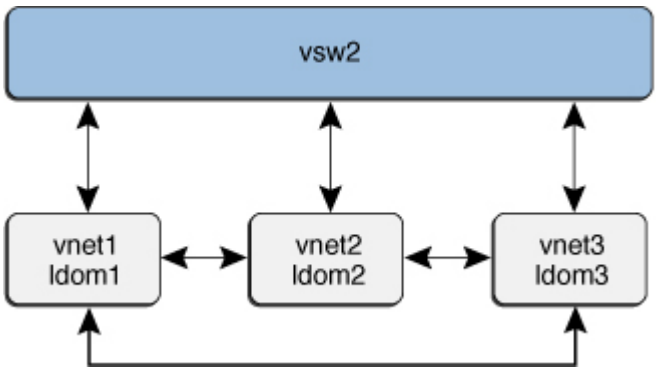
When `inter-vnet-link=auto`, the `ldm list` output shows the value as `on/auto` or `off/auto` depending on the active state of `inter-vnet` LDC channels for the virtual switch.

The following figures show typical virtual switch configurations when `inter-vnet-link=on` and `inter-vnet-link=off`, respectively.

The figure titled Virtual Switch Configuration That Uses Inter-Vnet Channels 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.

This figure also represents the case where `inter-vnet-link=auto` and the number of virtual networks connected to the same virtual switch is less than or equal to the maximum value set by the `ldmd/auto_inter_vnet_link_limit` SMF property.

Virtual Switch Configuration That Uses Inter-Vnet Channels

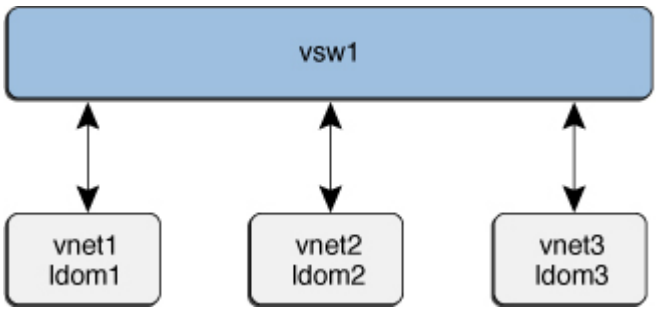


The figure titled Virtual Switch Configuration That Does Not Use Inter-Vnet Channels shows the same virtual switch configuration with the `inter-vnet-link` property set to `off`. The inter-vnet LDC channels are not allocated. Fewer LDC channels are used than when the `inter-vnet-link` property is set to `on`. In this configuration, guest-to-guest communications between `vnet1` and `vnet2` must go through `vsw1`.

This figure also represents the case where `inter-vnet-link=auto` and the number of virtual networks connected to the same virtual switch exceeds the maximum value set by the `ldmd/auto_inter_vnet_link_limit` SMF property.

> **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.

Virtual Switch Configuration That Does Not Use Inter-Vnet Channels



For more information about LDC channels, see Using Logical Domain Channels.

Determining What Networks Are Present in Logical Domains

You can issue commands from the OpenBoot PROM (OBP) prompt to list network information for logical domains.

Run the `show-nets` command on any domain to list the networks that are available on that domain:

```
OK show-nets
```

Run the `watch-net-all` command on the control domain to list the available networks and to show network traffic:

```
OK watch-net-all
```

Finding the Oracle Solaris 11 Network Interface Name

You can use the `ldm list-netdev` command to find the Oracle Solaris OS network interface name that corresponds to a virtual switch or virtual network device. For more information, see the `ldm`(8) man page. For a domain that runs an OS older than Oracle Solaris 11.2 SRU 1, obtain this information by mapping the MAC address from the

combination of the output from the `ldm list-domain -o network` command and from the `dladm show-phys -m` command.

The following example shows the `ldm list-netdev` command. The `ldm list-netdev` output shows the corresponding Oracle Solaris OS interface name in the `NAME` column.

```
primary# ldm list-netdev -l ldg5
DOMAIN
ldg5

NAME CLASS MEDIA STATE   SPEED OVER  LOC
---- ----- ----- -----   ----- ----  ---
net0 VNET  ETHER up      1000  vnet0 primary-vsw0/vnet0-ldg5
    [/virtual-devices@100/channel-devices@200/network@0]
    MTU       : 1500 [60-1500]
    IPADDR    : 10.129.71.179/255.255.252.0
              : fe80::214:4fff:fef8:3dc/ffc0::
              : 2606:b400:418:17b2:214:4fff:fef8:3dc/ffff:ffff:ffff:ffff::
    MAC_ADDRS : 00:14:4f:f8:03:dc
net1 VNET  ETHER unknown 0R    vnet1 primary-vsw0/ldg5-vnet1
    [/virtual-devices@100/channel-devices@200/network@1]
    MTU       : 1500 [60-1500]
    MAC_ADDRS : 00:14:4f:f8:9f:eb
net2 VNET  ETHER unknown 0R    vnet3 primary-vsw0/ldg5-vnet2
    [/virtual-devices@100/channel-devices@200/network@2]
    MTU       : 1500 [60-1500]
    MAC_ADDRS : 00:14:4f:f8:54:97
```

The corresponding Oracle Solaris OS network interface to the virtual device, `ldg5-vnet0`, is `net0`.

To verify that the `ldm list-netdev` output is correct, run the `dladm show-phys` command from the `ldg5` domain:

```
ldg5# dladm show-phys -m
LINK SLOT     ADDRESS            INUSE CLIENT
net0 primary  0:14:4f:f8:3:dc    yes   net0
net1 primary  0:14:4f:f8:9f:eb   no    --
net2 primary  0:14:4f:f8:54:97   no    --
```

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 13-1 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      1000  --        primary-vsw0/vnet0_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      10000 --        ldg1-vsw1/vnet1_ldg1
net6              VNET  ETHER up      10000 --        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 13-2 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      1000  --   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]
    MAC_ADDRS : a0:36:9f:0a:c5:d2

net5              VNET     ETHER    up      10000 --   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      10000 --   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 13-3 Listing Network Device Statistics

The `ldm list-netstat` command shows network statistics for one or more domains in the system.

The following example shows the default network statistics for all domains in the system.

```
primary# ldm list-netstat

DOMAIN
primary

NAME               IPACKETS     RBYTES       OPACKETS     OBYTES
----               --------     ------       --------     ------
net3               0            0            0            0
net0               2.72M        778.27M      76.32K       6.01M
net4               2.72M        778.27M      76.32K       6.01M
net6               2            140          1.30K        18.17K
net7               0            0            0            0
net2               0            0            0            0
net1               0            0            0            0
aggr1              0            0            0            0
ldoms-vsw0.vport0  935.40K      74.59M       13.15K       984.43K
ldoms-vsw0.vport1  933.26K      74.37M       11.42K       745.15K
ldoms-vsw0.vport2  933.24K      74.37M       11.46K       747.66K
ldoms-vsw1.vport1  202.26K      17.99M       179.75K      15.69M
ldoms-vsw1.vport0  202.37K      18.00M       189.00K      16.24M
------------------------------------------------------------------------
DOMAIN
ldg1
```

```
NAME              IPACKETS     RBYTES       OPACKETS     OBYTES
----              --------     ------       --------     ------
net0              5.19K        421.57K      68           4.70K
net3              0            0            2.07K        256.93K
net4              0            0            4.37K        560.17K
net1              0            0            2.29K        303.24K
net5              149          31.19K       78           17.00K
net6              147          30.51K       78           17.29K
aggr2             0            0            0            0
ldoms-vsw0.vport3 162          31.69K       52           14.11K
ldoms-vsw0.vport2 163          31.74K       51           13.76K
ldoms-vsw0.vport1 176          42.99K       25           1.50K
ldoms-vsw0.vport0 158          40.19K       45           4.42K
------------------------------------------------------------------------
DOMAIN
ldg2

NAME              IPACKETS     RBYTES       OPACKETS     OBYTES
----              --------     ------       --------     ------
net0              5.17K        418.90K      71           4.88K
net1              2.70K        201.67K      2.63K        187.01K
net2              132          36.40K       1.51K        95.07K
------------------------------------------------------------------------
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 ensures that one guest domain does not take over the available physical network bandwidth and leave none for the others. This feature is supported on a service domain that runs at least the Oracle Solaris 11 OS and is configured with a virtual switch.

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

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 13-4 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 13-5 Setting the Bandwidth Limit on an Existing Virtual Network Device

The following command 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 13-6 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 13-7 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 `vnet3` virtual network device has a bandwidth limit of 15 Mbps. If no bandwidth limit is set, the `MAXBW` field is blank.

```
primary# ldm ls-bindings -e -o network ldg3
NAME
ldg3

MAC
    00:14:4f:f8:5b:12

NETWORK
    NAME         SERVICE                 MACADDRESS PVID|PVLAN|VIDs
    ----         -------                 ---------- ---------------
    vnet3        primary-vsw0@primary    00:14:4f:fa:ba:b9 1|--|--
            DEVICE     :network@0        ID   :0
            LINKPROP   :--               MTU  :1500
            MAXBW      :15M              MODE :--
            CUSTOM     :disable
            PRIORITY   :--               COS  :--
            PROTECTION :--

        PEER                    MACADDRESS          PVID|PVLAN|VIDs
        ----                    ----------          ---------------
        primary-vsw0@primary    00:14:4f:f9:08:28   1|--|--
            LINKPROP   :--              MTU  :1500
            MAXBW      :--              LDC  :0x0
            MODE       :--
```

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 and the `ldm list-domain -o network` 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 -e -o network primary
VSW
    NAME          MACADDRESS          NET-DEV       DVID|PVID|VIDs
    ----          ----------          -------       --------------
    primary-vsw0  00:14:4f:fb:86:af   net0          1|1|--
DEVICE          :switch@0        ID   :0
            LINKPROP        :phys-state      MTU  :1500
            INTER-VNET-LINK :on/auto         MODE :--
            VSW-RELAY-MODE  :local

        PEER          MACADDRESS           PVID|PVLAN|VIDs
        ----          ----------           ---------------
        vnet1@ldg1    00:14:4f:f9:41:af    1|--|--
            LINKPROP   :phys-state      MTU  :1500
            MAXBW      :--              LDC  :0xa
            MODE       :--
            CUSTOM     :disable
            PRIORITY   :--              COS  :--
            PROTECTION :--

        vnet0@ldg1    00:14:4f:f9:41:fb    1|--|--
            LINKPROP   :phys-state      MTU  :1500
            MAXBW      :--              LDC  :0xc
            MODE       :--
            CUSTOM     :disable
            PRIORITY   :--              COS  :--
            PROTECTION :--
```

The following example shows that the `ldg1` domain has two virtual network devices: `vnet0` and `vnet1`. The `vnet1` device has a device number of `0` (`network@0`) and the `vnet0` device has a device number of `1` (`network@1`).

```
primary# ldm list-domain -e -o network ldg1
NETWORK
    NAME         SERVICE                MACADDRESS          PVID|PVLAN|VIDs
    ----         -------                ----------          ---------------
    vnet1        primary-vsw0@primary   00:14:4f:f9:41:af   1|--|--
            DEVICE     :network@0       ID   :0
            LINKPROP   :phys-state      MTU  :1500
            MAXBW      :--              MODE :--
            CUSTOM     :disable
```

```
            PRIORITY   :--              COS  :--
            PROTECTION :--

NAME          SERVICE                  MACADDRESS          PVID|PVLAN|VIDs
 ----          -------                  ----------          ---------------
 vnet0         primary-vsw0@primary     00:14:4f:f9:41:fb   1|--|--
            DEVICE     :network@1       ID   :1
            LINKPROP   :phys-state      MTU  :1500
            MAXBW      :--              MODE :--
            CUSTOM     :disable
            PRIORITY   :--              COS  :--
            PROTECTION :-
```

When a domain with a virtual network device is running the Oracle Solaris 11 OS, the virtual network device has a network interface, `net` *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.

Managing MAC Addresses With Oracle VM Server for SPARC

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`, `set-domain`, `add-vsw`, `set-vsw`, `add-vnet`, `set-vnet`, and `set-io`. 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. 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.

Detecting MAC Address Collisions

A detection check for duplicate MAC addresses is performed when the logical domain or network device is created, when the logical domain is started with the `-m` option, and when the `ldmd/mac_collision_check` SMF property is set to `true`.

The `mac_collision_check` SMF property controls whether a MAC address collision check is performed when the Logical Domains Manager starts. The check is performed when the property value is `true`. The default value is `false`.

The following command enables the MAC address collision check during Logical Domains Manager startup by setting the `mac_collision_check` SMF property:

```
primary# svccfg -s ldmd setprop ldmd/mac_collision_check=true
primary# svcadm refresh ldmd
primary# svcadm restart ldmd
```

By default, MAC address collision checks are disabled. If the `mac_collision_check` SMF property is enabled, a warning message is logged when collisions are detected.

> **Note:**
>
> The MAC address collision check slows down the start of the Logical Domains Manager process depending on the number of MAC addresses to check.

In addition to the Logical Domains Manager performing MAC address collision checks, you can perform this check when a domain starts by using the `ldm start-domain -m` command. If a MAC address collision is detected, the `ldm start-domain` command fails.

> **Note:**
>
> The MAC address collision check slows down the start of a domain depending on the number of MAC addresses to check.

The following command to enable the MAC address collision check when the `ldg1` domain starts up fails with an error:

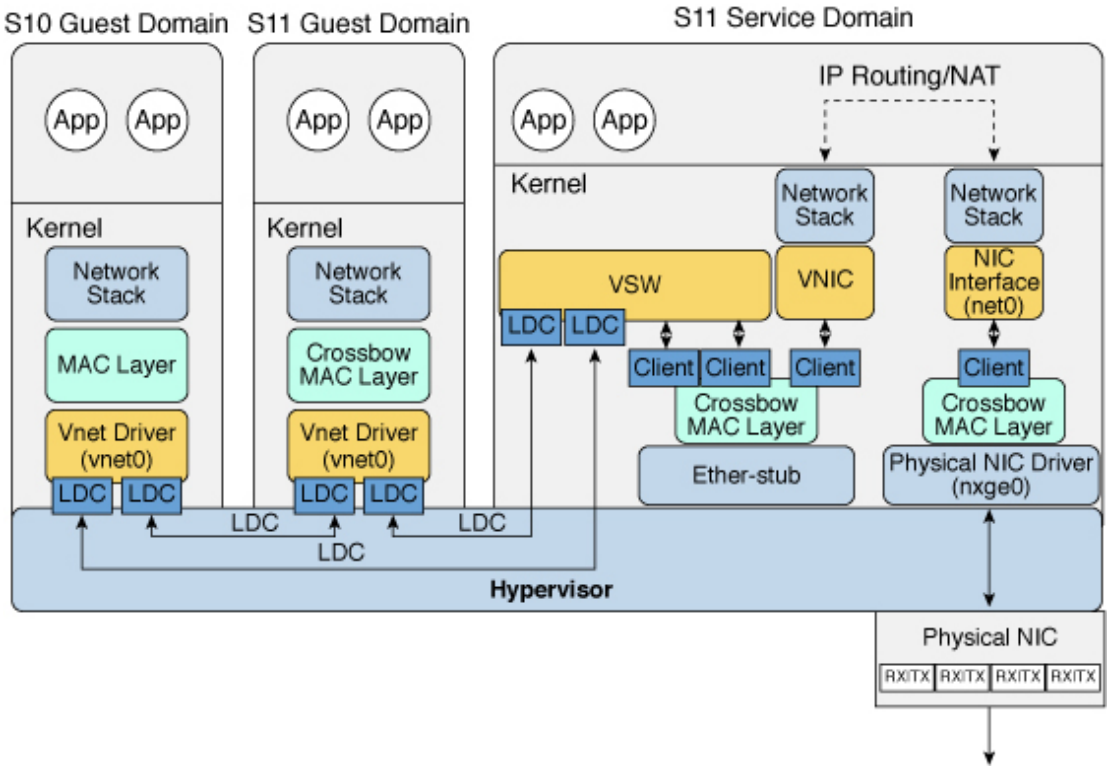
```
primary# ldm start-domain -m ldg1
MAC address 00:14:4f:fb:9d:32 is already in use.
```

Configuring a Virtual Switch and the Service Domain for NAT and Routing

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 network endpoints, including zones, in the service domain. By enabling IP routing in the service domain, virtual networks can communicate outside the machine by using the service domain as a router. Subsequently, configure NAT in the service domain to provide external connectivity to guest domains by means of a private IP address that is not externally routable. 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.

Virtual Network Routing



How to Set Up a Virtual Switch to Enable NAT 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 to provide external connectivity to the domains.**

   See Customizing IP Interface Properties and Addresses in *Configuring and Managing Network Components in Oracle Solaris 11.4* and Packet Forwarding and Routing on IPv4 Networks in *Oracle Solaris Administration: IP Services*.

Configuring IPMP in an Oracle VM Server for SPARC Environment

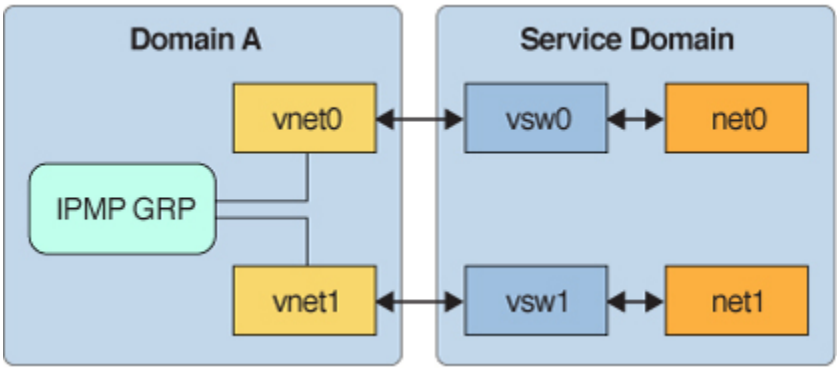
The Oracle VM Server for SPARC software supports link-based IP network multipathing (IPMP) with virtual network devices. When configuring an IPMP group with virtual network devices, configure the group to use link-based detection. If you are using older versions of the Oracle VM Server for SPARC (Logical Domains) software, you can only configure probe-based detection with virtual network devices.

Configuring Virtual Network Devices Into an IPMP Group in an Oracle Solaris 11 Domain

The figure titled Two Virtual Networks Connected to Separate Virtual Switch Instances (Oracle Solaris 11) 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.

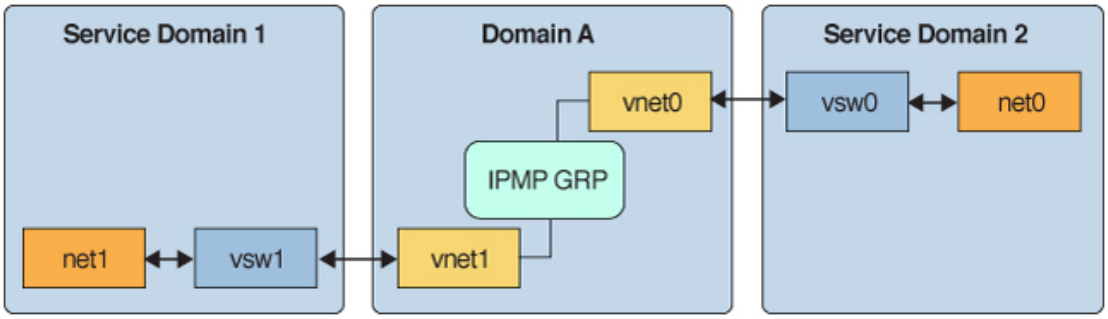
If a physical link failure occurs in the service domain, the virtual switch device that is bound to that physical device detects the link failure. Then, the virtual switch device propagates the failure to the corresponding virtual network device that is bound to this virtual switch. The virtual network device sends notification of this link event to the IP layer in the guest domain A, which results in failover to the other virtual network device in the IPMP group.

Two Virtual Networks Connected to Separate Virtual Switch Instances (Oracle Solaris 11)



The figure titled Virtual Network Devices Each Connected to Different Service Domains (Oracle Solaris 11) 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, guest domain A can detect virtual network failure and trigger a failover following a service domain crash or shutdown.

Virtual Network Devices Each Connected to Different Service Domains (Oracle Solaris 11)

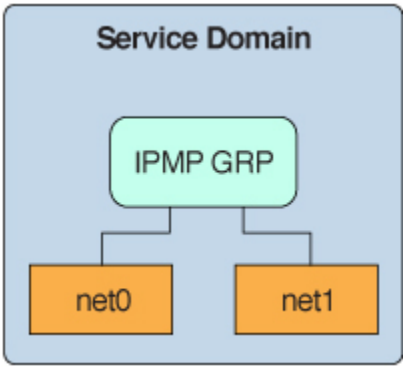


For more information, see “Administering Oracle Solaris Networks” in the Oracle Solaris 11.4 Information Library.

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. The figure titled Two Physical NICs Configured as Part of an IPMP Group (Oracle Solaris 11) shows two physical interfaces, which are `net0` and `net1`.

Two Physical NICs Configured as Part of an IPMP Group (Oracle Solaris 11)



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) and physical link. This configuration is enabled by default and does not require you to perform additional configuration steps.

> **Note:**
>
> The `linkprop` property is set to `phys-state` by default depending on backing device support. You do not need to perform the tasks in this section unless you disabled the `linkprop` property manually and are now attempting to set the property to the `phys-state` value.

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 releases prior to Oracle Solaris 11.4, see the `dladm`(1M), `ipadm`(1M), and `ipmpstat`(1M) man pages. For Oracle Solaris 11.4, see the `dladm`(8), `ipadm`(8), and `ipmpstat`(8) man pages.

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:**
>
> 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 11.4, see Chapter 1, About Using Rights to Control Users and Processes in *Securing Users and Processes in Oracle Solaris 11.4*.

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.

     ```
     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
```

Configuring Link-Based IPMP

The following example shows how to configure two virtual network devices on a domain to use link-based IPMP. Each virtual network device is connected to a separate virtual switch device on the service domain.

> **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. If the virtual switch has a physical network device assigned, physical link state updates are enabled automatically. Otherwise, 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 192.168.1.1 netmask + broadcast + group ipmp0 up
  # ifconfig vnet1 plumb group ipmp0
  ```

- **Oracle Solaris 11 OS:** Use the `ipadm` command.

  Note that `net0` and `net1` are the Oracle Solaris 11 vanity names for `ldom1_vnet0` and `ldom1_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
  ```

Configuring DLMP Aggregations Over Virtual Network Devices

The Oracle VM Server for SPARC software supports datalink multipathing (DLMP) aggregation over virtual network devices on logical domains.

To configure DLMP aggregations over virtual network devices in a domain, ensure that both the service domain and the guest domain run at least the Oracle Solaris 11.4 OS.

While the DLMP aggregation feature is similar to IPMP, IPMP manages IP addresses on a set of network interfaces while DLMP manages the virtual NIC. Both of these features provide high-availability capabilities that enable network connections to remain up even when a service domain becomes unavailable due to a reboot or a panic.

DLMP Aggregation Limitations

The DLMP aggregation feature has the following limitations:

- The DLMP aggregation over virtual network devices is not operable if one of the connected virtual switches is configured over an Ethernet stub.
- The underlying physical device of each connected virtual switch (as specified by the `net-dev` property) must have the same link speed.

How to Configure a DLMP Aggregation in a Domain

Before You Begin

Before you can configure a DLMP aggregation over virtual network devices in a domain, you must ensure that the `linkprop` property for each virtual network device is set to `phys-state` and that the `custom` property set to `enable`. For more information, see How to Configure Physical Link Status Updates, Configuring Trusted Virtual Networks, and the `ldm`(8) man page.

The steps you perform in this procedure are run in two domains, the domain in which you create the DLMP aggregation and the `primary` domain.

The shell prompt indicates the domain in which to run the commands: `gdom#` for the DLMP aggregation domain and `primary#` for the `primary` domain.

1. **Determine which virtual network devices to use.**

   Any of the virtual network devices that you choose cannot be in use.

   ```
   gdom# dladm show-phys -m
   ```

2. **Ensure that the virtual network devices you choose for the DLMP aggregation have the `custom` property set to enable and the `linkprop` property set to `phys-state`.**

   ```
   primary# ldm list -o network domain-name
   ```

3. **Configure the DLMP aggregation.**

   The following command assumes that two virtual network devices are used for the DLMP aggregation.

   ```
   gdom# dladm create-aggr -m dlmp -l net-name1 -l net-name2 aggr-name
   ```

   > **Note:**
   >
   > Ensure that the network devices that are configured into the DLMP aggregation do not have any IP interface configured over them.

4. **Configure an IP interface for the DLMP aggregation.**

```
gdom# ipadm create-ip aggr-name
```

5. **Configure an IP address for the DLMP aggregation.**

```
gdom# ipadm create-addr -a IP-addr/24 aggr-name
```

6. **Verify the configuration of the DLMP aggregation and verify that the network devices used by the aggregation are attached.**

```
gdom# dladm show-aggr -x
```

7. **Verify the configuration of the DLMP aggregation and the network devices.**

```
primary# ldm list-netdev domain-name
```

8. **Enable probe-based failure detection for the DLMP aggregation.**

```
gdom# dladm set-linkprop -p probe-ip=+ aggr-name
```

9. **Verify the detailed probe information about the DLMP aggregation configuration.**

```
gdom# dladm show-aggr -nS
```

Example 13-8 Configuring a DLMP Aggregation in a Domain

This example follows the steps in the procedure to create a DLMP aggregation.

You can determine that `net1` and `net2` are the Oracle Solaris 11 vanity names for `vnet1` and `vnet2` by matching the MAC addresses.

```
gdom# dladm show-phys -m
LINK                SLOT     ADDRESS            INUSE CLIENT
net0                primary  0:14:4f:fb:68:f1   yes   net0
net1                primary  0:14:4f:fa:20:68   no    --
net2                primary  0:14:4f:fa:42:a8   no    --
```

Verify that the `custom` property is set to `enable` and that the `linkprop` property is set to `phys-state` on the virtual network devices that will be used for the DLMP aggregation. Also, verify that the `vnet1` and `vnet2` virtual network devices are associated with different virtual switches.

```
primary# ldm list -o network gdom
NAME         SERVICE                MACADDRESS          PVID|PVLAN|VIDs
----         -------                ----------          ---------------
vnet1        primary-vsw0@primary   00:14:4f:fa:20:68   1|--|--
 DEVICE     :network@3       ID   :3
LINKPROP   :phys-state      MTU  :1500
 MAXBW      :--              MODE :--
CUSTOM     :enable
 MAX-CUSTOM-MACS:4096        MAX-CUSTOM-VLANS:4096
 PRIORITY   :--              COS  :--
 PROTECTION :--

NAME         SERVICE                MACADDRESS          PVID|PVLAN|VIDs
----         -------                ----------          ---------------
vnet2        primary-vsw1@primary   00:14:4f:fa:42:a8   1|--|--
 DEVICE     :network@3       ID   :3
LINKPROP   :phys-state      MTU  :1500
 MAXBW      :--              MODE :--
CUSTOM     :enable
 MAX-CUSTOM-MACS:4096        MAX-CUSTOM-VLANS:4096
 PRIORITY   :--              COS  :--
 PROTECTION :--
```

Note that the underlying device of the `primary-vsw0` and `primary-vsw1` virtual switches must have the same link speed to support DLMP aggregation over the `vnet1` and `vnet2` virtual network devices. The single service domain shown in this example is not a high-availability configuration. If the `primary` domain fails, DLMP aggregation fails. A high-availability configuration must use at least two service domains.

Configure the DLMP aggregation.

```
gdom# dladm create-aggr -m dlmp -l net1 -l net2 aggr0
```

Configure an IP interface for the DLMP aggregation.

```
gdom# ipadm create-ip aggr0
```

Configure an IP address for the DLMP aggregation.

```
gdom# ipadm create-addr -a 192.168.10.14/24 aggr0
```

Verify the configuration of the DLMP aggregation and verify that the network devices used by the aggregation are attached.

```
gdom# dladm show-aggr -x
LINK        PORT    SPEED   DUPLEX   STATE   ADDRESS           PORTSTATE
aggr0       --      1000Mb  full     up      2:8:20:d4:52:cc   --
net1                1000Mb  full     up      0:14:4f:fa:20:68  attached
net2                1000Mb  full     up      0:14:4f:fa:42:a8  attached
```

Verify the configuration of the DLMP aggregation and the network devices.

```
primary# ldm list-netdev gdom
DOMAIN
gdom

NAME         CLASS    MEDIA    STATE    SPEED    OVER     LOC
----         -----    -----    -----    -----    ----     ---
aggr         AGGR     ETHER    up       1G       net1,net2 --
net0         VNET     ETHER    up       1G       vnet0    primary-vsw0/vnet0
net1         VNET     ETHER    up       1G       vnet1    primary-vsw0/vnet1
net2         VNET     ETHER    up       1G       vnet2    primary-vsw1/vnet2
```

Enable probe-based failure detection for DLMP and verify the detailed probe information about the DLMP aggregation configuration.

```
gdom# dladm set-linkprop -p probe-ip=+ aggr0
gdom# dladm show-aggr -nS
LINK        PORT           FLAGS STATE   TARGETS         XTARGETS
aggr0       net1           u--3  active  192.168.10.1    net2
--          net2           u-2-  active  --              net1
```

Example 13-9 Configuring a High-Availability DLMP Aggregation

This example follows the steps in the procedure to create a high-availability DLMP aggregation. The difference is that the `net2` (`vnet2`) network device is associated with the `secondary-vsw0` virtual switch instead of with the `primary-vsw1` virtual switch. Each of these virtual switches are in different service domains.

```
gdom# dladm show-phys -m
LINK                SLOT     ADDRESS            INUSE CLIENT
net0                primary  0:14:4f:fb:68:f1   yes   net0
net1                primary  0:14:4f:fa:20:68   no    --
net2                primary  0:14:4f:fa:42:a8   no    --
```

Verify that the `custom` property is set to `enable` and that the `linkprop` property is set to `phys-state` on the virtual network devices that will be used for the DLMP aggregation. Also, verify that the `vnet1` and `vnet2` virtual network devices are associated with different virtual switches.

```
primary# ldm list -o network gdom
NAME          SERVICE                 MACADDRESS          PVID|PVLAN|VIDs
----          -------                 ----------          ---------------
vnet1         primary-vsw0@primary    00:14:4f:fa:20:68   1|--|--
 DEVICE      :network@3        ID   :3
LINKPROP    :phys-state       MTU  :1500
 MAXBW       :--               MODE :--
CUSTOM      :enable
 MAX-CUSTOM-MACS:4096          MAX-CUSTOM-VLANS:4096
 PRIORITY    :--               COS  :--
 PROTECTION :--

NAME          SERVICE                 MACADDRESS          PVID|PVLAN|VIDs
----          -------                 ----------          ---------------
vnet2         secondary-vsw0@secondary   00:14:4f:fa:42:a8   1|--|--
 DEVICE      :network@3        ID   :3
LINKPROP    :phys-state       MTU  :1500
 MAXBW       :--               MODE :--
CUSTOM      :enable
 MAX-CUSTOM-MACS:4096          MAX-CUSTOM-VLANS:4096
 PRIORITY    :--               COS  :--
 PROTECTION :--
```

Note that the underlying device of the `primary-vsw0` and `secondary-vsw0` virtual switches must have the same link speed to support DLMP aggregation over the `vnet1` and `vnet2` virtual network devices. For more information about Oracle Solaris DLMP aggregation, see Chapter 2, Configuring High Availability by Using Link Aggregations in *Managing Network Datalinks in Oracle Solaris 11.4*.

Configure the DLMP aggregation.

```
gdom# dladm create-aggr -m dlmp -l net1 -l net2 aggr0
```

Configure an IP interface for the DLMP aggregation.

```
gdom# ipadm create-ip aggr0
```

Configure an IP address for the DLMP aggregation.

```
gdom# ipadm create-addr -a 192.168.10.14/24 aggr0
```

Verify the configuration of the DLMP aggregation and verify that the network devices used by the aggregation are attached.

```
gdom# dladm show-aggr -x
LINK        PORT    SPEED   DUPLEX   STATE   ADDRESS           PORTSTATE
aggr0       --      1000Mb  full     up      2:8:20:d4:52:cc   --
net1                1000Mb  full     up      0:14:4f:fa:20:68  attached
net2                1000Mb  full     up      0:14:4f:fa:42:a8  attached
```

Verify the configuration of the DLMP aggregation and the network devices.

```
primary# ldm list-netdev gdom
DOMAIN
gdom
```

```
NAME        CLASS    MEDIA    STATE    SPEED    OVER     LOC
----        -----    -----    -----    -----    ----     ---
aggr        AGGR     ETHER    up       1G       net1,net2 --
net0        VNET     ETHER    up       1G       vnet0    primary-vsw0/vnet0
net1        VNET     ETHER    up       1G       vnet1    primary-vsw0/vnet1
net2        VNET     ETHER    up       1G       vnet2    secondary-vsw0/vnet2
```

Enable probe-based failure detection for DLMP and verify the detailed probe information about the DLMP aggregation configuration.

```
gdom# dladm set-linkprop -p probe-ip=+ aggr0
gdom# dladm show-aggr -nS
LINK        PORT           FLAGS STATE   TARGETS         XTARGETS
aggr0       net1           u--3  active  192.168.10.1    net2
--          net2           u-2-  active  --              net1
```

Example 13-10 Configuring a DLMP Aggregation For a Virtual Switch on an Ethernet Stub Is Inoperable

The following example output shows that a DLMP aggregation you create for a virtual switch on an Ethernet stub is inoperable. In such a situation, the `net3` and `net4` ports are in `standby` state, which causes the Ethernet stub to report a speed value of zero.

```
gdom# dladm show-aggr -x
LINK   PORT           SPEED DUPLEX   STATE     ADDRESS            PORTSTATE
aggr1  --             0Mb   unknown  down      2:8:20:91:f3:79    --
net3                  0Mb   unknown  up        0:14:4f:fa:cb:a4   standby
net4                  0Mb   unknown  up        0:14:4f:f9:ab:a5   standby
```

Example 13-11 Configuring a DLMP Aggregation Fails When the `custom` Property Is Not Set to `enable`

This example shows that an attempt to create a DLMP aggregation fails if the `custom` property is not set to `enable` on the virtual network devices used for the DLMP aggregation.

The following commands set the `custom` property to `disable` for the `vnet` and `vnet6` virtual network devices in the `gdom` domain:

```
primary# ldm add-vnet custom=disable linkprop=phys-state vnet5 primary-vsw0 gdom
primary# ldm add-vnet custom=disable linkprop=phys-state vnet6 primary-vsw1 gdom
```

The following command attempts to create the `aggr2` DLMP aggregation on the `gdom` domain. The command fails because the `custom` property should be set to `enable` on the virtual network devices.

```
gdom# dladm create-aggr -m dlmp -l net5 -l net6 aggr2
dladm: create operation failed: operation not supported
```

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 11.4*.

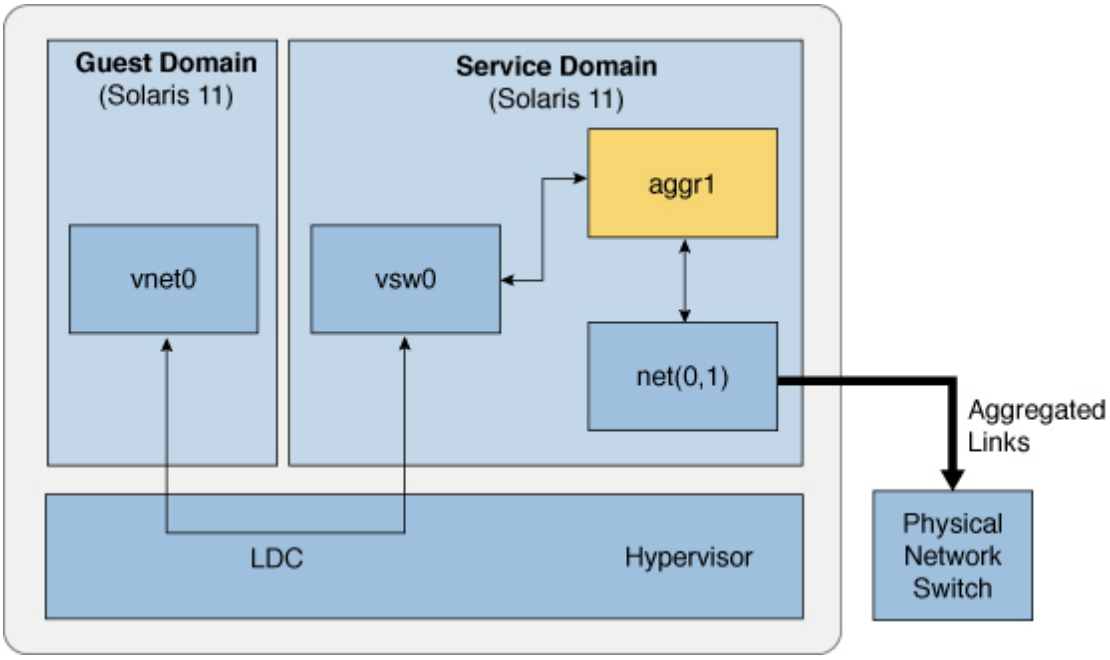
After you create a link aggregation, you can assign it to the virtual switch. Making this assignment is similar to assigning a physical network device to a virtual switch. Use the `ldm add-vswitch` or `ldm set-vswitch` command to set the `net-dev` property.

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.

The figure titled Configuring a Virtual Switch to Use a Link Aggregation (Oracle Solaris 11) shows a virtual switch configured to use an aggregation, `aggr1`, over physical interfaces `net0` and `net1`, and `nxge0` and `nxge1`, respectively.

Configuring a Virtual Switch to Use a Link Aggregation (Oracle Solaris 11)



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`(8) man page. 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 can 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 VLAN IDs 2 through 4096 are valid. VLAN ID 1 is reserved as the `default-vlan-id`.

Use the `ldm add-vnet`, `ldm set-vnet`, `ldm add-vsw`, or `ldm set-vsw` command to specify the `pvid` and `vid` property values.

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 192.169.2.100/24 net0
   # dladm create-vlan -l net0 -v 20 vlan20
   # ipadm create-ip vlan20
   # ipadm create-addr -T static -a 192.168.1.100/24 vlan20
   ```

   For more information about how to configure VLAN interfaces in the Oracle Solaris 11 OS, refer to Chapter 4, Configuring Virtual Networks by Using Virtual Local Area Networks in *Managing Network Datalinks in Oracle Solaris 11.4*.

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.

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 13-1 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 LDC channels feature supports the communication restrictions of isolated and community PVLANs. Inter-vnet LDC channels 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`(8) 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 set-vnet pvid=3 pvlan=300,community vnet2 ldg1
  ```

- Use `ldm set-vnet` to remove a PVLAN:

  ```
  ldm set-vnet pvlan= if-name domain-name
  ```

  The following command removes the PVLAN configuration for the `vnet0` virtual network. To revert the `vnet0` virtual network to a regular VLAN ID, you must remove the PVLAN ID first.

  ```
  primary# ldm set-vnet pvlan= vnet0 ldg1
  ```

Viewing PVLAN Information

You can view information about a PVLAN by using several of the Logical Domains Manager listing subcommands. See the `ldm`(8) 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 the 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:fb:22:79

  NETWORK
  ```

```
     NAME         SERVICE                 MACADDRESS PVID|PVLAN|VIDs
     ----         -------                 ---------- ---------------
     vnet0        primary-vsw0@primary    00:14:4f:f8:6e:d9 2|
300,community|--
             DEVICE           :network@0       ID   :0
             LINKPROP       :--                  MTU  :1500
             MAXBW          :--                 MODE :--
             CUSTOM        :disable
             PRIORITY       :--                    COS :--
             PROTECTION :--
```

- 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 1.19
DOMAIN|name=ldg1|
MAC|mac-addr=00:14:4f:fb:22:79
VNET|name=vnet0|dev=network@0|service=primary-vsw0@primary|mac-
addr=00:14:4f:f8:6e:d9|mode=|pvid=2|vid=|mtu=1500|linkprop=|id=0|alt-mac-
addrs=|maxbw=|pvlan=300,community|protection=|priority=|cos=|
custom=disable|max-mac-addrs=|max-vlans=
```

- 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 -o network ldg1
NAME
ldg1

MAC
     00:14:4f:fb:22:79

NETWORK
     NAME         SERVICE                 MACADDRESS PVID|PVLAN|VIDs
     ----         -------                 ---------- ---------------
     vnet0        primary-vsw0@primary    00:14:4f:f8:6e:d9 2|
300,community|--

         PEER                   MACADDRESS          PVID|PVLAN|VIDs
         ----                   ----------          ---------------
         primary-vsw0@primary   00:14:4f:f9:08:28   1|--|--
```

- The following `ldm list-bindings` command shows PVLAN configuration information in a parseable form for the `ldg1` domain.

```
primary# ldm list-bindings -o network -p ldg1
VERSION 1.19
DOMAIN|name=ldg1|
MAC|mac-addr=00:14:4f:fb:22:79
VNET|name=vnet0|dev=network@0|service=primary-vsw0@primary|mac-
addr=00:14:4f:f8:6e:d9|mode=|pvid=2|vid=|mtu=1500|linkprop=|id=0|alt-mac-
addrs=|maxbw=|pvlan=300,community|protection=|priority=|cos=|
custom=disable|max-mac-addrs=|max-vlans=
|peer=primary-vsw0@primary|mac-addr=00:14:4f:f9:08:28|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`(8) man page.

Example 13-12 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-dhcp-cids=oracle@system1.company.com, \
00:14:4f:fb:22:79,system2,00:14:4f:fb:22:56 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-domain -o network ldg3
NAME
ldg3

MAC
     00:14:4f:f8:5b:12
     NAME         SERVICE                MACADDRESS PVID|PVLAN|VIDs
     ----         -------                ---------- ---------------
     vnet3_ldg3   primary-vsw0@primary   00:14:4f:f8:dd:96 1|--|--
             DEVICE     :network@1       ID   :1
             LINKPROP   :phys-state      MTU  :1500
             MAXBW      :--              MODE :--
             CUSTOM     :disable
             PRIORITY   :high            COS  :7
             PROTECTION :mac-nospoof
                         restricted
                         ip-nospoof
                         [192.168.100.1
                          192.168.100.2]
                        dhcp-nospoof
                        [oracle@system1.company.com
                         00:14:4f:fb:22:79
                         system2
                         00:14:4f:fb:22:56]
```

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.

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
```

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 11.4, see Chapter 1, About Using Rights to Control Users and Processes in *Securing Users and Processes in Oracle Solaris 11.4*.
3. **Determine the value of MTU that you want to use for the virtual network.**

   You can specify an MTU value up to 16000 bytes. The specified MTU must match the MTU of the physical network device that is assigned to the virtual switch.

   Use the `ldm list-netdev -l` command to obtain the MTU value of the physical network device.

   ```
   primary# ldm list-netdev -l -o net0 primary

   DOMAIN
   primary

   NAME CLASS   MEDIA  STATE  SPEED OVER LOC
   ---- -----   -----  -----  ----- ---- ---
   net0 PHYS    ETHER  up     1G    igb0 /SYS/RIO/NET0
        [/pci@400/pci@1/pci@0/pci@2/network@0]
        MTU       : 1500 [60-9216]
        IPADDR    : 10.129.68.118/255.255.255.0
                  : fe80::210:e0ff:fe0e:e0c0/ffc0::
                  : 2606:b400:418:17b2:210:e0ff:fe0e:e0c0/ffff:ffff:ffff:ffff::
        MAC_ADDRS : 00:10:e0:0e:e0:c0
   ```
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 13-13 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 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 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
  ```

Example 13-14 Changing the MTU Value of a Network Interface

The following example shows how to change the MTU value of the network 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
```

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.

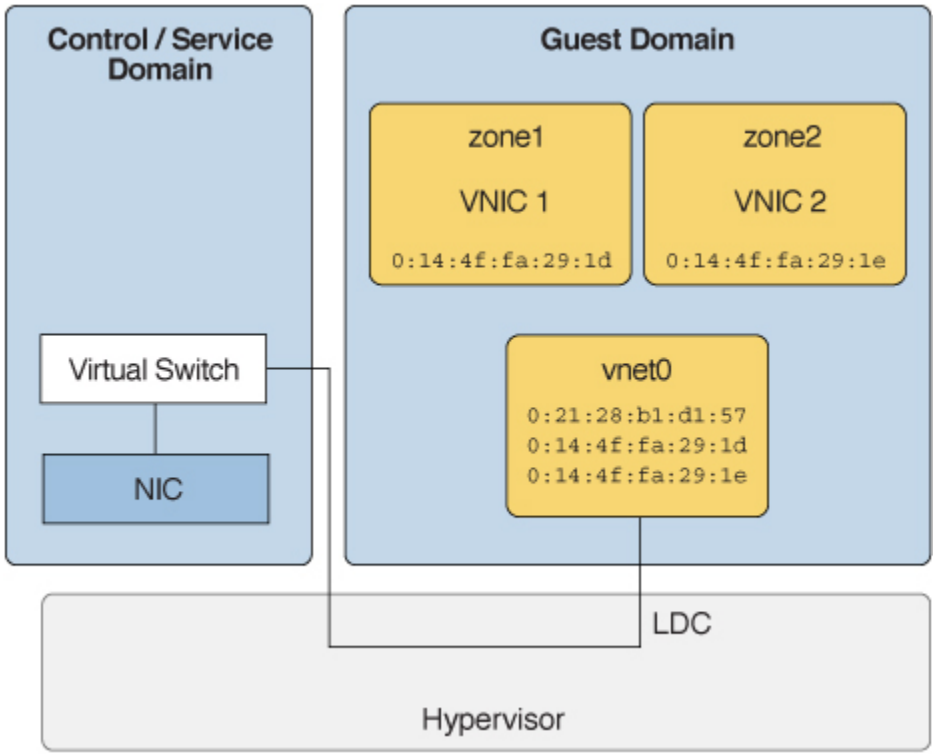
Oracle Solaris 11 improves on the Oracle Solaris 10 “shared IP” zone model in which zones inherit network properties from the global zone and cannot set their own network address or other properties. Now, by using zones with virtual network devices, you can configure multiple isolated virtual NICs, associate 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 11.4 information library (http://docs.oracle.com/cd/E37838_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.

Within the `domain1` domain are Oracle Solaris 11 zones: `zone1` and `zone2`. Each zone is connected to the network by a virtual NIC based on the `vnet0` virtual network device.

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
- Creating Oracle Solaris 11 Zones in a Domain

For information about using virtual NICs on Ethernet SR-IOV virtual functions, see the following sections:

- Creating Ethernet Virtual Functions
- Modifying Ethernet SR-IOV Virtual Functions
- Creating Virtual NICs on SR-IOV Virtual Functions

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 `vnet0` virtual network device on the `domain1` domain and makes three system-generated MAC addresses available to the device.

  ```
  primary# ldm add-vnet auto-alt-mac-addrs=3 vnet0 primary-vsw0 domain1
  primary# ldm add-vnet alt-mac-addrs=auto,auto,auto vnet0 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 vnet0 primary-vsw0 domain1
  primary# ldm set-vnet alt-mac-addrs=auto,auto,auto vnet0 domain1
  primary# ldm set-vnet auto-alt-mac-addrs=3 vnet0 domain1
  ```

Dynamically Updating Alternate MAC Addresses

You can use the `ldm set-vnet` command to perform an update on the alternate MAC address of a virtual network device dynamically. You can make this change when the update increases the total number of alternate MAC addresses of the virtual network device.

Both of the following commands are examples of dynamically adding an alternate MAC address to the `vnet1` virtual network device on the `ldg1` domain:

```
primary# ldm set-vnet alt-mac-addrs=+auto vnet1 ldg1

primary# ldm set-vnet auto-alt-mac-addrs=+1 vnet1 ldg1
```

The following `ldm list` output shows the MAC addresses that are associated with the `vnet0` virtual network device on the `ldg1` domain. `00:14:4f:f9:8a:c2` is the primary MAC address for `vnet0` and `00:14:4f:f8:1c:a5` and `00:14:4f:f8:2c:22` are its two alternate MAC addresses.

```
primary# ldm list -o network ldg1
NETWORK
    NAME    SERVICE              MACADDRESS        PVID|PVLAN|VIDs
    ----    -------              ----------        ---------------
    vnet0   primary-vsw0@primary 00:14:4f:f9:8a:c2 1|--|--
                                 00:14:4f:f8:1c:a5
                                 00:14:4f:f8:2c:22
```

If you log in to the `ldg1` domain, you can use the `dladm show-phys -m` command to view the MAC addresses that are associated with the `net0` network device.

```
ldg1# dladm show-phys -m
LINK                SLOT     ADDRESS            INUSE CLIENT
net0                primary  0:14:4f:f9:8a:c2   yes   net0
                    1        0:14:4f:f8:1c:a5   no    --
                    2        0:14:4f:f8:2c:22   no    --
```

The `dladm show-vnic` command shows the alternate MAC address (`00:14:4f:f8:2c:22`) that is used to configure the virtual NIC:

```
ldg1# dladm show-vnic
LINK            OVER           SPEED  MACADDRESS        MACADDRTYPE IDS
vnic1           net0           0      0:14:4f:f8:2c:22  fixed       VID:0
```

While you can use the `ldm set-vnet` command to increase the number of alternate MAC addresses dynamically, you cannot update or remove existing alternate MAC addresses dynamically. If you modify or remove an alternate MAC address that is in use, the VNICs are left in an unusable state.

The following examples show the error you receive when attempting to dynamically remove or modify an existing alternate MAC addresses.

- The following example shows that attempting to dynamically remove the `00:15:4f:f9:41:c4` alternate MAC address from `vnet2` on the `ldg1` domain fails with an error:

  ```
  primary# ldm set-vnet alt-mac-addrs=-00:15:4f:f9:41:c4 vnet2 ldg1
  Please perform the operation while the LDom is bound or inactive
  ```

- The following example shows that attempting to modify an existing alternate MAC address with the `auto` value for `vnet1` dynamically fails with an error:

  ```
  primary# ldm set-vnet alt-mac-addrs=auto vnet1 ldg1
  Please perform the operation while the LDom is bound or inactive
  ```

Creating Oracle Solaris 11 Zones in a Domain

After creating the virtual NICs in Configuring Virtual NICs on Virtual Network Devices, 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.

Using Trusted Virtual Networks

The trusted virtual network feature extends privileges to trusted guest domains to assign custom alternate MAC addresses and alternate VLAN IDs to the `vnet` device dynamically. These MAC addresses and VLAN IDs are used to configure virtual devices. Prior to the introduction of this feature, you could make such assignments only from the Logical Domains Manager. Moreover, the alternate MAC addresses assignment also required that the domain hosting the virtual network device be in the bound state. This feature enables the dynamic creation of virtual devices such as VNICs and VLANs on top of virtual network devices.

To use the trusted virtual network feature on a `vnet` device, you must create or configure the device in trusted mode by using the Logical Domains Manager. By default, a `vnet` device is created with trusted mode disabled.

The trusted virtual network feature seamlessly supports the live migration, service domain reboot, and multiple service domain features.

Trusted Virtual Network Requirements and Restrictions

You can configure a trusted virtual network by using the `ldm add-vnet` and `ldm set-vnet` commands to set the `custom=enable` property. Note that you should provide values for the `custom/max-mac-addrs` and `custom/max-vlans` properties to ensure that the number of custom MAC addresses and VLAN are limited for the specified virtual network device. Both property values are set to 4096 by default.

The trusted virtual network feature requires at least the Oracle Solaris 11.3 SRU 8 OS.

Both guest domain that has the custom virtual network device and the service domain that has the corresponding virtual switch device require that latest level of the supported system firmware.

To configure a trusted virtual network, you must specify the following information:

- `custom` – Enable or disable the trusted virtual network feature. This feature enables a trusted entity to add custom alternate VLAN IDs and custom alternate MAC addresses dynamically.
- `custom/max-mac-addrs` – Specify the maximum number of custom alternate MAC addresses to be configured on a particular trusted virtual network device.
- `custom/max-vlans` – Specify the maximum number of custom alternate VLAN IDs to be configured on a particular trusted virtual network device.

The following restrictions are for the trusted virtual network feature:

- You cannot use the Logical Domains Manager to configure alternate MAC addresses or VLAN IDs on a given trusted virtual network.
- To modify custom or existing alternate MAC addresses, the domain must be in the bound state.
- You can increase the `custom/max-mac-addrs` and `custom/max-vlans` property values dynamically. However, the domain must be in the bound state to reduce these property values.

> **Note:**
>
> Reducing these property values might cause undesirable side effects. So, ensure that you delete any of the VNICs or VLANs created on the host that you do not need because you have no control over which MAC addresses or VLAN IDs the OS will retain. Also, set `custom=disable` on the virtual network device before using the `ldm set-vnet` command to reduce the number of maximum VLAN IDs and MAC addresses for the custom virtual network device.

> **Caution:**
>
> The effective use of this feature is to limit and control these properties.

- Ensure that any VNIC and VLAN devices that have been created are removed before you reduce the number of custom VLAN IDs or custom alternate MAC addresses. Otherwise,

the guest domain will have VNICs that cannot be configured and must be removed manually.

- The `dladm show-vnic -m` command shows the MAC addresses and VLAN IDs that are configured on the specified virtual network. The `dladm show-vnic -m` command shows the alternate MAC addresses and VLAN IDs in use on the guest domain. This is a departure from older releases where in all alternate MAC addresses and VLAN IDs were preconfigured on the virtual switch.
- The trusted virtual network feature is mutually exclusive with the PVLAN feature.
- The Logical Domains Manager attempts to validate the guest domain and service domain support for this feature before enabling the custom feature. If the guest domain is not running, you can enable this feature if the service domain supports it. However, if the guest domain does not support the feature you must set `custom=disabled` before you re-enable non-custom alternate MAC addresses and VLAN IDs.
- You can perform a live migration of a domain with trusted virtual networks only if the target service domain supports the trusted virtual network feature.

Configuring Trusted Virtual Networks

This section includes tasks that show how to create trusted virtual networks and how to obtain information about trusted virtual networks.

You can configure a trusted virtual network by setting the `custom` property value by using the `ldm add-vnet` or `ldm set-vnet` command. See the `ldm`(8) man page.

Example 13-15 Creating a Trusted Virtual Network

You can use the following commands to create a trusted virtual network `ldg1_vnet0` on the `primary-vsw0` virtual switch in the `ldg1` domain. The `custom/max-mac-addrs` and `custom/max-vlans` property values use the default values of 4096.

```
primary# ldm add-vnet custom=enable ldg1_vnet0 primary-vsw0 ldg1
primary# ldm list -o network ldg1
...
NETWORK
    NAME         SERVICE                 MACADDRESS          PVID|PVLAN|VIDs
    ----         -------                 ----------          ---------------
    ldg1-vnet0   primary-vsw0@primary    00:14:4f:fa:d7:5e   1|--|--
            DEVICE     :network@1        ID   :1
            LINKPROP   :phys-state       MTU  :1500
            MAXBW      :--               MODE :--
CUSTOM     :enable
            MAX-CUSTOM-MACS:4096         MAX-CUSTOM-VLANS:4096
            PRIORITY   :--               COS  :--
            PROTECTION :--
```

Example 13-16 Enabling the Trusted Virtual Network Feature on an Existing Virtual Network

The following example shows how to enable the trusted virtual network feature by setting `custom=enable` for the `ldg1_vnet0` virtual network device in the `ldg1` domain. The `custom/max-mac-addrs` and `custom/max-vlans` property values use the default values of 4096.

```
primary# ldm set-vnet custom=enabled ldg1_vnet0 ldg1
primary# ldm list -o network ldg1
```

```
...
NETWORK
    NAME         SERVICE                 MACADDRESS          PVID|PVLAN|VIDs
    ----         -------                 ----------          ---------------
    ldg1-vnet0   primary-vsw0@primary    00:14:4f:fa:d7:5e   1|--|--
            DEVICE     :network@1        ID   :1
            LINKPROP   :phys-state       MTU  :1500
            MAXBW      :--               MODE :--
CUSTOM     :enable
            MAX-CUSTOM-MACS:4096         MAX-CUSTOM-VLANS:4096
            PRIORITY   :--               COS  :--
            PROTECTION :--
```

Example 13-17 Setting the `custom/max-mac-addrs` and `custom/max-vlans` Properties

The following example sets the `custom/max-vlans` property value to 12 and the `custom/max-mac-addrs` property value to 13.

Because these new property values are lower than the previous values, you cannot change these settings dynamically. You can make these changes only to a bound or inactive domain.

```
primary# ldm stop ldg1
primary# ldm set-vnet custom/max-vlans=12 custom/max-mac-addrs=13 ldg1_vnet0 ldg1
primary# ldm list -o network ldg1
...
NETWORK
    NAME         SERVICE                 MACADDRESS          PVID|PVLAN|VIDs
    ----         -------                 ----------          ---------------
    ldg1-vnet0   primary-vsw0@primary    00:14:4f:fa:d7:5e   1|--|--
            DEVICE     :network@1        ID   :1
            LINKPROP   :phys-state       MTU  :1500
            MAXBW      :--               MODE :--
CUSTOM     :enable
            MAX-CUSTOM-MACS:13           MAX-CUSTOM-VLANS:12
            PRIORITY   :--               COS  :--
            PROTECTION :--
```

Example 13-18 Resetting the `custom/max-mac-addrs` and `custom/max-vlans` Properties

The following example shows how to reset the `custom/max-mac-addrs` property value to its default of 4096 by specifying a null value.

When `custom=enabled`, you can reset the `custom/max-vlans` property value, the `custom/max-mac-addrs` property value, or both.

```
primary# ldm set-vnet custom/max-mac-addrs= ldg1_vnet0 ldg1
primary# ldm list -o network ldg1
...
NETWORK
    NAME         SERVICE                 MACADDRESS          PVID|PVLAN|VIDs
    ----         -------                 ----------          ---------------
    ldg1-vnet0   primary-vsw0@primary    00:14:4f:fa:d7:5e   1|--|--
            DEVICE     :network@1        ID   :1
            LINKPROP   :phys-state       MTU  :1500
            MAXBW      :--               MODE :--
            CUSTOM     :enable
MAX-CUSTOM-MACS:4096          MAX-CUSTOM-VLANS:12
            PRIORITY   :--               COS  :--
            PROTECTION :--
```

Example 13-19 Changing the `custom/max-mac-addrs` and `custom/max-vlans` Property Values

The following example shows how to increase the `custom/max-vlans` property value and decrease the `custom/max-mac-addrs` property value. You can increase the `custom/max-vlans` property value to 24 dynamically, because 24 is larger than the previous value of 12. However, because you are reducing the maximum value for `custom/max-mac-addrs` from 4096 to 11, you must first stop the domain.

```
primary# ldm set-vnet custom/max-vlans=24 ldg1_vnet0 ldg1
primary# ldm stop ldg1
primary# ldm set-vnet custom/max-mac-addrs=11 ldg1_vnet0 ldg1
primary# ldm list -o network ldg1
...
NETWORK
    NAME         SERVICE                 MACADDRESS          PVID|PVLAN|VIDs
    ----         -------                 ----------          ---------------
    ldg1-vnet0   primary-vsw0@primary    00:14:4f:fa:d7:5e   1|--|--
            DEVICE     :network@1        ID   :1
            LINKPROP   :phys-state       MTU  :1500
            MAXBW      :--               MODE :--
            CUSTOM     :enable
MAX-CUSTOM-MACS:11          MAX-CUSTOM-VLANS:24
            PRIORITY   :--               COS  :--
            PROTECTION :--
```

Example 13-20 Disabling the Trusted Virtual Network Feature

The following example shows how to disable the `custom` property for the `ldg1_vnet0` virtual network device in the `ldg1` domain.

```
primary# ldm set-vnet custom=disabled ldg1_vnet0 ldg1
...
NETWORK
    NAME         SERVICE                 MACADDRESS          PVID|PVLAN|VIDs
    ----         -------                 ----------          ---------------
    ldg1-vnet0   primary-vsw0@primary    00:14:4f:fa:d7:5e   1|--|--
            DEVICE     :network@1        ID   :1
            LINKPROP   :phys-state       MTU  :1500
            MAXBW      :--               MODE :--
CUSTOM     :disable
            PRIORITY   :--               COS  :--
            PROTECTION :--
```

Viewing Trusted Virtual Network Information

You can obtain information about trusted virtual network settings by using several of the Logical Domains Manager list subcommands. See the `ldm`(8) man page.

The following examples use the `ldm list-domain -o network`, `ldm list-bindings`, and `ldm list-constraints` commands to show information about a trusted virtual network configuration.

- The following example shows how to use the `ldm list-domain` command to view trusted virtual network configuration information for the `ldg1` domain:

  ```
  primary# ldm list-domain -o network ldg1
  ...
  NETWORK
      NAME         SERVICE                 MACADDRESS          PVID|PVLAN|VIDs
  ```

```
    ----         -------                ----------          ---------------
    ldg1-vnet0   primary-vsw0@primary   00:14:4f:fa:d7:5e   1|--|--
           DEVICE     :network@1        ID   :1
           LINKPROP   :phys-state       MTU  :1500
           MAXBW      :--               MODE :--
           CUSTOM     :enable
           MAX-CUSTOM-MACS:11           MAX-CUSTOM-VLANS:24
           PRIORITY   :--               COS  :--
           PROTECTION :--
```

- The following examples shows how to use the `ldm list-domain` command to view trusted virtual network configuration information in a parseable form for the `ldg1` domain:

```
primary# ldm list-domain -o network -p ldg1
VERSION 1.19
DOMAIN|name=ldg1|
MAC|mac-addr=00:14:4f:f9:4b:d0
VNET|name=ldg1-vnet0|dev=network@1|service=primary-vsw0@primary|mac-
addr=00:14:4f:fa:d7:5e|mode=|pvid=1|vid=|mtu=1500|linkprop=phys-state|id=1|alt-mac-
addrs=|maxbw=|pvlan=|protection=|priority=|cos=|custom=enable|max-mac-addrs=11|max-
vlans=24
```

- The following examples shows how to use the `ldm list-bindings` command to view trusted virtual network configuration information for the `ldg1` domain:

```
primary# ldm list-bindings -e -o network ldg1
...
NETWORK
    NAME         SERVICE                MACADDRESS          PVID|PVLAN|VIDs
    ----         -------                ----------          ---------------
    ldg1-vnet0   primary-vsw0@primary   00:14:4f:fa:d7:5e   1|--|--
           DEVICE     :network@1        ID   :1
           LINKPROP   :phys-state       MTU  :1500
           MAXBW      :--               MODE :--
           CUSTOM     :enable
           MAX-CUSTOM-MACS:11           MAX-CUSTOM-VLANS:24
           PRIORITY   :--               COS  :--
           PROTECTION :--

        PEER                   MACADDRESS          PVID|PVLAN|VIDs
        ----                   ----------          ---------------
        primary-vsw0@primary   00:14:4f:f9:08:28   1|--|--
            LINKPROP   :--               MTU  :1500
            MAXBW      :--               LDC  :0x5
            MODE       :--
```

- The following examples shows how to use the `ldm list-bindings` command to view trusted virtual network configuration information in a parseable form for the `ldg1` domain:

```
primary# ldm list-bindings -p ldg1
...
VNET|name=ldg1-vnet0|dev=network@1|service=primary-vsw0@primary|mac-
addr=00:14:4f:fa:d7:5e|mode=|pvid=1|vid=|mtu=1500|linkprop=phys-state|id=1|alt-mac-
addrs=|maxbw=|pvlan=|protection=|priority=|cos=|custom=enable|max-mac-addrs=11|max-
vlans=24
|peer=primary-vsw0@primary|mac-addr=00:14:4f:f9:08:28|mode=|pvid=1|vid=|mtu=1500|
maxbw=
```

- The following example shows how to generate XML by running the `ldm list-constraints -x` 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">ldg1-vnet0</gprop:GenericProperty>
    <gprop:GenericProperty key="service_name">primary-vsw0</
gprop:GenericProperty>
    <gprop:GenericProperty key="pvid">1</gprop:GenericProperty>
    <gprop:GenericProperty key="linkprop">phys-state</gprop:GenericProperty>
    <gprop:GenericProperty key="custom">enable</gprop:GenericProperty>
    <gprop:GenericProperty key="max-mac-addrs">11</gprop:GenericProperty>
    <gprop:GenericProperty key="max-vlans">24</gprop:GenericProperty>
    <gprop:GenericProperty key="device">network@1</gprop:GenericProperty>
    <gprop:GenericProperty key="id">1</gprop:GenericProperty>
  </Item>
</Section>
```

Using a Virtual Switch Relay

A virtual switch relay forces all domain traffic through an external physical network infrastructure. This configuration enforces network policies for access control lists (ACL) and packet monitoring, which can be configured on the external switch. The switch must support the virtual switch relay capability. Use the `ldm set-vsw` command to specify the `vsw-relay-mode` property value to one of the following values:

- `local` enables the network traffic between domains over the same physical NIC to be exchanged internally. This is the default value.
- `remote` enables the network traffic between domains over the same physical NIC to be exchanged through the external switch. This `remote` value requires that you manually disable the `inter-vnet-link` property on all the virtual networks that are connected to this switch.

  When resetting the value to `local`, you can reset the `inter-vnet-link` property value to `on` or `auto` depending on your needs.

> **Note:**
>
> You can set the value to `remote` on an Ethernet device only.

How to Set the Virtual Switch Mode to Remote

1. **Determine whether the `inter-vnet-link` property is set to `off`.**

   ```
   primary# ldm list -e domain-name
   VSW

       NAME         MACADDRESS        NET-DEV  DVID|PVID|VIDs
       ----         ----------        -------  --------------
       primary-vsw0 00:14:4f:fb:86:af net0     1|1|--
               ....
               INTER-VNET-LINK :on/auto              MODE :--
               VSW-RELAY-MODE  :local
   ```

2. **Disable inter-vnet LDC channels if the `inter-vnet-link` value is `auto` or `on`.**

```
primary# ldm set-vsw inter-vnet-link=off primary-vsw0
```

3. **Set the `vsw-relay-mode` property value to `remote`.**

```
primary# ldm set-vsw vsw-relay-mode=remote primary-vsw0
```

4. **Verify that the `vsw-relay-mode` property is set to `remote`.**

```
primary# ldm list -e domain-name
VSW

    NAME         MACADDRESS          NET-DEV   DVID|PVID|VIDs
    ----         ----------          -------   --------------
    primary-vsw0 00:14:4f:fb:86:af   net0      1|1|--
            ....
            INTER-VNET-LINK :off             MODE :--
            VSW-RELAY-MODE  :remote
```

Virtual Switch Relay Failure Cases

- The following command fails because you must disable `inter-vnet-link` before you set `vsw-relay-mode=remote`:

```
primary# ldm set-vsw vsw-relay-mode=remote primary-vsw0
Vswitch primary-vsw1 vsw-relay-mode is set to remote,
inter-vnet-link should be off.
```

- The following command verifies that the `linkprop` property is set correctly at the NIC level:

```
primary# dladm show-linkprop -p vswitchmode net0
LINK PROPERTY    PERM VALUE  EFFECTIVE DEFAULT POSSIBLE
net0 vswitchmode rw   remote remote    local   local,remote,auto
```

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:

- **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.

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.

- **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.

14
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 3.6 Release Notes* and related versions of the administration guide.

Introduction to Domain Migration

Domain migration enables you to move 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.

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 Migrating and Renaming a Guest Domain. 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.
- **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.
- **FIPS 140-2.** The Logical Domains Manager respects the Oracle Solaris FIPS 140-2 system configuration when performing domain migrations. See *Using a FIPS 140-2 Enabled System in Oracle Solaris 11.4*.
- **Host Name Matching Semantics.** The Oracle Solaris 11.4 SRU 48 OS introduces the `ldmd/tls_host_match` SMF property to control the strictness of host name and IP address matching semantics when validating SSL certificates:
  The default property value is `false`, which disables the stricter matching. To enable strict host name and IP address matching of the specified migration target against the target's certificate, set the property value to `true`. Then, refresh and restart the `ldmd` SMF service.

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 the source machine.

1. **Create the `/var/share/ldomsmanager/trust` directory if it does not already exist.**

```
source:primary# mkdir /var/share/ldomsmanager/trust
```

2. **Copy the `ldmd` certificate from the target server to the local 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`. Rename the remote certificate file *target-hostname* `.pem`, for example `tgt-primary.pem`.

3. **Create a symbolic link from the certificate in the trusted certificate directory to the `/etc/certs/CA` directory.**

   ```
   source:primary# ln -s /var/share/ldomsmanager/trust/tgt-primary.pem /etc/
   certs/CA/
   ```

4. **Restart the `svc:/system/ca-certificates` service.**

   ```
   source:primary# svcadm restart svc:/system/ca-certificates
   ```

5. **Verify that the symbolic links to `/etc/certs/CA/` that you created in Step 3 are correct.**

   ```
   source:primary# openssl verify /var/share/ldomsmanager/trust/tgt-primary.pem
   /var/share/ldomsmanager/trust/tgt-primary.pem: ok
   ```

6. **Verify that the `ca-certificates` service is online.**

   Restart or enable the service if required.

   ```
   source:primary# svcs ca-certificates
   /var/share/ldomsmanager/trust/tgt-primary.pem: ok
   STATE          STIME    FMRI
   online         0:22:38  svc:/system/ca-certificates:default
   ```

7. **Restart the `ldmd` daemon.**

   ```
   source:primary# svcadm restart ldmd
   ```

8. **Starting with Oracle Solaris 11.4 SRU 48, verify that the configuration of the `ldmd` certificates is correct.**

   ```
   source:primary# openssl verify -CApath /var/opt/SUNWldm/CA /var/opt/SUNWldm/
   trust/tgt-primary.pem
   /var/share/ldomsmanager/trust/tgt-primary.pem: ok
   ```

9. **Repeat these steps on the target server.**

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
```

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 System Firmware Versions in *Oracle VM Server for SPARC 3.6 Installation Guide*.

Cross-CPU Restrictions for Migration

You cannot perform live migration operations between a server that runs system firmware version 7.*x* and a server that runs system firmware version 8.*x*.

Therefore, the following live migration operations are not supported:

- From a server that runs version 7.*x* of the system firmware to a server that runs version 8.*x* of the system firmware
- From a server that runs version 8.*x* of the system firmware to a server that runs version 7.*x* of the system firmware

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 and the `ldm`(8) man page.

Forced Cross-CPU Migration Can Fail if Global Performance Counters are Enabled

In certain cross-CPU migration scenarios, a migration can fail with the following errors when you force the migration by using the `-f` option:

```
API group 0x20b v1.0 is not supported in the version of the firmware
 running on the target machine.
API group 0x214 v1.0 is not supported in the version of the firmware
 running on the target machine.
```

All of the following conditions must be present to encounter this problem:

- The domain has the `cpu-arch` property set to `generic` or `migration-class1`
- The domain has a `perf-counter` property setting that includes the `global` value
- The domain was booted on at least a SPARC M7 series server or a SPARC T7 series server
- The target machine is a platform released prior to the SPARC M7 series server or SPARC T7 series server

This problem occurs because a domain booted on at least a SPARC M7 series server or a SPARC T7 series server with a `perf-counter` property setting that includes the `global` value will register platform-specific performance counter Hypervisor interfaces that do not exist on older platforms. As part of the migration, a check is performed to ensure that all the interfaces used by the domain are present on the target machine. When these SPARC M7 series server or SPARC T7 series server specific interfaces are detected, the migration is aborted.

Do not set `perf-counter=global` if `cpu-arch` is note `native` and at least SPARC M7 series servers and SPARC T7 series servers are part of the migration pool.

Migration Restrictions for Setting `linkprop=phys-state`

You can migrate a virtual network device that has a physical NIC backing device and has `linkprop=phys-state` to a target domain that does not have a physical NIC as a backing device (`net-dev=`). Because the `linkprop=phys-state` constraint is not a hard requirement, if such a domain is migrated to a machine that does not have an available `net-dev` value, the constraint is preserved but not fulfilled. The `linkprop` property is still preserved as `phys-state` and the network device link state shows as link up.

Migration Restrictions for Domains That Have a Large Number of Virtual Devices

Sometimes, migrating a domain that has a large number of virtual devices causes the control domain on the target machine to be less responsive than usual. During this time, `ldm` commands appear to hang and standard Oracle Solaris OS commands take longer to complete than usual.

This interruption is caused by virtual servers processing the large number of incoming virtual devices associated with the migrated domain. After this processing completes, the control domain returns to normal and any stalled `ldm` commands complete.

You can minimize this sort of interruption by limiting the number of virtual devices used by a domain to no more than 1000.

Migration Restrictions for Silicon Secured Memory Servers

SPARC servers starting with the SPARC M7, SPARC T7, and SPARC S7 series server support a hardware capability called Silicon Secured Memory (SSM). When enabled through the Application Data Integrity (ADI) API, this feature enables software to assign versions to regions of memory. These versions are used to detect invalid or unauthorized memory accesses. These version tags are part of the guest virtual machine state and are migrated with the guest domain.

For more information about SSM and the ADI API, see *Software in Silicon: Enabling Secure Clouds for the Real-Time Enterprise* (http://www.oracle.com/technetwork/server-storage/sun-sparc-enterprise/documentation/sparc-t7-m7-server-architecture-2702877.pdf) and Using Application Data Integrity (ADI) in *Oracle Solaris 11.4 Programming Interfaces Guide*.

You can perform only a migration that preserves ADI tags between servers of the same CPU family. For example, you can perform a migration that preserves ADI tags from a SPARC M7 series server to another SPARC M7 series server or to a SPARC T7 series server. Or, you can perform a migration that preserves ADI tags from a SPARC T8 series server to another SPARC T8 series server or to a SPARC M8 series server.

> **Caution:**
>
> If you intend to perform a domain migration on your servers that support SSM, ensure that they run at least the Oracle VM Server for SPARC 3.5 software. When running previous versions of the Oracle VM Server for SPARC software on servers that support SSM, ADI version information is not migrated to the target machine. This situation can result in undefined application behavior if ADI is in use in the domain being migrated.

While it is best to run the latest version of the Oracle VM Server for SPARC software on all SSM-capable machines used in a migration, if an upgrade is not possible on either the source machine or the target machine, you can still perform a migration if you are certain that the domain to be migrated does not use ADI versioning.

To perform this sort of migration, set the `ldmd/migration_adi_legacy_compat` SMF property value to `true` on the machine that runs Oracle VM Server for SPARC 3.5. By setting this

property, the migration overrides the support checks and permits the migration to proceed. The migrated domain does not retain the ADI version tags.

Migration Restrictions for Running `cputrack` During a Migration

If the `cputrack` command is run on a guest domain while that domain is migrated to a SPARC T4 server, the guest domain might panic on the target machine after it has been migrated. So, do not run the `cputrack` command during the migration of a guest domain to a SPARC T4 server.

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. For information about migrating a bound or inactive domain, see Migrating Bound or Inactive Domains.

For information about the migration options and operands, see the `ldm`(8) man page.

> **Note:**
>
> After a domain migration completes, save a new SP 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.

Starting with Oracle VM Server for SPARC 3.5, an SP configuration can be saved automatically following a successful migration. For more information, see Saving Post-Migration SP Configurations Automatically.

> **Note:**
>
> In rare circumstances, a successful domain migration reports the following error:
>
> `Unable to send suspend request to domain` *`domain-name`*
>
> This issue occurs when the Logical Domains Manager detects an error while suspending the domain, and the Logical Domains Manager is able to recover and completes the migration. The exit status of the command is 0, reflecting the successful migration. Because the migration completes successfully, you can ignore the error message.

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.

> **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 Time-Related Services Key Tasks in *Introduction to Oracle Solaris 11.4 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 share the same CPU characteristics, such as CPUs that share the same processor core. `native` is the default value.
  - `migration-class1` is a cross-CPU migration family for SPARC platforms starting with the SPARC T4, SPARC M5, and SPARC S7 series server. These platforms support hardware cryptography during and after these migrations so that there is a lower bound to the supported CPUs.

Starting with the Oracle VM Server for SPARC 3.6 software, the `migration-class1` definition no longer includes support for a 2-Gbyte page size because this page size is not available on SPARC M8 and SPARC T8 series servers.

So, any migration that uses `migration-class1` on a source machine that runs software prior to Oracle VM Server for SPARC 3.6 is blocked if the target machine is a SPARC M8 or SPARC T8 series server that runs at least the Oracle VM Server for SPARC 3.6 software. If the target machine is not a SPARC M8 or SPARC T8 series server, the migration succeeds and the domain continues to have access to 2-Gbyte pages until any subsequent reboot. As part of this post-migration reboot, the domain inherits the new `migration-class1` definition and loses access to 2-Gbyte pages.

This value is not compatible with Fujitsu SPARC M12 platforms or Fujitsu M10 platforms.

- `migration-class2` is a cross-CPU migration family for SPARC T7, SPARC M7, SPARC S7, SPARC T8 and SPARC M8 series servers. These platforms support 16-Gbyte page sizes and the DAX co-processor, which this migration class preserves.

  This value is not compatible with Fujitsu SPARC M12 servers or Fujitsu M10 servers.

- `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 SPARC M12 servers or 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, SPARC M5, and SPARC S7 series servers, 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, but in all cases Oracle VM Server for SPARC must take into account memory page sizes during a migration operation. In particular, two page sizes are used when laying out the memory of a migrating domain on the target machine:

- **Hardware largest page size –** The largest page size that is supported by the hardware platform.
- **Effective largest page size –** The largest page size that is available for the domain to use. This page size is always less than or equal to the hardware largest page size.

The real address and physical address alignments are relative to the hardware largest supported page size and must be preserved for each memory block in the migrated domain.

For a guest domain that runs at least the Oracle Solaris 11.3 OS, the migrated domain's memory blocks might be split automatically during the migration so that the migrated domain can fit into smaller available free memory blocks. Memory blocks can only be split on boundaries that are aligned with the effective 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.

Use the `ldm list-domain -o domain` command to determine the hardware largest page size that is supported by the target machine and the effective largest page size that is supported by the domain.

The following example shows the read-only `effective-max-pagesize` and `hardware-max-pagesize` property values. The `effective-max-pagesize` property value is for the `ldg1` domain. The `hardware-max-pagesize` is for the platform.

```
primary# prtdiag|head -n 1
System Configuration:  Oracle Corporation  sun4v SPARC T7-2
primary# ldm list-domain -o domain ldg1 | grep pagesize
effective-max-pagesize=2G
hardware-max-pagesize=16G
```

Migration Requirements for Physical I/O Devices

Except for SR-IOV Ethernet virtual functions, domains that have direct access to physical devices cannot be migrated. However, virtual devices that are associated with physical devices can be migrated.

For information, see Migrating a Domain That Has an SR-IOV Ethernet Virtual Function Assigned.

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.

You cannot perform a domain migration on an I/O domain that is configured with PCIe SR-IOV Fibre Channel and InfiniBand virtual functions.

For information about the SR-IOV feature, see Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions.

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. Starting with Oracle VM Server for SPARC 3.5, explicit console constraints are preserved during the migration. Otherwise, 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.

Migrating While a Delayed Reconfiguration Is Active

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-domain 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-domain ldg1 system2
Migration is not supported while the domain ldg1 is in the 'Solaris debugging' state
Domain Migration of LDom ldg1 failed
```

Migrating a Domain That Uses Named Resources

> **Caution:**
>
> Do *not* assign named resources unless you are an expert administrator.

You can migrate a domain that is configured to use named resources by specifying the cores and memory ranges on the target machine to be used by the migrating domain. To migrate such a domain, ensure that the domain is in the `native` migration class and that it has the `whole-core` constraint applied.

The `ldm migrate-domain` command uses the `cidmap` and `mblockmap` properties to specify physical resource mappings between the source machine and the target machine.

```
ldm migrate-domain -c domain-name cidmap=core-ID:core-ID[,core-ID:core-ID,...] \
mblockmap=phys-addr:phys-addr[,phys-addr:phys-addr,...] target-machine
```

In the following example, the `ldm migrate-domain` command migrates the `ldg1` domain from the `system1` machine to the `system2` machine. The `ldg1` domain has named cores `8` and `9` and a named memory block at physical address `0x400000000`. The domain is migrated to the `system2` machine and will use cores `16` and `17` and a memory block at physical address `0xc00000000`:

```
system1:primary# ldm migrate-domain -c ldg1 cidmap=8:16,9:17 \
mblockmap=0x400000000:c00000000 system2
```

Ensure that the `cidmap` property specifies free, non-duplicate cores on the target machine and that the `mblockmap` property specifies free, non-overlapping physical address ranges on the target machine. The physical address ranges must meet the migration requirements for target machine memory. See Migration Requirements for Memory.

If you omit the `cidmap` and `mblockmap` properties from the `ldm migrate-domain` command, each core ID on the source machine is mapped to the same core ID on the target machine and each physical address range on the source machine is mapped to the same physical address range on the target machine. Thus, the following command migrates the `ldg1` domain to the `system2` machine and the migrated domain uses cores `8` and `9` and a memory block at physical address `0x400000000`:

```
system1:primary# ldm migrate-domain -c ldg1 system2
```

Migrating a Domain That Uses Kernel Zones

On a SPARC server, a running kernel zone within a guest domain blocks live migration of the domain with the following error message:

```
Guest suspension failed because Kernel Zones are active.
Stop Kernel Zones and retry.
```

Stop or suspend the running kernel zone prior to migrating the kernel zone:

- Stop running the kernel zone.

  ```
  # zoneadm -z zonename shutdown
  ```

- Suspend the kernel zone.

  ```
  # zoneadm -z zonename suspend
  ```

Then, perform a live migration of the kernel zone to another system before migrating the guest domain.

See Chapter 5, Migrating an Oracle Solaris Kernel Zone in *Creating and Using Oracle Solaris Kernel Zones*.

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 T4 to SPARC T7 platforms, Fujitsu SPARC M12 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 SP 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.

Starting with Oracle VM Server for SPARC 3.5, an SP configuration can be saved automatically following a successful migration. For more information, see Saving Post-Migration SP Configurations Automatically.

> **Caution:**
>
> When cold migrating a bound domain that has a large number of virtual devices, the operation might fail with the following message in the SMF log:
>
> ```
> warning: Timer expired: Failed to read feasibility response type (9) from
> target LDoms Manager
> ```
>
> This failure indicates that the Logical Domains Manager running on the source machine timed out while waiting for the domain to be bound on the target machine. The chances of encountering this problem increases as the number of virtual devices in the migrating domain increases. The timing of this failure results in a bound copy of the domain on both the source machine and the target machine. Do not start both copies of this domain. This action can cause data corruption because both domains reference the same virtual disk backends. After verifying that the copy of the migrated domain is correct on the target machine, manually unbind the copy of the domain on the source machine and destroy it.

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.

Migration Requirements for PCIe SR-IOV Virtual Functions

Except for SR-IOV Ethernet 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 Creating an I/O Domain by Using PCIe SR-IOV Virtual Functions.

For information about migrating a domain that has SR-IOV Ethernet virtual functions, see Migrating a Domain That Has an SR-IOV Ethernet Virtual Function Assigned.

Migrating a Domain That Has an SR-IOV Ethernet Virtual Function Assigned

> **Note:**
>
> While you can migrate a domain that has an SR-IOV Ethernet virtual function, you cannot migrate a domain that has SR-IOV Fibre Channel virtual functions or SR-IOV InfiniBand virtual functions.

Before you can migrate a domain that has an SR-IOV Ethernet virtual function, you must configure the guest domain on the source machine and make preparations on the target machine to ensure that the migration succeeds.

- **Source machine.** When you create an SR-IOV Ethernet virtual function or have one assigned to a guest domain, ensure that the virtual function has a user-assigned virtual function name by specifying it as the `name` property value. Use the `ldm set-io` or `ldm create-vf` command to specify the `name` property value.

  In addition, the domain to be migrated must be configured with IPMP in active or standby mode for the Ethernet virtual function and a virtual network device.

  > **Note:**
  >
  > Ensure that both of these virtual devices are able to access the same network.

  See How to Prepare a Domain With an SR-IOV Ethernet Virtual Function for Migration.

- **Target machine.** Create an Ethernet virtual function that uses the same user-assigned name as the Ethernet virtual function on the domain to be migrated. In addition, the SR-IOV Ethernet virtual function that you created on the target machine must be connected to the same network as the SR-IOV Ethernet virtual function on the source machine.

  See How to Prepare a Target Machine to Receive a Domain With an SR-IOV Ethernet Virtual Function.

  In addition, the Ethernet virtual function that you create on the target machine must be connected to the same network as the Ethernet virtual function on the source machine.

During the migration operation, the Ethernet virtual function is removed from the source domain dynamically and when the domain is created on the target machine, the Ethernet virtual function on the target machine is added to the migrated domain.

> **Note:**
>
> When the virtual function is removed, the traffic fails over to the standby virtual network path. After the migration succeeds, the communication returns to the active path. An application's performance might decline while using the standby path. After the migration succeeds and the traffic moves through the active path, the application returns to its previous level of performance.

How to Prepare a Domain With an SR-IOV Ethernet Virtual Function for Migration

When you create an SR-IOV Ethernet virtual function or assign it to a guest domain, you must ensure that the virtual function has a user-assigned virtual function name by specifying it as the `name` property value.

In addition, the domain to be migrated must be configured with a multipath route for the SR-IOV Ethernet virtual function and a virtual network device. Both of these virtual interfaces must be able to access the same network.

1. **Ensure the SR-IOV Ethernet virtual function is associated with a user-assigned virtual function name.**

   - New SR-IOV Ethernet virtual function: Create the virtual function with the user-assigned virtual function name.

     ```
     source:primary# ldm create-vf name=user-assigned-name
     pf-name
     ```

   - Existing SR-IOV Ethernet virtual function: Associate the user-assigned virtual function name with the virtual function.

     ```
     source:primary# ldm set-io name=user-assigned-name
     vf-name
     ```

2. **Assign the SR-IOV Ethernet virtual function to the guest domain.**

   ```
   source:primary# ldm add-io vf-name
   guest-domain
   ```

3. **Create an active-standby IPMP group for the SR-IOV Ethernet virtual function and the virtual network device.**

   The SR-IOV Ethernet virtual function is the active interface and the virtual network device is the standby interface.

   For more information, see How to Configure an Active-Standby IPMP Group in *Administering TCP/IP Networks, IPMP, and IP Tunnels in Oracle Solaris 11.4*.

How to Prepare a Target Machine to Receive a Domain With an SR-IOV Ethernet Virtual Function

Create an SR-IOV Ethernet virtual function on the target machine and ensure that uses the same user-assigned name as the SR-IOV Ethernet virtual function on the domain to be migrated. After the domain successfully migrates to the target machine, the SR-IOV Ethernet virtual function is assigned dynamically to the migrated domain.

- **Create an SR-IOV Ethernet virtual function on the target machine that uses the same user-assigned name as the SR-IOV Ethernet virtual function on the domain to be migrated.**

  ```
  target:primary# ldm create-vf name=user-assigned-name
  pf-name
  ```

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.

When migrating a domain that has SR-IOV Ethernet virtual functions, it might take some time after the migration has completed before the virtual functions are back in use by the guest domain. This situation occurs because the time it takes to assign virtual functions to the target machine depends on the number of virtual functions in the domain. However, IPMP handles the return to the active path after the assignment of virtual functions is complete.

The sixth column in the `FLAGS` field shows one of the following values:

- `s` – The domain that is the source of the migration.
- `t` – 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:

```
source:primary# 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:

```
source:primary# 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:

```
source:primary# 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.

If you cancel a live migration, the memory contents of the domain instance that is created on the target machine must be “scrubbed” by the hypervisor. This scrubbing process is performed for security reasons and must be complete before the memory can be returned to the pool of free memory. While this scrubbing is in progress, `ldm` commands become unresponsive. As a result, the Logical Domains Manager appears to be hung. So, wait for the scrubbing request to finish before you attempt to run other `ldm` commands. This process might take a long time. For example, a guest domain that has 500 Gbytes of memory might complete this process in up to 7 minutes on a SPARC T4 server.

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.

Saving Post-Migration SP Configurations Automatically

You can configure the system to save an SP configuration automatically to the SP on the source machine and the target machine following a successful migration. This behavior ensures that a powercycle following a successful migration does not result in the migrated domain being recreated on the source machine or removed from the target machine due to its presence or absence in the previously saved SP configuration.

When configured, an SP configuration is saved with the reserved name `@post-migration` following a successful migration. This SP configuration is overwritten after any successful migration and becomes the active configuration. You can use the `ldm set-spconfig` command and the `ldm remove-spconfig` command to manipulate this `@post-migration` SP configuration.

You can modify this behavior by setting the `migration_save_spconfig` SMF property or using the `-s` option to the `ldm migrate-domain` command.

The Boolean `migration_save_spconfig` SMF property controls whether to save an SP configuration following a successful incoming or outgoing migration.

- `migration_save_spconfig=false` – Prevents the saving of an SP configuration following a successful migration, which is the default value. Also, no SP configuration is saved if the `migration_save_spconfig` property is not specified.
- `migration_save_spconfig=true` – Saves the post migration SP configuration.

If the `migration_save_spconfig` value on the source machine differs from the value on the target machine, attempting an `ldm migrate-domain` command without the `-s` option fails immediately and no migration is performed.

> **Note:**
>
> To save these SP configurations automatically, the source machine and the target machine must both run at least the Oracle VM Server for SPARC 3.5 software. If one of the machines runs an older version of Oracle VM Server for SPARC, migrations succeed only if `migration_save_spconfig=false` on the Oracle VM Server for SPARC 3.5 machine.

Note that the automatic saving of a post-migration SP configuration interferes with the way that Oracle VM Manager maintains the system configuration. If Oracle VM Manager is running, the `migration_save_spconfig` property is ignored and no post-migration SP configuration is saved.

Use the `ldm migrate-domain -s` *spconfig-name* command to specify that you save a new SP configuration on the source machine and target machine following a migration. If you specify this option, it overrides the value of the `migration_save_spconfig` SMF property value. Using the `-s` option with no argument uses the default reserved name to save the SP configuration on both the source machine and the target machine. If you specify the `-s` option with *spconfig-name*, a new user SP configuration is created on both the source machine and the target machine with the specified name.

If the *spconfig-name* argument matches the name of an existing SP configuration on either the source machine or the target machine, the `ldm migrate-domain` command rejects the migration request.

If the target machine runs an older version of the Logical Domains Manager, and you specify the `-s` option, the `ldm migrate-domain` command rejects the migration request.

Migration Examples

Example 14-1 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.

```
source:primary# ldm migrate-domain -c ldg1 system2
```

Example 14-2 Migrating a Guest Domain

This example shows how to migrate the `ldg1` domain to a machine called `system2`.

```
source:primary# ldm migrate-domain ldg1 system2
Target Password:
```

To perform this migration without being prompted for the target machine password, use the following command:

```
source:primary# 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 14-3 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.

```
source:primary# ldm migrate ldg-src ldm-admin@system2:ldg-tgt
Target Password:
```

Example 14-4 Migration Failure Message

This example shows the error message that you might see if the target machine, `tsystem`, does not support the latest migration functionality.

```
source:primary# ldm migrate-domain ldg1 tsystem
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 14-5 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`.

```
source:primary# ldm list -o status ldg-tgt
NAME
ldg-tgt

STATUS
    OPERATION    PROGRESS    SOURCE
    migration    55%         t5-sys-1
```

Example 14-6 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`.

```
source:primary# ldm list -o status -p ldg-src
VERSION 1.6
DOMAIN|name=ldg-src|
STATUS
|op=migration|progress=42|error=no|target=system2
```

15

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, 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
- **CPU whole cores –** See Oracle Solaris OS Versions in *Oracle VM Server for SPARC 3.6 Installation Guide*
- **Virtual I/O devices –** Supported in at least the Oracle Solaris 10 10/08 OS and the Oracle Solaris 11 OS

- **Memory –** See Using Memory Dynamic Reconfiguration
- **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`(8) 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`(8) 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. To remove large amounts of memory from the control domain, see Decrease the Control Domain's Memory.

> **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 15-1 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. When you update the max-cores constraint on the control domain, the `ldm set-domain` command initiates a delayed reconfiguration automatically.

1. **Enable the max-cores constraint on the domain.**

   ```
   primary# ldm set-domain max-cores=max-number-of-CPU-cores domain-name
   ```

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 domain-name
   ```

   Now, you can use the domain with hard partitioning.

Example 15-2 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-domain 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
- Dynamic Resource Management

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 processors that are used in these systems have multiple CPU cores, each of which contains multiple CPU threads.
- **Hard partitioning and CPU whole cores.** 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 creates and configures the specified number of CPU cores and all its CPU threads in 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, see Hard Partitioning With Oracle VM Server for SPARC (http://www.oracle.com/technetwork/server-storage/vm/ovm-sparc-hard-partitioning-1403135.pdf).

  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 and max-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`(8) 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 max-cores. See the `ldm`(8) man page.

Max-cores 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 max-cores 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 `remove-core` subcommand.
- Set the max-cores 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 add-domain 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 remove-core` command.

5. **Bind and start the domain.**

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

Example 15-3 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.

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 add-domain ldg1
primary# ldm set-core 2 ldg1
...
primary# ldm bind ldg1
primary# ldm start-domain 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 domain-name
   ```

Example 15-4 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.

   You need to reboot the domain only if you want to modify the `max-cores` property.

Example 15-5 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 remove-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 15-6 Dynamically Adding Two CPU Whole Cores to a Domain

This example shows how to dynamically add two CPU whole cores to the `ldg1` domain. The `ldg1` domain is an active domain that has been configured with CPU whole cores. The first command shows that the `ldg1` domain is active. The second command shows that the `ldg1` domain is configured with CPU whole cores and a maximum of four CPU cores. The third and fifth commands show the CPU cores that are assigned to the domain before and after the addition of two CPU whole cores. The fourth command dynamically adds two CPU whole cores to the `ldg1` domain.

```
primary# ldm list ldg1
NAME    STATE   FLAGS   CONS  VCPU  MEMORY UTIL  UPTIME
ldg1    active  -n----  5000  16    2G     0.4%  5d 17h 49m
primary# ldm list -o resmgmt ldg1
NAME
ldg1

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.

CPU Weighted Mean Utilization

DRM uses weighted mean utilization to determine when to perform a CPU DR operation on a domain based on CPU utilization history. The weighted mean utilization value is an average of CPU utilization figures where the most recent utilization figure is assigned a greater weight than less recent utilization figures.

The weighted mean utilization is compared to the `util-upper` and `util-lower` DRM properties for each domain policy that is running. A CPU DR operation is performed only if the weighted mean utilization value falls outside of the upper and lower utilization bounds.

Power Management

You can set a separate power management (PM) policy for each hard-partitioned domain.

When PM detects that a domain is idle, it begins to skip cycles to save power. Skipping cycles reduces utilization, which affects DRM. As domain activity increases, PM stops skipping cycles and restores normalized utilization. This transition enables DRM to correctly calculate weighted mean utilization.

> **Note:**
>
> These PM transitions, along with DRM heuristics, take several seconds before DRM can dynamically reconfigure domain resources.

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 3.6 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. For information about physical resource restrictions, see Restrictions for Managing Physical Resources on Domains.

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 remove-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.

  You might encounter a situation where a guest domain OS binds a process running on a thread to a particular virtual CPU. If you later attempt to remove the core that is associated with that virtual CPU, the bound thread prevents the virtual CPU from being removed and leaves behind a partial core. As a result, you might see the following error message:

  ```
  Vcpu n: cpu has bound threads
  ```

  In this situation, re-add the core. Then, you can do one of the following actions:

  – Release the physical binding from the thread, and retry.

  – Remove a different named core.

- **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 remove-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 (`:`).

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. A physically assigned resource or a physically bound resource is also referred to as a named resource.

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 named 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. A non-physically assigned resource or a non-physically bound resource is also referred to as an anonymous resource.

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 named.**

   - 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 remove-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.

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

- **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 256-Mbyte-aligned. If you make a nonaligned request, the request is rounded up to be 256-Mbyte-aligned.

Memory DR Examples

The following examples show how to perform memory DR operations. For information about the related CLI commands, see the `ldm`(8) man page.

Example 15-7 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 remove-mem` command exits with an error because you must specify memory in multiples of 256 Mbytes. When you add the `--auto-adj` option to the same command, the memory removal succeeds because the amount of memory is rounded down to the next 256-Mbyte boundary.

```
primary# ldm remove-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 15-8 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 remove-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 remove-mem 100M ldom2
primary# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   3.3%  1d 22h 55m
ldom1            active     -n----  5000    2     2G       0.2%  1d 1h 25m
ldom2            bound      ------  5001    2     412M

primary# ldm remove-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 15-9 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
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-rsrc-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`(8) man page.

For examples, see Listing Resource Group Information.

Resource Group Requirements and Restrictions

The resource group feature is available only on servers starting with the SPARC T5, SPARC M5, and SPARC S7 series server and the Fujitsu M10 server.

The resource group feature has the following restrictions:

- It is not available on 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:

- 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):
>
> - 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).

> **Note:**
>
> Kernel zones and dynamic pools limitations prevent a DRM policy from working correctly with these features. Because DRM policies must not violate the limits that are established by these features, sometimes DRM does not appear to work when used with kernel zones or dynamic pools. For information about these limitations, see *Creating and Using Oracle Solaris Kernel Zones* and *Administering Resource Management in Oracle Solaris 11.4*.

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`(8) 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 15-10 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.
- Uses the default values for those properties that are not specified, such as `enable` and `sample-rate`. See the `ldm`(8) 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`(8) 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`(8) 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 by the `ldm list -l` command. The statistic is the percentage of time that the virtual CPU spends executing on behalf of the guest domain OS. A virtual CPU is considered to be executing on behalf of the guest domain OS except when the virtual CPU yields to the hypervisor.

> **Note:**
>
> The CPU utilization (`UTIL`) for an active domain that has a single CPU is `100%` regardless of load.

The utilization statistic reported for a logical domain is the average of the virtual CPU utilizations for the virtual CPUs in the domain. The per-virtual-CPU utilization statistic

is averaged over a 20-second interval. The logical domain utilization statistic is computed over a 10-second interval.

The normalized utilization statistic (`NORM`) is the percentage of time the virtual CPU spends executing on behalf of the guest domain OS. This value reflects the effect of CPUs running at lower frequency due to power management. Normalized virtualization is only available when your system runs at least version 8.2.0 of the system firmware.

When CPU power management is enabled, the utilization of CPUs is monitored and the effective frequency is adjusted in response to the workload. A guest domain that does not yield to the hypervisor and uses its full CPU cycles to run at full frequency has a normalized utilization of `100%`. A guest domain that has a lower workload still uses its CPUs, but they run at a lower effective frequency. The normalized utilization of such a guest domain reports a lower percentage. Thus, normalized utilization is processor-dependent and can vary on different platforms. Use the `ldm list` or `ldm list -l` command to show normalized utilization for both virtual CPUs and the guest domain 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
  - `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 and virtual logical domain channel client (`vldcc`))
- `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 SP configurations that have been stored on the SP:

  The `ldm list-spconfig` command lists the SP configurations that are stored on the service processor. When used with the `-r` option, this command lists those SP configurations for which autosave files exist on the control domain.

  For more information about SP configurations, see Managing SP Configurations. For more examples, see the `ldm`(8) man page.

  ```
  primary# ldm list-spconfig
  factory-default
  3guests
  foo [next poweron]
  primary
  reconfig-primary
  ```

  The labels next to the SP configuration name mean the following:

  - `[current]` – Last booted SP configuration, only as long as it matches the currently running SP configuration; that is, until you initiate a reconfiguration. After the reconfiguration, the annotation changes to `[next poweron]`.
  - `[next poweron]` – SP configuration to be used at the next power cycle.
  - `[degraded]` – SP configuration is a degraded version of the previously booted SP 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

> **Note:**
>
> While a DRM policy is actively running, the virtual CPU constraints are dynamic. Before you save the XML list of constraints, disable the running DRM policy so that these constraints become static.

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`(8) 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`(8) 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 15-11 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 15-12 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 15-13 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 15-14 Viewing Performance Access Information

The following examples show how to view performance access information by using the `ldm list -o domain` command.

- The following `ldm list -o domain` command shows that the `global` and `htstrand` performance values are specified on the `ldg0` domain:

```
primary# ldm list -o domain ldg0
NAME     STATE     FLAGS     UTIL
NORM
ldg0     active    -n----    0.0% 0.0%
```

```
SOFTSTATE
Solaris running

UUID
    062200af-2de2-e05f-b271-f6200fd3eee3

HOSTID
    0x84fb315d

CONTROL
    failure-policy=ignore
    extended-mapin-space=on
    cpu-arch=native
    rc-add-policy=
    shutdown-group=15
perf-counters=global,htstrand

DEPENDENCY
    master=

PPRIORITY    4000

VARIABLES
    auto-boot?=false
    boot-device=/virtual-devices@100/channel-devices@200/disk@0:a
      /virtualdevices@100/channel@200/disk@0
    network-boot-arguments=dhcp,hostname=solaris,
      file=http://10.129.241.238:5555/cgibin/wanboot-cgi
    pm_boot_policy=disabled=0;ttfc=2000;ttmr=0;
```

- The following `ldm list -p -o domain` command shows the same information as in the previous example but in the parseable form:

```
primary# ldm list -p -o domain ldg0
VERSION 1.12
DOMAIN|name=ldg0|state=active|flags=normal|util=|norm_util=
UUID|uuid=4e8749b9-281b-e2b1-d0e2-ef4dc2ce5ce6
HOSTID|hostid=0x84f97452
CONTROL|failure-policy=reset|extended-mapin-space=on|cpu-arch=native|rc-add-
policy=|
shutdown-group=15|perf-counters=global,htstrand
DEPENDENCY|master=
VARIABLES
|auto-boot?=false
|boot-device=/virtual-devices@100/channel-devices@200/disk@0
|pm_boot_policy=disabled=0;ttfc=2500000;ttmr=0;
```

Resource Management Issues

Removing a Large Number of CPUs From a Domain Might Fail

You might see the following error message when you attempt to remove a large number of CPUs from a guest domain:

```
Request to remove cpu(s) sent, but no valid response received
VCPU(s) will remain allocated to the domain, but might
not be available to the guest OS
Resource modification failed
```

To avoid this issue, remove fewer than 100 CPUs at a time from the domain.

Sometimes a Block of Dynamically Added Memory Can Be Dynamically Removed Only as a Whole

Due to the way in which the Oracle Solaris OS handles the metadata for managing dynamically added memory, you might later be able to remove only the entire block of memory that was previously dynamically added rather than a proper subset of that memory.

This situation could occur if a domain with a small memory size is dynamically grown to a much larger size, as shown in the following example.

```
primary# ldm list ldom1
NAME  STATE FLAGS   CONS VCPU MEMORY UTIL UPTIME
ldom1 active -n--   5000 2    2G     0.4% 23h

primary# ldm add-mem 16G ldom1

primary# ldm remove-mem 8G ldom1
Memory removal failed because all of the memory is in use.

primary# ldm remove-mem 16G ldom1

primary# ldm list ldom1
NAME  STATE FLAGS   CONS VCPU MEMORY UTIL UPTIME
ldom1 active -n--   5000 2    2G     0.4% 23h
```

To work around this issue, use the `ldm add-mem` command to sequentially add memory in smaller chunks rather than in chunks larger than you might want to remove in the future.

If you have experienced this problem, perform one of the following actions:

- Stop the domain, remove the memory, and then restart the domain.
- Reboot the domain, which causes the Oracle Solaris OS to reallocate its memory management metadata such that the previously added memory can now be removed dynamically in smaller chunks.

16

Managing SP Configurations

This chapter contains information about managing SP configurations.

This chapter covers the following topics:

- Managing SP Configurations
- Available Configuration Recovery Methods
- Configuration Management Issues

Managing SP Configurations

An *SP configuration* is a complete description of all the domains and their resource allocations within a single system. You can save and store SP configurations on the service processor (SP) for later use.

Saving an SP configuration on the SP makes it persist across system power cycles. You can save several SP configurations and specify which SP configuration to boot on the next power-on attempt.

When you power up a system, the SP boots the selected SP configuration. The system runs the same set of domains and uses the same virtualization and partitioning resource allocations that are specified in the SP configuration. The default SP configuration is the one that is most recently saved. You can also explicitly request a different SP 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 SP 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.

A local copy of the SP configuration and the Logical Domains constraint database is saved on the control domain whenever you save an SP 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 servers starting with the SPARC T5, SPARC M5, and SPARC S7 series server and the 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 SP 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 servers and SPARC S-series servers. Or, it might be either the entire system or a subset of the system as is the case of supported SPARC M-series servers.

Available Configuration Recovery Methods

Oracle VM Server for SPARC supports the following configuration recovery methods:

- The autosave method, used when the SP configuration is not available on the SP.

  This situation might occur in one of the following circumstances:

  - The hardware that holds the saved SP configurations has been replaced
  - The SP configuration is not up to date because you neglected to save the latest SP configuration changes to the SP or an unexpected power cycle occurred
- The `ldm add-domain -i` 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.

> **Note:**
>
> When using a boot device other than the factory default boot device, perform the steps in After Dropping Into factory-default, Recovery Mode Fails if the System Boots From a Different Device Than the One Booted in the Previously Active Configuration. Performing the steps ensure that recovery mode can recover the configuration on SPARC T4, SPARC T5, and SPARC M6 series servers that run a system firmware version prior to 9.5.3.

Restoring Configurations By Using Autosave

A copy of the current SP configuration is automatically saved on the control domain whenever the configuration of the domain 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 of the domain is not explicitly saved on the SP
- When the configuration change is not made until after the affected domain reboots

When a configuration is not explicitly saved to the SP or when SP configurations saved on the SP are lost, autosave operations enable the Logical Domains Manager to detect and report this situation, recover a configuration, or both. In these circumstances, the Logical Domains Manager, depending on the autorecovery policy value when it starts up, reports a configuration, restores a configuration, or both if it is newer than the SP 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 SP configurations. By default, when an autosave configuration is newer than the corresponding running SP 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 SP configuration on the SP or create a new one based on the autosave data. Note that you must perform a power cycle after using this command to have the running SP configuration match the newly updated SP configuration and so complete the manual recovery.

> **Note:**
>
> When a delayed reconfiguration is pending, the SP configuration changes are immediately autosaved. As a result, if you run the `ldm list-spconfig -r` command, the autosave configuration is shown as being newer than the current SP configuration.

For information about how to use the `ldm *-spconfig` commands to manage SP configurations and to manually recover autosave files, see the `ldm`(8) man page.

For information about how to select an SP configuration to boot, see Using Oracle VM Server for SPARC With the Service Processor. You can also use the `ldm set-spconfig` command, which is described on the `ldm`(8) man page.

Autorecovery Policy

The autorecovery policy specifies how to handle the recovery of an SP configuration when one SP configuration that is automatically saved on the control domain is newer than the corresponding running SP configuration.

Regardless of the policy you choose, the specified action occurs only when the `ldmd` service first starts.

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 SP configuration. These messages are logged in the `ldmd` SMF log file. You must manually perform any SP configuration recovery. This is the default policy.

- `autorecovery_policy=2` – Displays a notification message if an autosave configuration is newer than the corresponding running SP 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 SP configuration recovery.
- `autorecovery_policy=3` – Automatically updates the SP configuration if any autosave configuration is newer than its corresponding saved SP configuration. This action overwrites the SP configuration that will be used during the next power cycle, but does not make any changes to the currently running SP configuration. To update the running SP configuration to match the newly updated SP configuration, you must perform another power cycle. A message is also logged that states that a newer SP 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 11.4, see Chapter 1, About Using Rights to Control Users and Processes in *Securing Users and Processes in Oracle Solaris 11.4*.
3. **View the `autorecovery_policy` property value.**

   ```
   primary# svccfg -s ldmd listprop ldmd/autorecovery_policy
   ```
4. **Stop the `ldmd` service.**

   ```
   primary# svcadm disable ldmd
   ```
5. **Change the `autorecovery_policy` property value.**

   ```
   primary# svccfg -s ldmd setprop ldmd/autorecovery_policy=value
   ```

   For example, to set the policy to perform autorecovery, set the property value to `3`:

   ```
   primary# svccfg -s ldmd setprop ldmd/autorecovery_policy=3
   ```
6. **Refresh and restart the `ldmd` service.**

   ```
   primary# svcadm refresh ldmd
   primary# svcadm enable ldmd
   ```

Example 16-1 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.

```
primary# svccfg -s ldmd listprop ldmd/autorecovery_policy
ldmd/autorecovery_policy integer 1
primary# svcadm disable ldmd
primary# svccfg -s ldmd setprop ldmd/autorecovery_policy=3
primary# svcadm refresh ldmd
primary# svcadm enable ldmd
```

Saving Domain Configurations

You can save domain configurations (domain constraints) 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 domain configurations, 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 domain configurations for a single domain, create an XML file containing the domain's constraints.

  ```
  primary# 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:

  ```
  primary# ldm list-constraints -x ldg1 >ldg1.xml
  ```

- To save the domain configurations for all the domains on a system, create an XML file containing the constraints for all domains.

  ```
  primary# 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:

  ```
  primary# ldm list-constraints -x >config.xml
  ```

Restoring Domain Configurations

This section describes how to restore a domain configurations from an XML file for guest domains and for the control (`primary`) domain.

- To restore a domain constraints 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). 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).

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).

1. **Create the domain by using the XML file that you created as input.**

```
primary# ldm add-domain -i domain-name.xml
```

2. **Bind the domain.**

   ```
   primary# 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.**

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

Example 16-2 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.

```
primary# ldm add-domain -i ldg1.xml
primary# ldm bind ldg1
primary# ldm start-domain ldg1
```

How to Restore a Domain Configuration From an XML File (`ldm init-system`)

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.

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 Idm 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 3.6 Release Notes*.

1. **Log in to the primary domain.**
2. **Verify that the system is in the `factory-default` configuration.**

   ```
   primary# ldm list-spconfig | grep "factory-default"
   factory-default [current]
   ```

   If the system is not in the `factory-default` configuration, see How to Restore the factory-default Configuration.

3. **Become an administrator.**

   For Oracle Solaris 11.4, see Chapter 1, About Using Rights to Control Users and Processes in *Securing Users and Processes in Oracle Solaris 11.4*.

4. **Restore the domain configuration or configurations from the XML file.**

```
primary# 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` 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 16-3 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.

  ```
  primary# ldm init-system -r -i config.xml
  ```

  After the system reboots, the following commands bind and restart the `ldg1` and `ldg2` domains:

  ```
  primary# ldm bind ldg1
  primary# ldm start ldg1
  primary# ldm bind ldg2
  primary# ldm start ldg2
  ```

Addressing Service Processor Connection Problems

On a server starting with the SPARC T7, SPARC M7, and SPARC S7 series server, the ILOM interconnect is used to communicate between the `ldmd` service and the SP.

- **Servers Starting With the SPARC T7, SPARC M7, and SPARC S7 Series Server:** Check the ILOM interconnect state and re-enable the `ilomconfig-interconnect` service. See How to Verify the ILOM Interconnect Configuration and How to Re-Enable the ILOM Interconnect Service.
- **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.

Configuration Management Issues

`init-system` Does Not Restore Named Core Constraints for Guest Domains From Saved XML Files

If you assigned named core resources to a domain, using the `ldm init-system` command might fail to re-assign those named resources to that domain. This might occur because the `ldm init-system` command initiates a delayed reconfiguration for the `primary` domain and you can perform only one virtual CPU operation per delayed reconfiguration. So, this command fails to restore the named CPU core constraints for guest domains from a saved XML file.

Workaround: Perform the following steps:

1. Create an XML file for the primary domain.

   ```
   primary# ldm list-constraints -x primary > primary.xml
   ```

2. Create an XML file for the guest domain or domains.

   ```
   primary# ldm list-constraints -x domain-name[,domain-name][,...] > guest.xml
   ```

3. Power cycle the system and boot a `factory-default` configuration.
4. Apply the XML configuration to the `primary` domain.

   ```
   primary# ldm init-system -r -i primary.xml
   ```

5. Apply the XML configuration to the guest domain or domains.

   ```
   primary# ldm init-system -f -i guest.xml
   ```

After Dropping Into `factory-default`, Recovery Mode Fails if the System Boots From a Different Device Than the One Booted in the Previously Active Configuration

While triggering a recovery after dropping into `factory-default`, recovery mode fails if the system boots from a different device than the one booted in the previously active configuration. This failure might occur if the active configuration uses a boot device other than the `factory-default` boot device.

> **Note:**
>
> This problem applies to SPARC T4 series servers. This problem also applies to SPARC T5, SPARC M5, and SPARC M6 series servers that run a system firmware version prior to 9.5.3.

To work around the problem, perform the following steps any time you want to save a new configuration to the SP:

1. Determine the full PCI path to the boot device for the `primary` domain.

   Use this path for the `ldm set-var` command in Step 4.

2. Remove any currently set `boot-device` property from the `primary` domain.

   Performing this step is necessary only if the `boot-device` property has a value set. If the property does not have a value set, an attempt to remove the `boot-device` property results in the `boot-device not found` message.

   ```
   primary# ldm rm-var boot-device primary
   ```

3. Save the current configuration to the SP.

   ```
   primary# ldm add-spconfig config-name
   ```

4. Explicitly set the `boot-device` property for the `primary` domain.

   ```
   primary# ldm set-var boot-device=value primary
   ```

   If you set the `boot-device` property after saving the configuration to the SP as described, the specified boot device is booted when recovery mode is triggered.

If recovery mode has already failed as described, perform the following steps:

1. Explicitly set the boot device to the one used in the last running configuration.

   ```
   primary# ldm set-var boot-device=value primary
   ```

2. Reboot the `primary` domain.

   ```
   primary# reboot
   ```

   The reboot enables the recovery to proceed.

Guest Domain `eeprom` Updates Are Lost if an `ldm add-spconfig` Operation Is Not Complete

An attempt to set an OBP variable from a guest domain might fail if you use the `eeprom` or the OBP command before one of the following commands is completed:

- `ldm add-spconfig`
- `ldm remove-spconfig`
- `ldm set-spconfig`
- `ldm bind`

This problem might occur when these commands take more than 15 seconds to complete.

```
# /usr/sbin/eeprom boot-file\=-k
promif_ldom_setprop: promif_ldom_setprop: ds response timeout
eeprom: OPROMSETOPT: Invalid argument
boot-file: invalid property
```

If you encounter this error, retry the `eeprom` or OBP command after the `ldm` operation has completed.

To avoid this error, retry the `eeprom` or OBP command on the affected guest domain. You might be able to avoid the problem by using the `ldm set-var` command on the `primary` domain.

Trying to Connect to Guest Domain Console While It Is Being Bound Might Cause Input to Be Blocked

A domain's guest console might freeze if repeated attempts are made to connect to the console before and during the time the console is bound. For example, this might occur if you use an automated script to grab the console as a domain is being migrated onto the machine.

To unfreeze console, perform the following commands on the domain that hosts the domain's console concentrator (usually the control domain):

```
primary# svcadm disable vntsd
primary# svcadm enable vntsd
```

17

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 the SPARC T5, SPARC M5, and SPARC S7 series server:

- **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 SP configurations that cannot be booted because of faulty or missing resources.

The Fujitsu SPARC M12 platform and Fujitsu M10 platform also support recovery mode. While the blacklisting of faulty resources is not supported, the Fujitsu SPARC M12 platform and 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.

You can use the `ldm list-devices -B` command to view blacklisted resources or resources pending evacuation. The following command shows the blacklisted memory and core resources:

```
primary# ldm list-devices -B
CORE
ID STATUS DOMAIN
1 Blacklisted
2 Evac_pending ldg1
MEMORY
PA SIZE STATUS DOMAIN
0xa30000000 87G Blacklisted
0x80000000000 128G Evac_pending ldg1
```

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 list
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

Notice: the system is running in a degraded mode as domain <guest> could not
be started because required resources were blacklisted and evacuated.
```

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 server starting with the SPARC T5, SPARC M5, SPARC S7 series server, or the 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 M-series servers) is said to be in *recovery mode*. Recovery mode is enabled by default. See Controlling Recovery Mode.

> **Note:**
>
> The recovery operation does not preserve auto-allocated worldwide numbers (WWNs) for any devices when creating the degraded configuration. Instead, the recovery operation allocates new WWNs to those devices.

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 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.

> **Note:**
>
> You cannot delete the original configuration from the SP until the system successfully boots the original 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. If you assigned named resources to a domain and it is later recovered in recovery mode, there is no attempt to re-assign those named resources to that domain.

> **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 servers starting with the SPARC T5, SPARC M5, and SPARC S7 series server and the Fujitsu M10 server.
- **Firmware Requirements –** At least version 9.1.0.a of the system firmware for the SPARC T5, SPARC M5, and SPARC M6 server. At least version 9.4.3 of the system firmware for the SPARC T7 and SPARC M7 series server. Any released version of the system firmware for the SPARC S7, SPARC T8, and SPARC M8 series server. At least version XCP2230 of the system firmware for the Fujitsu M10 server. At least version XCP3021 of the system firmware for the Fujitsu SPARC M12 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.

> **Note:**
>
> You cannot delete the original configuration from the SP until the system successfully boots the original configuration.

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.

18
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

In general, Logical Domains Manager names can be up to 256 characters in length.

The following sections describe the naming restrictions in the Logical Domains Manager CLI.

- Variable names
  - First character must be a letter, a number, or a forward slash (/).
  - Subsequent letters must be letters, numbers, or punctuation.
- File names that are used in virtual disk back ends, virtual switch device names, and path file names

  The names must contain only letters, numbers, or punctuation.

- SP configuration names

  The length of SP configuration names (or SP configuration names) is limited by the SP. The limit is currently around 69 characters but could vary by platform.

  The following error results if you specify an SP configuration name that is too long:

```
primary# ldm add-spconfig \
test5678901234567892123456789312345678941234567895123456789612345678 97
Error: Operation failed because an invalid configuration name was given
```

- Virtual device service and client names

  Virtual device names are used to create the `devalias` property, which is used by the OpenBoot PROM. However, the OpenBoot PROM does not support `devalias` names longer than 31 characters.

  If you specify a virtual device name that exceeds 31 characters, the command succeeds but the corresponding `devalias` property is not created. The command also issues the following warning:

```
primary# ldm add-vds primary-vds0123456789012345678 90 primary
Warning: Device name primary-vds012345678901234567890 is too long to create
devalias
```

- Hardware path names

  These names are paths to physical resources. These names are used to specify the iport in the `ldm add-vsan` command and the resource group in the `ldm list-rsrc-group` command.

- All other names
  - 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

Starting with the Oracle Solaris 11 OS, 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

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`(8) 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 how the Oracle Solaris OS behavior changes when you run an Oracle VM Server for SPARC configuration.

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 the configuration of your current logical domains to the SP first.

To save your SP configuration to the SP, use the following command:

```
ldm add-spconfig config-name
```

Starting a Domain

Use the `ldm start-domain` command to start one or more domains. You can also specify an XML configuration file to start a logical domain.

Use the `ldm start-domain -f` command to enable a guest domain and its I/O service domains to boot simultaneously. The `-f` option starts a guest domain even when its SR-IOV service domain or domains are not running. The guest domain does not boot successfully if the missing I/O services are essential to the OS boot process. Note that applications running on the guest domain might not function properly if the missing I/O services are essential to the application's operation.

Stopping a Domain

The `ldm stop-domain` command stops one or more running domains.

If the domain runs the Logical Domains Manager agent provided by at least the Oracle Solaris 10 1/13 or Oracle Solaris 11.1 OS, the `ldm stop-domain` command sends a `shutdown` request to the domain.

If the domain does not run the appropriate Logical Domains Manager agent, the `ldm stop-domain` command sends a `uadmin` request to the domain.

You can also specify a number of seconds in which the `shutdown` command must complete. If this timeout value is exceeded, an `ldm stop-domain -q` command is issued.

You can use SMF properties to change the default behavior. See the `ldmd`(8) man page.

For more information about the `ldm stop-domain` command, see the `ldm`(8) man page.

> **Note:**
>
> If you run the `ldm stop-domain -f` command for a domain that is at the `kmdb` prompt, the command fails with the following error message:
>
> ```
> LDom domain-name stop notification failed
> ```

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.

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 SP configuration on the next power on to an different SP 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 SP configurations that have been created with the Logical Domains Manager using the `ldm add-spconfig` command and stored on the service processor (SP). The name you specify in the Logical Domains Manager `ldm add-spconfig`

command can be used to select that SP configuration with the ILOM `bootmode` command. For example, assume you stored the SP 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 SP configuration.

See the `ldm`(8) man page for more information about the `ldm add-spconfig` 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-domain` command)
- `reset` immediately stops and then restarts any slave domains (similar to running the `ldm stop-domain -f` command and then the `ldm start-domain` command)
- `stop` stops any slave domains (similar to running the `ldm stop-domain -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.

For domain dependency XML examples, see Domain Information (ldom_info) Resource in *Oracle VM Server for SPARC 3.6 Developer's Guide*.

Domain Dependency Examples

The following examples show how to configure domain dependencies.

Example 18-1 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 18-2 Modifying a Domain to Assign a Master Domain

This example shows how to use the `ldm set-domain` command to modify the `orange` domain to assign `primary` as the master domain. The second command uses the `ldm set-domain` command to assign `orange` and `primary` as master domains for the `tangerine` domain. The third command lists information about all of these domains.

```
primary# ldm set-domain master=primary orange
primary# ldm set-domain master=orange,primary tangerine
primary# ldm list -o domain
NAME             STATE      FLAGS   UTIL
primary          active     -n-cv-  0.2%

SOFTSTATE
Solaris running

HOSTID
    0x83d8b31c

CONTROL
    failure-policy=ignore

DEPENDENCY
```

```
    master=

------------------------------------------------------------------------------
NAME             STATE      FLAGS   UTIL
orange           bound      ------

HOSTID
    0x84fb28ef

CONTROL
    failure-policy=ignore

DEPENDENCY
    master=primary

------------------------------------------------------------------------------
NAME             STATE      FLAGS   UTIL
tangerine        bound      ------

HOSTID
    0x84f948e9

CONTROL
    failure-policy=ignore

DEPENDENCY
    master=orange,primary
```

Example 18-3 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.

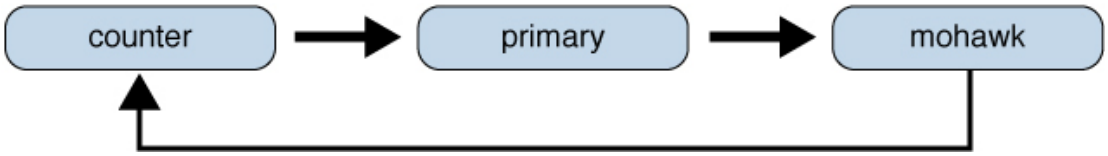
Single Domain Dependency

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.

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.

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 associated with the stanza for the domain, and the virtual CPU number within the domain is specified 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 associated with the stanza for the domain and the corresponding real address within the domain is specified 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 18-4 Determining the Configuration of Domains

The following command produces a long parseable list of logical domains configurations.

```
primary# ldm list -l -p
VERSION 1.6
DOMAIN|name=primary|state=active|flags=normal,control,vio-service|
cons=SP|ncpu=4|mem=1073741824|util=0.6|uptime=64801|
softstate=Solaris running
VCPU
|vid=0|pid=0|util=0.9|strand=100
|vid=1|pid=1|util=0.5|strand=100
|vid=2|pid=2|util=0.6|strand=100
|vid=3|pid=3|util=0.6|strand=100
MEMORY
|ra=0x8000000|pa=0x8000000|size=1073741824
IO
|dev=pci@780|alias=bus_a
|dev=pci@7c0|alias=bus_b
...
DOMAIN|name=ldg1|state=active|flags=normal|cons=5000|
ncpu=2|mem=805306368|util=29|uptime=903|
softstate=Solaris running
VCPU
|vid=0|pid=4|util=29|strand=100
|vid=1|pid=5|util=29|strand=100
MEMORY
|ra=0x8000000|pa=0x48000000|size=805306368
...
DOMAIN|name=ldg2|state=active|flags=normal|cons=5001|
ncpu=3|mem=1073741824|util=35|uptime=775|
```

```
softstate=Solaris running
VCPU
|vid=0|pid=6|util=35|strand=100
|vid=1|pid=7|util=34|strand=100
|vid=2|pid=8|util=35|strand=100
MEMORY
|ra=0x8000000|pa=0x78000000|size=1073741824
...
```

Example 18-5 Determining the Virtual CPU That Corresponds to a Physical CPU Number

The logical domain configuration is shown in Determining the Configuration of Domains. 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 in the stanza for 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 in the stanza for domain `ldg2`.

```
ra=0x8000000|pa=0x78000000|size=1073741824
```

Where 0x78000000 <= 0x7e816000 <= (0x78000000 + 1073741824 - 1); that is, *pa* <= PA <= (*pa* + *size* - 1). Hence, the PA is in domain `ldg2` and the corresponding real address is 0x8000000 + (0x7e816000 - 0x78000000) = 0xe816000.

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 add-domain 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`(8) 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 on servers starting with the SPARC T4, SPARC M5, and SPARC S7 series server and the Fujitsu M10 server. The number of LDCs in the pool is based on the platform type as follows:

- **SPARC T4 Server –** 1984 LDC endpoints per logical domain
- **SPARC T5 Server –** 4080 LDC endpoints per logical domain
- **SPARC T7 Series Server –** 4080 LDC endpoints per logical domain
- **SPARC T8 Series Server –** 4080 LDC endpoints per logical domain
- **SPARC M5 Server –** 4080 LDC endpoints per logical domain
- **SPARC M6 Server –** 4080 LDC endpoints per logical domain
- **SPARC M7 Series Server –** 4080 LDC endpoints per logical domain
- **SPARC M8 Series Server –** 4080 LDC endpoints per logical domain

- **Fujitsu M10 Server –** 4080 LDC endpoints per logical domain
- **Fujitsu SPARC M12 Server –** 4080 LDC endpoints per logical domain
- **SPARC S7 Series Server –** 4080 LDC endpoints per logical domain

> **Note:**
>
> Starting with the Oracle SPARC T5, SPARC M5, and SPARC S7 series server and the Fujitsu M10 server, the LDC pool contains 4080 LDCs when the domain runs at least the Oracle Solaris 11.3 SRU 8 OS. Otherwise, LDC the pool contains 1984 LDCs.

The required system firmware to support the LDC endpoint pool is as follows:

- 8.5.2 for SPARC T4 servers
- 9.2.1 for SPARC T5, SPARC M5, and SPARC M6 servers
- Any released version for SPARC T7 and SPARC M7 series servers
- Any released version for SPARC S7 series servers
- Any released version for SPARC T8 and SPARC M8 series servers
- XCP2240 for Fujitsu M10 servers
- Any released version of the system firmware for Fujitsu SPARC M12 servers

The following LDC endpoint limits still apply if you run an older version of the system firmware on a supported platform:

- **SPARC T4 Server –** 768 LDC endpoints per logical domain
- **SPARC T5 Server –** 768 LDC endpoints per logical domain
- **SPARC M5 Server –** 768 LDC endpoints per logical domain
- **SPARC M6 Server –** 768 LDC endpoints per logical domain
- **Fujitsu M10 Server –** 768 LDC endpoints per logical domain

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

Virtual network devices that you create by using at least the Oracle VM Server for SPARC 3.4 software have `inter-vnet-link=auto` by default. This feature automatically turns off inter-vnet-links when the number exceeds the threshold. However, any virtual network devices that you created with `inter-vnet-link=on` must be explicitly modified to change `inter-vnet-link=off` to reduce the number of LDC channels. For more information, see Inter-Vnet LDC Channels.

You can still set `inter-vnet-link=off` to reduce 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`(8) 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 SPARC M12 servers per physical partition
- Up to 256 on Fujitsu M10 servers per physical partition
- Up to 128 on SPARC M8 series servers per physical domain
- 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 T8 series servers
- Up to 128 on SPARC T7 series servers

- Up to 128 on SPARC T5 servers
- Up to 128 on SPARC T4 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*, *Manually Installing an Oracle Solaris 11.4 System*, and *Automatically Installing Oracle Solaris 11.4 Systems*.

Memory and swap space usage increases in a guest domain when the `vsw` services used by the domain provide services to many virtual networks in multiple domains. This increase is due to the peer-to-peer links between all the `vnet` instances connected to the `vsw`. The service domain benefits from having extra memory. The recommended minimum is four Gbytes when running more than 64 domains. Start domains in groups of 10 or fewer and wait for them to boot before starting the next batch. The same advice applies to installing operating systems on domains. You can reduce the number of links by disabling inter-vnet links. See Inter-Vnet LDC Channels.

Cleanly Shutting Down and Power Cycling an Oracle VM Server for SPARC System

If you have made any configuration changes since last saving an SP configuration to the SP, 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.

2. **Use the SP to power on the system.**

Logical Domains Variable Persistence

Any updates that you make to variables, including the date and time, in the control domain or in any guest domain persist across a reboot.

If you save a new logical domain configuration to the SP prior to performing a power cycle, the variable, date, and time updates are preserved across power cycles. So, each time you update a variable, the date or time, save a new configuration to the SP after you make the change.

Starting with the SPARC T4, SPARC M5, and SPARC S7 series servers, any updates you make to variables in the control domain are preserved automatically across a power cycle. On these servers, you do not need to save a configuration to the SP prior to a power cycle. On older-generation SPARC servers, only updates made at the OpenBoot prompt are automatically preserved.

> **Note:**
>
> If you reboot the control domain while all guest domains are unbound and while no delayed reconfiguration is in progress, the SP performs a power cycle of the system.

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 servers starting with the SPARC M7, SPARC T7, and SPARC S7 series server.

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 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.

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
00001000170199f8 emlxs#1 MSI-X  No       32     8    8
000010001400ca28 emlxs#2 MSI-X  No       32     8    8
0000100016151328 igb#3   MSI-X  No       10     3    3
0000100019549d30 igb#2   MSI-X  No       10     3    3
0000040000e0f878 igb#1   MSI-X  No       10     3    3
000010001955a5c8 igb#0   MSI-X  No       10     3    3
```

The default limit in this example is 8 interrupts per device, which is not enough interrupts to accommodate attaching the final `emlxs3` device to the system. Assuming that all `emlxs` instances behave in the same way, `emlxs3` probably requested 8 interrupts.

By subtracting the 12 interrupts used by all of the `igb` devices from the total pool size of 36 interrupts, 24 interrupts are available for the `emlxs` devices. Dividing the 24

interrupts by 4 suggests that 6 interrupts per device would enable all `emlxs` devices to attach with equal performance. So, the following adjustment is added to the `/etc/system` file:

```
set ddi_msix_alloc_limit = 6
```

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

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
```

Handling an Exhausted Interrupt Supply While Attaching I/O Device Drivers

This following warning on the Oracle Solaris console means that the interrupt supply was exhausted while attaching I/O device drivers:

```
WARNING: ddi_intr_alloc: cannot fit into interrupt pool
```

This limitation applies only to the supported SPARC systems prior to the SPARC M7 series servers and SPARC T7 series servers.

The hardware provides a finite number of interrupts, so Oracle Solaris limits how many each device can use. A default limit is designed to match the needs of typical system configurations, however this limit may need adjustment for certain system configurations.

Specifically, the limit may 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 interrupts into smaller sets given to guest domains. If too many I/O devices are assigned to a guest domain, its supply might be too small to give each device the default limit of interrupts. Thus, it exhausts its supply before it completely attaches all the drivers.

Some drivers provide an optional callback routine which allows Oracle Solaris to automatically adjust their interrupts. The default limit does not apply to these drivers.

To work around this issue, use the `::irmpools` and `::irmreqs` MDB macros to determine how interrupts are used. The `::irmpools` macro shows the overall supply of interrupts 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 the driver is given.

The macros do not show information about drivers that failed to attach. However, the information that is shown helps calculate the extent to which you can adjust the default limit.

Any device that uses more than one interrupt without providing a callback routine can be forced to use fewer interrupts by adjusting the default limit. Reducing the default limit below the amount that is used by such a device results in freeing of interrupts for use 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.

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
00001000170199f8 emlxs#1 MSI-X  No       32     8    8
000010001400ca28 emlxs#2 MSI-X  No       32     8    8
0000100016151328 igb#3   MSI-X  No       10     3    3
0000100019549d30 igb#2   MSI-X  No       10     3    3
0000040000e0f878 igb#1   MSI-X  No       10     3    3
000010001955a5c8 igb#0   MSI-X  No       10     3    3
```

The default limit in this example is eight interrupts per device, which is not enough interrupts to accommodate the attachment of 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
```

When the system successfully boots without warnings, the `::irmpools` and `::irmreqs` macros show the following updated information:

```
# 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 servers starting with the SPARC T7, SPARC M7, and SPARC S7 series server, 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.

> **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`(8) 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 Logical Domains Manager Configuration Data

Before you reinstall the operating system on the control domain, save the Logical Domains Manager configuration data. This data includes the autosave configuration

information in the `/var/share/ldomsmanager/autosave-`*`autosave-name`* directories.

> **Note:**
>
> Each autosave directory includes a timestamp for the last SP configuration update for the related SP 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 SP Configurations.

For information about methods you can use to recover the domains you configured, see Available Configuration Recovery Methods.

How to Save Logical Domains Manager Configuration Data on the Control Domain

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

- **Save the Logical Domains Manager configuration data in the `/var/share/ldomsmanager` directory.**

  The following example shows how to use the `tar` command to create the `configdata.tar` archive file:

```
primary# cd /
primary# tar -cvpf configdata.tar var/share/ldomsmanager/
```

The `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 the `factory-default` 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 SP configurations, and revert the SP configuration to the factory default settings.

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 SP Configurations

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

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

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

   Use the following command for each such SP configuration:

```
primary# ldm remove-spconfig config-name
```

   After you remove all the SP 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` SP configuration.**

```
primary# ldm set-spconfig 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
   ```

Logging Oracle VM Server for SPARC Events

The `ldm set-logctl`, `ldm list-logctl`, and `ldm list-history` commands enable you to specify fine-grained logging characteristics, view the logging settings, and list `ldm` command history, respectively. See the `ldm`(8) man page.

Oracle VM Server for SPARC logs messages to its standard log, `/var/svc/log/ldoms-ldmd:default.log`.

This section covers the following topics:

Controlling Oracle VM Server for SPARC Logging Operations

Use the `ldm set-logctl` command to specify the fine-grained logging characteristics that control the messages written to the log. Note that you cannot disable the logging of fatal or warning messages. See the `ldm`(8) man page.

Example 18-6 Controlling Event Logging Operations

The following examples show ways in which to use the `ldm set-logctl` command:

- Enable logging for `notice` messages, which indicate that an event requires user attention.

  ```
  primary# ldm set-logctl notice=on
  ```

- Specify the number of messages that the `ldm list-history` command outputs.

  The following command sets the number to 20.

  ```
  primary# ldm set-logctl history=20
  ```

- Reset logging properties to the default values.

  ```
  primary# ldm set-logctl defaults
  ```

Example 18-7 Logging Output

Use the `ldm set-logctl` command to specify the logging characteristics for Oracle VM Server for SPARC events to be written to the log.

- The following command logs `ldm` commands:

  ```
  primary# ldm set-logctl cmd=on
  ```

  When you configure command logging, the following types of information are logged:

  - The following entry is logged for an `ldm list` command:

    ```
    cmd: ldm list
    cmd: OK
    ```

  - The following entry is logged for a badly formed `ldm list` command:

    ```
    cmd: ldm list -x
    cmd: USAGE
    ```

  - The following entry is logged for an `ldm list` command that specifies a non-existent domain:

    ```
    cmd: ldm list non-existent-domain-name
    cmd: ERROR:
    cmd: LDom "non-existent-domain-name" was not found
    ```

- The following command logs `ldm` commands and command responses:

  ```
  primary# ldm set-logctl cmd=resp
  ```

  For example, when you issue the `ldm list` command, the following command information is logged:

  ```
  cmd: ldm list
  cmd: OK:
  cmd: NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM
  UPTIME
  cmd: primary          active     -n-cv-  SP      16    9248M    0.8%  118d 3h
  cmd: ldg1             active     -t----  5000    16    4G       6.2     %
  2d 22h 24m
  ```

Controlling Logging Capabilities by Using SMF

The `ldmd/logctl` property specifies the property values that you can also set by using the `ldm set-logctl` command. Separate each property with a colon character (`:`).

For example, use the `svccfg` command to set the `cmd` property value to `resp` and to set the `debug` property value to `on`. Then, make the changes take affect by refreshing and restarting the `ldmd` service as follows:

```
primary# svccfg -s ldmd setprop ldmd/logctl = "cmd=resp:debug=on"
primary# svcadm refresh ldmd
primary# svcadm restart ldmd
```

Viewing Oracle VM Server for SPARC Logging Capabilities

The `ldm list-logctl` command shows you the current behavior of the logging types.

- View settings for all logging types.

  ```
  primary# ldm list-logctl
  ```

- View the settings for the `notice` and `cmd` logging types.

  ```
  primary# ldm list-logctl notice cmd
  ```

- View the default settings.

  ```
  primary# ldm list-logctl -d
  ```

- View the logging capability values for all logging types and the number of commands output by the `ldm list-history` command.

  ```
  primary# ldm list-logctl -a
  ```

Viewing Oracle VM Server for SPARC Command History

Use the `ldm list-history` command to view the Oracle VM Server for SPARC command history log. This log captures `ldm` commands and commands that are issued through the XMPP interface. By default, the number of the commands shown by the `ldm list-history` command is ten.

The short form of the `ldm list-history` command is the `ldm history` command.

To change the number of commands output by the `ldm list-history` command, use the `ldm set-logctl` command to set the `history` property value. If you set `history=0`, the saving of command history is disabled. You can re-enable this feature by setting the `history` property to a non-zero value.

Note that enabling and disabling command history is logged in the command log.

A
Using Power Management

This appendix contains information about using power management (PM) on Oracle VM Server for SPARC systems.

Starting with the Oracle VM Server for SPARC 3.5.0.1 software, Logical Domains Manager-based PM is disabled by default for Oracle SPARC platforms.

To enable PM, set the `ldmd/pm_enabled` SMF property value to `true`. See the `ldmd`(8) man page.

If you do not plan to override this new default, perform the following steps before you upgrade to the Oracle VM Server for SPARC software. Performing these steps ensures that the system runs at full power and with the highest performance.

Perform the following steps that apply to your SPARC server:

- **Up to and including the SPARC M6 and SPARC T5 series servers.**
  1. Disable Service Processor power management, also known as the platform policy.

     Run the following ILOM command as an administrative user:

     ```
     --> set /SP/powermgmt/ policy=disabled
     ```

  2. Set the PM administrative control to `platform` on the control domain as superuser:

     ```
     primary# poweradm set active_control/administrative-authority=platform
     ```

- **Starting with the SPARC M7, SPARC T7, and SPARC S7 series servers.** Perform one of the following steps:
  - Perform a power cycle of the server.
  - Update the server with at least system firmware version 9.8.6.

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.

> **Note:**
>
> The `pm_boot_policy` property value is shown in some list output. This value contains internal PM data and is not to be interpreted or modified.

For information about preventing a user to suspend or power down a system from a system console, see How to Remove Power Management Capability From Users in *Securing Users and Processes in Oracle Solaris 11.4*.

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 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 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, SPARC T8, SPARC M7, SPARC M8, and SPARC S7 series 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 and SPARC T8 series servers, the clock frequency is reduced on SCCs. This feature is available only on servers starting with the SPARC T5, SPARC M5, and SPARC S7 series server.

- **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 servers starting with the SPARC T4, SPARC M5, and SPARC S7 series server 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 servers starting with the SPARC T5, SPARC M5, and SPARC S7 series server 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`(8) 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`(8) 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.

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.

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 A-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 A-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
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 A-3 Viewing Processor Power-Consumption Data

The following examples show how to use the `ldmpower` to report processor power-consumption data for your domains.

- The following command shows the 15-second, 30-second, and 60-second rolling average processor power-consumption data for all domains:

```
primary# ldmpower
Processor Power Consumption in Watts
DOMAIN  15_SEC_AVG  30_SEC_AVG  60_SEC_AVG
primary 75          84          86
gdom1   47          24          19
gdom2   10          24          26
```

- The following command shows extrapolated power-consumption data for all the domains: `primary`, `gdom1`, and `gdom2`.

```
primary# ldmpower -x
System Power Consumption in Watts
DOMAIN  15_SEC_AVG  30_SEC_AVG  60_SEC_AVG
primary 585/57.47%  701/68.96%  712/70.22%
gdom1   132/12.97%  94/9.31%    94/9.30%
gdom2   298/29.27%  218/21.47%  205/20.22%
```

- The following command shows the instantaneous processor power-consumption data for the `gdom2` and `gdom5` domains. It reports the data every ten seconds for five times.

```
primary# ldmpower -itl gdom2,gdom5 10 5
Processor Power Consumption in Watts
DOMAIN          TIMESTAMP               INSTANT
gdom2           2013.05.17 11:14:45     13
gdom5           2013.05.17 11:14:45     24

gdom2           2013.05.17 11:14:55     18
gdom5           2013.05.17 11:14:55     26

gdom2           2013.05.17 11:15:05     9
gdom5           2013.05.17 11:15:05     16

gdom2           2013.05.17 11:15:15     15
gdom5           2013.05.17 11:15:15     19

gdom2           2013.05.17 11:15:25     12
gdom5           2013.05.17 11:15:25     18
```

- The following command shows the average power-consumption data for the last 12 hours for all domains. Data is shown at one-hour intervals starting from the last requested hourly calculation.

```
primary# ldmpower -eto 12
Per domain MINIMUM and MAXIMUM power consumption ever recorded:
primary         2013.05.17 08:53:06     3               Min Processors
primary         2013.05.17 08:40:44     273             Max Processors
gdom1           2013.05.17 09:56:35     2               Min Processors
gdom1           2013.05.17 08:53:06     134             Max Processors
gdom2           2013.05.17 10:31:55     2               Min Processors
gdom2           2013.05.17 08:56:35     139             Max Processors

primary         2013.05.17 08:53:06     99              Min Memory
primary         2013.05.17 08:40:44     182             Max Memory
gdom1           2013.05.17 09:56:35     13              Min Memory
gdom1           2013.05.17 08:53:06     20              Max Memory
gdom2           2013.05.17 10:31:55     65              Min Memory
gdom2           2013.05.17 08:56:35     66              Max Memory

Processor Power Consumption in Watts
12 hour's worth of data starting from 2013.05.16 23:17:02
DOMAIN          TIMESTAMP               1 HOUR AVG
primary         2013.05.17 09:37:35     112
gdom1           2013.05.17 09:37:35     15
gdom2           2013.05.17 09:37:35     26

primary         2013.05.17 10:37:35     96
gdom1           2013.05.17 10:37:35     12
gdom2           2013.05.17 10:37:35     21

primary         2013.05.17 11:37:35     85
gdom1           2013.05.17 11:37:35     11
gdom2           2013.05.17 11:37:35     23
...
```

Glossary

API

Application programming interface.

API

Application programming interface.

auditreduce

Command to merge and select audit records from audit trail files (see the auditreduce(8) 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.

bge

Broadcom Gigabit Ethernet driver on Broadcom BCM57xx devices.

BSM

Basic Security Module.

bsmconv

Command to enable the BSM (see the bsmconv(8) man page).

bsmunconv

Command to disable the BSM (see the bsmunconv(8) man page).

CMT

Chip multithreading.

compliance

Determining whether a system's configuration is in compliance with a predefined security profile.

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.

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`(8) man page).

DRM

Dynamic resource management.

DS

Domain Services module.

DVD

Digital versatile disc.

EFI

Extensible firmware interface.

ETM

Encoding Table Management module.

FC_AL

Fiber Channel Arbitrated Loop.

FMA

Fault Management Architecture.

fmd

Oracle Solaris OS fault manager daemon (see the `fmd`(8) man page).

fmthard

Command to populate label on hard disks (see the `fmthard`(8) man page).

`format`

Disk partitioning and maintenance utility (see the `format`(8) man page).

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.

hardening

Modifying Oracle Solaris OS configuration to improve security.

hypervisor

Firmware layer interposed between the operating system and the hardware layer.

I/O

Input/output devices, such as internal disks and PCIe controllers and their attached adapters and devices.

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.

`ioctl`

Input/output control call.

IPMP

Internet Protocol Network Multipathing.

`kaio`

Kernel asynchronous input/output.

KB

Kilobyte.

KU

Kernel update.

LAN

Local-area network.

LDAP

Lightweight Directory Access Protocol.

LDC

Logical domain channel.

`ldm`

Logical Domains Manager utility (see the `ldm`(8) 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.

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`(8) man page).

`metaset`

Command to configure disk sets (see the `metaset`(8) man page).

mhd

Command to perform multihost disk control operations (see the `mhd`(4I) 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.

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.

OID

Object identifier, which is a sequence of numbers that uniquely identifies each object in a MIB.

OVF

Open Virtualization Format.

P2V

Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool. See Chapter 19, Oracle VM Server for SPARC Physical-to-Virtual Conversion Tool in *Oracle VM Server for SPARC 3.3 Administration Guide*.

PA

Physical address.

PCI

Peripheral component interconnect bus.

PCI-X

PCI Extended bus.

PCIe

PCI EXPRESS 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 and SPARC S-series servers. Or, it might be either the entire system or a subset of the system as is the case of supported SPARC M-series servers.

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`(8) man page).

PM

Power management of virtual CPUs and memory.

`praudit`

Command to print contents of an audit trail file (see the `praudit`(8) man page).

PRI

Priority.

RA

Real address.

RAID

Redundant Array of Inexpensive Disks, which enables you to combine independent disks into a logical unit.

RPC

Remote Procedure Call.

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.

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.

service processor (SP)

The SP, also known as the system controller (SC), monitors and runs the physical machine.

SMA

System Management Agent.

SMF

Service Management Facility.

SMI

Structure of Management Information, which defines and groups managed objects for use by a MIB.

SNMP

Simple Network Management Protocol.

SP configuration

Name of the logical domain SP configuration that is saved on the service processor.

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`(8) man page).

SunVTS

Sun Validation Test Suite.

`svcadm`

Manipulates service instances (see the `svcadm`(8) man page).

system controller (SC)

Also see service processor.

TLS

Transport Layer Security.

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`(4I) man page).

UTP

Unshielded twisted pair.

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.

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

A VNIC is a virtual NIC. When configured by using the `dladm create-vnic` command, a virtual NIC behaves like a physical NIC. You can configure a virtual NIC on an SR-IOV virtual function and on a virtual network device (VNET). See Creating Virtual NICs on SR-IOV Virtual Functions and Configuring Virtual NICs on Virtual Network Devices.

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`(8) man page).

volfs

Volume Management file system (see the `volfs`(4FS) 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.

XFP

eXtreme Fast Path.

XML

Extensible Markup Language.

XMPP

Extensible Messaging and Presence Protocol.

ZFS

Zettabyte File System.

`zpool`

ZFS storage pool (see the `zpool`(8) man page).

ZVOL

ZFS Volume Emulation Driver.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Z