Oracle® VM Server for SPARC 2.2 Reference Manual

Part No: E23809
May 2012

120627@25097

Contents

System Administration 5

`ldm`(1M) 6

`ldmconfig`(1M) 65

`ldmd`(1M) 66

`ldmp2v`(1M) 68

R E F E R E N C E

System Administration

Name ldm – command-line interface for the Logical Domains Manager

Synopsis

```
ldm or ldm --help [subcommand]
ldm -V
ldm add-domain -i file
ldm add-domain [cpu-arch=generic|native] [mac-addr=num] [hostid=num]
  [failure-policy=ignore|panic|reset|stop] [extended-mapin-space=on]
  [master=master-ldom1,...,master-ldom4] [max-cores=[num|unlimited]]
  [uuid=uuid] [threading=max-ipc] ldom
ldm add-domain ldom...
ldm set-domain -i file
ldm set-domain [cpu-arch=generic|native] [mac-addr=num] [hostid=num]
  [failure-policy=ignore|panic|reset|stop] [extended-mapin-space=[on|off]]
  [master=[master-ldom1,...,master-ldom4]] [max-cores=[num|unlimited]]
  [threading=[max-throughput|max-ipc]] ldom
ldm remove-domain -a
ldm remove-domain ldom...
ldm list-domain [-e] [-l] [-o format] [-p] [ldom...]
ldm migrate-domain [-f] [-n] [-p filename] source-ldom [user@]target-host[:target-ldom]
ldm add-vcpu [-c] number ldom
ldm set-vcpu [-c] number ldom
ldm remove-vcpu [-c] number ldom
ldm add-core num ldom
ldm add-core cid=core-ID[,core-ID[,...]] ldom
ldm set-core num ldom
ldm set-core cid=[core-ID[,core-ID[,...]]] ldom
ldm remove-core [-f] num ldom
ldm remove-core cid=core-ID[,core-ID[,...]] ldom
ldm add-crypto number ldom
ldm set-crypto [-f] number ldom
ldm remove-crypto [-f] number ldom
ldm add-memory [--auto-adj] size[unit] ldom
ldm add-memory mblock=PA-start:size[,PA-start:size[,...]] ldom
ldm set-memory [--auto-adj] size[unit] ldom
ldm set-memory mblock=[PA-start:size[,PA-start:size[,...]]] ldom
ldm remove-memory [--auto-adj] size[unit] ldom
ldm remove-memory mblock=PA-start:size[,PA-start:size[,...]] ldom
ldm start-reconf ldom
ldm cancel-reconf ldom
ldm cancel-operation (migration | reconf | memdr) ldom
ldm add-io (bus | device | vf-name) ldom
ldm set-io name=value [name=value ...] pf-name
ldm set-io [mac-addr=num] [alt-mac-addrs=[auto|num1,[auto|num2,...]]] [pvid=[pvid]]
  [vid=[vid1,vid2,...]] [mtu=size] [name=value...] vf-name
ldm remove-io (bus | device | vf-name) ldom
ldm list-io [-l] [-p] [pf-name]
ldm list-io -d pf-name
ldm add-vsw [-q] [default-vlan-id=vlan-id] [pvid=port-vlan-id] [vid=vlan-id1,vlan-id2,...]
  [linkprop=phys-state] [mac-addr=num] [net-dev=device] [mode=sc] [mtu=size]
  [id=switch-id] [inter-vnet-link=on|off] vswitch-name ldom
ldm set-vsw [-q] [pvid=port-vlan-id] [vid=vlan-id1,vlan-id2,...] [mac-addr=num]
  [net-dev=device] [linkprop=[phys-state]] [mode=[sc]] [mtu=size]
  [inter-vnet-link=[on|off]] vswitch-name
ldm remove-vsw [-f] vswitch-name
ldm add-vnet [mac-addr=num] [mode=hybrid] [pvid=port-vlan-id] [vid=vlan-id1,vlan-id2,...]
  [linkprop=phys-state] [id=network-id] [mtu=size] if-name vswitch-name ldom
ldm set-vnet [mac-addr=num] [vswitch=vswitch-name] [mode=[hybrid]] [pvid=port-vlan-id]
  [linkprop=[phys-state]] [vid=vlan-id1,vlan-id2,...] [mtu=size] if-name ldom
ldm remove-vnet [-f] if-name ldom
ldm add-vds service-name ldom
ldm remove-vds [-f] service-name
ldm add-vdsdev [-f] [-q] [options={ro,slice,excl}] [mpgroup=mpgroup] backend
  volume-name@service-name
ldm set-vdsdev [-f] options=[{ro,slice,excl}] [mpgroup=mpgroup]
  volume-name@service-name
ldm remove-vdsdev [-f] volume-name@service-name
ldm add-vdisk [timeout=seconds] [id=disk-id] disk-name volume-name@service-name ldom
ldm set-vdisk [timeout=seconds] [volume=volume-name@service-name] disk-name ldom
ldm remove-vdisk [-f] disk-name ldom
ldm add-vdpcs vdpcs-service-name ldom
ldm remove-vdpcs [-f] vdpcs-service-name
ldm add-vdpcc vdpcc-name vdpcs-service-name ldom
ldm remove-vdpcc [-f] vdpcc-name ldom
ldm add-vcc port-range=x-y vcc-name ldom
ldm set-vcc port-range=x-y vcc-name
ldm remove-vcc [-f] vcc-name
ldm set-vcons [port=[port-num]] [group=group] [service=vcc-server] ldom
ldm create-vf [mac-addr=num] [alt-mac-addrs=[auto|num1,[auto|num2,...]]] [pvid=pvid]
  [vid=vid1,vid2,...] [mtu=size] [name=value...] pf-name
ldm destroy-vf vf-name
ldm add-variable var-name=[value]... ldom
ldm set-variable var-name=[value]... ldom
ldm remove-variable var-name... ldom
ldm list-variable [var-name...] ldom
ldm start-domain (-a | -i file | ldom...)
ldm stop-domain [-f] (-a | ldom...)
ldm panic-domain ldom
ldm bind-domain [-f] [-q] (-i file | ldom)
ldm unbind-domain ldom
ldm list-bindings [-e] [-p] [ldom...]
ldm add-spconfig config-name
ldm add-spconfig -r autosave-name [new-config-name]
ldm set-spconfig config-name
ldm set-spconfig factory-default
ldm remove-spconfig [-r] config-name
ldm list-spconfig [-r [autosave-name]]
ldm list-constraints ([-x] | [-e] [-p]) [ldom...]
ldm list-devices [-a] [-p] [core] [cpu] [crypto] [memory] [io]
ldm list-services [-e] [-p] [ldom...]
ldm add-policy [enable=yes|no] [priority=value] [attack=value] [decay=value]
  [elastic-margin=value] [sample-rate=value] [tod-begin=hh:mm[:ss]]
  [tod-end=hh:mm[:ss]] [util-lower=percent] [util-upper=percent] [vcpu-min=value]
  [vcpu-max=value] name=policy-name ldom...
ldm set-policy [enable=[yes|no]] [priority=[value]] [attack=[value]] [decay=[value]]
  [elastic-margin=[value]] [sample-rate=[value]] [tod-begin=[hh:mm:ss]]
  [tod-end=[hh:mm:ss]] [util-lower=[percent]] [util-upper=[percent]] [vcpu-min=[value]]
  [vcpu-max=[value]] name=policy-name ldom...
ldm remove-policy [name=]policy-name... ldom
ldm init-system [-frs] -i file
```

Description The `ldm` command interacts with the Logical Domains Manager and is used to create and manage logical domains. There can be only one Logical Domains Manager per server. The Logical Domains Manager runs on the control domain, which is the initial domain created by the service processor. The control domain is named `primary`.

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

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

Control domain	Creates and manages other logical domains and services by communicating with the hypervisor.
Service domain	Provides services to other logical domains, such as a virtual network switch or a virtual disk service.
I/O domain	Has direct access to a physical I/O device, such as a network card in a PCI EXPRESS (PCIe) controller or a single-root I/O virtualization (SR-IOV) virtual function. An I/O domain can own a PCIe root complex, or it can own a PCIe slot or on-board PCIe device by using the direct I/O (DIO) feature and an SR-IOV virtual function by using the SR-IOV feature. 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	Has a PCIe root complex assigned to it. This domain owns the PCIe fabric and all connected devices, and provides all fabric-related services, such as fabric error handling. A root domain owns all of the SR-IOV physical functions from which you can create virtual functions and

	assign them to I/O domains. A root domain is also an I/O domain, as it owns and has direct access to physical I/O devices. The number of root domains that you can have depends on your platform architecture. For example, if you are using a Sun SPARC Enterprise T5440 server from Oracle, you can have up to four root domains.
Guest domain	Uses services from the I/O and service domains and is managed by the control domain.

You can use the Logical Domains Manager to establish dependency relationships between domains.

Master domain	A domain that has one or more domains that depend on it. A master domain specifies a failure policy to be enacted by its slave domains when the master domain fails. For instance, depending on the master domain's failure policy, a slave can be left as-is, panicked, rebooted, or stopped when the master domain fails.
Slave domain	A domain that depends on another domain. A domain can specify up to four master domains that dictate the failure policy to enact when one or more of the master domains fail.

Subcommand Summaries

Following are the supported subcommands along with a description and required authorization for each. For information about setting up authorization for user accounts, see "Creating Authorizations and Profiles and Assigning Roles to User Accounts" in *Oracle VM Server for SPARC 2.2 Administration Guide*.

Subcommand	Description	Authorization
`add-spconfig`	Adds a logical domain configuration to the service processor (SP).	`solaris.ldoms.write`
`add-domain`	Creates a logical domain.	`solaris.ldoms.write`
`add-policy`	Adds a resource management policy to an existing logical domain.	`solaris.ldoms.write`
`add-`*resource*	Adds a resource to an existing logical domain. See RESOURCES for resource definitions.	`solaris.ldoms.write`
`bind-domain`	Binds resources to a created logical domain.	`solaris.ldoms.write`
`cancel-operation`	Cancels an operation, such as a delayed reconfiguration (`reconf`), memory dynamic reconfiguration (DR) removal (`memdr`), or domain migration (`migration`).	`solaris.ldoms.write`

Subcommand	Description	Authorization
`cancel-reconf`	Cancels a delayed reconfiguration operation on the primary domain.	`solaris.ldoms.write`
`create-vf`	Creates a virtual function.	`solaris.ldoms.write`
`destroy-vf`	Destroys a virtual function.	`solaris.ldoms.write`
`list-`*type*	Lists server resources, including bindings, constraints, devices, services, and configurations for logical domains.	`solaris.ldoms.read`
`list-domain`	Lists logical domains and their states.	`solaris.ldoms.read`
`list-variable`	Lists variables for logical domains.	`solaris.ldoms.read`
`migrate-domain`	Migrates a logical domain from one machine to another.	`solaris.ldoms.write`
`panic-domain`	Panics the Oracle Solaris OS on a specified logical domain.	`solaris.ldoms.write`
`remove-`*resource*	Removes a resource from an existing logical domain. See RESOURCES for resource definitions.	`solaris.ldoms.write`
`remove-domain`	Deletes a logical domain.	`solaris.ldoms.write`
`remove-policy`	Removes a resource management policy from an existing logical domain.	`solaris.ldoms.write`
`remove-spconfig`	Removes a logical domain configuration from the service processor.	`solaris.ldoms.write`
`remove-variable`	Removes one or more variables from an existing logical domain.	`solaris.ldoms.write`

Subcommand	Description	Authorization
`set-`*resource*	Specifies a resource for an existing logical domain. This can be either a property change or a quantity change. This represents a quantity change when applied to the resources `core`, `vcpu`, `memory`, or `crypto`. For a quantity change, the subcommand becomes a dynamic or a delayed reconfiguration operation, where the quantity of the specified resource is assigned to the specified logical domain. If there are more resources assigned to the logical domain than are specified in this subcommand, some are removed. If there are fewer resources assigned to the logical domain than are specified in this subcommand, some are added. See RESOURCES for resource definitions.	`solaris.ldoms.write`
`set-domain`	Sets properties on a logical domain.	`solaris.ldoms.write`
`set-io`	Modifies a physical function or a virtual function.	`solaris.ldoms.write`
`set-policy`	Sets properties for a resource management policy to an existing logical domain.	`solaris.ldoms.write`
`set-spconfig`	Specifies a logical domain configuration to use.	`solaris.ldoms.write`
`set-variable`	Sets one or more variables for an existing logical domain.	`solaris.ldoms.write`
`start-domain`	Starts one or more logical domains.	`solaris.ldoms.write`
`start-reconf`	Enters delayed reconfiguration mode on the primary domain.	`solaris.ldoms.write`
`stop-domain`	Stops one or more running logical domains.	`solaris.ldoms.write`
`unbind-domain`	Unbinds or releases resources from a logical domain.	`solaris.ldoms.write`

Note – Not all subcommands are supported on all resources types.

Aliases The following table shows the three kinds of aliases for `ldm` subcommands.

Alias Type	Short Form	Long Form
Action alias (verb)	`ls`	`list`

Alias Type	Short Form	Long Form
Action alias (verb)	`rm`	`remove`
Resource alias (noun)	`config`	`spconfig`
Resource alias (noun)	`crypto`	`mau`
Resource alias (noun)	`dom`	`domain`
Resource alias (noun)	`mem`	`memory`
Resource alias (noun)	`var`	`variable`
Resource alias (noun)	`vcc`	`vconscon`
Resource alias (noun)	`vcons`	`vconsole`
Resource alias (noun)	`vdpcc`	`ndpsldcc`
Resource alias (noun)	`vdpcs`	`ndpsldcs`
Resource alias (noun)	`vds`	`vdiskserver`
Resource alias (noun)	`vdsdev`	`vdiskserverdevice`
Resource alias (noun)	`vsw`	`vswitch`
Subcommand shortcut	`bind`	`bind-domain`
Subcommand shortcut	`cancel-op`	`cancel-operation`
Subcommand shortcut	`create`	`add-domain`
Subcommand shortcut	`destroy`	`remove-domain`
Subcommand shortcut	`list`	`list-domain`
Subcommand shortcut	`migrate`	`migrate-domain`
Subcommand shortcut	`modify`	`set-domain`
Subcommand shortcut	`panic`	`panic-domain`
Subcommand shortcut	`start`	`start-domain`
Subcommand shortcut	`stop`	`stop-domain`
Subcommand shortcut	`unbind`	`unbind-domain`

Note – In the syntax and examples in the remainder of this man page, the short forms of the action and resource aliases are used.

Resources The following resources are supported:

`core`	CPU cores.

`crypto`	Any supported cryptographic unit on a supported server. Currently, the two cryptographic units supported are the Modular Arithmetic Unit (MAU) and the Control Word Queue (CWQ).
`io`	I/O devices, such as PCIe root complexes and their attached adapters and devices. Also direct I/O-assignable devices and PCIe SR-IOV virtual functions.
`mem`, `memory`	Default memory size in bytes. Or specify gigabytes (`G`), kilobytes (`K`), or megabytes (`M`). Virtualized memory of the server that can be allocated to guest domains.
`vcc`, `vconscon`	Virtual console concentrator service with a specific range of TCP ports to assign to each guest domain at the time it is created.
`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`	Each virtual CPU represents one CPU thread of a server. For example, an 8-core Sun SPARC Enterprise T5120 server has 64 CPU threads (virtual CPUs) that can be allocated among the logical domains.
`vdisk`	Virtual disks are generic block devices backed by different types of physical devices, volumes, or files. A virtual disk is not synonymous with a SCSI disk and, therefore, excludes the target ID (`t`*N*) in the disk name. Virtual disks in a logical domain have the following format: `c`*N*`d`*N*`s`*N*, where `c`*N* is the virtual controller, `d`*N* is the virtual disk number, and `s`*N* is the slice.
`vds`, `vdiskserver`	Virtual disk server that allows you to export virtual disks to other logical domains.
`vdsdev`, `vdiskserverdevice`	Device exported by the virtual disk server. The device can be an entire disk, a slice on a disk, a file, or a disk volume.
`vdpcc`	Virtual data plane channel client. Only of interest in a Netra Data Plane Software (NDPS) environment.
`vdpcs`	Virtual data plane channel service. Only of interest in a Netra Data Plane Software (NDPS) environment.
`vnet`	Virtual network device that implements a virtual Ethernet device and communicates with other `vnet` devices in the system using the virtual network switch (`vsw`).

`vsw`, `vswitch`	Virtual network switch that connects the virtual network devices to the external network and also switches packets between them.

List Types

The following list types are supported:

`bindings`	Lists the resources bound to a logical domain.
`config`	Lists the logical domain configurations stored on the service processor (SP). On SPARC T3 based servers, the configurations are shown in the order of creation from the oldest to the newest.
`constraints`	Lists the constraints used to create a logical domain.
`devices`	Lists all free devices for the server.
`services`	Lists all services exported by a logical domain.

Options

The following table describes the `ldm` command options. The short form of the option is followed by the long form, if applicable.

`-a`	`--all`	Operates on all of the operand types.
	`--auto-adj`	Specifies that the `add-memory`, `set-memory`, and `remove-memory` subcommands align memory changes on a 256-Mbyte boundary. The behavior of the `--auto-adj` option depends on the state of the affected domain. ■ **Active domain.** For *dynamic reconfigurations*, this option aligns the amount of memory to be added or removed to a 256-Mbyte boundary. The amount is rounded up for an `add-memory` operation and rounded down for a `remove-memory` operation. A `set-memory` operation is treated as either an `add-memory` or `remove-memory` operation. For any of these subcommands, the `--auto-adj` option ensures that the *resulting size* of the domain's memory is greater than or equal to the requested size. For *delayed reconfigurations*, this option has the same behavior as a bound or inactive domain. A delayed reconfiguration can occur under the following conditions: ■ The domain initiates a delayed reconfiguration. ■ A delayed reconfiguration is outstanding in the domain. ■ **Bound domain or inactive domain.** This option aligns the *resulting size* of the domain by rounding up to the next 256-Mbyte boundary. This alignment occurs in `add-memory`, `set-memory`, and `remove-memory` operations.

`-c` *number* `--core` *number*

Performs the following discrete CPU operations:

- Sets the allocation unit for the domain from threads to cores, if not already set, and sets the allocation to the specified number of cores.
- If the domain is inactive, sets a cap on the number of cores that can be allocated to the domain when it is bound or active. A cap is set on the `primary` domain *only* if the domain is in a delayed reconfiguration mode.

If any allocation request results in more cores being assigned to a domain than is permitted by the cap, the command fails.

This option configures hard partitioning on your Oracle VM Server for SPARC system. See "Configuring the System With Hard Partitions" in *Oracle VM Server for SPARC 2.2 Administration Guide*.

You can change the allocation unit from cores to threads and remove the cap. Make these changes by issuing an `add-vcpu`, `set-vcpu`, or `rm-vcpu` command without the `-c` option on an inactive domain or on the `primary` domain that is in delayed reconfiguration mode.

Starting with the Oracle VM Server for SPARC 2.2 release, the CPU cap and the allocation of CPU cores is handled by separate commands. By using these commands you can independently allocate CPU cores, set a cap, or both. The allocation unit can be set to cores even when no cap is in place. However, running the system in this mode is *not* acceptable for configuring hard partitioning on your Oracle VM Server for SPARC system.

- Allocate the specified number of CPU cores to a domain by using the `add-core`, `set-core`, and `rm-core` subcommands.
- Set the cap by using the `create-domain` or `set-domain` subcommand to specify the `max-cores` property value.

`-e` `--extended`

Generates an extended listing containing services and devices that are automatically set up, that is, not under your control.

`-f` `--force`

Attempts to force an operation.

`-i` *file* `--input` *file*

Specifies the XML configuration file to use in creating a logical domain.

`-l` `--long`

Generates a long listing.

`-n` `--dry-run`

Makes a dry run of a migration to check to see if the migration will succeed. Does not actually migrate the domain.

`-o` *format*	`--output` *format*	Specifies one or more of the following formats for an `ldm list` command, depending on what you want to see: `console`, `core`, `cpu`, `crypto`, `disk`, `domain`, `memory`, `network`, `physio`, `resmgmt`, `serial`, and `status`. If you specify more than one format, delimit each format by a comma with no spaces.
`-p`	`--parseable`	Generates a machine-readable version of the output.
`-q`		Disables the validation of network or disk back-end devices so that the command runs more quickly.
`-r`		For the `add-spconfig`, `list-spconfig`, and `remove-spconfig` subcommands: Performs a manual configuration recovery.
`-r`	`--reboot`	For the `init-system` subcommand: Reboots the system after configuration.
`-s`	`--services-only`	Restores only the virtual services configuration (`vds`, `vcc`, and `vsw`).
`-x` *file*	`--xml` *file*	Specifies that an XML file containing the constraints for the logical domain be written to standard output (`stdout`). Can be used as backup file.
`-V`	`--version`	Displays version information.
	`--help`	Displays usage statements.

Properties

Note – You can use various `ldm set-*` commands to reset any property to its default value by specifying an empty value. For example, the following `ldm set-policy` command resets the `attack` property to its default value:

```
# ldm set-policy attack= high-policy ldom1
```

The following properties are supported:

`alt-mac-addrs=auto|`*num1*`,[auto|`*num2*`,...]`
Specifies a comma-separated list of alternate MAC addresses. Valid values are numeric MAC addresses and the `auto` keyword, which can be used one or more times to request that the system generate an alternate MAC address. The `auto` keyword can be mixed with numeric MAC addresses.

`attack=`*value*
Specifies the maximum number of resources to be added during any one resource control cycle. If the number of available resources is less than the specified value, all of the available resources are added. By default, the attack is unlimited so that you can add as many CPU threads as are available. Valid values are from 1 to the number of free CPU threads on the system.

`cid=`*core-ID*

Specifies the physical core IDs to assign to or remove from a domain. To remove all named cores, omit *core-ID* values for the `cid` property by running the `ldm set-core cid=` command.

The `cid` property should *only* be used 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.

`cpu-arch=generic|native`

Specifies one of the following values:

- `generic` uses common CPU hardware features to enable a guest domain to perform a CPU-type-independent migration.
- `native` uses CPU-specific hardware features to enable a guest domain to migrate *only* between platforms that have the same CPU type. `native` is the default value.

Using the `generic` value might result in reduced performance compared to the `native` value. This occurs because the guest domain does not use some features that are only present in newer CPU types. By not using these features, the `generic` setting enables the flexibility of migrating the domain between systems that use newer and older CPU types.

`decay=`*value*

Specifies the maximum number of resources to be removed during any one resource control cycle. Only the number of currently bound CPU threads minus the value of `vcpu-min` can be removed even if the value specified by this property is larger. By default, the value is 1. Valid values are from 1 to the total number of CPU threads minus 1.

`default-vlan-id=`

Specifies the default virtual local area network (VLAN) to which a virtual network device or virtual switch needs to be a member, in tagged mode. The first VLAN ID (*vid1*) is reserved for the `default-vlan-id`.

`elastic-margin=`*value*

Specifies the amount of buffer space between `util-lower` and the number of free CPU threads to avoid oscillations at low CPU thread counts. Valid values are from 0 to 100. The default value is 5.

`enable=yes|no`

Enables or disables resource management for an individual domain. By default, `enable=yes`.

`extended-mapin-space=on|off`

Enables or disables extended mapin space for a domain. By default, `extended-mapin-space=off`.

The extended mapin space refers to the additional LDC shared memory space. This memory space is required to support a large number of virtual I/O devices that use direct-mapped shared memory. This extended mapin space is also used by virtual network devices to improve performance and scalability.

`failure-policy=`
Specifies the master domain's failure policy, which controls how slave domains behave when the master domain fails. This property is set on a master domain. The default value is `ignore`. Following are the valid property values:

- `ignore` ignores failures of the master domain (slave domains are unaffected).
- `panic` panics any slave domains when the master domain fails.
- `reset` resets any slave domains when the master domain fails.
- `stop` stops any slave domains when the master domain fails.

`group=`
Specifies a group to which to attach a console. The group argument allows multiple consoles to be multiplexed onto the same TCP connection.

`hostid=`
Specifies the host ID for a particular domain. If you do not specify a host ID, the Logical Domains Manager assigns a unique host ID to each domain.

`id=`
Specifies an ID for a new virtual disk device, virtual network device, and virtual switch device, respectively.

`inter-vnet-link=on|off`
Specifies whether to assign a channel between each virtual network device. The default value is `on`.

When `inter-vnet-link=on`, the Logical Domains Manager assigns a channel between each pair of virtual network devices that are connected to the same virtual switch for better guest-to-guest performance.

When `inter-vnet-link=off`, the Logical Domains Manager *only* assigns channels for communications between virtual network devices and virtual switches. In this case, guest-to-guest communications traffic goes through the virtual switch. This setting reduces the number of channels that are used for virtual network devices. Thus, the maximum number of virtual devices that you can add to the system is increased.

`linkprop=phys-state`
Specifies whether the virtual device reports its link status based on the underlying physical network device. When `linkprop=phys-state` is specified on the command line, the virtual device link status reflects the physical link state. By default, the virtual device link status does not reflect the physical link state.

`mac-addr=`
Defines a MAC address. The number must be in standard octet notation, for example, `80:00:33:55:22:66`.

`master=`

Specifies the name of up to four master domains for a slave domain. This property is set on a slave domain. By default, there are no masters for the domain. The domain must already exist prior to an `ldm add-domain` operation.

Note – The Logical Domains Manager does not permit you to create domain relationships that result in a dependency cycle.

`max-cores=`*num*`|unlimited`

Specifies the maximum number of cores that are permitted to be assigned to a domain. If the value is `unlimited`, there is no constraint on the number of CPU cores that can be allocated.

`mblock=`*PA-start*`:`*size*

Specifies one or more physical memory blocks to assign to or remove from a domain. *PA-start* specifies the starting physical address of the memory block in hexadecimal format. *size* is the size of the memory block, including a unit, to be assigned to or removed from the domain. To remove all named memory blocks, omit *PA-start*:*size* values from the `mblock` property by running the `ldm set-memory mblock=` command.

The `mblock` property should *only* be used 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.

`mode=`

For `add-vsw` and `set-vsw` subcommands:

Omit this option when you are not running Oracle Solaris Cluster software in guest domains because you could impact virtual network performance.

Otherwise, specify one of the following:

- Set `mode=sc` to enable virtual networking support for prioritized processing of Oracle Solaris Cluster heartbeat packets in a Logical Domains environment.
- Leave the `mode=` argument blank in the `set-vsw` subcommand to stop special processing of heartbeat packets.

For `add-vnet` and `set-vnet` subcommands:

Omit this option when you do not want to use NIU Hybrid I/O.

Otherwise, specify one of the following:

- Set `mode=hybrid` to request the system to use NIU Hybrid I/O if possible. If it is not possible, the system reverts to virtual I/O. See "Using NIU Hybrid I/O" in *Oracle VM Server for SPARC 2.2 Administration Guide*.
- Leave the `mode=` argument blank in the `set-vnet` subcommand to disable NIU Hybrid I/O.

`mpgroup=`
Defines the multipath group name for several virtual disk server devices (`vdsdev`). So, when a virtual disk cannot communicate with a virtual disk server device, a failover is initiated to another virtual disk server device in the multipath group.

`mtu=`
Specifies the maximum transmission unit (MTU) of a virtual switch, virtual network devices that are bound to the virtual switch, or both. Valid values are in the range of `1500-16000`. The `ldm` command issues an error if an invalid value is specified.

`name=`*policy-name*
Specifies the resource management policy name.

`net-dev=`
Defines the path name of the actual network device.

`options=`
Specifies all or a subset of the following options for a specific virtual disk server device. Separate two or more options with commas and no spaces, such as `ro,slice,excl`.

- `ro` – Specifies read-only access
- `slice` – Exports a back end as a single slice disk
- `excl` – Specifies exclusive disk access

Omit the `options=` argument or leave it blank in an `add-vdsdev` subcommand to have the default values of disk, not exclusive, and read/write. Leave the options= argument blank in the `set-vdsdev` subcommand to turn off any previous options specified.

`port=`
Specifies a specific port number or, left blank, lets the Logical Domains Manager set the port number.

`port-range=`
Defines a range of TCP ports.

`priority=`*value*
Specifies a priority for dynamic resource management (DRM) policies. Priority values are used to determine the relationship between DRM policies in a single domain and between DRM-enabled domains in 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 whether a pool of free CPU resources is available, 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 in 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.

`pvid=`
: Specifies the VLAN to which the virtual network device needs to be a member, in untagged mode.

`sample-rate=`*value*
: Specifies the cycle time, in seconds, which is the sample rate for DRM. Valid values are from 1 to 9999. The default and recommended value is 10.

`service=`
: Specifies the name of the existing virtual console concentrator that you want to handle the console connection.

`threading=max-ipc|max-throughput`
: The `threading` property specifies the workflow throughput of the domain.

The following are valid values for the `threading` property:

- `max-ipc`. Only one thread is active for each CPU core that is assigned to the domain, which maximizes the number of instructions per cycle. Selecting this mode requires that the domain is also configured with the whole-core constraint. See the `add-vcpu` and `set-vcpu` subcommand descriptions.
- `max-throughput`. Activates all threads that are assigned to the domain, which maximizes throughput. This mode is used by default and is also selected if you do not specify any mode (`threading=`).

`timeout=`
: Defines the number of seconds for establishing a connection between a virtual disk client (`vdc`) and a virtual disk server (`vds`). If there are multiple virtual disk (`vdisk`) paths, then

the `vdc` can try to connect to a different `vds`, and the timeout ensures that a connection to any `vds` is established within the specified amount of time. Specify `0` to disable the timeout in the `set-vdisk` subcommand.

`tod-begin=`*hh*`:`*mm*`[:`*ss*`]`
Specifies the effective start time of a policy in terms of hour, minute, and optional second. This time must be earlier than the time specified by `tod-end` in a period that begins at midnight and ends at 23:59:59. The default value is `00:00:00`.

`tod-end=`*hh*`:`*mm*`[:`*ss*`]`
Specifies the effective stop time of a policy in terms of hour, minute, and optional second. This time must be later than the time specified by `tod-begin` in a period that begins at midnight and ends at 23:59:59. The default value is `23:59:59`.

`util-lower=`*percent*
Specifies the lower utilization level at which policy analysis is triggered. Valid values are from 1 to `util-upper` minus 1. The default value is 60.

`util-upper=`*percent*
Specifies the upper utilization level at which policy analysis is triggered. Valid values are from `util-lower` plus 1 to 99. The default value is 85.

`uuid=`*uuid*
Specifies the universally unique identifier (UUID) for the domain. *uuid* is a hexadecimal string, such as `12345678-1234-abcd-1234-123456789abc`, which consists of five hexadecimal numbers separated by dashes. Each number must have the specified number of hexadecimal digits: 8, 4, 4, 4, and 12, as follows:

xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx

`vcpu-max=`*value*
Specifies the maximum number of CPU thread resources for a domain. By default, the maximum number of CPU threads is unlimited. Valid values are from `vcpu-min` plus 1 to the total number of free CPU threads on the system.

`vcpu-min=`*value*
Specifies the minimum number of CPU thread resources for a domain. Valid values are from 1 to `vcpu-max` minus 1. The default value is 1.

`vid=`
Specifies the VLAN to which a virtual network device or virtual switch needs to be a member, in tagged mode.

`volume=`
Changes a volume name for a virtual disk.

`vswitch=`
Changes a virtual switch name for a virtual network.

Flags in `list` Subcommand Output

Following are definitions of the flags in the `list` subcommand output:

`-`	Placeholder
`c`	Control domain
`d`	Delayed reconfiguration
`e`	Error
`n`	Normal
`r`	Memory DR in progress
`s`	Column 1 – starting or stopping Column 6 – source domain
`t`	Column 2 – transition Column 6 – target domain
`v`	Virtual I/O service domain

The list flag values are position dependent. Following are the values that can appear in each of the five columns from left to right.

TABLE 1 List Flag Positions

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
`s` or `-`	`n` or `t`	`d`, `r`, or `-`	`c` or `-`	`v` or `-`	`s`, `t`, or `e`

Subcommand Usage

This section contains descriptions of every supported command-line interface (CLI) operation, that is, every subcommand and resource combination.

Add, Set, Remove, and Migrate Domains

Add Domains

This subcommand adds one or more logical domains by specifying one or more logical domain names or by using an XML configuration file. You can also specify property values to customize the domain, such as the MAC address, the host ID, a list of master domains, and a failure policy. If you do not specify these property values, the Logical Domains Manager automatically assigns default values.

```
ldm add-dom -i file
ldm add-dom [cpu-arch=generic|native] [mac-addr=num] [hostid=num]
  [failure-policy=ignore|panic|reset|stop] [extended-mapin-space=on]
  [master=master-ldom1,...,master-ldom4] [max-cores=[num|unlimited]]
  [uuid=uuid] [threading=max-ipc] ldom
ldm add-dom ldom...
```

where:

- `-i` *file* specifies the XML configuration file to use in creating the logical domain.
- `cpu-arch=generic|native` specifies one of the following values:
  - `generic` configures a guest domain for a CPU-type-independent migration.
  - `native` configures a guest domain to migrate only between platforms that have the same CPU type. `native` is the default value.
- `mac-addr=`*num* is the MAC address for this domain. The number must be in standard octet notation, for example, `80:00:33:55:22:66`.
- `hostid` specifies the host ID for a particular domain. If you do not specify a host ID, the Logical Domains Manager assigns a unique host ID to each domain.
- `failure-policy` specifies the master domain's failure policy, which controls how slave domains behave when the master domain fails. This property is set on a master domain. The default value is `ignore`. Following are the valid property values:
  - `ignore` ignores failures of the master domain (slave domains are unaffected).
  - `panic` panics any slave domains when the master domain fails.
  - `reset` resets any slave domains when the master domain fails.
  - `stop` stops any slave domains when the master domain fails.
- `extended-mapin-space=on` enables the extended mapin space for the specified domain. By default, the extended mapin space is disabled.
- `master` specifies the name of up to four master domains for a slave domain. This property is set on a slave domain. By default, there are no masters for the domain. The master domain must exist prior to an `ldm add-domain` operation.

  Note – The Logical Domains Manager does not permit you to create domain relationships that result in a dependency cycle.
- Setting the `threading` property specifies the workflow throughput of the domain.

  The following are valid values for the `threading` property:
  - `max-ipc`. Only one thread is active for each CPU core that is assigned to the domain, which maximizes the number of instructions per cycle. Selecting this mode requires that the domain is also configured with the whole-core constraint. See the `add-vcpu` and `set-vcpu` subcommand descriptions.
  - `max-throughput`. Activates all threads that are assigned to the domain, which maximizes throughput. This mode is used by default and is also selected if you do not specify any mode (`threading=`).
- `uuid=`*uuid* specifies the universally unique identifier (UUID) for the domain. *uuid* is a hexadecimal string, such as `12345678-1234-abcd-1234-123456789abc`, which consists of five hexadecimal numbers separated by dashes. Each number must have the specified number of hexadecimal digits: 8, 4, 4, 4, and 12, as follows:

  xxxxxxxx-*xxxx*-*xxxx*-*xxxx*-*xxxxxxxxxxxx*

- `max-cores=[`*num*`|unlimited]` specifies the maximum number of cores that are permitted to be assigned to a domain. If the value is `unlimited`, there is no constraint on the number of CPU cores that can be allocated.
- *ldom* specifies the logical domain to be added.

Set Options for Domains

This subcommand enables you to modify *only* the `mac-addr`, `hostid`, `failure-policy`, `extended-mapin-space`, `master`, `max-cores`, and `threading` properties of each domain. You *cannot* use this command to update resource properties.

Note – If the slave domain is bound, all of its specified master domains must also be bound prior to invoking the `ldm set-domain` command.

```
ldm set-dom -i file
ldm set-dom [cpu-arch=generic|native] [mac-addr=num] [hostid=num]
  [failure-policy=ignore|panic|reset|stop] [extended-mapin-space=[on|off]]
  [master=[master-ldom1,...,master-ldom4]] [max-cores=[num|unlimited]]
  [threading=[max-throughput|max-ipc]] ldom
```

where:

- `-i` *file* specifies the XML configuration file to use in creating the logical domain.

  Only the `ldom_info` nodes specified in the XML file are parsed. Resource nodes, such as `vcpu`, `mau`, and `memory`, are ignored.
- `cpu-arch=generic|native` specifies one of the following values:
  - `generic` configures a guest domain for a CPU-type-independent migration.
  - `native` configures a guest domain to migrate only between platforms that have the same CPU type. `native` is the default value.
- `mac-addr=`*num* is the MAC address for this domain. The number must be in standard octet notation, for example, `80:00:33:55:22:66`.
- `hostid` specifies the host ID for a particular domain. If you do not specify a host ID, the Logical Domains Manager assigns a unique host ID to each domain.
- `failure-policy` specifies the master domain's failure policy, which controls how slave domains behave when the master domain fails. This property is set on a master domain. The default value is `ignore`. Following are the valid property values:
  - `ignore` ignores failures of the master domain (slave domains are unaffected).
  - `panic` panics any slave domains when the master domain fails.
  - `reset` resets any slave domains when the master domain fails.
  - `stop` stops any slave domains when the master domain fails.
- `extended-mapin-space` enables or disables the extended mapin space for the specified domain. By default, the `extended-mapin-space=off`, which is equivalent to setting `extended-mapin-space=`.

- `master` specifies the name of up to four master domains for a slave domain. This property is set on a slave domain. By default, there are no masters for the domain. The master domain must already exist prior to this operation.

  Note – The Logical Domains Manager does not permit you to create domain relationships that result in a dependency cycle.
- Setting the `threading` property specifies the workflow throughput of the domain.

  The following are valid values for the `threading` property:
  - `max-ipc`. Only one thread is active for each CPU core that is assigned to the domain, which maximizes the number of instructions per cycle. Selecting this mode requires that the domain is also configured with the whole-core constraint. See the `add-vcpu` and `set-vcpu` subcommand descriptions.
  - `max-throughput`. Activates all threads that are assigned to the domain, which maximizes throughput. This mode is used by default and is also selected if you do not specify any mode (`threading=`).
- `max-cores=[`*num*`|unlimited]` specifies the maximum number of cores that are permitted to be assigned to a domain. If the value is `unlimited`, there is no constraint on the number of CPU cores that can be allocated.
- *ldom* specifies the name of the logical domain for which you want to set options.

Remove Domains

This subcommand removes one or more logical domains.

`ldm rm-dom -a`
`ldm rm-dom` *ldom*...

where:

- `-a` deletes all logical domains except the control domain.
- *ldom* specifies the logical domain to be deleted.

  In the event that the domain to be destroyed is specified as a master domain, references to this domain are removed from all slave domains.

Migrate Logical Domains

This subcommand migrates a domain from one location to another.

`ldm migrate-domain [-f] [-n] [-p` *filename*] *source-ldom* [*user*@]*target-host*[:*target-ldom*]

where:

- `-f` attempts to force the migration of the domain.
- `-n` performs a dry run on the migration to determine whether it will succeed. It does not actually migrate the domain.

- `-p` *filename* reads the password needed on the target machine from the first line of *filename*. This option enables you to perform non-interactive migrations that do not require you to provide the target machine password at a prompt.

  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).
- *source-ldom* is the logical domain that you want to migrate.
- *user* is the user name that is authorized to run the Logical Domains Manager on the target host. If no user name is specified, the name of the user running the command is used by default.
- *target-host* is the host where you want to place the *target-ldom*.
- *target-ldom* is the logical domain name to be used on the target machine. The default is to keep the domain name used on the source domain (*source-ldom*).

Reconfiguration Operations

Logical Domains supports the following types of reconfiguration operations:

- **Dynamic reconfiguration operations.** DR is the ability to add, set, or remove resources to or from an active domain. The ability to perform dynamic reconfiguration of a particular resource type is dependent on having support in the particular version of the OS running in the logical domain. For the control domain, if a DR cannot be done, a delayed reconfiguration operation is done instead. You can also manually enter delayed reconfiguration mode on the control domain by running the `ldm start-reconf primary` command.
- **Delayed reconfiguration operations.** In contrast to DR operations that take place immediately, delayed reconfiguration operations take effect after the next reboot of the OS or stop and start of the logical domain if no OS is running. Delayed reconfiguration operations can only be performed on the control domain. Other domains must be stopped prior to modifying resources that cannot be dynamically configured.

See "Resource Reconfiguration" in *Oracle VM Server for SPARC 2.2 Administration Guide* for more information about dynamic reconfiguration and delayed reconfiguration.

CPU Operations

You can allocate either CPU threads or CPU cores to a domain. To allocate CPU threads, use the `add-vcpu`, `set-vcpu`, and `remove-vcpu` subcommands. To allocate CPU cores, use the `add-core`, `set-core`, and `remove-core` subcommands.

Add CPU Threads

This subcommand adds the specified number of CPU threads or CPU cores to a logical domain. Note that a domain *cannot* be configured simultaneously with CPU cores and CPU threads. CPU core configurations and CPU thread configurations are mutually exclusive.

`ldm add-vcpu [-c]` *number ldom*

where:

- `-c` performs the following discrete CPU operations:
  - Sets the allocation unit for the domain from threads to cores, if not already set, and adds the specified number of cores to the domain.
  - If the domain is inactive, sets a cap on the number of cores that can be allocated to the domain when it is bound or active. A cap is set on the `primary` domain *only* if the domain is in a delayed reconfiguration mode.

  If any allocation request results in more cores being assigned to a domain than is permitted by the cap, the command fails.

  This option configures hard partitioning on your Oracle VM Server for SPARC system. See "Configuring the System With Hard Partitions" in *Oracle VM Server for SPARC 2.2 Administration Guide*.

  You can change the allocation unit from cores to threads and remove the cap. Make these changes by issuing an `add-vcpu`, `set-vcpu`, or `rm-vcpu` command without the `-c` option on an inactive domain or on the `primary` domain that is in delayed reconfiguration mode.

  Starting with the Oracle VM Server for SPARC 2.2 release, the CPU cap and the allocation of CPU cores is handled by separate commands. By using these commands you can independently allocate CPU cores, set a cap, or both. The allocation unit can be set to cores even when no cap is in place. However, running the system in this mode is *not* acceptable for configuring hard partitioning on your Oracle VM Server for SPARC system.
  - Allocate the specified number of CPU cores to a domain by using the `add-core`, `set-core`, and `rm-core` subcommands.
  - Set the cap by using the `create-domain` or `set-domain` subcommand to specify the `max-cores` property value.
- When the `-c` option is *not* specified, *number* is the number of CPU threads to be added to the logical domain. When the `-c` option is specified, *number* is the number of CPU cores to be added to the logical domain.
- *ldom* specifies the logical domain where the CPU threads are to be added.

Set CPU Threads

This subcommand specifies the number of CPU threads or CPU cores to be set in a logical domain. Note that a domain *cannot* be configured simultaneously with CPU cores and CPU threads. CPU core configurations and CPU thread configurations are mutually exclusive.

`ldm set-vcpu [-c]` *number ldom*

where:

- `-c` performs the following discrete CPU operations:
    - Sets the allocation unit for the domain from threads to cores, if not already set, and sets the allocation to the specified number of cores.
    - If the domain is inactive, sets a cap on the number of cores that can be allocated to the domain when it is bound or active. A cap is set on the `primary` domain *only* if the domain is in a delayed reconfiguration mode.

    If any allocation request results in more cores being assigned to a domain than is permitted by the cap, the command fails.

    This option configures hard partitioning on your Oracle VM Server for SPARC system. See "Configuring the System With Hard Partitions" in *Oracle VM Server for SPARC 2.2 Administration Guide*.

    You can change the allocation unit from cores to threads and remove the cap. Make these changes by issuing an `add-vcpu`, `set-vcpu`, or `rm-vcpu` command without the `-c` option on an inactive domain or on the `primary` domain that is in delayed reconfiguration mode.

    Starting with the Oracle VM Server for SPARC 2.2 release, the CPU cap and the allocation of CPU cores is handled by separate commands. By using these commands you can independently allocate CPU cores, set a cap, or both. The allocation unit can be set to cores even when no cap is in place. However, running the system in this mode is *not* acceptable for configuring hard partitioning on your Oracle VM Server for SPARC system.

    - Allocate the specified number of CPU cores to a domain by using the `add-core`, `set-core`, and `rm-core` subcommands.
    - Set the cap by using the `create-domain` or `set-domain` subcommand to specify the `max-cores` property value.
- When the `-c` option is *not* specified, *number* is the number of CPU threads to be set in a logical domain. When the `-c` option is specified, *number* is the number of CPU cores to be set in a logical domain.
- *ldom* is the logical domain where the number of CPU threads are to be set.

Remove CPU Threads

This subcommand removes the specified number of CPU threads or CPU cores from a logical domain. Note that a domain *cannot* be configured simultaneously with CPU cores and CPU threads. CPU core configurations and CPU thread configurations are mutually exclusive.

`ldm rm-vcpu [-c]` *number ldom*

where:

- `-c` performs the following discrete CPU operations:
  - Sets the allocation unit for the domain from threads to cores, if not already set, and removes the specified number of cores from the domain.
  - If the domain is inactive, sets a cap on the number of cores that can be allocated to the domain when it is bound or active. A cap is set on the `primary` domain *only* if the domain is in a delayed reconfiguration mode.

  If any allocation request results in more cores being assigned to a domain than is permitted by the cap, the command fails.

  This option configures hard partitioning on your Oracle VM Server for SPARC system. See "Configuring the System With Hard Partitions" in *Oracle VM Server for SPARC 2.2 Administration Guide*.

  You can change the allocation unit from cores to threads and remove the cap. Make these changes by issuing an `add-vcpu`, `set-vcpu`, or `rm-vcpu` command without the `-c` option on an inactive domain or on the `primary` domain that is in delayed reconfiguration mode.

  Starting with the Oracle VM Server for SPARC 2.2 release, the CPU cap and the allocation of CPU cores is handled by separate commands. By using these commands you can independently allocate CPU cores, set a cap, or both. The allocation unit can be set to cores even when no cap is in place. However, running the system in this mode is *not* acceptable for configuring hard partitioning on your Oracle VM Server for SPARC system.
  - Allocate the specified number of CPU cores to a domain by using the `add-core`, `set-core`, and `rm-core` subcommands.
  - Set the cap by using the `create-domain` or `set-domain` subcommand to specify the `max-cores` property value.
- When the `-c` option is *not* specified, *number* is the number of CPU threads to be removed from the logical domain. When the `-c` option is specified, *number* is the number of CPU cores to be removed from the logical domain.
- *ldom* specifies the logical domain where the CPU threads are to be removed.

Add CPU Cores

This subcommand adds the specified number of CPU cores to a domain. When you specify the number of CPU cores, the cores to be assigned are automatically selected. However, when you specify a *core-ID* value to the `cid` property, the specified cores are explicitly assigned.

The `cid` property should *only* be used 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 explicitly assign cores to a domain as long as power management is *not* using the elastic policy.

ldm add-core *num ldom*
ldm add-core cid=*core-ID***[,***core-ID***[,...]]** *ldom*

where:

- *num* specifies the number of CPU cores to assign to a domain.
- `cid=`*core-ID*`[,...]` specifies one or more physical CPU cores to assign to a domain.
- *ldom* specifies the domain to which the CPU cores are assigned.

Set CPU Cores

This subcommand specifies the number of CPU cores to assign to a domain. When you specify the number of CPU cores, the cores to be assigned are automatically selected. However, when you specify a *core-ID* value to the `cid` property, the specified cores are explicitly assigned.

You can explicitly assign cores to a domain as long as power management is *not* using the elastic policy.

ldm set-core *num ldom*
ldm set-core cid=[*core-ID***[,***core-ID***[,...]]]** *ldom*

where:

- *num* specifies the number of CPU cores to assign to a domain.
- `cid=`*core-ID*`[,...]` specifies one or more physical CPU cores to assign to a domain. `cid=` removes all named CPU cores.
- *ldom* specifies the domain to which the CPU cores are assigned.

Remove CPU Cores

This subcommand specifies the number of CPU cores to remove from a domain. When you specify the number of CPU cores, the cores to be removed are automatically selected. However, when you specify a *core-ID* value to the `cid` property, the specified cores are explicitly removed.

You can explicitly remove cores from a domain as long as power management is *not* using the elastic policy.

ldm remove-core [-f] *num ldom*
ldm remove-core cid=[*core-ID***[,***core-ID***[,...]]]** *ldom*

where:

- `-f` attempts to force the removal of one or more cores from an active domain.
- *num* specifies the number of CPU cores to remove from a domain.
- `cid=`*core-ID*`[,...]` specifies one or more physical CPU cores to remove from a domain.
- *ldom* specifies the domain from which the CPU cores are removed.

Cryptographic Unit Operations

The cryptographic unit subcommands only pertain to SPARC platforms that have discrete cryptographic units.

Add Cryptographic Units

This subcommand specifies the number of cryptographic units to be added to a logical domain. Currently, the supported cryptographic units on supported servers are the Modular Arithmetic Unit (MAU) and the Control Word Queue (CWQ).

`ldm add-crypto` *number ldom*

where:

- *number* is the number of cryptographic units to be added to the logical domain.
- *ldom* specifies the logical domain where the cryptographic units are to be added.

Set Cryptographic Units

This subcommand specifies the number of cryptographic units to be set in a logical domain. If you want to remove all cryptographic units from an active domain, you must specify the `-f` option.

To remove the last cryptographic unit from the `primary` domain when domains are active, you must do one of the following:

- Use DR and specify the `-f` option
- Use delayed reconfiguration

`ldm set-crypto [-f]` *number ldom*

where:

- `-f` forces the removal of the last cryptographic unit in the domain if *number* is `0`.

  The `-f` option is only necessary in the following situations:
  - When the guest domain is active
  - On the `primary` domain, but *only* if at least one active guest domain exists on the system
- *number* is the number of cryptographic units to be set in the logical domain.
- *ldom* specifies the logical domain where the number of cryptographic units are to be set.

Remove Cryptographic Units

This subcommand removes the specified number of cryptographic units from a logical domain. If you want to remove all cryptographic units from an active domain, you must specify the `-f` option.

To remove the last cryptographic unit from the `primary` domain when domains are active, you must do one of the following:

- Use DR and specify the `-f` option
- Use delayed reconfiguration

`ldm rm-crypto [-f]` *number ldom*

where:

- `-f` forces the removal of the last cryptographic unit in the domain if *number* is equal to the number of cryptographic units in the domain.

  The `-f` option is only necessary in the following situations:

  - When the guest domain is active
  - On the `primary` domain, but *only* if at least one active guest domain exists on the system
- *number* is the number of `cryptographic` units to be removed from the logical domain.
- *ldom* specifies the logical domain where the cryptographic units are to be removed.

Memory Operations

Add Memory

This subcommand adds the specified amount of memory to a domain. When you specify a memory block size, the memory block to be assigned is automatically selected. However, when you specify a *PA-start*`:`*size* value to the `mblock` property, the specified memory blocks are explicitly assigned.

The `mblock` property should *only* be used 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.

`ldm add-mem [--auto-adj]` *size*`[`*unit*`]` *ldom*
`ldm add-mem mblock=`*PA-start*`:`*size*`[,`*PA-start*`:`*size*`[,...]]` *ldom*

where:

- `--auto-adj` specifies that the amount of memory to be added to an active domain is automatically 256Mbyte-aligned, which might increase the requested memory size. If the domain is inactive, bound, or in a delayed reconfiguration, this option automatically aligns the resulting size of the domain by rounding up to the next 256-Mbyte boundary.
- *size* is the size of memory to be added to a logical domain.
- *unit* is the unit of measurement. The default is bytes. If you want a different unit of measurement, specify one of the following (the *unit* is not case-sensitive):
  - `G` is gigabytes
  - `K` is kilobytes

  - `M` is megabytes
- `mblock=`*PA-start*`:`*size* specifies one or more physical memory blocks to assign to a domain. *PA-start* specifies the starting physical address of the memory block in hexadecimal format. *size* is the size of the memory block, including a unit, to be assigned to the domain.
- *ldom* specifies the logical domain where the memory is to be added.

Set Memory

This subcommand sets a specific amount of memory in a domain. Depending on the amount of memory specified, this subcommand is treated as an `add-memory` or `remove-memory` operation.

When you specify a memory block size, the memory block to be assigned is automatically selected. However, when you specify a *PA-start*`:`*size* value to the `mblock` property, the specified memory blocks are explicitly assigned.

`ldm set-mem [--auto-adj]` *size*`[`*unit*`]` *ldom*
`ldm set-mem mblock=`*PA-start*`:`*size*`[,`*PA-start*`:`*size*`[,...]]` *ldom*

where:

- `--auto-adj` specifies that the amount of memory to be set on an active domain is automatically 256Mbyte-aligned, which might increase the requested memory size. If the domain is inactive, bound, or in a delayed reconfiguration, this option automatically aligns the resulting size of the domain by rounding up to the next 256-Mbyte boundary.
- *size* is the size of memory to be set in the logical domain.
- *unit* is the unit of measurement. The default is bytes. If you want a different unit of measurement, specify one of the following (the *unit* is not case-sensitive):
  - `G` is gigabytes
  - `K` is kilobytes
  - `M` is megabytes
- `mblock=`*PA-start*`:`*size* specifies one or more physical memory blocks to assign to a domain. *PA-start* specifies the starting physical address of the memory block in hexadecimal format. *size* is the size of the memory block, including a unit, to be assigned to the domain.
- *ldom* specifies the logical domain where the memory is to be modified.

Remove Memory

This subcommand removes the specified amount of memory from a logical domain. When you specify a memory block size, the memory block to be removed is automatically selected. However, when you specify a *PA-start*`:`*size* value to the `mblock` property, the specified memory blocks are explicitly removed.

`ldm rm-mem [--auto-adj]` *size*`[`*unit*`]` *ldom*
`ldm rm-mem mblock=`*PA-start*`:`*size*`[,`*PA-start*`:`*size*`[,...]]` *ldom*

where:

- `--auto-adj` specifies that the amount of memory to be removed from an active domain is automatically 256Mbyte-aligned, which might increase the requested memory size. If the domain is inactive, bound, or in a delayed reconfiguration, this option automatically aligns the resulting size of the domain by rounding up to the next 256-Mbyte boundary.
- *size* is the size of memory to be removed from the logical domain.
- *unit* is the unit of measurement. The default is bytes. If you want a different unit of measurement, specify one of the following (the *unit* is not case-sensitive):
  - `G` is gigabytes
  - `K` is kilobytes
  - `M` is megabytes
- `mblock=`*PA-start*`:`*size* specifies one or more physical memory blocks to remove from a domain. *PA-start* specifies the starting physical address of the memory block in hexadecimal format. *size* is the size of the memory block, including a unit, to be removed from the domain.
- *ldom* specifies the logical domain where the memory is to be removed.

Enter Delayed Reconfiguration Mode

This subcommand enables the domain to enter delayed reconfiguration mode. `primary` is the only valid value for *ldom*.

```
ldm start-reconf ldom
```

Cancel a Delayed Reconfiguration Operation

This subcommand cancels a delayed reconfiguration. `primary` is the only valid value for *ldom*.

```
ldm cancel-reconf ldom
```

Cancel Operations

This subcommand cancels a delayed reconfiguration (`reconf`), memory DR removal (`memdr`), or domain migration (`migration`) for a logical domain. For the `reconf` operation, `primary` is the only valid value for *ldom*.

```
ldm cancel-op migration ldom
ldm cancel-op reconf ldom
ldm cancel-op memdr ldom
```

Input/Output Devices

Add Input/Output Device

This subcommand adds a PCI bus or device to a specified logical domain.

```
ldm add-io (bus | device | vf-name) ldom
```

where:

- *bus*, *device*, and *vf-name* are a PCIe bus, a direct I/O-assignable device, and a PCIe SR-IOV virtual function, respectively. Although the operand can be specified as a device path or as a pseudonym, using the device pseudonym is recommended. The pseudonym is based on the ASCII label that is printed on the chassis to identify the corresponding I/O card slot and is platform specific.

  The following are examples of the pseudonyms that are associated with the device names:
  - **PCIe bus.** The `pci_0` pseudonym matches the `pci@400` device path.
  - **Direct I/O-assignable device.** The `PCIE1` pseudonym matches the `pci@400/pci@0/pci@c` device path.
  - **PCIe SR-IOV virtual function.** The `/SYS/MB/NET0/IOVNET.PF0.VF0` pseudonym matches the `pci@400/pci@2/pci@0/pci@6/network@0` device path.

  The specified guest domain must be in the inactive or bound state. If you specify the `primary` domain, this command initiates a delayed reconfiguration.
- *ldom* specifies the logical domain where the bus or device is to be added.

Set a Property for a Virtual Function

This subcommand modifies the current configuration of a virtual function by changing the property values or by passing new properties. This command can modify both the class-specific properties and the device-specific properties.

You can change most network class-specific properties without requiring a reboot of the `primary` domain. Changes to the `mtu` property do require a reboot. You *must* stop the domain if you change a virtual function MAC address.

- All device-specific properties initiate a delayed reconfiguration so that those properties can be updated during the attach operation of the physical function device driver. As a result, the `primary` domain must be rebooted.
- This command only succeeds when the physical function driver can successfully validate the resulting configuration.

```
ldm set-io [mac-addr=num] [alt-mac-addrs=[auto|num1,[auto|num2,...]]]
  [pvid=[port-vlan-id]] [vid=[vlan-id1,vlan-id2,...]] [mtu=size]
  [name=value...] vf-name
```

where:

- *name*=*value* is the name-value pair of a property to set.
- *vf-name* is the name of the virtual function.

Set a Property for a Physical Function

This subcommand modifies the physical function configuration. Only the physical function device-specific properties are supported. Any change to the properties causes a delayed reconfiguration because the properties are applied during the attach operation of the physical function device driver.

The property values must be an integer or a string. Run the `ldm list-io -d` command to determine the property value type and whether a particular property can be set.

Note that the `ldm set-io` command succeeds only when the physical function driver successfully validates the resulting configuration.

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

where:

- *name*=*value* is the name-value pair of a property to set.
- *pf-name* is the name of the physical function.

Remove Input/Output Device

This subcommand removes a PCI bus or device from a specified logical domain.

`ldm rm-io (`*bus* **`|`** *device* **`|`** *vf-name***`)`** *ldom*

where:

- *bus*, *device*, and *vf-name* are a PCIe bus, a direct I/O-assignable device, and a PCIe SR-IOV virtual function, respectively. Although the operand can be specified as a device path or as a pseudonym, using the device pseudonym is recommended. The pseudonym is based on the ASCII label that is printed on the chassis to identify the corresponding I/O card slot and is platform specific.

  The following are examples of the pseudonyms that are associated with the device names:

  - **PCIe bus.** The `pci_0` pseudonym matches the `pci@400` device path.
  - **Direct I/O-assignable device.** The `PCIE1` pseudonym matches the `pci@400/pci@0/pci@c` device path.
  - **PCIe SR-IOV virtual function.** The `/SYS/MB/NET0/IOVNET.PF0.VF0` pseudonym matches the `pci@400/pci@2/pci@0/pci@6/network@0` device path.

  The specified guest domain must be in the inactive or bound state. If you specify the `primary` domain, this command initiates a delayed reconfiguration.
- *ldom* specifies the logical domain where the bus or device is to be removed.

Virtual Network Server

Add a Virtual Switch

This subcommand adds a virtual switch to a specified logical domain.

```
ldm add-vsw [-q] [default-vlan-id=vlan-id] [pvid=port-vlan-id] [vid=vlan-id1,vlan-id2,...]
  [linkprop=phys-state] [mac-addr=num] [net-dev=device] [mode=sc] [mtu=size]
  [id=switch-id] [inter-vnet-link=on|off] vswitch-name ldom
```

where:

- `-q` disables the validation of the path to the network device that is specified by the `net-dev` property. This option enables the command to run more quickly, especially if the logical domain is not fully configured.
- `default-vlan-id=`*vlan-id* specifies the default VLAN to which a virtual switch and its associated virtual network devices belong to implicitly, in untagged mode. It serves as the default port VLAN ID (*pvid*) of the virtual switch and virtual network devices. Without this option, the default value of this property is `1`. Normally, you would not need to use this option. It is provided only as a way to change the default value of `1`.
- `pvid=`*port-vlan-id* specifies the VLAN to which the virtual switch device needs to be a member, in untagged mode. This property also applies to the `set-vsw` subcommand. See "Using VLAN Tagging" in *Oracle VM Server for SPARC 2.2 Administration Guide*.
- `linkprop=phys-state` specifies whether the virtual device reports its link status based on the underlying physical network device. When `linkprop=phys-state` is specified on the command line, the virtual device link status reflects the physical link state. By default, the virtual device link status does not reflect the physical link state.
- `vid=`*vlan-id* specifies one or more VLANs to which a virtual network device or virtual switch needs to be a member, in tagged mode. This property also applies to the `set-vsw` subcommand. See "Using VLAN Tagging" in *Oracle VM Server for SPARC 2.2 Administration Guide* for more information.
- `mac-addr=`*num* is the MAC address to be used by this switch. The number must be in standard octet notation, for example, `80:00:33:55:22:66`. If you do not specify a MAC address, the switch is automatically assigned an address from the range of public MAC addresses allocated to the Logical Domains Manager.
- `net-dev=`*device* is the path to the network device over which this switch operates. The system validates that the path references an actual network device unless the `-q` option is specified.
- `mode=sc` enables virtual networking support for prioritized processing of Oracle Solaris Cluster heartbeat packets in a Logical Domains environment. Applications like Oracle Solaris Cluster need to ensure that high priority heartbeat packets are not dropped by congested virtual network and switch devices. This option prioritizes Oracle Solaris Cluster heartbeat frames and ensures that they are transferred in a reliable manner.

  You must set this option when running Oracle Solaris Cluster in a Logical Domains environment and using guest domains as Oracle Solaris Cluster nodes. Do *not* set this option when you are not running Oracle Solaris Cluster software in guest domains because you could impact virtual network performance.
- `mtu=`*size* specifies the maximum transmission unit (MTU) of a virtual switch device. Valid values are in the range of `1500-16000`.

- `id=`*switch-id* is the ID of a new virtual switch device. By default, ID values are generated automatically, so set this property if you need to match an existing device name in the OS.
- `inter-vnet-link=on|off` specifies whether to assign a channel between each pair of virtual network devices that are connected to the same virtual switch. This behavior improves guest-to-guest performance. The default value is `on`.
- *vswitch-name* is the unique name of the switch that is to be exported as a service. Clients (network) can attach to this service.
- *ldom* specifies the logical domain in which to add a virtual switch.

Set Options for a Virtual Switch

This subcommand modifies the properties of a virtual switch that has already been added.

```
ldm set-vsw [-q] [pvid=port-vlan-id] [vid=vlan-id1,vlan-id2,...] [mac-addr=num]
  [net-dev=device] [linkprop=[phys-state]] [mode=[sc]] [mtu=size]
  [inter-vnet-link=[on|off]] vswitch-name
```

where:

- `-q` disables the validation of the path to the network device that is specified by the `net-dev` property. This option enables the command to run more quickly, especially if the logical domain is not fully configured.
- `pvid=`*port-vlan-id* specifies the VLAN to which the virtual switch device needs to be a member, in untagged mode. See "Using VLAN Tagging" in *Oracle VM Server for SPARC 2.2 Administration Guide*.
- `vid=`*vlan-id* specifies one or more VLANs to which a virtual network device or virtual switch needs to be a member, in tagged mode. See "Using VLAN Tagging" in *Oracle VM Server for SPARC 2.2 Administration Guide*.
- `mac-addr=`*num* is the MAC address used by the switch. The number must be in standard octet notation, for example, `80:00:33:55:22:66`.
- `net-dev=`*device* is the path to the network device over which this switch operates. The system validates that the path references an actual network device unless the `-q` option is specified.
- `linkprop=phys-state` specifies whether the virtual device reports its link status based on the underlying physical network device. When `linkprop=phys-state` is specified on the command line, the virtual device link status reflects the physical link state. By default, the virtual device link status does not reflect the physical link state. The default situation occurs when the `linkprop` property is unspecified or when you run the `ldm set-vsw` command with the `linkprop=` argument.
- `mode=sc` enables virtual networking support for prioritized processing of Oracle Solaris Cluster heartbeat packets in a Logical Domains environment. Applications like Oracle Solaris Cluster need to ensure that high priority heartbeat packets are not dropped by congested virtual network and switch devices. This option prioritizes Oracle Solaris Cluster heartbeat frames and ensures that they are transferred in a reliable manner.

`mode=` (left blank) stops special processing of heartbeat packets.

You must set this option when running Oracle Solaris Cluster in a Logical Domains environment and using guest domains as Oracle Solaris Cluster nodes. Do *not* set this option when you are not running Oracle Solaris Cluster software in guest domains because you could impact virtual network performance.

- `mtu=`*size* specifies the maximum transmission unit (MTU) of a virtual switch device. Valid values are in the range of `1500-16000`.
- `inter-vnet-link=on|off` specifies whether to assign a channel between each pair of virtual network devices that are connected to the same virtual switch. This behavior improves guest-to-guest performance. The default value is `on`.
- *vswitch-name* is the unique name of the switch that is to exported as a service. Clients (network) can be attached to this service.

Remove a Virtual Switch

This subcommand removes a virtual switch.

`ldm rm-vsw [-f]` *vswitch-name*

where:

- `-f` attempts to force the removal of a virtual switch. The removal might fail.
- *vswitch-name* is the name of the switch that is to be removed as a service.

Virtual Network – Client

Add a Virtual Network Device

This subcommand adds a virtual network device to the specified logical domain.

`ldm add-vnet [mac-addr=`*num***`] [mode=hybrid] [pvid=`***port-vlan-id***`] [vid=`***vlan-id1,vlan-id2,...***`]`**
  `[linkprop=phys-state] [id=`*network-id***`] [mtu=`***size***`]`** *if-name vswitch-name ldom*

where:

- `mac-addr=`*num* is the MAC address for this network device. The number must be in standard octet notation, for example, `80:00:33:55:22:66`.
- `mode=hybrid` requests the system to use NIU Hybrid I/O on this `vnet` if possible. If it is not possible, the system reverts to virtual I/O. This hybrid mode is considered a delayed reconfiguration if set on an active `vnet` on a control domain. See "Using NIU Hybrid I/O" in *Oracle VM Server for SPARC 2.2 Administration Guide*.
- `pvid=`*port-vlan-id* specifies the VLAN to which the virtual network device needs to be a member, in untagged mode. See "Using VLAN Tagging" in *Oracle VM Server for SPARC 2.2 Administration Guide*.
- `vid=`*vlan-id* specifies one or more VLANs to which a virtual network device needs to be a member, in tagged mode. See "Using VLAN Tagging" in *Oracle VM Server for SPARC 2.2 Administration Guide*.

- `mtu=`*size* specifies the maximum transmission unit (MTU) of a virtual network device. Valid values are in the range of `1500-16000`.
- `linkprop=phys-state` specifies whether the virtual network device reports its link status based on the underlying physical network device. When `linkprop=phys-state` is specified on the command line, the virtual network device link status reflects the physical link state. By default, the virtual network device link status does not reflect the physical link state.
- `id=`*network-id* is the ID of a new virtual network device. By default, ID values are generated automatically, so set this property if you need to match an existing device name in the OS.
- *if-name* is a unique interface name to the logical domain, which is assigned to this virtual network device instance for reference on subsequent `set-vnet` or `rm-vnet` subcommands.
- *vswitch-name* is the name of an existing network service (virtual switch) to which to connect.
- *ldom* specifies the logical domain to which to add the virtual network device.

Set Options for a Virtual Network Device

This subcommand sets options for a virtual network device in the specified logical domain.

```
ldm set-vnet [mac-addr=num] [vswitch=vswitch-name] [mode=[hybrid]] [pvid=port-vlan-id]
  [linkprop=[phys-state]] [vid=vlan-id1,vlan-id2,...] [mtu=size] if-name ldom
```

where:

- `mac-addr=`*num* is the MAC address for this network device. The number must be in standard octet notation, for example, `80:00:33:55:22:66`.
- `vswitch=`*vswitch-name* is the name of an existing network service (virtual switch) to which to connect.
- `mode=hybrid` enables NIU Hybrid I/O operations on this `vnet`. This option is considered a delayed reconfiguration if set on an active `vnet` on a control domain. Leave the `mode=` argument blank to disable NIU Hybrid I/O.
- `pvid=`*port-vlan-id* specifies the VLAN to which the virtual network device needs to be a member, in untagged mode. See "Using VLAN Tagging" in *Oracle VM Server for SPARC 2.2 Administration Guide*.
- `linkprop=phys-state` specifies whether the virtual device reports its link status based on the underlying physical network device. When `linkprop=phys-state` is specified on the command line, the virtual device link status reflects the physical link state. By default, the virtual device link status does not reflect the physical link state. The default situation occurs when the `linkprop` property is unspecified or when you run the `ldm set-vnet` command with the `linkprop=` argument.

- `vid=`*vlan-id* specifies one or more VLANs to which a virtual network device needs to be a member, in tagged mode. See "Using VLAN Tagging" in *Oracle VM Server for SPARC 2.2 Administration Guide*.
- `mtu=`*size* specifies the maximum transmission unit (MTU) of a virtual network device. Valid values are in the range of `1500-16000`.
- *if-name* is the unique interface name assigned to the virtual network device that you want to set.
- *ldom* specifies the logical domain in which to modify the virtual network device.

Remove a Virtual Network Device

This subcommand removes a virtual network device from the specified logical domain.

`ldm rm-vnet [-f]` *if-name ldom*

where:

- `-f` attempts to force the removal of a virtual network device from a logical domain. The removal might fail.
- *if-name* is the unique interface name assigned to the virtual network device that you want to remove.
- *ldom* specifies the logical domain from which to remove the virtual network device.

Virtual Disk – Service

Add a Virtual Disk Server

This subcommand adds a virtual disk server to the specified logical domain.

`ldm add-vds` *service-name ldom*

where:

- *service-name* is the service name for this instance of the virtual disk server. The *service-name* must be unique among all virtual disk server instances on the server.
- *ldom* specifies the logical domain in which to add the virtual disk server.

Remove a Virtual Disk Server

This subcommand removes a virtual disk server.

`ldm rm-vds [-f]` *service-name*

where:

- `-f` attempts to force the removal of a virtual disk server. The removal might fail.
- *service-name* is the unique service name for this instance of the virtual disk server.

Caution – The `-f` option attempts to unbind all clients before removal, which might cause loss of disk data if writes are in progress.

Add a Device to a Virtual Disk Server

This subcommand adds a device to a virtual disk server. The device can be an entire disk, a slice on a disk, a file, or a disk volume. See Chapter 7, "Using Virtual Disks," in *Oracle VM Server for SPARC 2.2 Administration Guide*.

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

where:

- `-f` attempts to force the creation of an additional virtual disk server when specifying a block device path that is already part of another virtual disk server. If specified, the `-f` option must be the first in the argument list.
- `-q` disables the validation of the virtual disk back end that is specified by the *backend* operand. This option enables the command to run more quickly, especially if the logical domain or the back end is not fully configured.
- `options=` are as follows:
  - `ro` – Specifies read-only access
  - `slice` – Exports a back end as a single slice disk
  - `excl` – Specifies exclusive disk access

  Omit the `options=` argument to have the default values of disk, not exclusive, and read/write. If you add the `options=` argument, you must specify one or more of the options for a specific virtual disk server device. Separate two or more options with commas and no spaces, such as `ro,slice,excl`.
- `mpgroup=`*mpgroup* is the disk multipath group name used for virtual disk failover support. You can assign the virtual disk several redundant paths in case the link to the virtual disk server device currently in use fails. To do this, you would group multiple virtual disk server devices (`vdsdev`) into one multipath group (`mpgroup`), all having the same `mpgroup` name. When a virtual disk is bound to any virtual disk server device in a multipath group, the virtual disk is bound to all the virtual disk server devices that belong to the `mpgroup`.
- *backend* is the location where data of a virtual disk are stored. The back end can be a disk, a disk slice, a file, a volume (including ZFS, Solaris Volume Manager, or VxVM), or any disk pseudo device. The disk label can be SMI VTOC, EFI, or no label at all. 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. When adding a device, the *volume-name* must be paired with the *backend*. The system validates that the location specified by *backend* exists and can be used as a virtual disk back end unless the `-q` option is specified.
- *volume-name* is a unique name that you must specify for the device being added to the virtual disk server. The *volume-name* must be unique for 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 *backend*.

- *service-name* is the name of the virtual disk server to which to add this device.

Set Options for a Virtual Disk Server Device

This subcommand sets options for a virtual disk server. See the *Oracle VM Server for SPARC 2.2 Administration Guide*.

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

where:

- `-f` removes the read-only restriction when multiple volumes in the same logical domain are sharing an identical block device path in read-only mode (`option=ro`). If specified, the `-f` option must be the first in the argument list.
- `options=` are as follows:
  - `ro` – Specifies read-only access
  - `slice` – Exports a back end as a single slice disk
  - `excl` – Specifies exclusive disk access
  - Leave the `options=` argument blank to turn off any previous options specified. You can specify all or a subset of the options for a specific virtual disk server device. Separate two or more options with commas and no spaces, such as `ro,slice,excl`.
- `mpgroup=`*mpgroup* is the disk multipath group name used for virtual disk failover support. You can assign the virtual disk several redundant paths in case the link to the virtual disk server device currently in use fails. To do this, you would group multiple virtual disk server devices (`vdsdev`) into one multipath group (`mpgroup`), all having the same `mpgroup` name. When a virtual disk is bound to any virtual disk server device in a multipath group, the virtual disk is bound to all the virtual disk server devices that belong to the `mpgroup`.
- *volume-name* is the name of an existing volume exported by the service named by *service-name*.
- *service-name* is the name of the virtual disk server being modified.

Remove a Device From a Virtual Disk Server

This subcommand removes a device from a virtual disk server.

ldm rm-vdsdev [-f] *volume-name***@***service-name*

where:

- `-f` attempts to force the removal of the virtual disk server device. The removal might fail.
- *volume-name* is the unique name for the device being removed from the virtual disk server.
- *service-name* is the name of the virtual disk server from which to remove this device.

Caution – Without the `-f` option, the `rm-vdsdev` subcommand does not allow a virtual disk server device to be removed if the device is busy. Using the `-f` option can cause data loss for open files.

Virtual Disk – Client

Add a Virtual Disk

This subcommand adds a virtual disk to the specified logical domain. An optional timeout property allows you to specify a timeout for a virtual disk if it cannot establish a connection with the virtual disk server.

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

where:

- `timeout=`*seconds* is the number of seconds for establishing a connection between a virtual disk client (`vdc`) and a virtual disk server (`vds`). If there are multiple virtual disk (`vdisk`) paths, then the `vdc` can try to connect to a different `vds`, and the timeout ensures that a connection to any `vds` is established within the specified amount of time.

  Omit the `timeout=` argument or set `timeout=0` to have the virtual disk wait indefinitely.
- `id=`*disk-id* is the ID of a new virtual disk device. By default, ID values are generated automatically, so set this property if you need to match an existing device name in the OS.
- *disk-name* is the name of the virtual disk.
- *volume-name* is the name of the existing virtual disk server device to which to connect.
- *service-name* is the name of the existing virtual disk server to which to connect.
- *ldom* specifies the logical domain in which to add the virtual disk.

Set Options for a Virtual Disk

This subcommand sets options for a virtual disk in the specified logical domain. An optional timeout property allows you to specify a timeout for a virtual disk if it cannot establish a connection with the virtual disk server.

`ldm set-vdisk [timeout=`*seconds***`] [volume=`***volume-name***`@`***service-name***`]`** *disk-name ldom*

where:

- `timeout=`*seconds* is the number of seconds for establishing a connection between a virtual disk client (`vdc`) and a virtual disk server (`vds`). If there are multiple virtual disk (`vdisk`) paths, then the `vdc` can try to connect to a different `vds`, and the timeout ensures that a connection to any `vds` is established within the specified amount of time.

  Set `timeout=0` to disable the timeout.

  Do not specify a `timeout=` argument to have the virtual disk wait indefinitely.
- `volume=`*volume-name* is the name of the virtual disk server device to which to connect. *service-name* is the name of the virtual disk server to which to connect.

- *disk-name* is the name of the existing virtual disk.
- *ldom* specifies the existing logical domain where the virtual disk was previously added.

Remove a Virtual Disk

This subcommand removes a virtual disk from the specified logical domain.

`ldm rm-vdisk [-f]` *disk-name ldom*

where:

- `-f` attempts to force the removal of the virtual disk. The removal might fail.
- *disk-name* is the name of the virtual disk to be removed.
- *ldom* specifies the logical domain from which to remove the virtual disk.

Virtual Data Plane Channel – Service

Add a Virtual Data Plane Channel Service

This subcommand adds a virtual data plane channel service to the specified logical domain. This subcommand should only be used in a Netra Data Plane Software (NDPS) environment.

`ldm add-vdpcs` *vdpcs-service-name ldom*

where:

- *vdpcs-service-name* is the name of the virtual data plane channel service that is to be added.
- *ldom* specifies the logical domain to which to add the virtual data plane channel service.

Remove a Virtual Data Plane Channel Service

This subcommand removes a virtual data plane channel service. This subcommand should only be used in a Netra Data Plane Software (NDPS) environment.

`ldm rm-vdpcs [-f]` *vdpcs-service-name*

where:

- `-f` attempts to force the removal of the virtual data plane channel service. The removal might fail.
- *vdpcs-service-name* is the name of the virtual data plane channel service that is to be removed.

Virtual Data Plane Channel – Client

Add a Virtual Data Plane Channel Client

This subcommand adds a virtual data plane channel client to the specified logical domain. This subcommand should only be used in a Netra Data Plane Software (NDPS) environment.

`ldm add-vdpcc` *vdpcc-name vdpcs-service-name ldom*

where:

- *vdpcc-name* is the unique name of the virtual data plane channel service client.
- *vdpcs-service-name* is the name of the virtual data plane channel service to which to connect this client.
- *ldom* specifies the logical domain to which to add the virtual data plane channel client.

Remove a Virtual Data Plane Channel Client

This subcommand removes a virtual data plane channel client from the specified logical domain. This subcommand should only be used in a Netra Data Plane Software (NDPS) environment.

`ldm rm-vdpcc [-f]` *vdpcc-name ldom*

where:

- `-f` attempts to force the removal of the virtual data plane channel client. The removal might fail.
- *vdpcc-name* is the unique name assigned to the virtual data plane channel client that is to be removed.
- *ldom* specifies the logical domain from which to remove the virtual data plane channel client.

Virtual Console

Add a Virtual Console Concentrator

This subcommand adds a virtual console concentrator to the specified logical domain.

`ldm add-vcc port-range=`*x***`-`***y* *vcc-name ldom*

where:

- `port-range=`*x-y* is the range of TCP ports to be used by the virtual console concentrator for console connections.
- *vcc-name* is the name of the virtual console concentrator that is to be added.
- *ldom* specifies the logical domain to which to add the virtual console concentrator.

Set Options for a Virtual Console Concentrator

This subcommand sets options for a specific virtual console concentrator.

`ldm set-vcc port-range=`*x***`-`***y* *vcc-name*

where:

- `port-range=`*x-y* is the range of TCP ports to be used by the virtual console concentrator for console connections. Any modified port range must encompass all the ports assigned to clients of the concentrator.
- *vcc-name* is the name of the virtual console concentrator that is to be set.

Remove a Virtual Console Concentrator

This subcommand removes a virtual console concentrator from the specified logical domain.

`ldm rm-vcc [-f]` *vcc-name*

where:

- `-f` attempts to force the removal of the virtual console concentrator. The removal might fail.
- *vcc-name* is the name of the virtual console concentrator that is to be removed.

Caution – The `-f` option attempts to unbind all clients before removal, which might cause loss of data if writes are in progress.

Set Options for a Virtual Console

This subcommand sets a specific port number and group in the specified logical domain. You can also set the attached console's service. This subcommand can be used only when a domain is inactive.

`ldm set-vcons [port=[`*port-num***`]] [group=`***group***`] [service=`***vcc-server***`]`** *ldom*

where:

- `port=`*port-num* is the specific port to use for this console. Leave the *port-num* blank to have the Logical Domains Manager automatically assign the port number.
- `group=`*group* is the new group to which to attach this console. The group argument allows multiple consoles to be multiplexed onto the same TCP connection. Refer to the Oracle Solaris OS `vntsd(1M)` man page for more information about this concept. When a group is specified, a service must also be specified.
- `service=`*vcc-server* is the name for the existing virtual console concentrator that should handle the console connection. A service must be specified when a group is specified.
- *ldom* specifies the logical domain in which to set the virtual console concentrator.

Physical Functions and Virtual Functions

Virtual Functions

The PCIe single-root I/O virtualization (SR-IOV) standard enables the efficient sharing of PCIe devices among I/O domains. This standard is implemented in the hardware to achieve near-native I/O performance. SR-IOV creates a number of virtual functions that are virtualized instances of the physical device or function. The virtual functions are directly assigned to I/O domains so that they can share the associated physical device and perform I/O without CPU and hypervisor overhead.

PCIe *physical functions* have complete access to the hardware and provide the SR-IOV capability to create, configure, and manage virtual functions. A PCIe component on the system board or a PCIe plug-in card can provide one or more physical functions. An Oracle Solaris driver interacts with the physical functions that provide access to the SR-IOV features.

PCIe *virtual functions* contain the resources that are necessary for data movement. An I/O domain that has a virtual function can access hardware and perform I/O directly by means of an Oracle Solaris virtual function driver. This behavior avoids the overhead and latency that is involved in the virtual I/O feature by removing any bottlenecks in the communication path between the applications that run in the I/O domain and the physical I/O device in the root domain.

Some of these commands require that you specify an identifier for a physical function or virtual function as follows:

```
pf-name ::= pf-pseudonym | pf-path
vf-name ::= vf-pseudonym | vf-path
```

Use the pseudonym form when referring to a corresponding device. This is the form of the name that is shown in the `NAME` column of the `ldm list-io` output. When you run the `ldm list-io -l` command, the path form of the name appears in the output. The `ldm list-io -p` output shows the pseudonym form as the value of the `alias=` token and the path form as the value of the `dev=` token.

Create a Virtual Function

This subcommand creates a virtual function from a specified physical function by incrementing the number of virtual functions in the specified physical function by one. The new virtual function is assigned the highest number in the sequence of virtual function numbers. When you create a new virtual function, a delayed reconfiguration is initiated for the `primary` domain.

Network class virtual functions *must* have a MAC address assigned, which is assigned by default. To override the default MAC address value, specify another value for the `mac-addr` property.

You can also set class-specific properties and device-specific properties when you create a virtual function. This command succeeds only when the physical function driver successfully validates the resulting configuration. By default, a new virtual function is not assigned to any domain. The virtual function can only be assigned (bound) to an I/O domain after the `primary` domain is rebooted and the virtual function is instantiated in the hardware. Plan ahead by determining whether you want to create multiple virtual functions. If you do, create them one after the other to avoid performing multiple reboots.

The device-specific properties depend on the properties that are exported by the physical function driver. For more information, use the `ldm list-io -d` command. When the command is successful, you see a message about a delayed reconfiguration.

`ldm create-vf [mac-addr=`*`num`***`] [alt-mac-addrs=auto|`***`num1`***`,[auto|`***`num2`***`,...]]`**
  `[pvid=`*`port-vlan-id`***`] [vid=`***`vlan-id1`***`,`***`vlan-id2`***`,...] [mtu=`***`size`***`]`**
  `[`*`name`***`=`***`value`***`...]`** *`pf-name`*

where:

- `mac-addr=`*num* is the primary MAC address of the virtual function
- `alt-mac-addrs=auto|`*num1*`,[auto|`*num2*`,...]` is a comma-separated list of alternate MAC addresses. Valid values are numeric MAC addresses and the `auto` keyword, which can be used one or more times to request that the system generate an alternate MAC address. The `auto` keyword can be mixed with numeric MAC addresses.
- `pvid=`*port-vlan-id* is the port VLAN ID (no default value)
- `vid=`*vlan-id1*`,`*vlan-id*`2...` is a comma-separated list of integer VLAN IDs.
- `mtu=`*size* is the maximum transmission unit (in bytes) for the virtual function.
- *name*=*value* is the name-value pair of a property to specify.
- *pf-name* is the name of the physical function.

Destroy a Virtual Function

This subcommand destroys a virtual function from the specified physical function. This command succeeds *only* if the following are true:

- The specified virtual function is not currently assigned to any domain.
- The specified virtual function is the last virtual function in the corresponding physical function.
- The resulting configuration is successfully validated by the physical function driver.
- A successful operation triggers a delayed reconfiguration, as the change to the number of virtual functions can only be done as part of rebooting. See the `create-vf` subcommand for more information.

`ldm destroy-vf` *`vf-name`*

where *vf-name* is the name of the virtual function.

Variables

Add Variable

This subcommand adds one or more variables for a logical domain.

`ldm add-var` *`var-name`***`=[`***`value`***`]...`** *`ldom`*

where:

- *var-name*=*value* is the name-value pair of a variable to add. The value is optional.
- *ldom* specifies the logical domain in which to add the variable.

Set Variable

This subcommand sets variables for a logical domain.

ldm set-var *var-name*=**[***value***]...** *ldom*

where:

- *var-name*=*value* is the name-value pair of a variable to set. The value is optional.
- *ldom* specifies the logical domain in which to set the variable.

Note – Leaving *value* blank, sets *var-name* to no value.

Remove Variable

This subcommand removes a variable for a logical domain.

ldm rm-var *var-name*... *ldom*

where:

- *var-name* is the name of a variable to remove.
- *ldom* specifies the logical domain from which to remove the variable.

Other Operations

Start Domains

This subcommand starts one or more logical domains.

ldm start -a
ldm start -i *file*
ldm start *ldom***...**

where:

- `-a` starts all bound logical domains.
- `-i` *file* specifies an XML configuration file to use in starting the logical domain.
- *ldom* specifies one or more logical domains to start.

Stop Domains

This subcommand stops one or more running logical domains. The subcommand sends a `uadmin` request to the logical domain if the Oracle Solaris OS is booted. to stop a domain in a more "graceful" manner, perform a `shutdown` or `init` operation in the domain that you want to stop. See the `shutdown`(1M) and `init`(1M) man page.

```
ldm stop [-f] -a
ldm stop [-f] ldom...
```

where:

- `-f` attempts to force a running logical domain to stop. Use only if the domain cannot be stopped by any other means.
- `-a` stops all running logical domains except the control domain.
- *ldom* specifies one or more running logical domains to stop.

Panic Oracle Solaris OS

This subcommand panics the Oracle Solaris OS on a specified logical domain, which provides a back trace and crash dump if you configure the Oracle Solaris OS to do that. The `dumpadm`(1M) command provides the means to configure the crash dump.

```
ldm panic ldom
```

ldom specifies the logical domain to panic.

Provide Help Information

This subcommand provides usage for all subcommands or the subcommand that you specify. You can also use the `ldm` command alone to provide usage for all subcommands.

```
ldm --help [subcommand]
```

subcommand specifies the `ldm` subcommand about which you want usage information.

Provide Version Information

This subcommand provides version information.

```
ldm --version
ldm -V
```

Bind Resources to a Domain

This subcommand binds, or attaches, configured resources to a logical domain.

```
ldm bind-dom [-f] [-q] -i file
ldm bind-dom [-f] [-q] ldom
```

where:

- `-f` attempts to force the binding of the domain even if invalid network or disk back-end devices are detected.
- `-q` disables the validation of network or disk back-end devices so that the command runs more quickly.
- `-i` *file* specifies an XML configuration file to use in binding the logical domain.
- *ldom* specifies the logical domain to which to bind resources.

Unbind Resources From a Domain

This subcommand releases resources bound to configured logical domains.

`ldm unbind-dom` *ldom*

ldom specifies the logical domain from which to unbind resources.

Configure Operations

Add a Logical Domain Configuration

This subcommand adds a logical domain configuration, either based on the currently active configuration or on a previously autosaved configuration. The configuration is stored on the service processor (SP).

`ldm add-config` *config-name*
`ldm add-config -r` *autosave-name* **[***new-config-name***]**

where:

- *config-name* is the name of the logical domain configuration to add.
- `-r` *autosave-name* applies the autosave configuration data to one of the following:
  - Configuration on the SP that has the same name
  - Newly created configuration, *new-config-name*, which does not exist on the SP

  If the target configuration does not exist on the SP, a configuration of that name is created and saved to the SP based on the contents of the corresponding autosave configuration. After the autosave configuration data is applied, those autosave files are deleted from the control domain. If *autosave-name* does not represent the currently selected configuration, or if *new-config-name* is specified, the state of the current configuration on the SP and any autosave files for it on the control domain are unaffected.

  To recover an autosave configuration that is known to be corrupted, you must specify `-r` *new-config-name*. You are not permitted to overwrite an existing configuration with one that is known to be corrupted.
- *new-config-name* is the name of the logical domain configuration to add.

Set a Logical Domain Configuration

This subcommand enables you to specify a logical domain configuration to use. The configuration is stored on the SP.

`ldm set-config` *config-name*

config-name is the name of the logical domain configuration to use.

The default configuration name is `factory-default`. To specify the default configuration, use the following:

`ldm set-config factory-default`

Remove a Logical Domain Configuration

This subcommand removes a logical domain configuration that is stored on the SP, as well as any corresponding autosave configuration from the control domain.

`ldm rm-config [-r]` *config-name*

where:

- `-r` only removes autosave configurations from the control domain.
- *config-name* is the name of the logical domain configuration to remove.

List Operations

List Domains and States

This subcommand lists logical domains and their states. If you do not specify a logical domain, all logical domains are listed.

`ldm ls-dom [-e] [-l] [-o` *format***`] [-p][ [`***ldom***`...]`**

where:

- `-e` generates an extended listing containing services and devices that are automatically set up, that is, not under your control.
- `-l` generates a long listing.
- `-o` limits the output *format* to one or more of the following subsets. If you specify more than one format, delimit each format by a comma with no spaces.
  - `console` – Output contains the virtual console (`vcons`) and virtual console concentrator (`vcc`) service.
  - `core` – Output contains information about cores, core ID and physical CPU set.
  - `cpu` – Output contains information about the CPU thread (`vcpu`), physical CPU (`pcpu`), and core ID (`cid`).
  - `crypto` – Cryptographic unit output contains the Modular Arithmetic Unit (`mau`) and any other supported cryptographic unit, such as the Control Word Queue (CWQ).
  - `disk` – Output contains the virtual disk (`vdisk`) and virtual disk server (`vds`).
  - `domain` – Output contains the variables (`var`), host ID (`hostid`), domain state, flags, universally unique identifier (UUID), software state, utilization percentage, a slave's master domains, and the master domain's failure policy.
  - `memory` – Output contains `memory`.
  - `network` – Output contains the media access control (`mac`) address , virtual network switch (`vsw`), and virtual network (`vnet`) device.
  - `physio` – Physical input/output contains the peripheral component interconnect (`pci`) and network interface unit (`niu`).

  - `resmgmt` – Output contains DRM policy information, indicates which policy is currently running, and indicates whether the `whole-core`, `max-core`, and `threading` constraints are enabled.
  - `serial` – Output contains the virtual logical domain channel (`vldc`) service, virtual logical domain channel client (`vldcc`), virtual data plane channel client (`vdpcc`), and virtual data plane channel service (`vdpcs`).
  - `status` – Output contains the status of a migrating domain.
- `-p` generates the list in a parseable, machine-readable format.
- *ldom* is the name of the logical domain for which to list state information.

List Bindings for Domains

This subcommand lists bindings for logical domains. If no logical domains are specified, all logical domains are listed.

```
ldm ls-bindings [-e] [-p] [ldom...]
```

where:

- `-e` generates an extended listing containing services and devices that are automatically set up, that is, not under your control.
- `-p` generates the list in a parseable, machine-readable format.
- *ldom* is the name of the logical domain for which you want binding information.

List Services for Domains

This subcommand lists all the services exported by logical domains. If no logical domains are specified, all logical domains are listed.

```
ldm ls-services [-e] [-p] [ldom...]
```

where:

- `-e` generates an extended listing containing services and devices that are automatically set up, that is, not under your control.
- `-p` generates the list in a parseable, machine-readable format.
- *ldom* is the name of the logical domain for which you want services information.

List Constraints for Domains

This subcommand lists the constraints for the creation of one or more logical domains. If no logical domains are specified, all logical domains are listed.

```
ldm ls-constraints [-x] [ldom...]
ldm ls-constraints [-e] [-p] [ldom...]
```

where:

- `-x` writes the constraint output in XML format to the standard output (`stdout`) format. This output can be used as a backup.
- *ldom* is the name of the logical domain for which you want to list constraints.
- `-e` generates an extended listing containing services and devices that are automatically set up, that is, not under your control.
- `-p` writes the constraint output in a parseable, machine-readable form.

List Devices

This subcommand lists either free (unbound) resources or all server resources. The default is to list all free resources.

```
ldm ls-devices [-a] [-p] [core] [cpu] [crypto] [memory] [io]
```

where:

- `-a` lists all server resources, bound and unbound.
- `-p` writes the constraint output in a parseable, machine-readable form.
- `core` lists information about cores, the core ID and physical CPU set, and specifies which CPUs in the core are still unallocated.
- `cpu` lists CPU thread and physical CPU resources.
- `crypto` lists only the modular arithmetic unit resources.
- `memory` lists only memory resources.
- `io` lists only input/output resources, such as a PCI bus, a network, or direct I/O-assignable devices.

In the power management column (`PM`) or field (`pm=`), `yes` means that the CPU thread is power-managed, and `no` means that the CPU thread is powered on. It is assumed that 100 percent-free CPUs are power-managed by default.

List I/O Devices

This subcommand lists the I/O devices that are configured on the system. The list of devices includes I/O buses (including NIUs) and direct I/O-assignable devices.

The output is divided into the following sections:

- **I/O bus information.** The `IO` column lists the device path of the bus or network device, and the `PSEUDONYM` column shows the associated pseudonym for the bus or network device. The `DOMAIN` column indicates the domain to which the device is currently bound.
- **Direct I/O-assignable devices.** The `PCIE` column lists the device path of the device, and the `PSEUDONYM` column shows the associated pseudonym for the device.

  The `STATUS` column applies to slots that accept plug-in cards. The value is either `OCC` (occupied), `EMP` (empty), or `UNK` (unknown). Slots that represent on-board devices always have the status of `OCC`. If the root domain does not support the direct I/O (DIO) feature, the slot status is `UNK`.

  The `DOMAIN` column indicates the domain to which the device is currently bound. If this field is empty, the corresponding device is not bound. If this field shows a hyphen character (`-`), the device cannot currently be removed from the primary domain for one of the following reasons:

  - The OS that is running on the root domain does not support the DIO feature. In this case, the `STATUS` field for all the devices shown in the `PCIE` column of the output is `UNK`.
  - The corresponding slot is empty so that no PCIe option card is present. Note that empty slots cannot be assigned to direct I/O domains.
  - The root complex to which the device is connected is assigned to another domain by means of a split PCI configuration. In this case, the `STATUS` column for the device is `UNK`.

```
ldm list-io [-l] [-p] [pf-name]
ldm list-io -d pf-name
```

where:

- `-l` lists information about subdevices that are hosted by direct I/O-assignable devices. Note that this output indicates which devices will be loaned with the direct I/O-assignable device to the receiving domain. The subdevice names *cannot* be used for command input.
- `-p` writes the output in a parseable, machine-readable form.
- `-d` *pf-name* lists information about the specified physical function.
- *pf-name* is the name of the physical function.

List Logical Domain Configurations

This subcommand lists the logical domain configurations stored on the service processor (SP).

```
ldm ls-config [-r [autosave-name]]
```

`-r` [*autosave-name*] lists those configurations for which autosave files exist on the control domain. If *autosave-name* is specified, it only reports on *autosave-name*. The output also notes whether an autosave file is newer than the corresponding SP configuration.

Note – When a delayed reconfiguration is pending, the configuration changes are immediately autosaved. As a result, if you run the `ldm ls-config -r` command, the autosave configuration is shown as being newer than the current configuration.

List Variables

This subcommand lists one or more variables for a logical domain. To list all variables for a domain, leave the *var-name* blank.

```
ldm ls-var [var-name...] ldom
```

where:

- *var-name* is the name of the variable to list. If you do not specify any name, all variables will be listed for the domain.
- *ldom* is the name of the logical domain for which to list one or more variables.

Add, Set, and Remove Resource Management Policies

Add a Resource Management Policy

This subcommand enables you to add a resource management policy for one or more logical domains. A resource management policy consists of optional properties and their values.

You can enable a resource management policy in an active domain that supports CPU DR as long as power management is *not* using the elastic policy.

```
ldm add-policy [enable=yes|no] [priority=value] [attack=value] [decay=value]
  [elastic-margin=value] [sample-rate=value] [tod-begin=hh:mm[:ss]]
  [tod-end=hh:mm[:ss]] [util-lower=percent] [util-upper=percent] [vcpu-min=value]
  [vcpu-max=value] name=policy-name ldom...
```

where:

- The properties are described in the Properties section.
- *ldom* specifies the logical domain for which to add a resource management policy.

Modify a Resource Management Policy

This subcommand enables you to modify a resource management policy for one or more logical domains by specifying values for optional properties.

```
ldm set-policy [enable=[yes|no]] [priority=[value]] [attack=[value]] [decay=[value]]
  [elastic-margin=[value]] [sample-rate=[value]] [tod-begin=[hh:mm:ss]]
  [tod-end=[hh:mm:ss]] [util-lower=[percent]] [util-upper=[percent]] [vcpu-min=[value]]
  [vcpu-max=[value]] name=policy-name ldom...
```

where:

- The properties are described in the Properties section.
- *ldom* specifies the logical domain for which to modify the resource management policy.

Remove a Resource Management Policy

This subcommand enables you to remove a resource management policy from a logical domain by specifying one or more policy names.

```
ldm remove-policy [name=]policy-name... ldom
```

where:

- The `name` property specifies the name of the resource management policy, *policy-name*.
- *ldom* specifies the logical domain on which to remove the resource management policy.

Configure or Reconfigure a Domain From an XML File

This subcommand enables you to use an existing configuration to configure one or more guest domains, the control domain, or both types of domains. The `ldm init-system` command takes an XML file (such as the output of `ldm ls-constraints -x`) as input, configures the specified domains, and reboots the control domain. Run this command with the factory default configuration.

```
ldm init-system [-frs] -i file
```

where:

- `-i` *file* specifies the XML configuration file to use to create the logical domain.
- `-f` skips the factory-default configuration check and continues irrespective of what was already configured on the system.

  Use the `-f` option with caution. `ldm init-system` assumes that the system is in the factory-default configuration, and so *directly* applies the changes that are specified by the XML file. Using `-f` when the system is in a configuration other than the factory default will likely result in a system that is not configured as specified by the XML file. One or more changes might fail to be applied to the system depending on the combination of changes in the XML file and the initial configuration.
- `-r` reboots the system after configuration.
- `-s` restores only the virtual services configuration (`vds`, `vcc`, and `vsw`).

Examples

EXAMPLE 1 Create Default Services

Set up the three default services, virtual disk server, virtual switch, and virtual console concentrator so that you can export those services to the guest domains.

```
# ldm add-vds primary-vds0 primary
# ldm add-vsw net-dev=nxge0 primary-vsw0 primary
# ldm add-vcc port-range=5000-5100 primary-vcc0 primary
```

EXAMPLE 2 List Services

You can list services to ensure they have been created correctly or to see what services you have available.

EXAMPLE 2 List Services *(Continued)*

```
# ldm ls-services primary
VCC
    NAME         LDOM    PORT-RANGE
    primary-vcc0 primary 5000-5100
VSW
    NAME         LDOM    MAC               NET-DEV   DEVICE     DEFAULT-VLAN-ID PVID VID MODE
    primary-vsw0 primary 00:14:4f:f9:68:d0 nxge0 switch@0 1               1
VDS
    NAME         LDOM    VOLUME         OPTIONS      MPGROUP   DEVICE
    primary-vds0 primary
```

EXAMPLE 3 Set Up the Control Domain Initially

The control domain, named `primary`, is the initial domain that is present when you install the Logical Domains Manager. The control domain has a full complement of resources, and those resources depend on what server you have. Set only those resources you want the control domain to keep so that you can allocate the remaining resources to the guest domains. Then save the configuration on the service processor. You must reboot so the changes take effect.

If you want to enable networking between the control domain and the other domains, you must plumb the virtual switch on the control domain. You must enable the virtual network terminal server daemon, `vntsd`(1M), to use consoles on the guest domains.

```
# ldm start-reconf primary
# ldm set-crypto 1 primary
# ldm set-vcpu 4 primary
# ldm set-mem 4G primary
# ldm add-config initial
# shutdown -y -g0 -i6
# ifconfig -a
# ifconfig vsw0 plumb
# ifconfig nxge0 down unplumb
# ifconfig vsw0 IP-of-nxge0 netmask netmask-of-nxge0 broadcast + up
# svcadm enable vntsd
```

EXAMPLE 4 List Bindings

You can list bindings to see if the control domain has the resources you specified, or what resources are bound to any domain.

```
# ldm ls-bindings primary
NAME             STATE    FLAGS    CONS    VCPU  MEMORY  UTIL  UPTIME
primary          active   -t-cv            4     4G      12%   11m

MAC
    08:00:90:11:11:10
```

EXAMPLE 4 List Bindings *(Continued)*

```
VCPU
    VID    PID    UTIL  STRAND
    0      0       18%   100%
    1      1       13%   100%
    2      2      9.8%   100%
    3      3      5.4%   100%

MEMORY
    RA                PA             SIZE
    0x4000000         0x4000000      4G

IO
DEVICE           PSEUDONYM        OPTIONS
pci@400          pci_0
pci@500          pci_1
pci@400/pci@0/pci@9  PCIE2
pci@400/pci@0/pci@9  MB/SASHBA
pci@500/pci@0/pci@8  MB/NET0

VCC
    NAME             PORT-RANGE
    primary-vcc0     5000-5100

VSW
    NAME           MAC                 NET-DEV   DEVICE     MODE
    primary-vsw0   00:14:4f:f9:68:d0   nxge0   switch@0   prog,promisc

VDS
    NAME             VOLUME         OPTIONS          DEVICE
    primary-vds0
```

EXAMPLE 5 Create a Logical Domain

Ensure that you have the resources to create the desired guest domain configuration, add the guest domain, add the resources and devices that you want the domain to have, set boot parameters to tell the system how to behave on startup, bind the resources to the domain, and save the guest domain configuration in an XML file for backup. You also might want to save the primary and guest domain configurations on the SC. Then you can start the domain, find the TCP port of the domain, and connect to it through the default virtual console service.

```
# ldm ls-devices
# ldm add-dom ldg1
# ldm add-vcpu 4 ldg1
# ldm add-mem 1g ldg1
# ldm add-vnet vnet1 primary-vsw0 ldg1
# ldm add-vdsdev /dev/dsk/c0t1d0s2 vol1@primary-vds0
# ldm add-vdisk vdisk1 vol1@primary-vds0 ldg1
```

EXAMPLE 5 Create a Logical Domain *(Continued)*

```
# ldm set-var auto-boot\?=false ldg1
# ldm set-var boot-device=vdisk1 ldg1
# ldm bind-dom ldg1
# ldm ls-constraints -x ldg1 > ldg1.xml
# ldm add-config ldg1_4cpu_1G
# ldm start ldg1
# ldm ls -l ldg1
# telnet localhost 5000
```

EXAMPLE 6 Use One Terminal for Many Guest Domains

Normally, each guest domain you create has its own TCP port and console. Once you have created the first guest domain (`ldg1` in this example), you can use the `ldm set-vcons` command to attach all the other domains (second domain is `ldg2` in this example) to the same console port. Note that the `set-vcons` subcommand works only on an inactive domain.

```
# ldm set-vcons group=ldg1 service=primary-vcc0 ldg2
```

If you use the `ldm ls -l` command after performing the `set-vcons` commands on all guest domains except the first, you can see that all domains are connected to the same port. See the `vntsd`(1M) man page for more information about using consoles.

EXAMPLE 7 Add a Virtual PCI Bus to a Logical Domain

I/O domains are a type of service domain that have direct ownership of and direct access to physical I/O devices. The I/O domain then provides the service to the guest domain in the form of a virtual I/O device. This example shows how to add a virtual PCI bus to a logical domain.

```
# ldm add-io pci@7c0 ldg1
```

EXAMPLE 8 Add Virtual Data Plane Channel Functionality for Netra Only

If your server has a Netra Data Plane Software (NDPS) environment, you might want to add virtual data plane channel functionality. First, you would add a virtual data plane channel service (`primary-vdpcs0`, for example) to the service domain, in this case, the `primary` domain.

```
# ldm add-vdpcs primary-vdpcs0 primary
```

Now that you have added the service to the service domain (`primary`), you can add the virtual data plane channel client (`vdpcc1`) to a guest domain (`ldg1`).

```
# add-vdpcc vdpcc1 primary-vdpcs0 ldg1
```

EXAMPLE 9 Cancel Delayed Reconfiguration Operations for a Control Domain

A delayed reconfiguration operation blocks configuration operations on all other domains. There might be times when you want to cancel delayed configuration operations for a control

EXAMPLE 9 Cancel Delayed Reconfiguration Operations for a Control Domain *(Continued)*

domain. For example, you might do this so that you can perform other configuration commands on that domain or other domains. With this command, you can undo the delayed reconfiguration operation and do other configuration operations on this or other domains.

```
# ldm cancel-op reconf primary
```

EXAMPLE 10 Migrate a Domain

You can migrate a logical domain to another machine. This example shows a successful migration.

```
# ldm migrate ldg1 root@dt90-187:ldg
Target password:
```

EXAMPLE 11 List Configurations

The following examples show how to view the configurations. The first command shows the configurations that are stored on the SP. The second command shows the configurations on the SP as well as information about the autosave configurations on the control domain.

```
# ldm ls-config
factory-default
3guests [current]
data1
reconfig_primary
split1
# ldm ls-config -r
3guests [newer]
data1 [newer]
reconfig_primary
split1
unit
```

Both the current `3guests` configuration and the `data1` configuration have autosaved changes that have not been saved to the SP. If the system is powercycled while in this state, the Logical Domains Manager would perform the `3guests` autosave recovery based on the specified policy. The autosave recovery action is taken for `3guests` because it is marked as current.

The `reconfig_primary` and `split1` autosave configurations are identical to the versions on the SP, not newer versions.

The `unit` configuration only exists as an autosave configuration on the control domain. There is no corresponding configuration for `unit` on the SP. This situation might occur if the configuration was lost from the SP. A configuration can be lost if the SP is replaced or if a problem occurred with the persistent version of the configuration on the SP. Note that using the `rm-config` command to explicitly remove a configuration also removes the autosave version on the control domain. As a result, no remnants of the configuration remain on either

EXAMPLE 11 List Configurations *(Continued)*

the control domain or on the SP.

EXAMPLE 12 List I/O Devices

The following example lists the I/O devices on the system. The first section shows information about the PCIe bus on the `primary` domain. The second section shows information about the direct I/O-assignable devices.

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

PCIE                 PSEUDONYM  STATUS  DOMAIN
----                 ---------  ------  ------
pci@400/pci@0/pci@c  PCIE1      EMP     -
pci@400/pci@0/pci@9  PCIE2      OCC     ldg1
pci@400/pci@0/pci@d  PCIE3      OCC     ldg2
pci@400/pci@0/pci@8  MB/SASHBA  OCC     primary
pci@500/pci@0/pci@9  PCIE0      EMP     -
pci@500/pci@0/pci@d  PCIE4      OCC     ldg2
pci@500/pci@0/pci@c  PCIE5      OCC     ldg1
pci@500/pci@0/pci@8  MB/NET0    OCC     primary
```

Exit Status

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

Attributes

See the `attributes(5)` man page for a description of the following attributes.

Attribute Type	Attribute Value
Availability	`SUNWldm`
Interface Stability	Uncommitted

See Also

`dumpadm(1M)`, `ifconfig(1M)`, `shutdown(1M)`, `vntsd(1M)`, `attributes(5)`

Oracle VM Server for SPARC 2.2 Administration Guide

Name ldmconfig – Oracle VM Server for SPARC Configuration Assistant

Synopsis `ldmconfig [-cdh]`

Description The `ldmconfig` utility, the Oracle VM Server for SPARC Configuration Assistant, is a terminal-based application that streamlines the setup of systems that can run Oracle VM Server for SPARC. Only chip multithreaded-based (CMT) systems can be used to run Oracle VM Server for SPARC software.

Note that the `ldmconfig` command is supported *only* on Oracle Solaris 10 systems.

`ldmconfig` inspects the system to provide the user with a default set of choices to generate a valid configuration. After gathering the setup property values, `ldmconfig` creates a configuration that is suitable for setting up a logical domain.

You can run the `ldmconfig` utility by means of a console connection, remote terminal emulator, or `ssh` session.

The Configuration Assistant uses the following options:

`-c`	Checks Oracle Solaris OS media for valid packages
`-d`	Specifies debug mode, which retains run and error logs after completion
`-h`	Displays usage message

Exit Status The following exit values are returned:

0	Successful completion.
>0	An error occurred.

Attributes See the `attributes(5)` man page for a description of the following attributes.

Attribute Type	Attribute Value
Availability	`SUNWconfig`
Interface Stability	Uncommitted

See Also `ldm`(1M), `attributes`(5)

Oracle VM Server for SPARC 2.2 Administration Guide

Name ldmd – Logical Domains Manager daemon

Synopsis `/opt/SUNWldm/bin/ldmd`

Description The `ldmd` daemon is referred to as the Logical Domains Manager. It is the daemon program for the `ldm` command, which is used to create and manage logical domains. There can be only one Logical Domains Manager per server. The `ldmd` daemon runs on the control domain, which is the initial domain created by the service processor (SP). The control domain is named `primary`.

A logical domain is a discrete logical grouping with its own operating system, resources, and identity within a single system. Each logical domain can be created, destroyed, reconfigured, and rebooted independently, without requiring a power cycle of the server. You can use logical domains to run a variety of applications in different domains and keep them independent for security purposes.

SMF Properties You can use the `svccfg` command to modify the following properties:

`ldmd/autorecovery_policy`
Specifies the autorecovery policy. This property can have one of the following values:

- `autorecovery_policy=1` – Logs warning messages when an autosave configuration is newer than the corresponding running configuration. These messages are logged in the `ldmd` SMF log file. The user must manually perform any configuration recovery. This is the default policy.
- `autorecovery_policy=2` – Displays a notification message if an autosave configuration is newer than the corresponding running configuration. This notification message is printed in the output of any `ldm` command the first time an `ldm` command is issued after each restart of the Logical Domains Manager. The user must manually perform any configuration recovery.
- `autorecovery_policy=3` – Automatically updates the configuration if an autosave configuration is newer than the corresponding running configuration. This action overwrites the SP configuration that will be used during the next power cycle. This configuration is updated with the newer configuration that is saved on the control domain. This action does not impact the currently running configuration. It only impacts the configuration that will be used during the next power cycle. A message is also logged, which states that a newer configuration has been saved on the SP and that it will be booted the next time the system performs a power cycle. These messages are logged in the `ldmd` SMF log file.

`ldmd/incoming_migration_enabled`
Enables a guest domain migration from another system to this system if `xmpp_enabled` is also set to `true`. The default value is `true`.

`ldmd/outgoing_migration_enabled`
Enables a guest domain migration from this system to another system if `xmpp_enabled` is also set to `true`. The default value is `true`.

`ldmd/xmpp_enabled`
Enables the `ldmd` XMPP server to listen for configuration requests from third-party management applications. Also, permits the `ldmd` daemon to communicate with the `ldmd` daemon on another system to coordinate a migration between the two systems. The default value is `true`.

Attributes

See the `attributes(5)` man page for a description of the following attributes.

Attribute Type	Attribute Value
Availability	`SUNWldm`
Interface Stability	Uncommitted

See Also

`svcs(1)`, `drd(1M)`, `ldm(1M)`, `ldmad(1M)`, `svcadm(1M)`, `vntsd(1M)`, `attributes(5)`, `smf(5)`

Notes

The `ldmd` service is managed by the Service Management Facility (SMF) and uses the `svc:/ldoms/ldmd:default` service identifier. See the `smf(5)` man page.

Use the `svcadm` command to perform administrative actions on this service, such as enabling, disabling, or requesting a restart. Use the `svcs` command to query the service's status.

For more information about the `ldmd` SMF properties, see the *Oracle VM Server for SPARC 2.2 Administration Guide*.

Name ldmp2v – command-line interface for the Oracle VM Server for SPARC Physical-to-Virtual (P2V) Conversion Tool

Synopsis

```
ldmp2v collect [-a flash|none] [-O "flarcreate-options"] [-v] [-x mount-point [-x ...]]
  -d data-dir
ldmp2v prepare [-b zvol|file|disk] [-B backend:volume:vdisk [-B ...]] [-c cpu]
  [-m mount-point:size [-m ...]] [-M memsize] [-o keep-hostid] [-o keep-mac] [-p prefix]
  [-s] [-v] [-x no-auto-adjust-fs] [-x remove-unused-slices] -d data-dir domain
ldmp2v prepare -R guest-root  [-c cpu] [-M memsize] [-o keep-hostid] [-o keep-mac]
  [-v] -d data-dir domain
ldmp2v prepare -C domain
ldmp2v convert -i install-image -d data-dir [-v] [-x skip-ping-test] domain
ldmp2v convert [-j] -n interface -d data-dir [-v] [-x skip-ping-test] domain
```

Description The Oracle VM Server for SPARC Physical-to-Virtual (P2V) Conversion Tool automatically converts an existing physical system to a virtual system that runs the Oracle Solaris 10 OS in a logical domain on a chip multithreading (CMT) system. The source system can be any `sun4u` SPARC system that runs at least the Solaris 8, Solaris 9, or Oracle Solaris 10 OS, or a non-Logical Domains `sun4v` system that runs the Oracle Solaris 10 OS.

The conversion from a physical system to a virtual system is performed in the following phases:

- **Collection phase.** Runs on the physical source system. `collect` creates a file system image of the source system based on the configuration information that it collects about the source system.
- **Preparation phase.** Runs on the control domain of the target system. `prepare` creates the logical domain on the target system based on the configuration information collected in the `collect` phase. The file system image is restored to one or more virtual disks. The image is modified to enable it to run as a logical domain.
- **Conversion phase.** Runs on the control domain of the target system. In the `convert` phase, the created logical domain is converted into a logical domain that runs the Solaris 10 OS by using the standard Solaris upgrade process.

The following sections describe how the conversion from a physical system to a virtual system is performed in phases.

Collection Phase

```
ldmp2v collect [-a flash|none] [-O "flarcreate-options"] [-v] [-x mount-point [-x ...]]
  -d data-dir
```

The `ldmp2v collect` command uses the following options:

`-a flash|none`	Specifies the archiving method to use. Valid values are `flash` or `none`. The default is `flash`.
`-O` "*flarcreate-options*"	Specifies a quoted list of options to pass to the `flarcreate` command. The only `flarcreate` options permitted are `-c` and `-x`. The `-c` option compresses the archive, and the `-x` option excludes

files or directories from the archive. You can specify more than one `flarcreate` option. The `-O` option can only be used when you use `-a flash` to specify the flash archive method.

`-v`	Uses verbose mode, which increases the verbosity of the messages that are issued by `ldmp2v`.
`-x` *mount-point*	Excludes the file system, mounted on *mount-point*, from the archive.
`-d` *data-dir*	Specifies the per-system directory in which to store P2V files. For the collection phase, this directory must be writable by `root`. Any intermediate directories are created automatically.

Preparation Phase

```
ldmp2v prepare [-b zvol|file|disk] [-B backend:volume:vdisk [-B ...]] [-c cpu]
  [-m mount-point:size [-m ...]] [-M memsize] [-o keep-hostid] [-o keep-mac]
  [-p prefix] [-s] [-v] [-x no-auto-adjust-fs] [-x remove-unused-slices]
  -d data-dir domain
ldmp2v prepare -R guest-root  [-c cpu] [-M memsize] [-o keep-hostid] [-o keep-mac]
  [-v] -d data-dir domain
ldmp2v prepare -C domain
```

The `ldmp2v prepare` command uses the following operand and options:

domain	Specifies the logical domain on which to operate.
`-b zvol|file|disk`	Specifies the back-end type to use. The virtual disks can be backed by ZFS volumes, `zvol`, plain files, `file`, or physical disks or volume manager volumes, `disk`. This option overrides the setting for `BACKEND_TYPE` in `/etc/ldmp2v.conf`.
`-B` *backend*`:`*volume*`:`*vdisk*	Specifies the name of the back-end device and, optionally, the name of the volume and virtual disk to create. If the *volume* or *vdisk* value is omitted, a default name is used. You can omit values by specifying the colon character (`:`) for each value to omit. For example, these are valid uses of the `-B` option: `-B ::vdisk01` and `-B :volume001`. This option is required for the `disk` back end and should at least specify a back-end device, such as `/dev/dsk/c0t2d0s2` or `/dev/md/dsk/d100`. For the `disk` back end, specify one `-B` option for each disk that is present in the manifest for the physical system. For the `zvol` and `file` back ends, you can use *backend* to specify a file or ZFS dataset that `ldmp2v` should create for the virtual disk. For example, `-B data/ldom1/disk0`. Use the `-B` option to specify the back-end name and override the default

name. The default name is generated by the `-p` option, or by the `BACKEND_PREFIX` setting in `/etc/ldmp2v.config` and the domain name.

`-c` *cpu*
: Allocates the number of VCPUs to the logical domain. By default, `ldmp2v` allocates a VCPU for each CPU on the physical system.

`-C`
: Cleans up the specified domain.

`-d` *data-dir*
: Specifies the per-system directory where the files required for P2V are located.

`-m` *mount-point*:*size*
: Resizes the underlying slice and disk for the file system at *mount-point*. The size is specified as *numunit*. *num* is the amount of space and *unit* is `b` for blocks, `k` for Kbytes, `m` for Mbytes, or `g` for Gbytes. You can specify this option more than one time. This option disables the automatic resizing of `/`, `/usr`, and `/var`. If *mount-point* is `swap`, the first configured swap device is resized to *size*.

`-M` *memsize*
: Specifies the amount of memory to allocate to the logical domain. The memory size is specified as *numunit*, where *num* is the amount of memory and *unit* is one of the following:

- `m` or `M` represents Mbytes
- `g` or `G` represents Gbytes

If *unit* is not specified, the unit is Mbytes.

By default, the `ldmp2v` command allocates the same amount of memory that is in the physical system to the logical domain. If required, the memory size specified by the `-M` option is adjusted upward to 1 Gbyte to satisfy the minimum memory size for a guest domain.

`-o keep-hostid`
: Transfers the host ID of the physical system to the logical domain. By default, the Logical Domains Manager assigns a new unique host ID.

`-o keep-mac`
: Transfers the MAC addresses of the physical system to the logical domain. By default the Logical Domains Manager assigns a new unique MAC address.

`-p` *prefix*
: Specifies the location where backend devices will be created. Denotes the ZFS dataset for the `zvol` backend, or a directory relative to `/` for the `file` backend. This option overrides the `BACKEND_PREFIX` parameter in `/etc/ldmp2v.conf`.

`-R` *guest-root*	Selects non-automatic mode. The OS image modification steps are applied to the file system rooted at *guest-root*. Updates the `/etc/vfstab` of the logical domain to match the file system layout below *guest-root*.
`-s`	Creates sparse backend devices. This option overrides the `BACKEND_SPARSE` parameter in `/etc/ldmp2v.conf`.
`-v`	Uses verbose mode, which increases the verbosity of the messages that are issued by `ldmp2v`.
`-x no-auto-adjust-fs`	Prevents the automatic size adjustment of the `/`, `/usr`, and `/var` file systems to 10 Gbytes total. Use this option with care because the size of the existing file systems might not be sufficient to upgrade to a newer Solaris release. You can manually resize file system sizes by using the `-m` option.
`-x remove-unused-slices`	Reduces the size of the virtual disk by not creating slices that do not hold a file system or a swap device.

Conversion Phase

`ldmp2v convert -i` *install-image* `-d` *data-dir* `[-v] [-x skip-ping-test]` *domain*
`ldmp2v convert [-j] -n` *interface* `-d` *data-dir* `[-v] [-x skip-ping-test]` *domain*

The `ldmp2v convert` command uses the following options:

`-d` *data-dir*	Specifies the per-system directory where the files required for P2V are located.
`-i` *install-image*	Specifies the path to the Solaris 10 OS DVD ISO image to use for upgrade.
`-j`	Uses Custom JumpStart, which requires that a JumpStart server and JumpStart client are properly configured.
`-n` *interface*	Specifies the virtual network interface from which to boot when using a network install server.
`-v`	Uses verbose mode, which increases the verbosity of the messages issued by `ldmp2v`.
`-x skip-ping-test`	Skips the `ping` test that is performed to determine whether the IP addresses of the source system are up. Use this option *only* if you are certain that no duplicate IP addresses will exist, such as when the original system is not active.

Caution – Before you begin the conversion phase, shut down the original physical system, as the logical domain uses the same IP addresses, and possibly also MAC addresses, as the physical system.

If any IP address of the physical system is active, the `ldmp2v convert` command exits with an error message.

Examples This section includes examples for the three phases.

EXAMPLE 1 Collection Phase Examples

The following examples show how you might use the `ldmp2v collect` command.

- **Sharing an NFS-mounted file system.** The following example shows the simplest way to perform the `collect` phase, where the source and target systems share an NFS-mounted file system.

  ```
  # ldmp2v collect -d /home/dana/p2v/volumia
  ```

- **Not sharing an NFS-mounted file system.** When the source and target systems do not share an NFS-mounted file system, the file system image can be written to local storage and then later copied to the control domain. Use the flash archiving method that is provided by `ldmp2v`. The flash tool automatically excludes the archive it creates.

  ```
  # ldmp2v collect -d /home/dana/p2v/volumia -a flash
  ```

- **Skip file-system backup step.** If backups of the system are already available using a third-party backup tool such as NetBackup, you can skip the file system backup step by using the `none` archiving method. When you use this option, only the system configuration manifest is created.

  ```
  # ldmp2v collect -d /home/dana/p2v/volumia -a none
  ```

  Note – If the directory specified by `-d` is not shared by the source and target systems, copy the contents of that directory to the control domain. The directory contents must be copied to the control domain prior to beginning the preparation phase.

- **Exclude a file or directory from the flash archive.** When using the flash archiving method, you can exclude a file or directory from the archive by passing options to the `flarcreate` command. This capability requires that you have at least the following patch revisions installed on the source system:

  - **Solaris 8 OS:** Patch ID 109318-34
  - **Solaris 9 OS:** Patch ID 113434-06

  ```
  # ldmp2v collect -d /home/dana/p2v/volumia -a flash
    -O "-x /path/to/file -x /some/dir"
  ```

EXAMPLE 2 Preparation Phase Examples

The following examples show how you might use the `ldmp2v prepare` command.

- The following example creates a logical domain called `volumia` by using the defaults configured in `/etc/ldmp2v.conf` while keeping the MAC addresses of the physical system:

  ```
  # ldmp2v prepare -d /home/dana/p2v/volumia -o keep-mac volumia
  ```

EXAMPLE 2 Preparation Phase Examples *(Continued)*

- The following shows how to completely remove a domain and its backend devices by using the `-C` option:

  ```
  # ldmp2v prepare -C volumia
  ```

- The following shows how to resize a file system and the swap device during P2V by using the `-m` option:

  ```
  # ldmp2v prepare -d /home/dana/p2v/normaal -m /:8g -m swap:4g normaal
  ```

- The following shows how to use Solaris Volume Manager metadevices `d100` and `d101` as back-end devices for the guest domain and to set the name of the first virtual disk to `vdisk100`:

  ```
  # ldmp2v prepare -b disk -B /dev/md/dsk/d100::vdisk100 -B /dev/md/dsk/d101
    -d /p2v/volumia volumia
  ```

- The following shows how to use ZFS volumes with non-default ZFS volume names:

  ```
  # ldmp2v prepare -b zvol -B tank/ldom1/zvol1 -B tank/ldom1/zvol2 -d /p2v/volumia
    volumia
  ```

EXAMPLE 3 Conversion Phase Examples

The following examples show how you might use the `ldmp2v convert` command.

- **Using a network install server.** The `ldmp2v convert` command boots the logical domain over the network by using the specified virtual network interface. You must run the `setup_install_server` and `add_install_client` scripts on the install server.

  Optionally, you can use the Custom JumpStart feature to perform a completely hands-off conversion.

  The following shows how to use a network install server to upgrade your system:

  ```
  # ldmp2v convert -n vnet0 -d /p2v/volumia volumia
  ```

  The following shows how to use Custom JumpStart to upgrade your system:

  ```
  # ldmp2v convert -j -n vnet0 -d /p2v/volumia volumia
  ```

- **Using an ISO image.** The `ldmp2v convert` command attaches the Solaris DVD ISO image to the logical domain and boots from it. To upgrade, answer all `sysid` prompts and select Upgrade.

  Note – The answers to the `sysid` questions are only used during the upgrade process, so you can select the simplest options (non-networked, no naming service, and so on). The system's original identity is preserved by the upgrade and takes effect on the reboot after the upgrade is complete. The time required to perform the upgrade depends on the Solaris cluster that is installed on the original system.

EXAMPLE 3 Conversion Phase Examples *(Continued)*

```
# ldmp2v convert -i /tank/iso/s10s_u5.iso -d /home/dana/p2v/volumia volumia
```

Exit Status

The following exit values are returned:

0	Successful completion.
>0	An error occurred.

Attributes

See the `attributes(5)` man page for a description of the following attributes.

Attribute Type	Attribute Value
Availability	`SUNWldmp2v`
Interface Stability	Uncommitted

See Also

`ldm(1M)`, `attributes(5)`

Oracle VM Server for SPARC 2.2 Administration Guide