Oracle® VM Server for SPARC 2.0 管理指南

文件号码 821–2934–10
2010 年 9 月

ORACLE

110322@25097

目录

前言 13

1 Oracle VM Server for SPARC 软件概述 17
虚拟机管理程序和 Logical Domains 17
Logical Domains Manager 19
域的角色 19
命令行界面 20
虚拟输入/输出 20
资源配置 21
持久性配置 21
Oracle VM Server for SPARC 物理机到虚拟机转换工具 22
Oracle VM Server for SPARC Configuration Assistant 22

2 安装和启用软件 23
在新系统上安装 Oracle VM Server for SPARC 软件 24
更新 Oracle Solaris OS 24
升级系统固件 24
下载 Logical Domains Manager 26
安装 Logical Domains Manager 27
启用 Logical Domains Manager 守护进程 29
升级已使用 Oracle VM Server for SPARC 的系统 30
升级 Oracle Solaris OS 30
升级 Logical Domains Manager 和系统固件 31
升级至 Oracle VM Server for SPARC 2.0 软件 32
出厂默认配置和禁用 Logical Domains 33
▼ 删除所有来宾域 33
▼ 删除所有逻辑域配置 33

▼恢复出厂默认配置 34
▼禁用 Logical Domains Manager 34
▼删除 Logical Domains Manager 34
▼从服务处理器恢复出厂默认配置 35

3 安全 37
Logical Domains Manager 授权 37
创建授权和配置文件并为用户帐户分配角色 38
管理用户授权 38
管理用户配置文件 39
给用户分配角色 40
配置 RBAC 以进行来宾控制台访问 41
启用并使用 BSM 审计 41
▼启用 BSM 审计 42
▼检验 BSM 审计是否已启用 42
▼禁用 BSM 审计 42
▼显示审计输出 42
▼轮转审计日志 42

4 设置服务和控制域 43
输出消息 43
创建默认服务 44
▼创建默认服务 44
控制域的初始配置 45
▼设置控制域 45
重新引导以使用 Logical Domains 46
▼重新引导 46
启用控制/服务域与其他域之间的联网 46
▼将虚拟交换机配置为主接口 47
启用虚拟网络终端服务器守护进程 47
▼启用虚拟网络终端服务器守护进程 48

5 设置来宾域 49
创建和启动来宾域 49

▼创建和启动来宾域 49
在来宾域上安装 Oracle Solaris OS 52
▼在来宾域上使用 DVD 安装 Oracle Solaris OS 52
▼从 Oracle Solaris ISO 文件的来宾域上安装 Oracle Solaris OS 53
▼在来宾域上执行 JumpStart 操作 54

6 设置 I/O 域 57
I/O 域概述 57
分配 PCIe 总线 58
▼通过分配 PCIe 总线创建 I/O 域 59
分配 PCIe 端点设备 62
直接 I/O 硬件和软件要求 64
直接 I/O 限制 64
规划 PCIe 端点设备配置 64
重新引导 `primary` 域 66
更改 PCIe 硬件 66
▼通过分配 PCIe 端点设备创建 I/O 域 67

7 使用虚拟磁盘 73
虚拟磁盘简介 73
管理虚拟磁盘 74
▼添加虚拟磁盘 74
▼多次导出虚拟磁盘后端 75
▼更改虚拟磁盘选项 75
▼更改超时选项 75
▼删除虚拟磁盘 76
虚拟磁盘标识符和设备名称 76
虚拟磁盘外观 77
完整磁盘 77
具有单个分片的磁盘 77
虚拟磁盘后端选项 77
只读 (`ro`) 选项 77
独占 (`excl`) 选项 78
分片 (`slice`) 选项 78
虚拟磁盘后端 79

物理磁盘或磁盘 LUN 79
▼ 将物理磁盘作为虚拟磁盘导出 79
物理磁盘分片 80
▼ 将物理磁盘分片作为虚拟磁盘导出 80
▼ 导出分片 2 80
文件和卷 81
配置虚拟磁盘多路径 84
▼ 配置虚拟磁盘多路径 85
CD、DVD 和 ISO 映像 86
▼ 将 CD 或 DVD 从服务域导出到来宾域 87
▼ 从 primary 域导出 ISO 映像以安装来宾域 88
虚拟磁盘超时 89
虚拟磁盘和 SCSI 90
虚拟磁盘和 format(1M) 命令 90
将 ZFS 用于虚拟磁盘 90
在服务域中配置 ZFS 池 91
使用 ZFS 存储磁盘映像 91
创建磁盘映像的快照 92
使用克隆置备新域 93
在 Logical Domains 环境中使用卷管理器 94
在卷管理器之上使用虚拟磁盘 94
在虚拟磁盘之上使用卷管理器 96

8 使用虚拟网络 97
虚拟网络简介 97
虚拟交换机 98
虚拟网络设备 98
管理虚拟交换机 99
▼ 添加虚拟交换机 100
▼ 设置现有虚拟交换机的选项 101
▼ 移除虚拟交换机 101
管理虚拟网络设备 101
▼ 添加虚拟网络设备 101
▼ 设置现有虚拟网络设备的选项 102
▼ 移除虚拟网络设备 103

虚拟设备标识符和网络接口名称 103
▼ 查找 Oracle Solaris OS 网络接口名称 104
自动或手动分配 MAC 地址 105
分配给 Logical Domains 的 MAC 地址范围 105
自动分配算法 105
检测重复的 MAC 地址 106
释放的 MAC 地址 107
将网络适配器和 Logical Domains 结合使用 107
▼ 确定网络适配器是否与 GLDv3 兼容 107
针对 NAT 和路由配置虚拟交换机和服务域 108
▼ 设置虚拟交换机以实现与域的外部连接 109
在 Logical Domains 环境中配置 IPMP 110
在域中将虚拟网络设备配置到 IPMP 组中 110
在服务域中配置并使用 IPMP 111
在 Logical Domains 虚拟网络中使用基于链路的 IPMP 112
在 Logical Domains 1.3 之前的发行版中配置并使用 IPMP 114
使用 VLAN 标记 116
端口 VLAN ID (PVID) 116
VLAN ID (VID) 117
▼ 为虚拟交换机和虚拟网络设备分配 VLAN 117
▼ 安装服务器位于 VLAN 中时安装来宾域 118
使用 NIU 混合 I/O 118
▼ 配置虚拟交换机和 NIU 网络设备 120
▼ 启用混合模式 120
▼ 禁用混合模式 121
将链路聚合和虚拟交换机结合使用 121
配置巨型帧 122
▼ 配置虚拟网络和虚拟交换机设备以使用巨型帧 122
与 vnet 和 vsw 驱动程序的早期（巨型帧无感知）版本的兼容性 124

9 迁移域 127
域迁移介绍 127
迁移操作概述 128
软件兼容性 128
对迁移操作进行验证 128

迁移域 129
执行模拟运行 129
执行非交互式迁移 129
迁移活动域 129
迁移活动域中的 CPU 130
迁移活动域中的内存 130
迁移活动域中的物理 I/O 设备 131
迁移活动域中的虚拟 I/O 设备 131
迁移活动域中的 NIU 混合输入/输出 132
迁移活动域中的加密单元 132
活动域中的延迟重新配置 133
在活动域处于弹性模式时进行迁移 133
对其他域的操作 133
迁移绑定域或非活动域 133
迁移绑定域或非活动域中的 CPU 133
迁移绑定域或非活动域中的虚拟输入/输出 134
迁移绑定域或非活动域中的 PCIe 端点设备 134
监视正在进行的迁移 134
取消正在进行的迁移 135
从失败的迁移中恢复 135
迁移示例 135

10 管理资源 139
资源重新配置 139
动态重新配置 139
延迟重新配置 140
资源分配 140
CPU 分配 141
启用整体核心约束 141
禁用整体核心约束 142
将 CPU 分配到控制域 142
整体核心约束和其他域功能之间的交互作用 143
使用内存动态重新配置 144
添加内存 144
删除内存 145

部分内存 DR 请求 145
重新配置控制域内存 145
动态重新配置和延迟重新配置 146
内存对齐 146
内存 DR 示例 148
使用电源管理 151
列出受 CPU 电源管理的导线束 152
使用动态资源管理 154
列出域资源 156
计算机可读的输出 156
标志定义 156
使用率统计信息定义 157
查看各种列表 157
列出约束 160

11 管理配置 163
保存域配置用于将来重建 163
▼ 保存域配置 163
▼ 从 XML 文件恢复域配置 (`ldm add-domain`) 164
▼ 从 XML 文件恢复域配置 (`ldm init-system`) 164
管理 Logical Domains 配置 165
▼ 修改自动恢复策略 166

12 执行其他管理任务 169
在 CLI 中输入名称 169
文件名 (*file*) 和变量名 (*var-name*) 169
虚拟磁盘服务器 *backend* 和虚拟交换机设备名称 169
配置名称 (*config-name*) 170
所有其他名称 170
通过网络连接到来宾控制台 170
使用控制台组 170
▼ 将多个控制台组成一个组 171
停止高负载的域会超时 171
操作具有 Oracle VM Server for SPARC 的 Oracle Solaris OS 172
OpenBoot 固件在 Oracle Solaris OS 启动之后不可用 172

对服务器执行关开机循环 172
请勿对电源管理域中的活动 CPU 使用 psradm(1M) 命令 172
在 Oracle Solaris OS 中发出中断的结果 172
停止或重新引导控制域的结果 173
将 Logical Domains Manager 与服务处理器结合使用 173
▼ 将域配置重置为默认配置或其他配置 174
配置域依赖关系 174
域依赖关系示例 175
依赖关系循环 176
通过映射 CPU 和内存地址来确定出错位置 177
CPU 映射 177
内存映射 178
CPU 和内存映射示例 178
使用通用唯一标识符 179
虚拟域信息命令和 API 180

A Oracle VM Server for SPARC 物理机到虚拟机转换工具 181
Oracle VM Server for SPARC P2V 工具概述 181
收集阶段 182
准备阶段 182
转换阶段 182
后端设备 183
安装 Oracle VM Server for SPARC P2V 工具 184
先决条件 184
限制 184
▼ 安装 Oracle VM Server for SPARC P2V 工具 185
使用 ldmp2v 命令 186

B Oracle VM Server for SPARC Configuration Assistant 193
使用 Configuration Assistant (GUI) 193
使用 Configuration Assistant (ldmconfig) 194
安装 Configuration Assistant 194
ldmconfig 功能 194

C Logical Domains Manager 发现 197
发现运行 Logical Domains Manager 的系统 197
多播通信 197
消息格式 197
▼ 发现在子网上运行的 Logical Domains Manager 198

D 将 XML 接口与 Logical Domains Manager 结合使用 201
XML 传输 201
XMPP 服务器 202
本地连接 202
XML 协议 202
请求和响应消息 202
事件消息 206
注册和注销 206
<LDM_event> 消息 207
事件类型 208
Logical Domains Manager 操作 210
Logical Domains Manager 资源和属性 211
域信息 (ldom_info) 资源 211
CPU (cpu) 资源 212
MAU (mau) 资源 213
内存 (memory) 资源 213
虚拟磁盘服务器 (vds) 资源 214
虚拟磁盘服务器卷 (vds_volume) 资源 214
磁盘 (disk) 资源 215
虚拟交换机 (vsw) 资源 216
网络 (network) 资源 217
虚拟控制台集中器 (vcc) 资源 217
变量 (var) 资源 218
物理 I/O 设备 (physio_device) 资源 218
SP 配置 (spconfig) 资源 219
虚拟数据平面通道服务 (vdpcs) 资源 219
虚拟数据平面通道客户机 (vdpcc) 资源 220
控制台 (console) 资源 220
域迁移 221

E Logical Domains Manager XML 模式 223
LDM_interface XML 模式 223
LDM_Event XML 模式 225
ovf-envelope.xsd 模式 226
ovf-section.xsd 模式 228
ovf-core.xsd 模式 228
ovf-virtualhardware.xsc 模式 233
cim-rasd.xsd 模式 234
cim-vssd.xsd 模式 238
cim-common.xsd 模式 238
GenericProperty XML 模式 242
Binding_Type XML 模式 242

词汇表 245

索引 255

前言

《Oracle VM Server for SPARC 2.0 管理指南》提供了有关在支持的服务器、刀片和服务器模块上运行的 Oracle VM Server for SPARC 2.0 软件的详细信息，包括概述和安全注意事项方面的信息，以及安装、配置、修改和执行常规任务的具体过程。请参见《Oracle VM Server for SPARC 2.0 Release Notes》中的“Supported Platforms”。

本指南面向这些服务器的系统管理员，他们具有 UNIX 系统和 Oracle Solaris 操作系统 (Oracle Solaris OS) 的相关知识。

相关文档

下表显示了可用于 Oracle VM Server for SPARC 2.0 发行版的文档。这些文档以 HTML 和 PDF 格式提供，除非另有说明。

表 P–1 相关文档

应用	书名	文件号码
Oracle VM Server for SPARC 2.0 软件	《Oracle VM Server for SPARC 2.0 管理指南》	821-1485
	《Oracle VM Server for SPARC 2.0 Release Notes》	821-1487
	《Oracle VM Server for SPARC 2.0 Reference Manual》 Oracle Solaris 10 Reference Manual Collection ■ `drd`(1M) 手册页 ■ `vntsd`(1M) 手册页	821-1486
Logical Domains 软件基础	《Beginners Guide to LDoms: Understanding and Deploying Logical Domains Software》(PDF)	820-0832
Logical Domains 管理信息库 (Management Information Base, MIB)	《Logical Domains (LDoms) MIB 1.0.1 Administration Guide》	820-2319-10
	《Logical Domains (LDoms) MIB 1.0.1 Release Notes》	820-2320-10
Oracle Solaris OS：安装和配置	Oracle Solaris 10 9/10 Release and Installation Collection - Simplified Chinese	N/A

可以从 http://docs.sun.com 查找与您的服务器、软件或 Oracle Solaris OS 相关的文档。使用 "Search"（搜索）框查找所需的文档和信息。

文档、支持和培训

有关其他资源，请参见以下 Web 站点：

- 文档 (http://docs.sun.com)
- 支持 (http://www.oracle.com/us/support/systems/index.html)
- 培训 (http://education.oracle.com) – 单击左侧导航栏中的 Sun 链接。

Oracle 欢迎您提出意见

Oracle 欢迎您针对其文档质量和实用性提出意见和建议。如果您发现任何错误，或有其他任何改进建议，请转至 http://docs.sun.com，然后单击 Feedback（反馈）。请提供文档的标题和文件号码，以及章节和页码（如果有）。如果您需要回复，请告知。

Oracle 技术网络 (http://www.oracle.com/technetwork/index.html) 提供了一系列与 Oracle 软件相关的资源：

- 在讨论论坛 (http://forums.oracle.com) 上讨论技术问题和解决方案。
- 通过 Oracle By Example (http://www.oracle.com/technetwork/tutorials/index.html) 获得逐步骤实用教程。
- 下载样例代码 (http://www.oracle.com/technology/sample_code/index.html)。

印刷约定

下表介绍了本书中的印刷约定。

表 P–2 印刷约定

字体或符号	含义	示例
AaBbCc123	命令、文件和目录的名称；计算机屏幕输出	编辑 .login 文件。 使用 ls -a 列出所有文件。 machine_name% you have mail.
AaBbCc123	用户键入的内容，与计算机屏幕输出的显示不同	machine_name% **su** Password:
aabbcc123	要使用实名或值替换的命令行占位符	删除文件的命令为 rm *filename*。

表 P–2 印刷约定 （续）

字体或符号	含义	示例
AaBbCc123	保留未译的新词或术语以及要强调的词	这些称为 *Class* 选项。 **注意**：有些强调的项目在联机时以粗体显示。
新词术语强调	新词或术语以及要强调的词	**高速缓存**是存储在本地的副本。 请**勿**保存文件。
《书名》	书名	阅读《用户指南》的第 6 章。

命令中的 shell 提示符示例

下表列出了 C shell、Bourne shell 和 Korn shell 的缺省 UNIX 系统提示符和超级用户提示符。

表 P–3 shell 提示符

shell	提示符
C shell 提示符	`machine_name%`
C shell 超级用户提示符	`machine_name#`
Bourne shell 和 Korn shell 提示符	`$`
Bourne shell 和 Korn shell 超级用户提示符	`#`

◆ ◆ ◆ 第 1 章

1 Oracle VM Server for SPARC 软件概述

本章对 Oracle VM Server for SPARC 软件进行了概述。

Oracle VM Server for SPARC 软件依赖于特定 Oracle Solaris OS 版本、所需的软件修补程序以及特定版本的系统固件。有关更多信息，请参见《Oracle VM Server for SPARC 2.0 Release Notes》中的“Required and Recommended Oracle Solaris OS”。

本章包括以下内容：

- 第 17 页中的 “虚拟机管理程序和 Logical Domains”
- 第 19 页中的 “Logical Domains Manager”
- 第 22 页中的 “Oracle VM Server for SPARC 物理机到虚拟机转换工具”
- 第 22 页中的 “Oracle VM Server for SPARC Configuration Assistant”

虚拟机管理程序和 Logical Domains

本节对支持 Logical Domains 的 SPARC 虚拟机管理程序进行了概述。

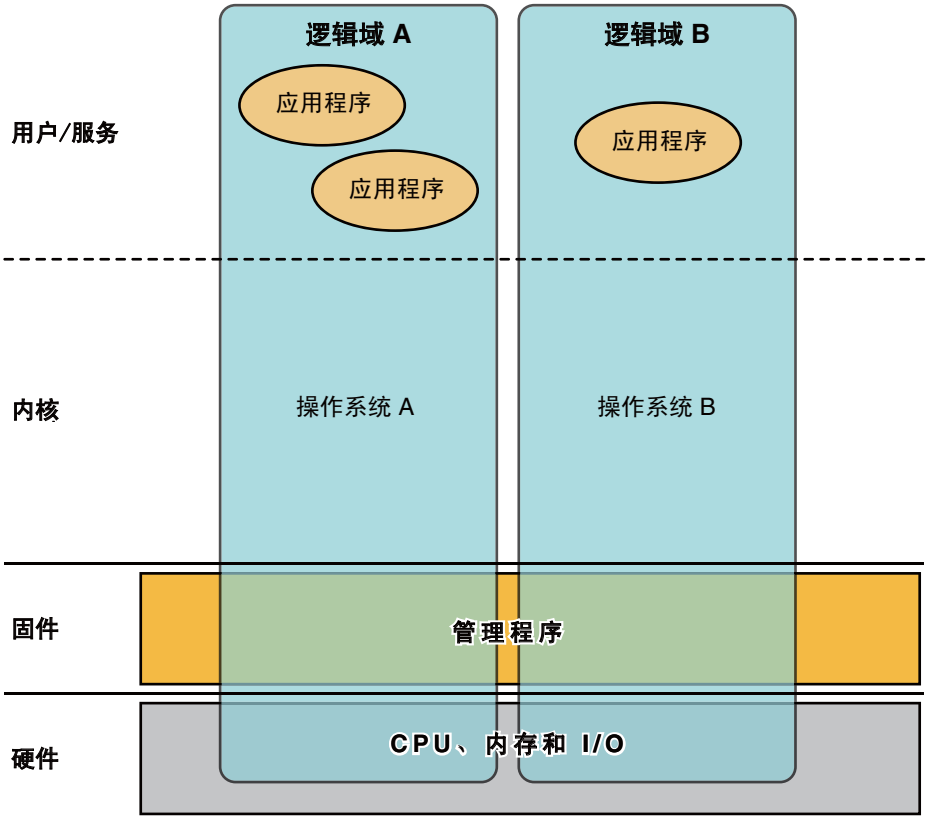
SPARC **虚拟机管理程序**是一个小固件层，提供了可以向其中写入操作系统的稳定虚拟机体系结构。使用虚拟机管理程序的 Oracle Sun 服务器提供的硬件功能，支持虚拟机管理程序对逻辑操作系统活动的控制。

逻辑域是由离散的逻辑分组资源组成的虚拟机。逻辑域在单个计算机系统内具有自己的操作系统和身份。可以单独创建、销毁、重新配置及重新引导每个逻辑域，而无需对服务器执行关开机循环。可以在不同的逻辑域中运行各种应用程序软件，并使其保持相互独立，以获得相应的性能和安全。

每个逻辑域只能发现由虚拟机管理程序设置为可用的那些服务器资源并与之交互。Logical Domains Manager 允许您指定虚拟机管理程序应通过控制域执行的操作。这样，虚拟机管理程序会执行服务器资源的划分，并向多个操作系统环境提供有限的子集。这种划分和置备操作是创建逻辑域的基本机制。下图显示了支持两个逻辑域的虚拟机管理程序。它还显示了构成 Logical Domains 功能的以下各层：

- 应用程序，即用户/服务
- 内核，即操作系统
- 固件，即虚拟机管理程序
- 硬件，包括 CPU、内存和 I/O

图 1–1 支持两个域的虚拟机管理程序



特定的 SPARC 虚拟机管理程序所支持的逻辑域的数量和功能是与服务器相关的特性。虚拟机管理程序可以向给定的逻辑域分配服务器的全部 CPU、内存和 I/O 资源的

子集。这样便可以同时支持多个操作系统，每个操作系统位于自己的逻辑域内。资源可以在不同逻辑域之间以任意粒度重新布置。例如，可以使用 CPU 线程的粒度将 CPU 分配给逻辑域。

每个逻辑域都可以作为完全独立的机器进行管理，并拥有自己的资源，例如：

- 内核、修补程序和调节参数
- 用户帐户和管理员
- 磁盘
- 网络接口、MAC 地址和 IP 地址

可以独立地停止、启动和重新引导每个逻辑域，而无需对服务器执行关开机循环。

虚拟机管理程序软件负责保持各个逻辑域之间的相互独立。虚拟机管理程序软件还提供逻辑域通道 (logical domain channel, LDC)，使逻辑域可以相互通信。使用 LDC，域可以相互提供服务，如联网或磁盘服务。

服务处理器 (service processor, SP)（也称为系统控制器 (system controller, SC)）监视和运行物理机，但不管理逻辑域。Logical Domains Manager 管理逻辑域。

Logical Domains Manager

Logical Domains Manager 用于创建和管理逻辑域，以及将逻辑域映射到物理资源。一台服务器上只能运行一个 Logical Domains Manager。

域的角色

所有逻辑域都是相同的，可以基于为其指定的角色将各个逻辑域区分开。以下是逻辑域可以承担的角色：

- **控制域。**Logical Domains Manager 在此域中运行，使您能够创建和管理其他逻辑域，并将虚拟资源分配给其他域。每台服务器只能有一个控制域。控制域是安装 Oracle VM Server for SPARC 软件时创建的第一个域。控制域名为 `primary`。
- **服务域。**服务域向其他域提供虚拟设备服务，如虚拟交换机、虚拟控制台集中器以及虚拟磁盘服务器。任何域都可以配置为服务域。
- **I/O 域。**I/O 域可以直接访问物理 I/O 设备，如 PCI EXPRESS (PCIe) 控制器中的网卡。I/O 域可以拥有 PCIe 根联合体，或可以通过使用直接 I/O (direct I/O, DIO) 功能拥有 PCIe 插槽或板载 PCIe 设备。请参见第 62 页中的 “分配 PCIe 端点设备”。

  当 I/O 域也用作服务域时，I/O 域能够以虚拟设备形式与其他域共享物理 I/O 设备。
- **根域。**根域已分配有 PCIe 根联合体。此域拥有 PCIe 结构并提供所有与结构相关的服务，如结构错误处理。根域也是 I/O 域，因为它拥有对物理 I/O 设备的直接访问权限。

  您可以拥有的根域的数量取决于您的平台体系结构。例如，如果使用的是 Sun SPARC Enterprise T5440 服务器，最多可以有四个根域。

- **来宾域**。来宾域是非 I/O 域，它使用一个或多个服务域提供的虚拟设备服务。来宾域没有任何物理 I/O 设备，只有虚拟 I/O 设备，如虚拟磁盘和虚拟网络接口。

可以在尚未配置 Logical Domains 的现有系统上安装 Logical Domains Manager。在这种情况下，OS 的当前实例会成为控制域。此外，系统会配置为只有控制域这一个域的 Logical Domains 系统。配置控制域之后，可以平衡其他域中应用程序的负载，从而最有效地利用整个系统。完成此操作的方法是：添加域并将这些应用程序从控制域移动到新域。

命令行界面

Logical Domains Manager 使用命令行界面 (command-line interface, CLI) 来创建和配置逻辑域。该 CLI 是具有多个子命令的单个命令 `ldm`。请参见 `ldm(1M)` 手册页。

必须运行 Logical Domains Manager 守护进程 `ldmd` 才能使用 Logical Domains Manager CLI。

虚拟输入/输出

在 Logical Domains 环境中，最多可以在 UltraSPARC T2 Plus 处理器系统和 SPARC T3 处理器系统上置备 128 个域。这些系统中的 I/O 总线和物理 I/O 插槽的数量有限。因此，无法向这些系统上的所有域提供对物理磁盘和网络设备的独占访问。可以将 PCIe 总线或端点设备分配给域，以为其提供对物理设备的访问。请注意，此解决方案不足以向所有域提供独占的设备访问。请参见第 6 章，设置 I/O 域。通过实施虚拟化 I/O 模型，可以解决对可直接访问的物理 I/O 设备数量的此限制。

没有物理 I/O 访问的所有逻辑域都配置有与服务域进行通信的虚拟 I/O 设备。服务域运行虚拟设备服务以提供对物理设备或其功能的访问。在此客户机-服务器模型中，虚拟 I/O 设备通过称为逻辑域通道 (logical domain channel, LDC) 的域际通信通道相互通信或与对应服务通信。虚拟化 I/O 功能包含对虚拟网络、存储和控制台的支持。

虚拟网络

Logical Domains 使用虚拟网络设备和虚拟网络交换机设备来实现虚拟网络。虚拟网络 (`vnet`) 设备可模仿以太网设备并通过使用点对点通道与系统中的其他 `vnet` 设备进行通信。虚拟交换机 (`vsw`) 设备主要充当所有虚拟网络的传入和传出包的多路复用器。`vsw` 设备可直接与服务域上的物理网络适配器进行通信，并代表虚拟网络发送和接收包。`vsw` 设备还充当简单的第二层交换机，在系统内与其连接的 `vnet` 设备之间交换包。

虚拟存储

虚拟存储基础结构采用客户机-服务器模型，以使逻辑域能够访问未直接分配给它们的块级存储。该模型使用以下组件：

- 虚拟磁盘客户机 (`vdc`)，导出块设备接口
- 虚拟磁盘服务 (`vds`)，代表虚拟磁盘客户机处理磁盘请求并将其提交给位于服务域上的后端存储。

虽然虚拟磁盘显示为客户机域上的常规磁盘，但是大多数磁盘操作都会转发给虚拟磁盘服务并在服务域上进行处理。

虚拟控制台

在 Logical Domains 环境中，`primary` 域的控制台 I/O 会定向至服务处理器。其他所有域的控制台 I/O 会重定向至运行虚拟控制台集中器 (`vcc`) 的服务域。运行 `vcc` 的域通常为 `primary` 域。虚拟控制台集中器服务可以充当所有域的控制台通信流量的集中器，还可以与虚拟网络终端服务器守护进程 (`vntsd`) 通信，以通过 UNIX 套接字提供对每个控制台的访问。

资源配置

运行 Oracle VM Server for SPARC 软件的系统可以配置资源，如虚拟 CPU、虚拟 I/O 设备、加密单元以及内存。某些资源可以在正在运行的域上动态地进行配置，而其他一些资源则必须在停止的域上进行配置。如果无法在控制域上动态地配置资源，必须首先启动延迟重新配置。延迟重新配置会将配置活动推迟到控制域进行重新引导后。有关更多信息，请参见第 139 页中的 “资源重新配置”。

持久性配置

可以使用 `ldm` 命令将逻辑域的当前配置存储在服务处理器上。可以添加配置、指定要使用的配置、删除配置，以及列出配置。请参见 `ldm`(1M) 手册页。还可以指定配置以从 SP 进行引导。请参见第 173 页中的 “将 Logical Domains Manager 与服务处理器结合使用”。

有关管理配置的信息，请参见第 165 页中的 “管理 Logical Domains 配置”。

Oracle VM Server for SPARC 物理机到虚拟机转换工具

Oracle VM Server for SPARC 物理机到虚拟机 (Physical-to-Virtual, P2V) 转换工具可以自动将现有物理系统转换为在芯片多线程 (chip multithreading, CMT) 系统上的逻辑域中运行的虚拟系统。源系统可以是以下任一种：

- 运行 Solaris 8 操作系统或更高版本的任何 `sun4u` SPARC 系统
- 运行 Oracle Solaris 10 OS 但不运行 Oracle VM Server for SPARC 软件的任何 `sun4v` 系统

有关该工具及其安装的信息，请参见附录 A，Oracle VM Server for SPARC 物理机到虚拟机转换工具。有关 `ldmp2v` 命令的信息，请参见 `ldmp2v(1M)` 手册页。

Oracle VM Server for SPARC Configuration Assistant

Oracle VM Server for SPARC Configuration Assistant 将引导您完成通过设置基本属性配置逻辑域的过程。可将其用于配置已安装 Oracle VM Server for SPARC 软件但尚未对其进行配置的任何系统。

收集配置数据之后，Configuration Assistant 将创建适合于作为逻辑域引导的配置。还可以使用 Configuration Assistant 选定的默认值创建可用的系统配置。

Configuration Assistant 可同时作为图形用户界面 (graphical user interface, GUI) 和基于终端的工具提供。

有关更多信息，请参见附录 B，Oracle VM Server for SPARC Configuration Assistant以及 `ldmconfig(1M)` 手册页。

◆ ◆ ◆ 第 2 章

2 安装和启用软件

本章介绍如何安装或升级启用 Oracle VM Server for SPARC 2.0 软件所需的不同软件组件。使用 Oracle VM Server for SPARC 软件需要以下组件：

- 支持的平台，有关支持的平台列表信息，请参阅《Oracle VM Server for SPARC 2.0 Release Notes》中的“Supported Platforms”。
- 控制域运行的操作系统至少等效于 Oracle Solaris 10 9/10 OS，且已安装《Oracle VM Server for SPARC 2.0 Release Notes 》中的“Required Software and Patches”中推荐的所有修补程序。请参见第 30 页中的 “升级 Oracle Solaris OS”。
- Sun UltraSPARC T2 或 T2 Plus 平台至少需要系统固件版本 7.3.0，SPARC T3 平台至少需要系统固件版本 8.0.0。请参见第 24 页中的 “升级系统固件”。
- 控制域上已安装并启用 Oracle VM Server for SPARC 2.0 软件。请参见第 27 页中的 “安装 Logical Domains Manager”。
- （可选）Oracle VM Server for SPARC 管理信息库 (Management Information Base, MIB) 软件包。有关使用 Logical Domains MIB 的更多信息，请参阅《Logical Domains (LDoms) MIB 1.0.1 Administration Guide》 。

在安装或升级 Logical Domains Manager 之前，必须先在服务器上安装或升级 Oracle Solaris OS 和系统固件。如果系统已经使用 Oracle VM Server for SPARC 软件，请参见第 30 页中的 “升级已使用 Oracle VM Server for SPARC 的系统”。否则，请参见第 24 页中的 “在新系统上安装 Oracle VM Server for SPARC 软件”。

本章包括以下内容：

- 第 24 页中的 “在新系统上安装 Oracle VM Server for SPARC 软件”
- 第 30 页中的 “升级已使用 Oracle VM Server for SPARC 的系统”
- 第 33 页中的 “出厂默认配置和禁用 Logical Domains”

注 – Solaris Security Toolkit (SST) 软件不再同 Oracle VM Server for SPARC 软件一起打包。如果您想要使用最新版本的 SST 软件，请参见《Oracle VM Server for SPARC 2.0 Release Notes》。

在新系统上安装 Oracle VM Server for SPARC 软件

支持 Oracle VM Server for SPARC 软件的 Oracle Sun 平台预安装了 Oracle Solaris 10 OS。最初，平台显示为仅管理一个操作系统的单个系统。安装 Oracle Solaris OS、系统固件以及 Logical Domains Manager 后，原有系统和 Oracle Solaris OS 实例将成为控制域。平台的该首个域被命名为 `primary`，无法更改该名称或销毁该域。从此处，可将平台重新配置为包括管理不同 Oracle Solaris OS 实例的多个域。

更新 Oracle Solaris OS

在全新系统上，您可能希望重新安装 OS，使其符合您的安装策略。此种情况下，要找到应该用于此版本的 Oracle VM Server for SPARC 软件的 Oracle Solaris 10 OS，请参阅《Oracle VM Server for SPARC 2.0 Release Notes》中的“Required and Recommended Oracle Solaris OS”。有关安装 Oracle Solaris OS 的完整说明，请参阅 Oracle Solaris 10 OS 安装指南。可以根据系统需要调整安装。

如果您的系统已经安装，则需要将其升级至应该用于此版本的 Oracle VM Server for SPARC 软件的相应 Oracle Solaris 10 OS。要找到应该用于此版本的 Oracle VM Server for SPARC 软件的 Oracle Solaris 10 OS 以及所需的和建议的修补程序，请参阅<《Oracle VM Server for SPARC 2.0 Release Notes》中的“Required Software and Patches”。有关升级 OS 的完整说明，请参阅 Oracle Solaris 10 9/10 Release and Installation Collection - Simplified Chinese (`http://docs.sun.com/app/docs/coll/1236.11`)。

升级系统固件

以下任务介绍如何使用 Integrated Lights Out Manager (ILOM) 软件升级固件。

有关使用 ILOM 软件升级系统固件的信息，请参见《Sun SPARC Enterprise T5120 and T5220 Servers Topic Set》中的 "Update the Firmware"（更新固件）和《Sun Integrated Lights Out Manager (ILOM) 3.0 CLI 过程指南》(`http://dlc.sun.com/pdf/820-6412-12/820-6412-12.pdf`) 中的“更新 ILOM 固件”。

▼ 升级系统固件

您可以在 `http://www.oracle.com/technetwork/systems/patches/firmware/index.html` 中找到适用于您平台的系统固件。

有关支持的服务器所需的系统固件的信息，请参见《Oracle VM Server for SPARC 2.0 Release Notes》中的“Required System Firmware Patches”。

要从控制域升级系统固件，请参阅系统固件发行说明。

有关为支持的服务器安装和升级系统固件的更多信息，请参阅这些服务器的管理指南或产品说明。

您也可以使用 ILOM Web 界面来升级系统固件，请参见《Sun Integrated Lights Out Manager (ILOM) 3.0 Web 界面过程指南》(http://dlc.sun.com/pdf/820-6411-12/820-6411-12.pdf) 中的“更新 ILOM 固件”。

1 **将系统固件映像下载到运行 `tftp` 服务的另一个系统中。**

   a. **确保 `tftp` 服务在服务器上处于联机状态。**

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

   b. **如果 `tftp` 服务未处于联机状态，则启用此服务。**

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

   c. **将系统固件映像下载到 `/tftpboot` 目录。**

2 **确保已配置了 ILOM 服务处理器网络管理端口。**

必须进行此配置，才能访问网络上的新闪存映像。请参见《Sun SPARC Enterprise T5120 and T5220 Servers Topic Set》中的 "To Configure the Service Processor Network Management Port"（配置服务处理器网络管理端口）和《Sun Integrated Lights Out Manager (ILOM) 3.0 CLI 过程指南》(http://dlc.sun.com/pdf/820-6412-12/820-6412-12.pdf) 中的“更新 ILOM 固件”。

3 **打开一个 SSH 会话以连接到服务处理器。**

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

4 **检验主机的电源是否已关闭。**

   a. **键入以下命令：**

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

   b. **如果主机的电源尚未关闭，键入以下命令：**

```
-> stop /SYS
```

5 **检验 `keyswitch_state` 参数是否设置为 `normal`。**

a. **键入以下命令：**

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

b. **如果该值不是 `normal`，请使用以下命令进行设置：**

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

6 **升级服务处理器闪存映像和主机固件。**

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

`-source` 选项指定系统固件闪存映像的 IP 地址和完整路径名 (URL)。

- *IP-addr* 是网络中可以访问闪存映像的 `tftp` 服务器的 IP 地址。
- *pathname* 是 `tftp` 服务器上的闪存映像的完整路径名。
- *x_x_x* 是系统固件的版本号
- *nn* 是应用于此发行版的内部版本号。
- *server-name* 是服务器的名称。

  例如，对于 SPARC Enterprise T5440 服务器，*server-name* 为 `SPARC_Enterprise_T5440`。

例如，`-source tftp://192.168.1.1/Sun_System_Firmware-7_3_0-SPARC_Enterprise_T5440.pkg` 选项指向 IP 地址为 `192.168.1.1` 的服务器上的 `/tftpboot/Sun_System_Firmware-7_3_0-SPARC_Enterprise_T5440.pkg` 文件。

升级闪存映像后，系统将自动复位。

服务处理器将进行复位、运行诊断程序，然后返回登录提示符（在串行控制台上）。

下载 Logical Domains Manager

▼ 下载软件

1 **下载 zip 文件 (`OVM_Server_SPARC-2_0.zip`)。**

您可以在 `http://www.oracle.com/virtualization` 中找到该软件。

2 **解压 zip 文件。**

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

有关该文件的结构及其所含内容的详细信息，请参见《Oracle VM Server for SPARC 2.0 Release Notes》中的“Location of Oracle VM Server for SPARC 2.0 Software”。

安装 Logical Domains Manager

安装 Logical Domains Manager 软件有三种方法：

- 使用安装脚本安装软件包和修补程序。这种方法可自动安装 Logical Domains Manager 软件。请参见第 27 页中的 “自动安装 Logical Domains Manager 软件”。
- 使用 JumpStart 在 Oracle Solaris 网络安装过程中安装软件包。请参见第 28 页中的 “使用 JumpStart 安装 Oracle VM Server for SPARC 2.0 软件”。
- 手动安装软件包。请参见第 28 页中的 “手动安装 Logical Domains Manager 软件”。

注 – 请切记，在安装 Oracle VM Server for SPARC 软件包后，需要手动安装 Logical Domains MIB 软件包。该软件包不会自动与其他软件包一起安装。有关安装和使用 Logical Domains MIB 的更多信息，请参阅《Logical Domains (LDoms) MIB 1.0.1 Administration Guide》。

自动安装 Logical Domains Manager 软件

如果使用 `install-ldm` 安装脚本，则可以有几种选择来指定脚本的运行方式。每种选择都在下面的过程中进行了说明。

- **使用不带任何选项的 `install-ldm` 脚本可自动执行以下操作：**
  - 检查 Oracle Solaris OS 发行版是否至少为 Oracle Solaris 10 9/10 OS
  - 检验是否存在软件包子目录 `SUNWldm/` 和 `SUNWldmp2v/`
  - 检验是否存在必备 Logical Domains 驱动程序软件包 `SUNWldomr` 和 `SUNWldomu`
  - 检验是否尚未安装 `SUNWldm` 和 `SUNWldmp2v` 软件包
  - 安装 Oracle VM Server for SPARC 2.0 软件
  - 检验是否安装了所有软件包
  - 如果已安装 SST (`SUNWjass`)，将提示您强化控制域上的 Oracle Solaris OS。
  - 确定是否使用 Oracle VM Server for SPARC Configuration Assistant (`ldmconfig`) 来执行安装。
- **使用带 `-c` 选项的 `install-ldm` 脚本可在安装软件后自动运行 Oracle VM Server for SPARC Configuration Assistant。**
- **使用带 `-s` 选项的 `install-ldm` 脚本可跳过 Oracle VM Server for SPARC Configuration Assistant 的运行。**
- **使用 `install-ldm` 脚本和以下选项以及 SST 软件可执行以下操作：**
  - `install-ldm -d`。可指定不是以 `-secure.driver` 结尾的 SST 驱动程序。该选项自动执行上述选择中列出的所有功能并使用您指定的 SST 自定义驱动程序强化控制域上的 Oracle Solaris OS，例如，`server-secure-myname.driver`。

- `install-ldm -d none`。指定您不希望使用 SST 强化控制域上运行的 Oracle Solaris OS。该选项会自动执行上述选择中所列的强化功能以外的所有功能。建议不要绕过 SST 的使用，只有在您打算使用其他过程强化控制域时，才应绕过 SST 的使用。
- `install-ldm -p`。指定您只想执行启用 Logical Domains Manager 守护进程 (`ldmd`) 和运行 SST 的安装后操作。例如，如果 `SUNWldm` 和 `SUNWjass` 软件包已预安装在服务器上，将使用该选项。

使用 JumpStart 安装 Oracle VM Server for SPARC 2.0 软件

有关使用 JumpStart 的完整信息，请参见《JumpStart Technology: Effective Use in the Solaris Operating Environment》。

▼ 安装 JumpStart 服务器

如果您尚未安装 JumpStart 服务器，则必须执行该操作。有关该过程的完整信息，请参见《Oracle Solaris 10 9/10 安装指南：自定义 JumpStart 和高级安装》。

1. **请参阅《Oracle Solaris 10 9/10 安装指南：自定义 JumpStart 和高级安装》。**

   请执行以下步骤：

   a. **请参见《Oracle Solaris 10 9/10 安装指南：自定义 JumpStart 和高级安装》中的"任务图：准备自定义 JumpStart 安装"。**

   b. **根据"为联网系统创建配置文件服务器"中所述过程设置联网系统。**

   c. **根据"创建配置文件"中所述的过程创建配置文件，并在配置文件中添加一行，以便使用 `package` 配置文件关键字安装 `SUNWldm.v` 软件包。**

      例如，在您的配置文件中添加以下行，可从 HTTP 服务器 `192.168.254.255` 的 `extra` 目录安装 `SUNWldm.v` 软件包。

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

   d. **根据"创建 `rules` 文件"中所述的过程创建 `rules` 文件。**

2. **根据"验证 `rules` 文件"中所述的过程验证 `rules` 文件。**

手动安装 Logical Domains Manager 软件

▼ 手动安装 Oracle VM Server for SPARC 2.0 软件

开始之前 下载 Oracle VM Server for SPARC 2.0 软件（`SUNWldm` 和 `SUNWldmp2v` 软件包）。有关具体说明，请参见第 26 页中的 "下载软件"。

1. **使用 pkgadd 命令安装 SUNWldm.v 和 SUNWldmp2v 软件包。**

   有关 pkgadd 命令的更多信息，请参见 pkgadd(1M) 手册页。

   -G 选项仅在全局区域中安装软件包，-d 选项指定包含 SUNWldm.v 和 SUNWldmp2v 软件包的目录的路径。

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

2. **对交互式提示中的所有问题回答 y（代表赞同）。**

3. **使用 pkginfo 命令检验是否安装了 Oracle VM Server for SPARC 2.0 软件包 SUNWldm 和 SUNWldmp2v。**

   有关 pkginfo 命令的更多信息，请参见 pkginfo(1) 手册页。

   下面显示的修订 (REV) 信息是一个示例。

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

启用 Logical Domains Manager 守护进程

install-ldm 安装脚本自动启用 Logical Domains Manager 守护进程 (ldmd)。当安装 SUNWldm 软件包时，也会自动启用 ldmd 守护进程。启用后，即可创建、修改和控制逻辑域。

▼ 启用 Logical Domains Manager 守护进程

如果已禁用 ldmd 守护进程，请执行以下过程启用此守护进程。

1. **使用 svcadm 命令启用 Logical Domains Manager 守护进程 ldmd。**

   有关 svcadm 命令的更多信息，请参见 svcadm(1M) 手册页。

   ```
   # svcadm enable ldmd
   ```

2. **使用 ldm list 命令检验 Logical Domains Manager 是否正在运行。**

   ldm list 命令应列出当前在系统上定义的所有域。特别是，primary 域应被列出并处于 active 状态。以下示例输出显示在系统上仅定义了 primary 域。

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

升级已使用 Oracle VM Server for SPARC 的系统

本节介绍在已经使用 Oracle VM Server for SPARC 软件的系统上升级 Oracle Solaris OS、固件以及 Logical Domains Manager 组件的过程。

如果您的系统已配置有 Oracle VM Server for SPARC 软件，则必须升级控制域。如果您希望使用 Oracle VM Server for SPARC 2.0 软件的所有功能，则也必须升级其他现有的域。

升级 Oracle Solaris OS

要找到应该用于此版本的 Oracle VM Server for SPARC 软件的 Oracle Solaris 10 OS 以及不同域的所需和建议修补程序，请参阅《Oracle VM Server for SPARC 2.0 Release Notes》中的“Required Software and Patches”。有关升级 Oracle Solaris OS 的完整说明，请参阅 Oracle Solaris 10 安装指南。

当在控制域中重新安装 Oracle Solaris OS 时，需要保存和恢复 Logical Domains 自动保存配置数据和约束数据库文件，如本节中所述。

保存和恢复自动保存配置目录

从 Logical Domains 1.2 发行版开始，可在控制域上重新安装操作系统之前保存和恢复自动保存配置目录。无论何时在控制域上重新安装操作系统，都必须保存和恢复 Logical Domains 自动保存配置数据，该数据可在 `/var/opt/SUNWldm/autosave-`*autosave-name* 目录中找到。

可使用 `tar` 或 `cpio` 命令保存和恢复整个目录内容。

注 – 每个自动保存目录均包含相关配置的上次 SP 配置更新的时间戳。如果恢复自动保存文件，则时间戳可能无法保持同步。在这种情况下，已恢复的自动保存配置以其上一状态显示，即 `[newer]` 或最新。

有关自动保存配置的更多信息，请参见第 165 页中的 “管理 Logical Domains 配置”。

▼ 保存和恢复自动保存目录

以下过程说明如何保存和恢复自动保存目录。

1 保存自动保存目录。

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

2 **（可选）删除现有的自动保存目录，以确保干净的恢复操作。**

有时，自动保存目录可能包含多余的文件（可能是以前配置中遗留下来的），这些文件可能会破坏已下载到 SP 的配置。这种情况下，在执行恢复操作之前请先清理自动保存目录，如以下示例所示：

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

3 **恢复自动保存目录。**

以下命令可恢复 `/var/opt/SUNWldm` 目录中的文件和目录。

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

保存和恢复 Logical Domains 约束数据库文件

无论何时对控制域升级操作系统，都必须保存和恢复 Logical Domains 约束数据库文件，该文件位于 `/var/opt/SUNWldm/ldom-db.xml` 中。

注 – 另外，在执行会破坏控制域的文件数据（如磁盘交换区）的任何其他操作时，也必须保存和恢复 `/var/opt/SUNWldm/ldom-db.xml` 文件。

使用即时升级时保留 Logical Domains 约束数据库文件

如果要对控制域使用即时升级，请考虑向 `/etc/lu/synclist` 文件添加以下行：

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

这样，在切换引导环境时即可将数据库从活动引导环境自动复制到新的引导环境。有关 `/etc/lu/synclist` 文件和在引导环境之间同步文件的更多信息，请参阅《Oracle Solaris 10 9/10 安装指南：Solaris Live 升级和升级规划》中的“在引导环境之间同步文件”。

从早于 Oracle Solaris 10 5/08 OS 的 Oracle Solaris 10 OS 升级

如果控制域从版本早于 Oracle Solaris 10 5/08 OS（或没有修补程序 127127-11）的 Oracle Solaris 10 OS 升级，并且将卷管理器卷导出为虚拟磁盘，则在升级 Logical Domains Manager 后，必须使用 `options=slice` 重新导出虚拟磁盘后端。有关更多信息，请参见第 83 页中的 “导出卷以及向后兼容性”。

升级 Logical Domains Manager 和系统固件

本节说明如何升级至 Oracle VM Server for SPARC 2.0 软件。

首先将 Logical Domains Manager 下载到控制域。请参见第 26 页中的 “下载 Logical Domains Manager”。

然后停止平台上运行的所有域（控制域除外）：

▼ 停止平台上运行的所有域，控制域除外

1. **使每个域进入 ok 提示符。**
2. **使用 -a 选项停止所有域。**

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

3. **从控制域对每个域发出 unbind-domain 子命令。**

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

升级至 Oracle VM Server for SPARC 2.0 软件

本节说明如何升级至 Oracle VM Server for SPARC 2.0 软件。

如果要将现有 Logical Domains 1.0 配置用于 Oracle VM Server for SPARC 2.0 软件，请执行《Oracle VM Server for SPARC 2.0 Release Notes》中的“Upgrade From Logical Domains 1.0 Software Only”过程。现有的 Logical Domains 1.0 配置对 Oracle VM Server for SPARC 2.0 软件**无效**。

如果是从较新版本的 Oracle VM Server for SPARC 软件升级，请执行第 32 页中的 “升级至 Oracle VM Server for SPARC 2.0 软件”过程。此类现有 Logical Domains 配置对 Oracle VM Server for SPARC 2.0 软件**有效**。

▼ 升级至 Oracle VM Server for SPARC 2.0 软件

1. **执行系统固件的闪存升级。**
   有关完整过程，请参见第 24 页中的 “升级系统固件”。
2. **禁用 Logical Domains Manager 守护进程 (ldmd)。**

```
# svcadm disable ldmd
```

3. **删除旧的 SUNWldm 软件包。**

```
# pkgrm SUNWldm
```

4. **添加新的 SUNWldm 软件包。**
   指定 -d 选项假定软件包位于当前目录中。

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

5 **使用 `ldm list` 命令检验 Logical Domains Manager 是否正在运行。**

`ldm list` 命令应列出当前在系统上定义的所有域。特别是，`primary` 域应被列出并处于 `active` 状态。以下示例输出显示在系统上仅定义了 `primary` 域。

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

出厂默认配置和禁用 Logical Domains

平台显示为仅管理一个操作系统的单个系统的初始配置被称为出厂默认配置。如果希望禁用逻辑域，则您可能也希望恢复此配置，以便系统能够重新获得可能已分配到其他域的所有资源（CPU、内存和 I/O）的访问权限。

本节说明如何删除所有来宾域、删除所有 Logical Domains 配置以及如何恢复出厂默认配置。

▼ 删除所有来宾域

1 **使用 `-a` 选项停止所有域。**

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

2 **取消绑定除 `primary` 域之外的所有域。**

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

注 – 如果某 I/O 域正提供控制域所需的服务，您可能无法取消绑定此 I/O 域。这种情况下，请跳过此步骤。

---

3 **销毁除 `primary` 域之外的所有域。**

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

▼ 删除所有逻辑域配置

1 **列出存储在服务处理器 (Service Processor, SP) 上的所有逻辑域配置。**

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

2 **删除除 `factory-default` 配置之外的以前保存到 SP 的所有配置 (*config-name*)。**

对每个此类配置使用以下命令：

```
primary# ldm rm-config config-name
```

删除以前保存到 SP 的所有配置后，当重新引导控制域 (`primary`) 时，`factory-default` 域是下一个要使用的域。

▼ 恢复出厂默认配置

1 **选择出厂默认配置。**

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

2 **停止控制域。**

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

3 **对系统执行关开机循环，以加载出厂默认配置。**

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

▼ 禁用 Logical Domains Manager

- **从控制域禁用 Logical Domains Manager。**

```
primary# svcadm disable ldmd
```

注 – 禁用 Logical Domains Manager 不会停止任何运行的域，但可以禁止创建新的域、更改现有域的配置以及监视域的状态。

注意 – 禁用 Logical Domains Manager，也会禁用一些服务，例如错误报告或电源管理。对于错误报告，如果您使用的是 `factory-default` 配置，则可重新引导控制域以恢复错误报告。不过，电源管理的情况不同。此外，某些系统管理或监视工具依赖于 Logical Domains Manager。

▼ 删除 Logical Domains Manager

恢复出厂默认配置并禁用 Logical Domains Manager 后，可删除 Logical Domains Manager 软件。

- **删除 Logical Domains Manager 软件。**

```
primary# pkgrm SUNWldm SUNWldmp2v
```

注 – 如果在恢复出厂默认配置之前删除 Logical Domains Manager，可从服务处理器恢复出厂默认配置，如以下过程所示。

▼ 从服务处理器恢复出厂默认配置

如果在恢复出厂默认配置之前删除 Logical Domains Manager，可从服务处理器恢复出厂默认配置。

1 **从服务处理器恢复出厂默认配置。**

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

2 **对系统执行关开机循环，以加载出厂默认配置。**

```
-> reset /SYS
```

◆ ◆ ◆ 第 3 章

3 安全

本章介绍可以在 Logical Domains 系统上启用的一些安全功能。

本章包括以下内容：

- 第 37 页中的 “Logical Domains Manager 授权”
- 第 38 页中的 “创建授权和配置文件并为用户帐户分配角色”
- 第 41 页中的 “配置 RBAC 以进行来宾控制台访问”
- 第 41 页中的 “启用并使用 BSM 审计”

Logical Domains Manager 授权

Logical Domains Manager 的授权有两个级别：

- 读－允许查看配置，但不能修改配置。
- 读写－允许查看和更改配置。

安装 Logical Domains Manager 后，这些更改不会应用到 Oracle Solaris OS 中，而是由软件包脚本 `postinstall` 将其添加到授权文件中。同样，授权条目也是由软件包脚本 `preremove` 来删除。

下表列出了 `ldm` 子命令以及执行这些命令所需的相应用户授权。

表 3–1 `ldm` 子命令和用户授权

ldm 子命令[1]	用户授权
`add-*`	`solaris.ldoms.write`
`bind-domain`	`solaris.ldoms.write`
`list`	`solaris.ldoms.read`
`list-*`	`solaris.ldoms.read`

[1] 涉及您可以添加、列出、删除或设置的所有资源。

表 3–1 ldm 子命令和用户授权 （续）

ldm 子命令[1]	用户授权
panic-domain	solaris.ldoms.write
remove-*	solaris.ldoms.write
set-*	solaris.ldoms.write
start-domain	solaris.ldoms.write
stop-domain	solaris.ldoms.write
unbind-domain	solaris.ldoms.write

[1] 涉及您可以添加、列出、删除或设置的所有资源。

创建授权和配置文件并为用户帐户分配角色

需要使用适用于 Logical Domains Manager 的 Oracle Solaris OS 基于角色的访问控制 (Role-Based Access Control, RBAC)，来设置授权和配置文件并为用户帐户分配角色。有关 RBAC 的更多信息，请参阅 Oracle Solaris 10 System Administrator Collection - Simplified Chinese (http://docs.sun.com/app/docs/coll/47.16)。

Logical Domains Manager 的授权有两个级别：

- 读－允许查看配置，但不能修改配置。
- 读写－允许查看和更改配置。

以下是自动添加到 Oracle Solaris OS /etc/security/auth_attr 文件中的 Logical Domains 条目：

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

管理用户授权

▼ 为用户添加授权

根据需要执行下列步骤，在 /etc/security/auth_attr 文件中为 Logical Domains Manager 用户添加授权。由于超级用户已经具有 solaris.* 授权，因此超级用户已经具有 solaris.ldoms.* 授权的相应权限。

1 为每个需要授权以使用 ldm(1M) 子命令的用户创建一个本地用户帐户。

注 – 要为用户添加 Logical Domains Manager 授权，必须为该用户创建一个本地（非 LDAP）帐户。有关详细信息，请参阅 Oracle Solaris 10 System Administrator Collection - Simplified Chinese (`http://docs.sun.com/app/docs/coll/47.16`)。

2 **根据希望用户能够访问的 `ldm`(1M) 子命令，执行以下操作之一。**

有关 `ldm`(1M) 命令及其用户授权的列表，请参见表 3–1。

- 使用 `usermod`(1M) 命令为用户添加只读授权。

  `# usermod -A solaris.ldoms.read` *username*
- 使用 `usermod`(1M) 命令为用户添加读写授权。

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

▼ 为用户删除所有授权

- **为本地用户帐户删除所有授权（唯一可能的选择）。**

  `# usermod -A ''` *username*

管理用户配置文件

`SUNWldm` 软件包会在 `/etc/security/prof_attr` 文件中添加两个系统定义的 RBAC 配置文件，以便授权非超级用户访问 Logical Domains Manager。这两个 Logical Domains 特定的配置文件为：

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

`SUNWldm` 软件包还定义了与 LDoms Management 配置文件相关联的以下执行属性：

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

可以使用以下过程将上述其中一个配置文件分配给某个用户帐户。

▼ 为用户添加配置文件

直接分配了 LDoms Management 配置文件的用户必须调用配置文件 shell 以运行具有安全属性的 `ldm` 命令。有关更多信息，请参见 Oracle Solaris 10 System Administrator Collection - Simplified Chinese。

- **为本地用户帐户添加管理配置文件，例如 LDoms Management。**

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

▼ 为用户删除所有配置文件

- **为本地用户帐户删除所有配置文件（唯一可能的选择）。**

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

给用户分配角色

使用此过程的优势是，只有被分配了特定角色的用户才能承担该角色。在承担角色的过程中，如果该角色指定有一个密码，则需要输入该密码。这样便实现了双层安全性。如果没有为用户分配某个角色，则此用户不能承担该角色（通过执行 su *role-name* 命令），即使此用户有正确的密码也是如此。

▼ 创建角色并将该角色分配给用户

1. **创建角色。**

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

2. **为该角色指定密码。**

   ```
   # passwd ldm_read
   ```

3. **将该角色分配给用户。**

   例如，user_1。

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

4. **为用户 (user_1) 指定密码。**

   ```
   # passwd user_1
   ```

5. **仅向 user_1 帐户分配访问权限，使其成为 ldm_read 帐户。**

   ```
   # su user_1
   ```

6. **出现提示时，键入用户密码。**

7. **检验用户 ID 和对 ldm_read 角色的访问权限。**

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

8. **向用户提供对具有读授权的 ldm 子命令的访问权限。**

   ```
   # su ldm_read
   ```

9. **出现提示时，键入用户密码。**

10 **键入 `id` 命令以显示该用户。**

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

配置 RBAC 以进行来宾控制台访问

`vntsd` 守护进程提供了名为 `vntsd/authorization` 的 SMF 属性。可以配置此属性以对域控制台或控制台组启用用户和角色的授权检查。要启用授权检查，请使用 `svccfg` 命令将此属性的值设置为 `true`。当此选项处于启用状态时，`vntsd` 将仅侦听并接受 `localhost` 上的连接。如果启用 `vntsd/authorization` 时 `listen_addr` 属性指定备用 IP 地址，`vntsd` 将忽略该备用 IP 地址并继续仅在 `localhost` 上侦听。

默认情况下，在启用 `vntsd` 服务时，将向 `auth_attr` 数据库添加用于访问所有来宾控制台的授权。

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

超级用户可以使用 `usermod` 命令将所需授权分配给其他用户或角色。这样就可以仅允许具有所需授权的用户或角色访问给定的域控制台或控制台组。

以下示例授予用户 `terry` 访问所有域控制台的授权：

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

以下示例为名为 `ldg1` 的特定域控制台添加一项新授权，并将该授权分配给用户 `sam`：

1. 将新授权条目添加到域 `ldg1` 的 `auth_attr` 文件。

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

2. 将此授权分配给用户 `sam`：

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

有关授权和 RBAC 的更多信息，请参见《系统管理指南：安全性服务》。

启用并使用 BSM 审计

Logical Domains Manager 使用 Oracle Solaris OS 基本安全模块 (Basic Security Module, BSM) 审计功能。BSM 审计功能提供了检查控制域中操作和事件历史记录以确定发生的情况的方法。这些历史记录保存在日志中，其中包括执行的操作、完成时间、执行者以及产生的后果。

要启用和禁用此审计功能，请使用 Oracle Solaris OS `bsmconv(1M)` 和 `bsmunconv(1M)` 命令。本节还包括说明如何检验审计功能、显示审计输出以及轮转审计日志的任务。有关 BSM 审计的详细信息，请参阅 Solaris 10《系统管理指南：安全性服务》。

▼ 启用 BSM 审计

1. 在 `/etc/security/audit_control` 文件的 `flags:` 行中添加 `vs`。
2. 运行 `bsmconv(1M)` 命令。

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

   有关此命令的更多信息，请参见 bsmconv(1M) 手册页。
3. 重新引导 Oracle Solaris OS 以使审计生效。

▼ 检验 BSM 审计是否已启用

1. 键入以下命令。

   ```
   # auditconfig -getcond
   ```
2. 检查输出中是否显示 `audit condition = auditing`。

▼ 禁用 BSM 审计

1. 运行 `bsmunconv` 命令以禁用 BSM 审计。

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

   有关此命令的更多信息，请参见 bsmunconv(1M) 手册页。
2. 重新引导 Oracle Solaris OS，以使禁用审计生效。

▼ 显示审计输出

- 使用以下方法之一显示 BSM 审计输出：
  - 使用 auditreduce(1M) 和 praudit(1M) 命令显示审计输出。

    ```
    # auditreduce -c vs | praudit
    # auditreduce -c vs -a 20060502000000 | praudit
    ```
  - 使用 `praudit -x` 命令显示 XML 输出。

▼ 轮转审计日志

- 使用 `audit -n` 命令轮转审计日志。

◆ ◆ ◆ 第 4 章

4 设置服务和控制域

本章介绍了设置默认服务和控制域所需的过程。

您也可以使用 Oracle VM Server for SPARC Configuration Assistant 配置逻辑域和服务。请参见附录 B，Oracle VM Server for SPARC Configuration Assistant。

本章包括以下内容：

- 第 43 页中的 “输出消息”
- 第 44 页中的 “创建默认服务”
- 第 45 页中的 “控制域的初始配置”
- 第 46 页中的 “重新引导以使用 Logical Domains”
- 第 46 页中的 “启用控制/服务域与其他域之间的联网”
- 第 47 页中的 “启用虚拟网络终端服务器守护进程”

输出消息

从 Oracle VM Server for SPARC 2.0 发行版开始，如果无法在控制域上动态配置资源，则最佳做法是先启动延迟重新配置。延迟重新配置会将配置活动推迟到控制域进行重新引导后。

在 `primary` 域上启动延迟重新配置时，您将收到以下消息：

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

在重新引导之前每次对 `primary` 域执行后续操作之后，将收到以下通知：

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

创建默认服务

必须创建下列虚拟设备服务，以便将控制域作为服务域以及为其他域创建虚拟设备：

- vcc－虚拟控制台集中器服务
- vds－虚拟磁盘服务器
- vsw－虚拟交换机服务

▼ 创建默认服务

1 **创建虚拟控制台集中器 (vcc) 服务，以供虚拟网络终端服务器守护进程 (vntsd) 使用，并将该服务器作为所有逻辑域控制台的集中器。**

例如，以下命令会将端口范围从 5000 到 5100 的虚拟控制台集中器服务 (primary-vcc0) 添加到控制域 (primary)。

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

2 **创建虚拟磁盘服务器 (vds)，以允许将虚拟磁盘导入到逻辑域中。**

例如，以下命令会将虚拟磁盘服务器 (primary-vds0) 添加到控制域 (primary)。

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

3 **创建虚拟交换机服务 (vsw)，以启用逻辑域中的虚拟网络 (vnet) 设备之间的联网。**

如果每个逻辑域都需要通过虚拟交换机与外界进行通信，请将 GLDv3 兼容的网络适配器分配给虚拟交换机。

例如，以下命令会将网络适配器驱动程序 nxge0 上的虚拟交换机服务 (primary-vsw0) 添加到控制域 (primary)。

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

此命令会自动将 MAC 地址分配给虚拟交换机。您可以指定将 MAC 地址作为 ldm add-vsw 命令的选项。但在那种情况下，您必须负责保证所指定的 MAC 地址不会与已经存在的 MAC 地址相冲突。

如果添加的虚拟交换机将取代底层物理适配器作为主网络接口，则必须为其分配物理适配器的 MAC 地址，以便动态主机配置协议 (Dynamic Host Configuration Protocol, DHCP) 服务器会为该域分配相同的 IP 地址。请参见第 46 页中的 “启用控制/服务域与其他域之间的联网”。

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

4 **检验是否已使用 list-services 子命令创建服务。**

输出应该与以下内容类似。

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

```
    primary-vds0

VCC
    NAME             PORT-RANGE
    primary-vcc0     5000-5100

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

控制域的初始配置

最初，所有系统资源都分配给控制域。要允许创建其他逻辑域，您必须释放其中一些资源。

不要尝试使用内存动态重新配置 (Dynamic Reconfiguration, DR) 执行控制域的初始配置。尽管无需重新引导就能够使用内存 DR 执行此配置，但是建议您**不要**这样做。内存 DR 方法可能会需要很长时间（比重新引导的时间更长），甚至可能会失败。更改内存配置前，应该使用 `ldm start-reconf` 命令将控制域置于延迟重新配置模式中。然后，您可以在完成所有配置步骤后重新引导控制域。

▼ 设置控制域

注 – 以下过程举例说明了为控制域设置的资源。这些数字仅为示例，并且所使用的值可能不适用于您的控制域。

---

1 确定控制域中是否有加密设备。

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

2 为控制域分配加密资源。

以下示例会为控制域 `primary` 分配加密资源。这会令其余的加密资源可供来宾域使用。

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

3 为控制域分配虚拟 CPU。

例如，以下命令会为控制域 `primary` 分配 8 个虚拟 CPU。这会令其余的虚拟 CPU 可供来宾域使用。

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

4 在控制域上启动延迟重新配置。

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

5 **为控制域分配内存。**

例如，以下命令会为控制域 `primary` 分配 4 千兆字节的内存。这会令其余的内存可供来宾域使用。

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

6 **为服务处理器 (service processor, SP) 添加逻辑域机器配置。**

例如，以下命令会添加名为 `initial` 的配置。

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

7 **检验该配置在下次重新引导时是否可以使用。**

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

此 list 子命令显示了执行开关机循环后将使用的 `initial` 配置集。

重新引导以使用 Logical Domains

必须重新引导控制域才能使配置更改生效，并释放资源供其他逻辑域使用。

▼ 重新引导

- **关闭并重新引导控制域。**

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

注 – 重新引导或开关机循环都能够实例化新配置。实际上，只有开关机循环才能引导保存到服务处理器 (service processor, SP) 的配置，该配置将反映在 `list-config` 输出中。

启用控制/服务域与其他域之间的联网

默认情况下，会禁用系统中控制域与其他域之间的联网。要启用此功能，应将虚拟交换机设备配置为网络设备。虚拟交换机可以取代底层物理设备（在此示例中为 `nxge0`）作为主接口，或者配置为域中的附加网络接口。

注 – 请从控制域的控制台执行下列配置步骤，因为该过程可能会暂时中断与域的网络连接。

▼ 将虚拟交换机配置为主接口

1. 显示所有接口的寻址信息。

```
primary# ifconfig -a
```

2. 激活 (plumb) 虚拟交换机。在此示例中，vsw0 是要配置的虚拟交换机。

```
primary# ifconfig vsw0 plumb
```

3. （可选）要获得域中所有虚拟交换机实例的列表，您可以列出它们。

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

4. 取消激活 (Unplumb) 分配给虚拟交换机 (net-dev) 的物理网络设备，在此示例中为 nxge0。

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

5. 要将物理网络设备 (nxge0) 的属性迁移到虚拟交换机 (vsw0) 设备，请执行以下操作之一：

   - 如果使用静态 IP 地址配置网络，请对 vsw0 重用 nxge0 的 IP 地址和网络掩码。

     ```
     primary# ifconfig vsw0 IP_of_nxge0 netmask netmask_of_nxge0 broadcast + up
     ```

   - 如果使用 DHCP 配置网络，请为 vsw0 启用 DHCP。

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

6. 进行必需的配置文件修改，使其成为永久性更改。

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

注 – 如有必要，也可以配置虚拟交换机和物理网络设备。这种情况下，需要像步骤 2 中一样激活 (plumb) 虚拟交换机，但不需要取消激活 (unplumb) 物理设备（跳过步骤 4）。然后，必须使用静态 IP 地址或动态 IP 地址配置虚拟交换机。可以从 DHCP 服务器获取动态 IP 地址。有关其他信息和这种情况的示例，请参见第 108 页中的 “针对 NAT 和路由配置虚拟交换机和服务域”。

启用虚拟网络终端服务器守护进程

必须启用虚拟网络终端服务器守护进程 (vntsd)，以提供对每个逻辑域的虚拟控制台的访问。有关如何使用此守护进程的信息，请参阅 vntsd(1M) 手册页。

▼ 启用虚拟网络终端服务器守护进程

注 – 请确保在启用 vntsd 之前已在控制域上创建默认服务 vconscon (vcc)。有关更多信息，请参见第 44 页中的 “创建默认服务”。

1 **使用 svcadm(1M) 命令启用虚拟网络终端服务器守护进程 vntsd(1M)。**

```
primary# svcadm enable vntsd
```

2 **使用 svcs(1) 命令检验是否已启用 vntsd 守护进程。**

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

◆ ◆ ◆ 第 5 章

5 设置来宾域

本章介绍了设置来宾域所必需的过程。

您也可以使用 Oracle VM Server for SPARC Configuration Assistant 配置逻辑域和服务。请参见附录 B，Oracle VM Server for SPARC Configuration Assistant。

本章包括以下内容：

- 第 49 页中的 “创建和启动来宾域”
- 第 52 页中的 “在来宾域上安装 Oracle Solaris OS”

创建和启动来宾域

来宾域必须运行既可以识别 sun4v 平台又可以识别由虚拟机管理程序提供的虚拟设备的操作系统。目前，这就意味着必须至少运行 Oracle Solaris 10 11/06 OS。通过运行 Oracle Solaris 10 9/10 OS，您可获得所有的 Oracle VM Server for SPARC 2.0 功能。有关任何可能需要的特定修补程序的信息，请参见《Oracle VM Server for SPARC 2.0 Release Notes》。一旦创建了默认服务并重新分配了控制域的资源，您就可以创建和启动来宾域。

▼ 创建和启动来宾域

1 **创建逻辑域。**

例如，以下命令可以创建名为 `ldg1` 的来宾域。

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

2 **为来宾域添加 CPU。**

例如，以下命令可以为来宾域 `ldg1` 添加八个虚拟 CPU。

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

3 **为来宾域添加内存。**

例如，以下命令可为来宾域 `ldg1` 添加 2 GB 的内存。

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

4 **为来宾域添加虚拟网络设备。**

例如，以下命令为来宾域 `ldg1` 添加具有下列特定信息的虚拟网络设备。

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

其中：

- `vnet1` 是逻辑域的唯一接口名称，它被分配到此虚拟网络设备实例，供在后续 `set-vnet` 或 `remove-vnet` 子命令上引用。
- `primary-vsw0` 是要连接到的现有网络服务（虚拟交换机）的名称。

注 – 步骤 5 和 6 是简化后的说明，主要说明如何将虚拟磁盘服务器设备 `vdsdev` 添加到主域以及如何将虚拟磁盘 `vdisk` 添加到来宾域。要了解如何将 ZFS 卷和文件系统用作虚拟磁盘，请参见第 82 页中的 “将 ZFS 卷作为具有单个分片的磁盘导出”和第 90 页中的 “将 ZFS 用于虚拟磁盘”。

---

5 **指定要由虚拟磁盘服务器导出并用作来宾域的虚拟磁盘的设备。**

可以将物理磁盘、磁盘分片、卷或文件导出为块设备。以下示例对物理磁盘和文件进行了说明。

- **物理磁盘示例。**第一个示例中添加了具有以下特定信息的物理磁盘。

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

  其中：

  - `/dev/dsk/c2t1d0s2` 是实际物理设备的路径名称。添加设备时，路径名必须与设备名称成对出现。
  - `vol1` 是必须为添加到虚拟磁盘服务器的设备指定的唯一名称。该卷名称对于此虚拟磁盘服务器实例必须是唯一的，因为该卷名称由此虚拟磁盘服务器导出到客户机以便进行添加。添加设备时，卷名称必须与实际设备的路径名成对出现。
  - `primary-vds0` 是要将此设备添加到的虚拟磁盘服务器的名称。
- **文件示例。**此第二个示例中正在将文件导出为块设备。

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

  其中：

  - *backend* 是导出为块设备的实际文件的路径名称。添加设备时，后端必须与设备名称成对出现。
  - `vol1` 是必须为添加到虚拟磁盘服务器的设备指定的唯一名称。该卷名称对于此虚拟磁盘服务器实例必须是唯一的，因为该卷名称由此虚拟磁盘服务器导出到客户机以便进行添加。添加设备时，卷名称必须与实际设备的路径名成对出现。

- `primary-vds0` 是要将此设备添加到的虚拟磁盘服务器的名称。

6 为来宾域添加虚拟磁盘。

以下示例会为来宾域 `ldg1` 添加虚拟磁盘。

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

其中：

- `vdisk1` 是虚拟磁盘的名称。
- `vol1` 是要连接到的现有卷的名称。
- `primary-vds0` 是要连接到的现有虚拟磁盘服务器的名称。

注 – 虚拟磁盘是与各种类型的物理设备、卷或文件相关联的通用块设备。虚拟磁盘与 SCSI 磁盘不同义，因此磁盘标号中不包含目标 ID。逻辑域中的虚拟磁盘的格式如下：c*N*d*N*s*N*，其中 c*N* 是虚拟控制器，d*N* 是虚拟磁盘号，s*N* 是磁盘分片。

---

7 为来宾域设置 `auto-boot?` 和 `boot-device` 变量。

第一个示例命令将来宾域 `ldg1` 的 `auto-boot?` 变量设置为 `true`。

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

第二个示例命令将来宾域 `ldg1` 的 `boot-device` 变量设置为 `vdisk`。

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

8 将资源绑定到来宾域 `ldg1`，然后列出该域以检验它是否已绑定。

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

9 要查找来宾域的控制台端口，可以查看上述 `list-domain` 子命令的输出。

在标题 `CONS` 下可以看见逻辑域来宾 1 (`ldg1`) 已将其控制台输出绑定到端口 `5000`。

10 通过登录到控制域并直接连接到本地主机的控制台端口，可从其他终端连接到来宾域的控制台。

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

11 启动来宾域 `ldg1`。

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

在来宾域上安装 Oracle Solaris OS

本节说明了在来宾域上安装 Oracle Solaris OS 的一些不同方法。

注意 – 在 Oracle Solaris OS 安装期间，**不要**与虚拟控制台断开连接。

▼ 在来宾域上使用 DVD 安装 Oracle Solaris OS

1. **在 DVD 驱动器中插入 Oracle Solaris 10 OS DVD。**

2. **停止 `primary` 域上的卷管理守护进程 `vold(1M)`。**

```
primary# svcadm disable volfs
```

3. **停止并取消绑定来宾域 (`ldg1`)。**

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

4. **将 DVD 与 DVD-ROM 介质添加为辅助卷和虚拟磁盘。**

以下命令将 `c0t0d0s2` 用作驻留有 Oracle Solaris 介质的 DVD 驱动器，将 `dvd_vol@primary-vds0` 用作辅助卷，将 `vdisk_cd_media` 用作虚拟磁盘。

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

5. **检查以确定 DVD 是否已添加为辅助卷和虚拟磁盘。**

```
primary# ldm list-bindings
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active   -n-cv   SP      4     4G       0.2%  22h 45m
...
VDS
   NAME             VOLUME         OPTIONS          DEVICE
   primary-vds0     vol1                            /dev/dsk/c2t1d0s2
   dvd_vol                                          /dev/dsk/c0t0d0s2
....
------------------------------------------------------------------------------
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg1             inactive -----           60    6G
...
DISK
   NAME             VOLUME                      TOUT DEVICE  SERVER
   vdisk1           vol1@primary-vds0
   vdisk_cd_media   dvd_vol@primary-vds0
....
```

6. **绑定并启动来宾域 (`ldg1`)。**

```
primary# ldm bind ldg1
primary# ldm start ldg1
```

```
LDom ldg1 started
primary# telnet localhost 5000
Trying 027.0.0.1...
Connected to localhost.
Escape character is '^]'.

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

7 在客户机 OpenBoot PROM 中显示设备别名。

在此示例中，可看到 vdisk_cd_media （即 Oracle Solaris DVD）的设备别名和 vdisk1 （即可安装 Oracle Solaris OS 的虚拟磁盘）的设备别名。

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

8 在来宾域的控制台上，将从分片 f 上的 vdisk_cd_media (disk@1) 进行引导。

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

9 继续执行 Oracle Solaris OS 安装菜单的剩余步骤。

▼ 从 Oracle Solaris ISO 文件的来宾域上安装 Oracle Solaris OS

1 停止并取消绑定来宾域 (ldg1)。

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

2 将 Oracle Solaris ISO 文件添加为辅助卷和虚拟磁盘。

以下命令将 solarisdvd.iso 用作 Oracle Solaris ISO 文件，将 iso_vol@primary-vds0 用作辅助卷，将 vdisk_iso 用作虚拟磁盘。

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

3 检查以确定 Oracle Solaris ISO 文件是否已添加为辅助卷和虚拟磁盘。

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

```
   primary-vds0     vol1                           /dev/dsk/c2t1d0s2
   iso_vol                                         /export/solarisdvd.iso
....
------------------------------------------------------------------------------
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg1             inactive -----           60    6G
...
DISK
   NAME             VOLUME                      TOUT DEVICE  SERVER
   vdisk1           vol1@primary-vds0
   vdisk_iso        iso_vol@primary-vds0
....
```

4 绑定并启动来宾域 (`ldg1`)。

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

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

5 在客户机 OpenBoot PROM 中显示设备别名。

在此示例中，可看到 `vdisk_iso`（即 Oracle Solaris ISO 映像）的设备别名和 `vdisk_install`（即磁盘空间）的设备别名。

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

6 在来宾域的控制台上，将从分片 `f` 上的 `vdisk_iso` (`disk@1`) 进行引导。

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

7 继续执行 Oracle Solaris OS 安装菜单的剩余步骤。

▼ 在来宾域上执行 JumpStart 操作

此过程介绍了如何在来宾域上执行 JumpStart 操作。此过程遵循常规 JumpStart 过程，但它介绍了不同的磁盘设备名称格式，以用于来宾域的 JumpStart 配置文件中。请参见《Oracle Solaris 10 9/10 安装指南：自定义 JumpStart 和高级安装》。

逻辑域中的虚拟磁盘设备名称不同于物理磁盘设备名称。因为虚拟磁盘设备名称不包含目标 ID (t*N*)。

相对于常用 c*N*t*N*d*N*s*N*，虚拟磁盘设备名称改用 c*N*d*N*s*N* 格式。其中 c*N* 是虚拟控制器，d*N* 是虚拟磁盘号，而 s*N* 是分片号码。

- **修改 JumpStart 配置文件，使其能够反映此更改。**

  虚拟磁盘可显示为完整磁盘或具有单个分片的磁盘。通过使用指定多个分区的常规 JumpStart 配置文件，可在完整磁盘上安装 Oracle Solaris OS。具有单个分片的磁盘仅有一个分区 `s0`，使用整个磁盘。要在单个磁盘上安装 Oracle Solaris OS，必须使用拥有单个分区 (`/`) 的配置文件，此分区使用整个磁盘。无法定义任何其他分区，例如交换区。有关完整磁盘和具有单个分片的磁盘的更多信息，请参见第 77 页中的 “虚拟磁盘外观”。

  - **用于安装 UFS 根文件系统的 JumpStart 配置文件。**

    请参见《Oracle Solaris 10 9/10 安装指南：自定义 JumpStart 和高级安装》。

    普通 UFS 配置文件

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

    用于在完整磁盘上安装域的实际 UFS 配置文件

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

    用于在具有单个分片的磁盘上安装域的实际 UFS 配置文件

    ```
    filesys c0d0s0 free /
    ```

  - **用于安装 ZFS 根文件系统的 JumpStart 配置文件。**

    请参见《Oracle Solaris 10 9/10 安装指南：自定义 JumpStart 和高级安装》中的第 9 章 “使用 JumpStart 安装 ZFS 根池”。

    普通 ZFS 配置文件

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

    用于安装域的实际 ZFS 配置文件

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

◆ ◆ ◆ 第 6 章

6 设置 I/O 域

本章介绍 I/O 域以及如何在 Logical Domains 环境中配置 I/O 域。

本章包括以下内容：

- 第 57 页中的 “I/O 域概述”
- 第 58 页中的 “分配 PCIe 总线”
- 第 62 页中的 “分配 PCIe 端点设备”

I/O 域概述

I/O 域对物理 I/O 设备具有直接拥有权和直接访问权。可通过将 PCI EXPRESS (PCIe) 总线或 PCIe 端点设备分配到域来创建 I/O 域。使用 `ldm add-io` 命令将总线或设备分配到域。

由于以下原因，可能需要配置 I/O 域：

- I/O 域对物理 I/O 设备具有直接访问权，这就避免了虚拟 I/O 所带来的性能开销。因此，I/O 域上的 I/O 性能更接近于裸机系统上的 I/O 性能。
- I/O 域可托管要由其他来宾域使用的虚拟 I/O 服务。

有关配置 I/O 域的信息，请参见以下内容：

- 第 58 页中的 “分配 PCIe 总线”
- 第 62 页中的 “分配 PCIe 端点设备”

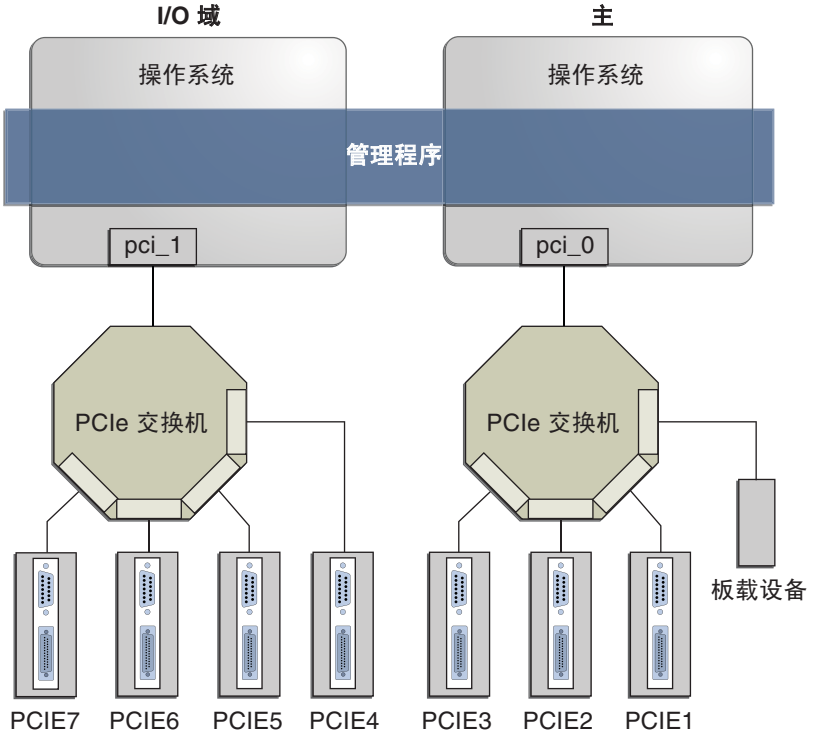
注 – 无法迁移配置有 PCIe 端点设备的 I/O 域。有关其他迁移限制的信息，请参见第 9 章，迁移域。

分配 PCIe 总线

可以使用 Oracle VM Server for SPARC 软件将完整的 PCIe 总线（也称为**根联合体**）分配到域。完整 PCIe 总线由 PCIe 总线本身及其所有 PCI 交换机和设备组成。服务器上的 PCIe 总线使用名称标识，例如 `pci@400` (`pci_0`)。配置有完整 PCIe 总线的 I/O 域也称为**根域**。

下图显示了具有两个 PCIe 总线 (`pci_0` and `pci_1`) 的系统。每个总线分配到不同的域。这样，系统就配置有两个 I/O 域。

图 6–1 将 PCIe 总线分配到 I/O 域



使用 PCIe 总线可以创建的最大 I/O 域数取决于服务器上可用的 PCIe 总线数。例如，如果使用 Sun SPARC Enterprise T5440 服务器，您最多可以有四个 I/O 域。

注 – 一些 Sun UltraSPARC 服务器只有一个 PCIe 总线。这种情况下，您可以通过将 PCIe 端点（或可分配的直接 I/O）设备分配到域来创建 I/O 域。请参见第 62 页中的 “分配 PCIe 端点设备”。如果系统具有网络接口单元 (Network Interface Unit, NIU)，也可以通过将 NIU 分配到域来创建 I/O 域。

将 PCIe 总线分配到 I/O 域时，该总线上的所有设备都归该 I/O 域所有。不允许将该总线上的任何 PCIe 端点设备分配到其他域。只可以将 PCIe 上已分配到 `primary` 域的 PCIe 端点设备分配到其他域。

在 Logical Domains 环境中最初配置服务器或使用 `factory-default` 配置时，`primary` 域可访问所有物理设备资源。这意味着 `primary` 域是系统上配置的唯一 I/O 域，所有 PCIe 总线都归其所有。

▼ 通过分配 PCIe 总线创建 I/O 域

此示例过程演示如何从初始配置创建 I/O 域，其中，几个总线归 `primary` 域所有。默认情况下，系统上的所有总线都归 `primary` 域所有。此示例适用于 Sun SPARC Enterprise T5440 服务器。在其他服务器上也可以使用此过程。虽然面向其他服务器的说明可能与这些说明稍有不同，但是您可以通过此处的示例了解基本原则。

首先，必须保留具有 `primary` 域的引导磁盘的总线。然后，将另一个总线从 `primary` 域删除并将其分配到另一个域。

注意 – 在支持的服务器上，所有内部磁盘都连接到一个 PCIe 总线。如果域从内部磁盘进行引导，请不要将该总线从域中删除。此外，请确保没有删除具有由域使用的设备（例如网络接口）的总线。如果错误地删除了总线，则域可能将无法访问所需的设备并变为不可用。要删除具有由域使用的设备的总线，请重新配置该域，以使用其他总线的设备。例如，可能需要重新配置该域，以使用其他板载网络端口或其他 PCIe 插槽中的 PCIe 卡。

在此示例中，`primary` 域仅使用 ZFS 池 (`rpool` (`c0t1d0s0`)) 和网络接口 (`nxge0`)。如果 `primary` 域使用多个设备，请对每个设备重复步骤 2 到步骤 4，以确保没有设备位于将要删除的总线上。

1 **检验 `primary` 域是否拥有多个 PCIe 总线。**

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

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

2 确定需要保留的引导磁盘的设备路径。

- **对于 UFS 文件系统，请运行 `df /` 命令，以确定引导磁盘的设备路径。**

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

- **对于 ZFS 文件系统，首先运行 `df /` 命令以确定池名称，然后运行 `zpool status` 命令以确定引导磁盘的设备路径。**

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

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

3 确定块设备连接到的物理设备。

以下示例使用块设备 `c1t0d0s0`：

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

在此示例中，域 `primary` 的引导磁盘的物理设备连接到总线 `pci@400`，该总线对应于前面列出的 `pci_0`。这意味着您**无法**将 `pci_0` (`pci@400`) 分配到其他域。

4 确定由系统使用的网络接口。

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

处于 `unknown` 状态的接口尚未配置，因此未使用。在此示例中，使用了 `nxge0` 接口。

5 **确定网络接口连接到的物理设备。**

以下命令使用 nxge0 网络接口：

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

在此示例中，域 primary 使用的网络接口的物理设备位于总线 pci@500 下，该总线对应于前面列出的 pci_1。这样，由于其他两个总线 pci_2 (pci@600) 和 pci_3 (pci@700) 未被 primary 域使用，因此可以将它们安全地分配到其他域。

如果 primary 域使用的网络接口位于您要分配到另一个域的总线上，则需要重新配置 primary 域以使用其他网络接口。

6 **将不包含引导磁盘或网络接口的总线从 primary 域删除。**

在此示例中，总线 pci_2 和总线 pci_3 将从 primary 域删除。可能会在 ldm 命令中看到消息，指示 primary 域正进入延迟重新配置模式。

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

7 **将此配置保存到服务处理器。**

在此示例中，配置为 io-domain。

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

此配置 io-domain 还设置为重新引导后要使用的下一个配置。

注 – 目前，SP 上可以保存的配置数限制为 8 个，不包括 factory-default 配置。

---

8 **重新引导 primary 域，以使更改生效。**

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

9 **停止要向其中添加 PCIe 总线的域。**

以下示例停止了 ldg1 域：

```
primary# ldm stop ldg1
```

10 **将可用总线添加到需要直接访问的域。**

可用总线为 pci_2，域为 ldg1。

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

11 **重新启动该域，以使更改生效。**

以下命令可重新启动 ldg1 域：

```
primary# ldm start ldg1
```

12 确认仍然为 primary 域分配了相应的总线，并为 ldg1 域分配了相应的总线。

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

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

此输出确认已将 PCIe 总线 `pci_0` 和 `pci_1` 及其下的设备分配到 `primary` 域，已将 `pci_2` 及其设备分配到 `ldg1`。

分配 PCIe 端点设备

从 Oracle VM Server for SPARC 2.0 发行版和 Oracle Solaris 10 9/10 OS 开始，可以将某个 PCIe 端点（或可分配的直接 I/O）设备分配到域。这种 PCIe 端点设备的使用增加了将设备分配到 I/O 域的粒度。这种功能是通过直接 I/O (Direct I/O, DIO) 功能提供的。

通过 DIO 功能，您可以在系统中创建比 PCIe 总线数更多的 I/O 域。可能的 I/O 域数当前仅受 PCIe 端点设备数限制。

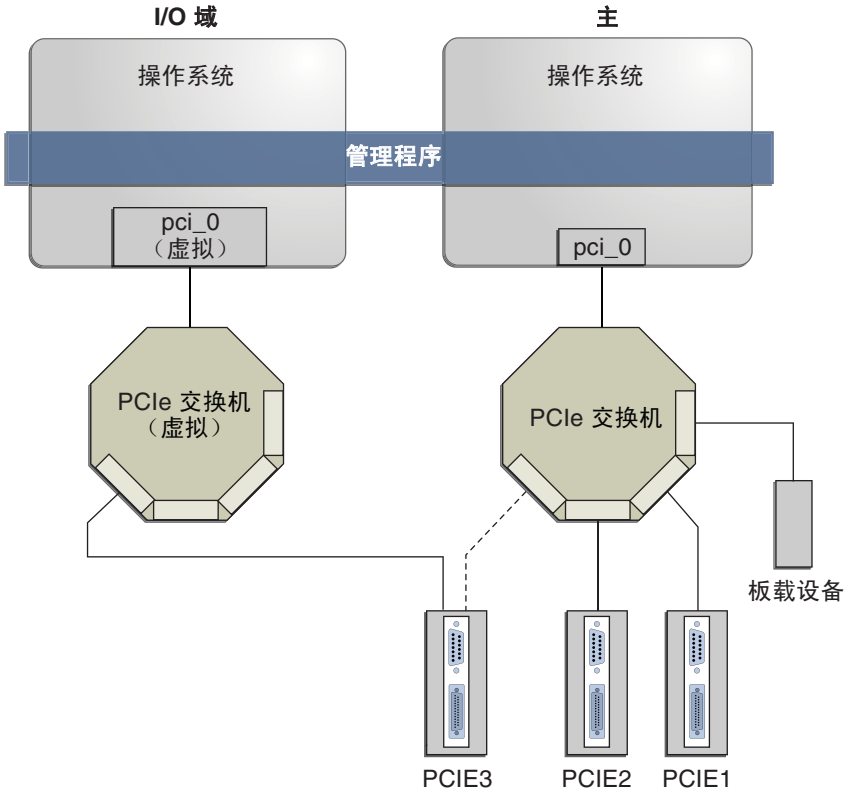
PCIe 端点设备可以是下列任意一个：

- 插槽中的 PCIe 卡
- 由平台标识的板载 PCIe 设备

下图显示了将 PCIe 端点设备 `PCIE3` 分配到 I/O 域。I/O 域中的总线 `pci_0` 和交换机都是虚拟的。不可再在 `primary` 域中访问`PCIE3` 端点设备。

在 I/O 域中，`pci_0` 块和交换机分别为虚拟根联合体和虚拟 PCIe 交换机。该块和交换机与 `primary` 域中的 `pci_0` 块和交换机十分类似。在 `primary` 域中，插槽 `PCIE3` 中的设备是原始设备的阴影格式，并标识为 `SUNW,assigned`。

图 6–2 将 PCIe 端点设备分配到 I/O 域



使用 `ldm list-io` 命令列出 PCIe 端点设备。

即使 DIO 功能允许将插槽中的任意 PCIe 卡分配到 I/O 域，也仅支持某些 PCIe 卡。请参见《Oracle VM Server for SPARC 2.0 Release Notes》中的“Direct I/O Hardware and Software Requirements”。

注 – 不支持具有交换机或桥的 PCIe 卡。也不支持 PCIe 函数级分配。将不受支持的 PCIe 卡分配到 I/O 域可能会导致不可预测的行为。

以下是一些有关 DIO 功能的重要详细信息：

- 仅当满足所有软件要求时才会启用此功能。请参见《Oracle VM Server for SPARC 2.0 Release Notes》中的“Direct I/O Hardware and Software Requirements”。
- 使用 DIO 功能，只可将连接到已分配至 `primary` 域的 PCIe 总线的 PCIe 端点分配到其他域。
- 仅当运行 `primary` 域时，使用 DIO 的 I/O 域才可访问 PCIe 端点设备。

- 重新引导 primary 域会对具有 PCIe 端点设备的 I/O 域产生影响。请参见第 66 页中的 “重新引导 primary 域”。primary 域还具有以下职责：
  - 初始化 PCIe 总线并管理该总线。
  - 处理所有由分配到 I/O 域的 PCIe 端点设备触发的总线错误。请注意，只有 primary 域才会收到所有与 PCIe 总线相关的错误。

直接 I/O 硬件和软件要求

要成功使用 DIO 功能，必须运行相应的软件，并仅将 DIO 功能支持的 PCIe 卡分配到 I/O 域。有关硬件和软件要求，请参见《Oracle VM Server for SPARC 2.0 Release Notes》中的“Direct I/O Hardware and Software Requirements”。

注 – primary 域支持平台上支持的所有 PCIe 卡。有关支持的 PCIe 卡列表，请参见平台的文档。但是，**仅**可将支持直接 I/O 的 PCIe 卡分配到 I/O 域。

直接 I/O 限制

有关如何解决以下限制的信息，请参见第 64 页中的 “规划 PCIe 端点设备配置”。

- 将 PCIe 端点设备分配到 primary 域或者将其从该域删除时会启动延迟重新配置，这意味着仅当 primary 重新引导后才会应用更改。

  重新引导 primary 域会对直接 I/O 产生影响，因此，请仔细规划直接 I/O 配置更改，以最大限度地增加对 primary 域的直接 I/O 相关更改，最大限度地减少 primary 域重新引导。
- 仅当域停止或处于非活动状态时，才允许将 PCIe 端点设备分配到任何其他域或删除该端点设备。

规划 PCIe 端点设备配置

分配或删除 PCIe 端点设备时提前仔细规划以避免产生 primary 域宕机。重新引导 primary 域不仅影响 primary 域本身可用的服务，还会影响已分配有 PCIe 端点设备的 I/O 域。即使对每个 I/O 域的更改不会影响其他域，提前规划有助于最大限度地减小对由该域所提供的服务的影响。

首次分配或删除设备时会启动延迟重新配置。因此，可以继续添加或删除更多设备，然后只重新引导一次 primary 域，以使所有更改生效。

有关示例，请参见第 67 页中的 “通过分配 PCIe 端点设备创建 I/O 域”。

下面介绍了要规划和执行 DIO 设备配置必须执行的步骤：

1. 了解和记录系统硬件配置。

   具体地说，记录系统中有关 PCIe 卡的部件号及其他详细信息的信息。

   使用 `ldm list-io -l` 和 `prtdiag -v` 命令获取信息并保存起来供将来参考。

2. 确定 `primary` 域中所需的 PCIe 端点设备。

   例如，确定提供对以下内容的访问权限的 PCIe 端点设备：

   - 引导磁盘设备
   - 网络设备
   - `primary` 域提供作为服务的其他设备

3. 删除可能会在 I/O 域中使用的所有 PCIe 端点设备。

   重新引导会对 I/O 域产生影响，此步骤将有助于您避免在 `primary` 域上执行后续重新引导操作。

   使用 `ldm rm-io` 命令删除 PCIe 端点设备。使用 pseudonyms 而非设备路径将设备指定到 `rm-io` 和 `add-io` 子命令。

   注 – 即使首次删除 PCIe 端点设备可能会启动延迟重新配置，您也可以继续删除设备。删除所需的所有设备后，仅需要重新引导 `primary` 域一次，使所有更改生效。

   ---

4. 将此配置保存到服务处理器 (service processor, SP)。

   使用 `ldm add-config` 命令。

5. 重新引导 `primary` 域以释放在步骤 3 中删除的 PCIe 端点设备。

6. 确认不再将删除的 PCIe 端点设备分配到 `primary` 域。

   使用 `ldm list-io -l` 命令检验删除的设备在输出中是否显示为 `SUNW,assigned-device`。

7. 将可用的 PCIe 端点设备分配到来宾域，以提供对物理设备的直接访问权。

   分配完成后，将无法再通过域迁移功能将来宾域迁移到其他物理系统。

8. 将 PCIe 端点设备添加到来宾域或者将其从来宾域删除。

   使用 `ldm add-io` 命令。

   通过减少重新引导操作并避免由 I/O 域提供的服务宕机来最大限度地减少对该域的更改。

9. （可选）更改 PCIe 硬件。

   请参见第 66 页中的“更改 PCIe 硬件”。

重新引导 primary 域

PCIe 总线归 primary 域所有，该域负责初始化和管理总线。primary 域必须处于活动状态，并运行支持 DIO 功能的 Oracle Solaris OS 版本。关闭、停止或重新引导 primary 域会中断对 PCIe 总线的访问。PCIe 总线不可用时，该总线上的 PCIe 设备会受到影响，可能会变为不可用。

在运行 I/O 域的同时如果重新引导 primary 域，则具有 PCIe 端点设备的那些 I/O 域的行为是不可预测的。例如，具有 PCIe 端点设备的 I/O 域可能会在重新引导过程中或在重新引导后出现紧急情况。primary 域重新引导时，您可能需要手动停止或启动每个域。

要解决这些问题，请执行以下步骤之一：

- 在关闭 primary 域**之前**，请在系统上手动关闭已将 PCIe 端点设备分配到其中的任意域。

  此步骤可确保您在关闭、停止或重新引导 primary 域之前完全关闭这些域。

  要查找已将 PCIe 端点设备分配到其中的所有域，请运行 ldm list-io 命令。通过此命令，可以列出系统上已分配到域的 PCIe 端点设备。这样，使用此信息可帮助您进行计划。有关此命令输出的详细说明，请参见 ldm(1M) 手册页。

  对于找到的每个域，请通过运行 ldm stop 命令停止它。

- 配置 primary 域和已分配有 PCIe 端点设备的域之间的域依赖关系。

  这种依赖关系可确保具有 PCIe 端点设备的域能够在 primary 域因故重新引导时自动重新启动。

  请注意，此依赖关系会强行重设那些域，并且那些域不能完全关闭。但是，此依赖关系不会对手动关闭的任何域产生影响。

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

更改 PCIe 硬件

以下步骤有助于避免错误地配置 PCIe 端点分配。有关安装和删除特定硬件的平台特定信息，请参见平台的文档。

- 如果要将 PCIe 卡安装到空插槽内，无需执行任何操作。此 PCIe 卡自动归拥有 PCIe 总线的域所有。

  要将新的 PCIe 卡分配到 I/O 域，请使用 ldm rm-io 命令，先将卡从 primary 域删除。然后，使用 ldm add-io 命令将卡分配到 I/O 域。

- 如果要将 PCIe 卡从系统中删除并将其分配到 primary 域，无须执行任何操作。
- 要删除已分配到某个 I/O 域的 PCIe 卡，请先将设备从 I/O 域删除。然后，在从系统中物理删除设备之前，将该设备添加到 primary 域。
- 要替换已分配到某个 I/O 域的 PCIe 卡，请检验 DIO 功能是否支持新卡。

如果支持，则会将新卡自动分配到当前 I/O 域而无需执行任何操作。

如果不支持，请先使用 `ldm rm-io` 命令将该 PCIe 卡从 I/O 域删除。然后，使用 `ldm add-io` 命令将 PCIe 卡重新分配到 `primary` 域。接着，使用其他 PCIe 卡物理替换已分配到 `primary` 域的 PCIe 卡。通过这些步骤，您可以避免不受 DIO 功能支持的配置。

▼ 通过分配 PCIe 端点设备创建 I/O 域

提前规划所有 DIO 部署，从而最大限度地缩短停机时间。

有关通过添加 PCIe 端点设备来创建 I/O 域的示例，请参见第 64 页中的 “规划 PCIe 端点设备配置”。

1 标识和归档系统上当前安装的设备。

`ldm list-io -l` 命令的输出显示了当前配置 I/O 设备的方式。可使用 `prtdiag -v` 命令获取更多详细信息。

注 – 将设备分配到 I/O 域之后，只可在 I/O 域中确定设备的标识。

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

PCIE                  PSEUDONYM  STATUS  DOMAIN
----                  ---------  ------  ------
pci@400/pci@0/pci@c   PCIE1      EMP     -
pci@400/pci@0/pci@9   PCIE2      OCC     primary
        network@0
        network@0,1
        network@0,2
        network@0,3
pci@400/pci@0/pci@d   PCIE3      OCC     primary
        SUNW,emlxs/fp/disk
        SUNW,emlxs@0,1/fp/disk
        SUNW,emlxs@0,1/fp@0,0
pci@400/pci@0/pci@8   MB/SASHBA  OCC     primary
        scsi@0/tape
        scsi@0/disk
        scsi@0/sd@0,0
        scsi@0/sd@1,0
pci@500/pci@0/pci@9   PCIE0      EMP     -
pci@500/pci@0/pci@d   PCIE4      OCC     primary
        network@0
        network@0,1
pci@500/pci@0/pci@c   PCIE5      OCC     primary
        SUNW,qlc@0/fp/disk
        SUNW,qlc@0/fp@0,0
```

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

2 确定需要保留的引导磁盘的设备路径。

- **对于 UFS 文件系统，请运行 `df /` 命令，以确定引导磁盘的设备路径。**

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

- **对于 ZFS 文件系统，首先运行 `df /` 命令以确定池名称，然后运行 `zpool status` 命令以确定引导磁盘的设备路径。**

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

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

3 确定块设备连接到的物理设备。

以下示例使用块设备 `c0t1d0s0`：

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

在此示例中，`primary` 域的引导磁盘的物理设备已连接到 PCIe 端点设备 (`pci@400/pci@0/pci@8`)，该端点设备对应于步骤 1 中列出的 `MB/SASHBA`。删除此设备可能会阻止 `primary` 域引导，因此请**不要**将此设备从 `primary` 域删除。

4 **确定由系统使用的网络接口。**

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

在此示例中，nxge0 接口用作 primary 域的网络接口。

5 **确定网络接口连接到的物理设备。**

以下命令使用 nxge0 网络接口：

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

在此示例中，primary 域使用的网络接口的物理设备已连接到 PCIe 端点设备 (pci@500/pci@0/pci@8)，该端点设备对应于步骤 1 中列出的 MB/NET0。因此，您**不**希望将此设备从 primary 域删除。由于所有其他 PCIe 设备都未被 primary 域使用，因此可以将它们安全地分配到其他域。

如果 primary 域使用的网络接口位于您要分配到其他域的总线上，则需要重新配置 primary 域以使用其他网络接口。

6 **删除可能会在 I/O 域中使用的 PCIe 端点设备。**

在此示例中，您可以删除 PCIE2、PCIE3、PCIE4 和 PCIE5 端点设备，因为它们没有被 primary 域使用。

a. **删除 PCIe 端点设备。**

注意 – 请**不要**删除 primary 域中使用的设备。

如果错误地删除了错误设备，请使用 ldm cancel-op reconf primary 命令取消 primary 域上的延迟重新配置。

可以一次删除多个设备以避免多次重新引导。

```
# ldm rm-io PCIE2 primary
Initiating a delayed reconfiguration operation on the primary domain.
All configuration changes for other domains are disabled until the primary
domain reboots, at which time the new configuration for the primary domain
will also take effect.
# ldm rm-io PCIE3 primary
------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
------------------------------------------------------------------------------
# ldm rm-io PCIE4 primary
------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
```

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

b. 将新配置保存到服务处理器 (service processor, SP)。

以下命令会将配置保存到名为 `dio` 的文件中：

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

c. 重新引导系统以反映 PCIe 端点设备的删除。

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

7 登录 `primary` 域，检验是否不再将 PCIe 端点设备分配到域。

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

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

注 – 对于删除的 PCIe 端点设备，`ldm list-io -l` 输出可能会显示 `SUNW,assigned-device`。不可再从 `primary` 域获取实际信息，但是要将设备分配到其中的域包含此信息。

---

8 将 PCIe 端点设备分配到域。

a. 将 `PCIE2` 设备添加到 `ldg1` 域。

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

b. 绑定并启动 `ldg1` 域。

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

9 登录 ldg1 域并检验设备是否可用。

使用 dladm show-dev 命令检验网络设备是否可用。然后，配置网络设备以在域中使用。

```
# dladm show-dev
vnet0           link: up        speed: 0     Mbps       duplex: unknown
nxge0           link: unknown   speed: 0     Mbps       duplex: unknown
nxge1           link: unknown   speed: 0     Mbps       duplex: unknown
nxge2           link: unknown   speed: 0     Mbps       duplex: unknown
nxge3           link: unknown   speed: 0     Mbps       duplex: unknown
```

◆ ◆ ◆ 第 7 章

使用虚拟磁盘

本章介绍如何将虚拟磁盘与 Oracle VM Server for SPARC 软件结合使用。

本章包括以下内容：

- 第 73 页中的“虚拟磁盘简介”
- 第 74 页中的“管理虚拟磁盘”
- 第 76 页中的“虚拟磁盘标识符和设备名称”
- 第 77 页中的“虚拟磁盘外观”
- 第 77 页中的“虚拟磁盘后端选项”
- 第 79 页中的“虚拟磁盘后端”
- 第 84 页中的“配置虚拟磁盘多路径”
- 第 86 页中的“CD、DVD 和 ISO 映像”
- 第 89 页中的“虚拟磁盘超时”
- 第 90 页中的“虚拟磁盘和 SCSI”
- 第 90 页中的“虚拟磁盘和 `format`(1M) 命令”
- 第 90 页中的“将 ZFS 用于虚拟磁盘”
- 第 94 页中的“在 Logical Domains 环境中使用卷管理器”

虚拟磁盘简介

虚拟磁盘包含两个组件：显示在来宾域中的虚拟磁盘本身，以及在其中存储数据和结束虚拟 I/O 的虚拟磁盘后端。虚拟磁盘后端由虚拟磁盘服务器 (`vds`) 驱动程序从服务域导出。`vds` 驱动程序通过虚拟机管理程序，借助于逻辑域通道 (logical domain channel, LDC) 与来宾域中的虚拟磁盘客户机 (`vdc`) 驱动程序进行通信。最终，虚拟磁盘在来宾域中显示为 `/dev/[r]dsk/cXdYsZ` 设备。

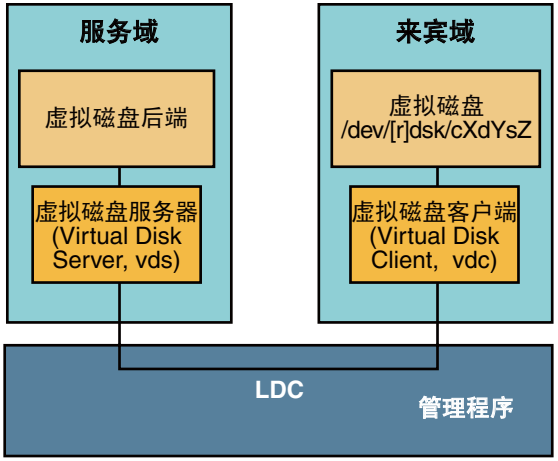
虚拟磁盘后端可以是物理设备，也可以是逻辑设备。物理设备可包括下列项：

- 物理磁盘或磁盘逻辑单元号 (logical unit number, LUN)
- 物理磁盘分片

逻辑设备可以是下列任意项：

- 文件系统（如 ZFS 或 UFS）上的文件
- 卷管理器（如 ZFS、VxVM 或 Solaris 卷管理器）中的逻辑卷
- 可从服务域访问的任意磁盘伪设备

图 7–1 具备 Logical Domains 的虚拟磁盘



管理虚拟磁盘

本节将介绍如何向来宾域添加虚拟磁盘、如何更改虚拟磁盘和超时选项，以及如何从来宾域中删除虚拟磁盘。有关虚拟磁盘选项的说明，请参见第 77 页中的 “虚拟磁盘后端选项”。有关虚拟磁盘超时的说明，请参见第 89 页中的 “虚拟磁盘超时”。

▼ 添加虚拟磁盘

1 **从服务域导出虚拟磁盘后端。**

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

2 **将后端指定给来宾域。**

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

您可以通过设置 `id` 属性来指定新虚拟磁盘设备的 ID。默认情况下，ID 值会自动生成，所以，如果您需要匹配 OS 中的现有设备名称，请设置此属性。请参见第 76 页中的 “虚拟磁盘标识符和设备名称”。

注 – 绑定来宾域 (*ldom*) 时，后端实际上是从服务域导出并指定给来宾域的。

▼ 多次导出虚拟磁盘后端

通过相同或不同的虚拟磁盘服务器可多次导出虚拟磁盘后端。每个导出的虚拟磁盘后端实例随后可指定给相同或不同的来宾域。

如果要多次导出虚拟磁盘后端，则不应使用独占 (`excl`) 选项将其导出。指定 `excl` 选项仅允许将后端导出一次。使用 `ro` 选项，可以将后端作为只读设备安全地导出多次。

注意 – 多次导出虚拟磁盘后端时，在来宾域上运行且使用该虚拟磁盘的应用程序将负责协调和同步并发写入访问，以确保数据一致性。

以下示例介绍如何通过同一个虚拟磁盘服务将同一个虚拟磁盘添加到两个不同的来宾域。

1. **使用下列命令将虚拟磁盘后端从服务域导出两次。**

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

   请注意，第二个 `ldm add-vdsdev` 命令将使用 `-f` 选项强制执行后端的第二次导出。将同一后端路径用于这两个命令且虚拟磁盘服务器位于同一服务域时，使用此选项。

2. **使用下列命令将导出的后端指定给每个来宾域。**

   对于 `ldom1` 和 `ldom2` 而言，*disk-name* 可以不同。

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

▼ 更改虚拟磁盘选项

有关虚拟磁盘选项的更多信息，请参见第 77 页中的 “虚拟磁盘后端选项”。

- **从服务域导出后端之后，可以通过使用以下命令更改虚拟磁盘选项。**

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

▼ 更改超时选项

有关虚拟磁盘选项的更多信息，请参见第 77 页中的 “虚拟磁盘后端选项”。

- **将虚拟磁盘指定给来宾域之后，可以通过使用以下命令更改虚拟磁盘的超时。**

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

▼ 删除虚拟磁盘

1 **使用以下命令从来宾域删除虚拟磁盘。**

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

2 **使用以下命令停止从服务域导出相应的后端。**

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

虚拟磁盘标识符和设备名称

当您使用 `ldm add-vdisk` 命令向域添加虚拟磁盘时，您可以通过设置 `id` 属性来指定其设备编号。

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

域的每个虚拟磁盘都有一个唯一的设备编号，该设备编号是在绑定域时指定的。如果使用显式设备编号（通过设置 `id` 属性获得）添加虚拟磁盘，则会使用指定的设备编号。否则，系统将自动指定最低的可用设备编号。在这种情况下，指定的设备编号将取决于向域添加虚拟磁盘的方式。绑定域后，最终指定给虚拟磁盘的设备编号会显示在 `ldm list-bindings` 命令的输出中。

当具有虚拟磁盘的域运行 Oracle Solaris OS 时，每个虚拟磁盘在该域中都将显示为 `c0d`*n* 磁盘设备，其中 *n* 是虚拟磁盘的设备编号。

在以下示例中，`ldg1` 域具有两个虚拟磁盘：`rootdisk` 和 `pdisk`。`rootdisk` 的设备编号为 `0`（`disk@0`），在域中显示为 `c0d0` 磁盘设备。`pdisk` 的设备编号为 `1`（`disk@1`），在域中显示为 `c0d1` 磁盘设备。

```
primary# ldm list-bindings ldg1
...
DISK
    NAME             VOLUME                          TOUT DEVICE  SERVER         MPGROUP
    rootdisk         dsk_nevada@primary-vds0              disk@0  primary
    pdisk            c3t40d1@primary-vds0                 disk@1  primary
...
```

注意 – 如果设备编号未显式指定给虚拟磁盘，则在域取消绑定并稍后重新绑定时，可更改其设备编号。在这种情况下，由该域中运行的 OS 指定的设备名称还可以更改和解除系统的现有配置。例如，从域的配置中删除虚拟磁盘时，可能会出现这种情况。

虚拟磁盘外观

将后端导出为虚拟磁盘之后，它可以在来宾域中显示为完整磁盘或具有单个分片的磁盘。它的显示方式取决于后端的类型以及导出后端时所使用的选项。

完整磁盘

将后端作为完整磁盘导出到某个域之后，它将在该域中显示为一个具有 8 个分片（s0 至 s7）的常规磁盘。使用 format(1M) 命令可显示此类磁盘。使用 fmthard(1M) 或 format(1M) 命令可更改磁盘的分区表。

在 OS 安装软件中也可以显示完整磁盘，并且能够将其选作可安装 OS 的磁盘。

除只能作为具有单个分片的磁盘导出的物理磁盘分片之外，所有后端都可作为完整磁盘导出。

具有单个分片的磁盘

将后端作为具有单个分片的磁盘导出到某个域之后，它将在该域中显示为一个具有 8 个分片（s0 至 s7）的常规磁盘。但是，只有第一个分片 (s0) 可用。使用 format(1M) 命令可显示此类磁盘，但无法更改磁盘的分区表。

在 OS 安装软件中也可以显示具有单个分片的磁盘，并且能够将其选作可安装 OS 的磁盘。在这种情况下，如果使用 UNIX 文件系统 (UNIX File System, UFS) 安装 OS，则必须只定义根分区 (/)，并且该分区必须占用所有磁盘空间。

除只能作为完整磁盘导出的物理磁盘之外，所有后端都可作为具有单个分片的磁盘导出。

注 – 在 Oracle Solaris 10 10/08 OS 发行版之前，具有单个分片的磁盘显示为具有一个分区 (s0) 的磁盘。使用 format(1M) 命令不能显示此类磁盘。在 OS 安装软件中也不会显示此类磁盘，并且也不能将其选作为可安装 OS 的磁盘设备。

虚拟磁盘后端选项

导出虚拟磁盘后端时可指定不同的选项。这些选项在 ldm add-vdsdev 命令的 options= 参数中以逗号分隔的列表形式表示。有效选项包括：ro、slice 和 excl。

只读 (ro) 选项

只读 (ro) 选项指定将后端导出为只读设备。在这种情况下，只能对指定给来宾域的虚拟磁盘进行读取访问操作，对该虚拟磁盘进行的任何写入操作都将失败。

独占 (excl) 选项

独占 (excl) 选项指定在将服务域中的后端作为虚拟磁盘导出到另一个域时，该后端只能通过虚拟磁盘服务器以独占方式打开。以独占方式打开后端时，服务域中的其他应用程序将无法访问该后端。这可以防止服务域中运行的应用程序无意中使用正在由来宾域使用的后端。

注 – 某些驱动程序不支持 `excl` 选项，并且不允许以独占方式打开某些虚拟磁盘后端。已知 `excl` 选项适用于物理磁盘和分片，但该选项不适用于文件。该选项可能适用于也可能不适用于伪设备（如磁盘卷）。如果后端的驱动程序不支持独占打开，则会忽略后端的 `excl` 选项，且不能以独占方式打开后端。

因为 `excl` 选项可防止服务域中运行的应用程序访问导出到来宾域的后端，因此，请勿在以下情况下设置 `excl` 选项：

- 当来宾域正在运行时，如果您希望能够使用某些命令（如 `format(1M)` 或 `luxadm(1M)`）管理物理磁盘，则不要使用 `excl` 选项导出这些磁盘。
- 当您导出 Solaris 卷管理器卷（如 RAID 或镜像卷）时，请勿设置 `excl` 选项。否则，在 RAID 或镜像卷的组件出现故障时，这可阻止 Solaris 卷管理器启动某恢复操作。有关更多信息，请参见第 95 页中的 “在 Solaris 卷管理器之上使用虚拟磁盘”。
- 如果 Veritas 卷管理器 (Veritas Volume Manager, VxVM) 已安装在服务域中，并且针对物理磁盘启用了 Veritas 动态多路径 (Veritas Dynamic Multipathing, VxDMP)，则在导出物理磁盘时不得使用 `excl` 选项（非默认选项）。否则，导出将失败，因为虚拟磁盘服务器 (virtual disk server, `vds`) 无法打开物理磁盘设备。有关更多信息，请参见第 95 页中的 “在安装了 VxVM 的情况下使用虚拟磁盘”。
- 如果要从同一个虚拟磁盘服务多次导出同一个虚拟磁盘后端，请参见第 75 页中的 “多次导出虚拟磁盘后端”以了解更多信息。

默认情况下，后端以非独占方式打开。这样，将后端导出到其他域时，后端仍可以由服务域中运行的应用程序使用。请注意，这是从 Oracle Solaris 10 5/08 OS 发行版开始新增的行为。在 Oracle Solaris 10 5/08 OS 发行版之前，磁盘后端始终以独占方式打开，且不可能以非独占方式打开。

分片 (slice) 选项

通常，后端既可以导出为完整磁盘，也可以导出为具有单个分片的磁盘，具体取决于后端的类型。如果指定了 `slice` 选项，则会强制将后端导出为具有单个分片的磁盘。

当您要导出后端的原始内容时，此选项非常有用。例如，如果存在其中存有数据的 ZFS 或 Solaris 卷管理器卷，且希望来宾域访问此数据，则应使用 `slice` 选项导出 ZFS 或 Solaris 卷管理器卷。

有关此选项的更多信息，请参见第 79 页中的 “虚拟磁盘后端”。

虚拟磁盘后端

虚拟磁盘后端是存储虚拟磁盘数据的位置。后端可以是磁盘、磁盘分片、文件或卷（如 ZFS、Solaris 卷管理器或 VxVM）。后端在来宾域中既可以显示为完整磁盘，也可以显示为具有单个分片的磁盘，具体取决于在从服务域导出后端时是否设置了 `slice` 选项。默认情况下，虚拟磁盘后端以非独占方式导出为可读写的完整磁盘。

物理磁盘或磁盘 LUN

物理磁盘或磁盘 LUN 始终作为完整磁盘导出。在这种情况下，虚拟磁盘驱动程序（`vds` 和 `vdc`）从虚拟磁盘转发 I/O，并充当到物理磁盘或磁盘 LUN 的传递通道。

通过在不设置 `slice` 选项的情况下导出与该磁盘分片 2 (`s2`) 相对应的设备，可从服务域导出物理磁盘或磁盘 LUN。如果使用 `slice` 选项导出某个磁盘的分片 2，则将只导出该分片，而非整个磁盘。

▼ 将物理磁盘作为虚拟磁盘导出

1. **将物理磁盘作为虚拟磁盘导出。**

   例如，要将物理磁盘 `c1t48d0` 作为虚拟磁盘导出，则必须导出该磁盘的分片 2 (`c1t48d0s2`)。

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

2. **将磁盘指定给来宾域。**

   例如，将磁盘 (`pdisk`) 指定给来宾域 `ldg1`。

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

3. **来宾域启动并运行 Oracle Solaris OS 之后，检验该磁盘是否可供访问且是否为完整磁盘。**

   完整磁盘是一个具有八 (8) 个分片的常规磁盘。

   例如，要检查的磁盘为 `c0d1`。

   ```
   ldg1# ls -1 /dev/dsk/c0d1s*
   /dev/dsk/c0d1s0
   /dev/dsk/c0d1s1
   /dev/dsk/c0d1s2
   /dev/dsk/c0d1s3
   /dev/dsk/c0d1s4
   /dev/dsk/c0d1s5
   /dev/dsk/c0d1s6
   /dev/dsk/c0d1s7
   ```

物理磁盘分片

物理磁盘分片始终作为具有单个分片的磁盘导出。在这种情况下，虚拟磁盘驱动程序（vds 和 vdc）从虚拟磁盘转发 I/O，并充当到物理磁盘分片的传递通道。

通过导出相应的分片设备可从服务域导出物理磁盘分片。如果该设备与分片 2 不同，无论您是否指定了 slice 选项，该设备都将自动导出为具有单个分片的磁盘。如果该设备是磁盘的分片 2，则必须设置 slice 选项，才能仅将分片 2 作为具有单个分片的磁盘导出；否则整个磁盘将作为完整磁盘导出。

▼ 将物理磁盘分片作为虚拟磁盘导出

1. **将物理磁盘的某个分片作为虚拟磁盘导出。**

   例如，要将物理磁盘 c1t57d0 的分片 0 作为虚拟磁盘导出，必须按照如下方式导出与分片 (c1t57d0s0) 相对应的设备。

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

   您无需指定 slice 选项，因为分片始终作为具有单个分片的磁盘导出。

2. **将磁盘指定给来宾域。**

   例如，将磁盘 (pslice) 指定给来宾域 ldg1。

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

3. **来宾域启动并运行 Oracle Solaris OS 之后，可列出磁盘（例如，c0d13），并且您可以查看该磁盘是否可供访问。**

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

   尽管有 8 个设备，但是，因为该磁盘是具有单个分片的磁盘，所以仅有第一个分片 (s0) 可用。

▼ 导出分片 2

- **要导出分片 2（例如，磁盘 c1t57d0s2），必须指定 slice 选项，否则，会导出整个磁盘。**

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

文件和卷

文件或卷（例如，来自 ZFS 或 Solaris 卷管理器中）既可以作为完整磁盘导出，也可以作为具有单个分片的磁盘导出，具体取决于是否设置了 `slice` 选项。

文件或卷作为完整磁盘导出

如果未设置 `slice` 选项，文件或卷会作为完整磁盘导出。在这种情况下，虚拟磁盘驱动程序（`vds` 和 `vdc`）从虚拟磁盘转发 I/O 并管理虚拟磁盘的分区。文件或卷最终会变成一个磁盘映像，其中含有虚拟磁盘所有分片中的数据以及用于管理分区和磁盘结构的元数据。

将空白文件或卷作为完整磁盘导出时，它将在来宾域中显示为未格式化的磁盘（即，无分区的磁盘）。然后，您需要在来宾域中运行 `format(1M)` 命令，以便定义可用的分区并写入有效的磁盘标号。如果虚拟磁盘未格式化，则对该磁盘进行的所有 I/O 操作都将失败。

注 – 在 Oracle Solaris 5/08 OS 发行版之前，将空白文件作为虚拟磁盘导出时，系统会写入默认磁盘标号并创建默认分区。但从 Oracle Solaris 5/08 OS 发行版开始，将不再有这种情况，您必须通过在来宾域中运行 `format(1M)` 才能创建分区。

▼ 将文件作为完整磁盘导出

1 在服务域中，创建一个文件（例如，`fdisk0`），用作虚拟磁盘。

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

该文件的大小定义虚拟磁盘的大小。此示例创建了一个 100 MB 的空白文件，以获取 100 MB 的虚拟磁盘。

2 在控制域中，将该文件导出为虚拟磁盘。

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

在此示例中，未设置 `slice` 选项，所以，该文件将作为完整磁盘导出。

3 在控制域中，将磁盘指定给来宾域。

例如，将磁盘 (`fdisk`) 指定给来宾域 `ldg1`。

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

4 来宾域启动并运行 Oracle Solaris OS 之后，检验该磁盘是否可供访问且是否为完整磁盘。

完整磁盘是一个具有 8 个分片的常规磁盘。

以下示例介绍如何列出磁盘 c0d5，并检验该磁盘是否可供访问且是否为完整磁盘。

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

文件或卷作为具有单个分片的磁盘导出

如果已设置 slice 选项，则文件或卷会作为具有单个分片的磁盘导出。在这种情况下，虚拟磁盘仅具有一个分区 (s0)，该分区直接映射到文件或卷后端。文件或卷仅包含写入到虚拟磁盘的数据，而不包含额外数据（如分区信息或磁盘结构）。

将文件或卷作为具有单个分片的磁盘导出时，系统会模拟一个假磁盘分区，这使得文件或卷看起来像磁盘分片。因为对磁盘分区进行了模拟，所以，您无需为该磁盘创建分区。

▼ 将 ZFS 卷作为具有单个分片的磁盘导出

1 创建 ZFS 卷，用作具有单个分片的磁盘。

以下示例介绍如何创建要用作具有单个分片的磁盘的 ZFS 卷 zdisk0。

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

卷的大小定义虚拟磁盘的大小。此示例创建了一个 100 MB 的卷，以获取 100 MB 的虚拟磁盘。

2 在控制域中，将相应的设备导出到该 ZFS 卷，并设置 slice 选项，以便将该卷作为具有单个分片的磁盘导出。

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

3 在控制域中，将卷指定给来宾域。

下面介绍如何将卷 zdisk0 指定给来宾域 ldg1。

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

4 来宾域启动并运行 Oracle Solaris OS 之后，可列出磁盘（例如，c0d9），并且您可以查看该磁盘是否可供访问且是否为具有单个分片的磁盘 (s0)。

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

```
/dev/dsk/c0d9s6
/dev/dsk/c0d9s7
```

导出卷以及向后兼容性

在 Oracle Solaris 10 5/08 OS 发行版之前，不存在 `slice` 选项，且卷会作为具有单个分片的磁盘导出。如果已存在将卷作为虚拟磁盘导出的配置，并且已将系统升级到 Oracle Solaris 10 5/08 OS，则卷现在将作为完整磁盘而非具有单个分片的磁盘导出。要保留旧行为并将卷作为具有单个分片的磁盘导出，您需要执行以下操作之一：

- 在 Oracle VM Server for SPARC 2.0 软件中使用 `ldm set-vdsdev` 命令，并针对要作为具有单个分片的磁盘导出的所有卷设置 `slice` 选项。有关此命令的更多信息，请参见 `ldm(1M)` 手册页。
- 将以下行添加到服务域上的 `/etc/system` 文件中。

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

  注 – 设置此可调参数会将所有卷强制作为具有单个分片的磁盘导出，且无法将任何卷作为完整磁盘导出。

不同类型的后端的导出方式汇总

后端	没有分片选项	设置了分片选项
磁盘（磁盘分片 2）	完整磁盘[1]	具有单个分片的磁盘[2]
磁盘分片（不是分片 2）	具有单个分片的磁盘[3]	具有单个分片的磁盘
文件	完整磁盘	具有单个分片的磁盘
卷（包括 ZFS、Solaris 卷管理器或 VxVM）	完整磁盘	具有单个分片的磁盘

[1] 导出整个磁盘。

[2] 仅导出分片 2

[3] 分片始终作为具有单个分片的磁盘导出。

将文件和磁盘分片作为虚拟磁盘导出的准则

本节包括将文件和磁盘分片作为虚拟磁盘导出的准则。

使用回送文件 (lofi) 驱动程序

可以使用回送文件 (`lofi`) 驱动程序将文件作为虚拟磁盘导出。但是，执行此操作会额外添加一个驱动程序层，从而影响虚拟磁盘的性能。您可以改为直接将文件作为完整磁盘或具有单个分片的磁盘导出。请参见第 81 页中的 “文件和卷”。

直接或间接导出磁盘分片

要将分片直接或间接（例如，通过 Solaris 卷管理器卷）作为虚拟磁盘导出，应通过使用 `prtvtoc(1M)` 命令确保该分片不是始于物理磁盘的第一个块（块 0）。

如果直接或间接导出了始于物理磁盘的第一个块的磁盘分片，可能会覆写物理磁盘的分区表，并导致该磁盘的所有分区不可访问。

配置虚拟磁盘多路径

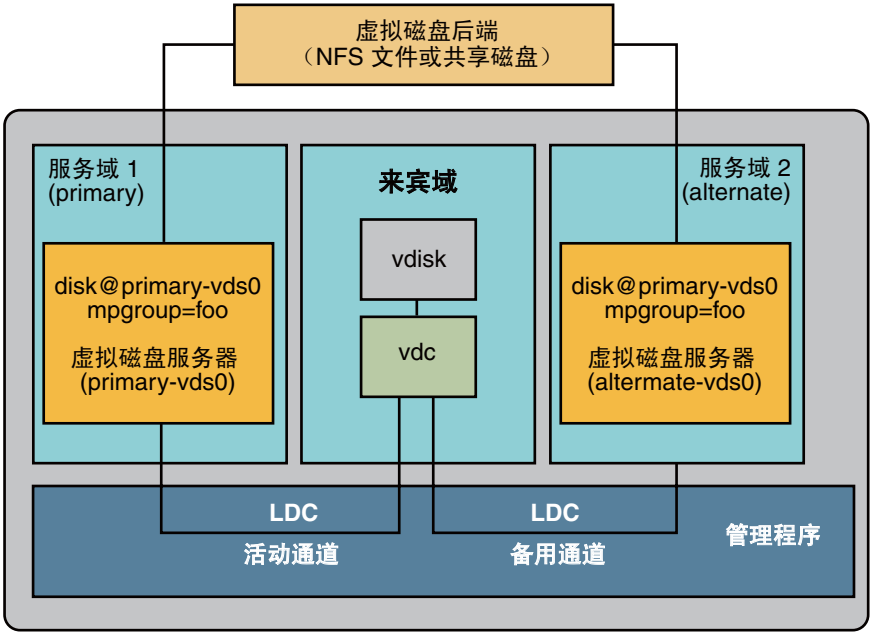
通过**虚拟磁盘多路径**可以在来宾域中配置可通过多个路径对其后端存储进行访问的虚拟磁盘。这些路径指向不同的能够对同一后端存储（如磁盘 LUN）提供访问权限的服务域。即使在其中一个服务域关闭的情况下，此功能也能够使来宾域中的虚拟磁盘处于可访问状态。例如，您可以设置一个虚拟磁盘多路径配置，用于访问网络文件系统 (network file system, NFS) 服务器上的文件。或者，您可以使用此配置访问已连接到多个服务域的共享存储中的 LUN。所以，当来宾域访问虚拟磁盘时，虚拟磁盘驱动程序将通过其中一个服务域访问后端存储。如果虚拟磁盘驱动程序无法连接到服务域，则该虚拟磁盘会尝试通过其他服务域访问后端存储。

注 – 从 Oracle VM Server for SPARC 2.0 发行版开始，虚拟磁盘多路径功能可以检测服务域无法访问后端存储的时间。在此类实例中，虚拟磁盘驱动程序将尝试通过另一个路径访问后端存储。

要启用虚拟磁盘多路径功能，必须从每个服务域导出虚拟磁盘后端，并将虚拟磁盘添加到同一多路径组 (`mpgroup`) 中。`mpgroup` 由一个名称标识，并在导出虚拟磁盘后端之后进行配置。

下图显示了虚拟磁盘多路径配置，在第 85 页中的 “配置虚拟磁盘多路径”过程中将以此配置为例进行说明。在此示例中，名为 `foo` 的多路径组用于创建虚拟磁盘，通过以下两个服务域可对其后端进行访问：`primary` 和 `alternate`。

图 7–2 配置虚拟磁盘多路径



▼ 配置虚拟磁盘多路径

1 **从 `primary` 服务域导出虚拟磁盘后端。**

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

其中 *backend-path1* 是指向 `primary` 域中虚拟磁盘后端的路径。

2 **从 `alternate` 服务域导出同一个虚拟磁盘后端。**

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

其中 *backend-path2* 是指向 `alternate` 域中虚拟磁盘后端的路径。

注 – *backend-path1* 和 *backend-path2* 是指向两个不同的域（`primary` 和 `alternate`）中同一虚拟磁盘后端的路径。这些路径可以相同或不同，具体取决于 `primary` 和 `alternate` 域的配置。用户可以选择 *volume* 名。对于两个命令，卷名可以相同，也可以不同。

---

3 **将虚拟磁盘导出到来宾域。**

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

注 – 尽管通过不同的服务域将虚拟磁盘后端多次导出，但只能为来宾域指定一个虚拟磁盘，并通过任意服务域将该磁盘与虚拟磁盘后端关联。

更多信息

配置虚拟磁盘多路径后所产生的结果

对虚拟磁盘配置多路径并启动来宾域之后，虚拟磁盘可通过与其关联的服务域（本示例中为 `primary` 域）访问其后端。如果此服务域不可用，则虚拟磁盘会尝试通过属于同一多路径组的其他服务域访问其后端。

注意 – 定义多路径组 (`mpgroup`) 时，应确保属于同一个 `mpgroup` 的虚拟磁盘后端实际上是同一个虚拟磁盘后端。如果将不同的后端添加到同一个 `mpgroup` 中，则可能会出现意外行为并有可能丢失或损坏存储在这些后端上的数据。

CD、DVD 和 ISO 映像

您可以按照与导出任意常规磁盘相同的方式导出光盘 (compact disc, CD) 或数字通用光盘 (digital versatile disc, DVD)。要将 CD 或 DVD 导出到来宾域，应将 CD 或 DVD 设备的分片 2 作为完整磁盘导出（即，不使用 `slice` 选项）。

注 – 您无法导出 CD 或 DVD 驱动器本身，而只能导出 CD 或 DVD 驱动器内部的 CD 或 DVD。因此，CD 或 DVD 必须存在于该驱动器内部，才可以将其导出。此外，要能够导出 CD 或 DVD，该 CD 或 DVD 在服务域中不能处于使用状态。特别是，卷管理文件系统 `volfs(7FS)` 服务不得使用该 CD 或 DVD。有关如何使 `volfs` 停止使用设备的说明，请参见第 87 页中的 “将 CD 或 DVD 从服务域导出到来宾域”。

如果存在存储在文件或卷中的 CD 或 DVD 的国际标准化组织 (International Organization for Standardization, ISO) 映像，并将该文件或卷作为完整磁盘导出，则它在来宾域中将显示为一个 CD 或 DVD。

导出 CD、DVD 或 ISO 映像后，它将在来宾域中自动显示为只读设备。但是，您不能在来宾域中执行任何 CD 控制操作；即，无法从来宾域启动、停止或弹出 CD。如果导出的 CD、DVD 或 ISO 映像可引导，则可以在相应的虚拟磁盘中对来宾域进行引导。

例如，如果导出 Oracle Solaris OS 安装 DVD，则可以在与该 DVD 相对应的虚拟磁盘上引导来宾域，并从该 DVD 安装该来宾域。为此，当来宾域进入 `ok` 提示符时，请使用以下命令。

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

其中 *n* 是表示已导出 DVD 的虚拟磁盘的索引。

注 – 如果导出 Oracle Solaris OS 安装 DVD，并在与该 DVD 相对应的虚拟磁盘上引导来宾域以对其进行安装，则在安装期间不能更改该 DVD。因此，您可能需要跳过任何请求其他 CD/DVD 的安装步骤，或者需要提供备用路径来访问请求的此介质。

▼ 将 CD 或 DVD 从服务域导出到来宾域

1. **在服务域中，检查卷管理守护进程 `vold(1M)` 是否正在运行且处于联机状态。**

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

2. **执行以下操作之一：**

   - 如果卷管理守护进程未运行或未处于联机状态，请跳至步骤 3。
   - 如果卷管理守护进程正在运行且处于联机状态（如步骤 1 的示例所示），请执行以下操作：

   a. **编辑 `/etc/vold.conf` 文件并注释掉以下列文字开头的行。**

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

   请参见 `vold.conf(4)` 手册页。

   b. **将 CD 或 DVD 插入 CD 或 DVD 驱动器。**

   c. **在服务域中，重新启动卷管理文件系统服务。**

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

3. **在服务域中，查找 CD-ROM 设备的磁盘路径。**

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

4. **将 CD 或 DVD 磁盘设备作为完整磁盘导出。**

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

5. **将导出的 CD 或 DVD 指定给来宾域。**

下面介绍如何将导出的 CD 或 DVD 指定给域 `ldg1`：

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

更多信息 多次导出 CD 或 DVD

CD 或 DVD 可以导出多次并可以指定给不同的来宾域。有关更多信息，请参见第 75 页中的 “多次导出虚拟磁盘后端”。

▼ 从 primary 域导出 ISO 映像以安装来宾域

以下过程介绍如何从 primary 域导出 ISO 映像并使用它安装来宾域。并假定 primary 域和来宾域均已配置。

例如，以下 ldm list 显示 primary 和 ldom1 域均已进行配置：

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

1. **添加虚拟磁盘服务器设备以导出 ISO 映像。**

   在本示例中，ISO 映像是 /export/images/sol-10-u8-ga-sparc-dvd.iso。

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

2. **停止来宾域。**

   在本示例中，逻辑域是 ldom1。

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

3. **将 ISO 映像的虚拟磁盘添加到逻辑域。**

   在本示例中，逻辑域是 ldom1。

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

4. **重新启动来宾域。**

   在本示例中，逻辑域是 ldom1。

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

   在本示例中，ldm list 命令显示 ldom1 域已刚刚启动。

5. **连接到来宾域。**

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

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

6 **验证 ISO 映像是否以虚拟磁盘形式存在。**

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

在本示例中，新添加的设备是 `/virtual-devices@100/channel-devices@200/disk@1`。

7 **引导要从 ISO 映像安装的来宾域。**

在本示例中，从 `/virtual-devices@100/channel-devices@200/disk@1` 磁盘的 `f` 分片引导。

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

虚拟磁盘超时

默认情况下，如果提供对虚拟磁盘后端访问的服务域已关闭，则从来宾域到相应虚拟磁盘的所有 I/O 都将被阻止。当服务域可以正常运行并能够向虚拟磁盘后端提供 I/O 请求服务时，I/O 可自动恢复。

但是，在某些情况下，文件系统或应用程序可能不希望阻止 I/O 操作，而是希望在服务域关闭时间过长时，该操作会失败并报告一个错误。现在，可以为每个虚拟磁盘设置一个连接超时期限，随后可将其用于在来宾域上的虚拟磁盘客户机和服务域上的虚拟磁盘服务器之间建立连接。当达到超时期限时，只要服务域关闭且未在虚拟磁盘客户机和服务器之间重新建立连接，所有暂挂的 I/O 和所有新的 I/O 都将失败。

通过执行下列操作之一可设置此超时：

- 使用 `ldm add-vdisk` 命令。

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

- 使用 `ldm set-vdisk` 命令。

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

请以秒为单位指定超时。如果将超时设置为 `0`，则会禁用超时，并在服务域关闭时阻止 I/O（这是默认设置和行为）。

或者，可通过将以下行添加到来宾域上的 `/etc/system` 文件中来设置超时。

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

注 – 如果已设置了此可调参数，它将覆写使用 `ldm` CLI 进行的任何超时设置。此外，此可调参数为来宾域中的所有虚拟磁盘设置超时。

虚拟磁盘和 SCSI

如果将物理 SCSI 磁盘或 LUN 作为完整磁盘导出，则相应的虚拟磁盘会支持用户 SCSI 命令接口 `uscsi`(7I) 和多主机磁盘控制操作 `mhd`(7i)。其他虚拟磁盘（如将文件或卷作为后端的虚拟磁盘）不支持这些接口。

因此，使用 SCSI 命令（例如 Solaris 卷管理器 `metaset` 或 Oracle Solaris Cluster `shared devices`）的应用程序或产品功能在来宾域中只能与将物理 SCSI 磁盘作为后端的虚拟磁盘一起使用。

注 – 因为服务域对用作虚拟磁盘后端的物理 SCSI 磁盘或 LUN 进行管理，所以服务域可有效地执行 SCSI 操作。特别是，该服务域设置了 SCSI 预留空间。因此，服务域和来宾域中运行的应用程序不应将 SCSI 命令发出至同一个物理 SCSI 磁盘；否则，这会导致出现意外的磁盘状态。

虚拟磁盘和 `format`(1M) 命令

`format`(1M) 命令可识别域中存在的所有虚拟磁盘。但是，对于作为具有单个分片的磁盘导出的虚拟磁盘，`format` 命令无法更改此类虚拟磁盘的分区表。诸如 `label` 之类的命令都将失败，除非尝试写入与已与虚拟磁盘关联的磁盘标号类似的磁盘标号。

其后端为 SCSI 磁盘的虚拟磁盘支持所有 `format`(1M) 子命令。其后端不是 SCSI 磁盘的虚拟磁盘不支持某些 `format`(1M) 子命令（如 `repair` 和 `defect`）。在这种情况下，`format`(1M) 的行为类似于集成驱动器电子 (Integrated Drive Electronics, IDE) 磁盘的行为。

将 ZFS 用于虚拟磁盘

本节介绍如何使用 Zettabyte 文件系统 (Zettabyte File System, ZFS) 存储已导出到来宾域的虚拟磁盘后端。ZFS 可提供一种简捷、有效的解决方案来创建和管理虚拟磁盘后端。下面是通过 ZFS 能够执行的操作：

- 将磁盘映像存储在 ZFS 卷或 ZFS 文件中
- 使用快照备份磁盘映像
- 使用克隆复制磁盘映像和置备更多的域

有关使用 ZFS 的更多信息，请参见《Oracle Solaris ZFS 管理指南》。

在下面的介绍和示例中，主域还是存储磁盘映像的服务域。

在服务域中配置 ZFS 池

要存储磁盘映像，首先应在服务域中创建一个 ZFS 存储池。例如，以下命令在 `primary` 域中创建了一个包含磁盘 `c1t50d0` 的 ZFS 存储池 `ldmpool`。

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

使用 ZFS 存储磁盘映像

以下命令为来宾域 `ldg1` 创建了一个磁盘映像。已为此来宾域创建了 ZFS 文件系统，从而此来宾域的所有磁盘映像都将存储在该文件系统上。

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

磁盘映像可以存储在 ZFS 卷或 ZFS 文件中。使用 `zfs create -V` 命令可快速创建 ZFS 卷，无论其大小为何。另一方面，必须使用 `mkfile` 命令创建 ZFS 文件。使用此命令完成此过程可能需要一些时间，尤其是创建的文件非常大时（通常是在创建磁盘映像时）。

ZFS 卷和 ZFS 文件都可以利用 ZFS 的功能（例如，快照和克隆功能），但 ZFS 卷是一个伪设备，而 ZFS 文件是一个常规文件。

如果将磁盘映像用作安装有 OS 的虚拟磁盘，则该磁盘映像的大小必须足以满足 OS 安装的要求。此大小取决于 OS 的版本以及所执行的安装类型。如果安装 Oracle Solaris OS，则使用的磁盘大小可以为 20 GB，它足以满足任意版本的 Oracle Solaris OS 的任意安装类型的需求。

使用 ZFS 存储磁盘映像的示例

下列示例可执行下列操作：

1. 在 ZFS 卷或文件上创建 20 GB 的映像。
2. 将 ZFS 卷或文件作为虚拟磁盘导出。导出 ZFS 卷或文件时使用的语法相同，但指向后端的路径不同。
3. 将导出的 ZFS 卷或文件指定给来宾域。

启动来宾域后，ZFS 卷或文件将显示为可安装 Oracle Solaris OS 的虚拟磁盘。

▼ 使用 ZFS 卷创建磁盘映像

- **例如，在 ZFS 卷上创建 20 GB 的磁盘映像。**

```
primary# zfs create -V 20gb ldmpool/ldg1/disk0
```

▼ 使用 ZFS 文件创建磁盘映像

- 例如，在 ZFS 卷上创建 20 GB 的磁盘映像。

```
primary# zfs create ldmpool/ldg1/disk0
primary# mkfile 20g /ldmpool/ldg1/disk0/file
```

▼ 导出 ZFS 卷

- 将 ZFS 卷作为虚拟磁盘导出。

```
primary# ldm add-vdsdev /dev/zvol/dsk/ldmpool/ldg1/disk0 ldg1_disk0@primary-vds0
```

▼ 导出 ZFS 文件

- 将 ZFS 文件作为虚拟磁盘导出。

```
primary# ldm add-vdsdev /ldmpool/ldg1/disk0/file ldg1_disk0@primary-vds0
```

▼ 将 ZFS 卷或文件指定给来宾域

- 将 ZFS 卷或文件指定给来宾域（在以下示例中，来宾域为 `ldg1`）。

```
primary# ldm add-vdisk disk0 ldg1_disk0@primary-vds0 ldg1
```

创建磁盘映像的快照

当磁盘映像存储在 ZFS 卷或 ZFS 文件上时，可通过使用 ZFS 快照命令创建该磁盘映像的快照。

创建磁盘映像的快照之前，请确保该磁盘在来宾域中当前未处于使用状态中，以确保当前存储在磁盘映像上的数据保持一致性。有几种方法可确保磁盘在来宾域中未处于使用状态。您可以执行以下操作之一：

- 停止并取消绑定来宾域。这是一种最安全的解决方案，也是在创建用作来宾域引导磁盘的磁盘映像的快照时，唯一可用的解决方案。
- 或者，您可以卸载要为其创建快照的磁盘的所有在来宾域处于使用状态的分片，从而确保来宾域中无任何分片处于使用状态。

在本示例中，由于 ZFS 布局，不管磁盘映像是存储在 ZFS 卷上还是 ZFS 文件上，创建磁盘映像的快照时所使用的命令都相同。

▼ 创建磁盘映像的快照

- 例如，创建已为 `ldg1` 域创建的磁盘映像的快照。

```
primary# zfs snapshot ldmpool/ldg1/disk0@version_1
```

使用克隆置备新域

创建完磁盘映像的快照后，可以通过使用 ZFS 克隆命令复制该磁盘映像。随后，克隆的映像会指定给另一个域。通过克隆引导磁盘映像可以为新的来宾域快速创建引导磁盘，而无需执行整个 Oracle Solaris OS 安装过程。

例如，如果已创建的 `disk0` 是域 `ldg1` 的引导磁盘，请执行以下操作克隆该磁盘，以为域 `ldg2` 创建引导磁盘。

```
primary# zfs create ldmpool/ldg2
primary# zfs clone ldmpool/ldg1/disk0@version_1 ldmpool/ldg2/disk0
```

随后，`ldompool/ldg2/disk0` 可作为虚拟磁盘导出并指定给新的 `ldg2` 域。域 `ldg2` 可从该虚拟磁盘直接引导，而无需执行 OS 整个安装过程。

克隆引导磁盘映像

克隆引导磁盘映像后，新的映像与原始引导磁盘完全相同，它包含在克隆映像之前存储在引导磁盘上的所有信息（如主机名、IP 地址、已装入的文件系统表或任意系统配置或可调参数）。

因为原始引导磁盘映像上已装入文件系统表与已克隆的磁盘映像上的已装入文件系统表相同，所以，将已克隆的磁盘映像指定给新域的的顺序必须与原始域上的相同。例如，如果引导磁盘映像已指定为原始域的第一个磁盘，则已克隆的磁盘映像也必须指定为新域的第一个磁盘。否则，无法引导新域。

如果原始域是使用静态 IP 地址配置的，则使用克隆映像的新域必须以相同的 IP 地址开始。在这种情况下，您可以通过使用 `sys-unconfig(1M)` 命令更改新域的网络配置。要避免此问题，您还可以创建未配置系统的磁盘映像的快照。

如果原始域是使用动态主机配置协议 (Dynamic Host Configuration Protocol, DHCP) 配置的，则使用克隆映像的新域也应使用 DHCP。在这种情况下，您无需更改新域的网络配置，因为该域在其引导时可自动接收 IP 地址及其网络配置。

注 – 域的主机 ID 没有存储在引导磁盘上，但在创建域时将由 Logical Domains Manager 指定。因此，在克隆磁盘映像时，新域不会保留原始域的主机 ID。

▼ 创建未配置系统的磁盘映像的快照

1. **绑定和启动原始域。**
2. **执行 `sys-unconfig` 命令。**
3. **执行完 `sys-unconfig` 命令后，该域将停止。**
4. **停止和取消绑定域；请勿重新引导该域。**

5 **创建域引导磁盘映像的快照。**

例如：

```
primary# zfs snapshot ldmpool/ldg1/disk0@unconfigured
```

此时，已创建了未配置系统的引导磁盘映像的快照。

6 **克隆此映像，以创建首次引导时要求对系统进行配置的新域。**

在 Logical Domains 环境中使用卷管理器

本节介绍在 Logical Domains 环境中使用卷管理器的情况。

在卷管理器之上使用虚拟磁盘

可以将任何 Zettabyte 文件系统 (Zettabyte File System, ZFS)、Solaris 卷管理器或 Veritas 卷管理器 (Veritas Volume Manager, VxVM) 卷作为虚拟磁盘从服务域导出到来宾域。卷可以作为具有单个分片的磁盘导出（如果使用 `ldm add-vdsdev` 命令指定了 `slice` 选项）或作为完整磁盘导出。

注 – 本节的其余部分将以 Solaris 卷管理器卷为例进行说明。但是，所讨论的内容也适用于 ZFS 卷和 VxVM 卷。

以下示例介绍如何将卷作为具有单个分片的磁盘导出。

来宾域中的虚拟磁盘（例如，`/dev/dsk/c0d2s0`）直接映射到相关联的卷（例如，`/dev/md/dsk/d0`），来宾域的虚拟磁盘上存储的数据直接存储到相关联的卷中，而没有额外的元数据。因此，也可以通过相关联的卷从服务域直接访问来宾域的虚拟磁盘上存储的数据。

示例

- 如果将 Solaris 卷管理器卷 `d0` 从 `primary` 域导出到 `domain1`，则 `domain1` 的配置需要一些额外的步骤。

  ```
  primary# metainit d0 3 1 c2t70d0s6 1 c2t80d0s6 1 c2t90d0s6
  primary# ldm add-vdsdev options=slice /dev/md/dsk/d0 vol3@primary-vds0
  primary# ldm add-vdisk vdisk3 vol3@primary-vds0 domain1
  ```

- 例如，绑定并启动 `domain1` 之后，导出的卷会显示为 `/dev/dsk/c0d2s0`，并且您可以使用它。

  ```
  domain1# newfs /dev/rdsk/c0d2s0
  domain1# mount /dev/dsk/c0d2s0 /mnt
  domain1# echo test-domain1 > /mnt/file
  ```

- 停止并取消绑定 `domain1` 后，可通过 Solaris 卷管理器卷 `d0` 从主域直接访问 `domain1` 的虚拟磁盘上存储的数据。

```
primary# mount /dev/md/dsk/d0 /mnt
primary# cat /mnt/file
test-domain1
```

在 Solaris 卷管理器之上使用虚拟磁盘

当 RAID 或镜像 Solaris 卷管理器卷由另一个域用作虚拟磁盘时，则在导出它时不得设置独占 (excl) 选项。否则，如果 Solaris 卷管理器卷的某个组件出现故障，就无法启动使用 metareplace 命令或热备份恢复 Solaris 卷管理器卷的过程。metastat 命令将卷视为正在进行重新同步，但并未进行重新同步。

例如，/dev/md/dsk/d0 是使用 excl 选项作为虚拟磁盘导出到另一个域的 RAID Solaris 卷管理器卷，并且 d0 配置有一些热备份设备。如果 d0 的组件出现故障，则 Solaris 卷管理器会将出现故障的组件替换为热备份，并重新同步 Solaris 卷管理器卷。但是，重新同步并不会启动。卷会被报告为正在进行重新同步，但并没有进行重新同步。

```
# metastat d0
d0: RAID
    State: Resyncing
    Hot spare pool: hsp000
    Interlace: 32 blocks
    Size: 20097600 blocks (9.6 GB)
Original device:
    Size: 20100992 blocks (9.6 GB)
Device                                          Start Block  Dbase   State Reloc
c2t2d0s1                                            330  No       Okay  Yes
c4t12d0s1                                           330  No       Okay  Yes
/dev/dsk/c10t600C0FF00000000000015153295A4B100d0s1  330  No  Resyncing  Yes
```

这种情况下，必须停止并取消绑定将 Solaris 卷管理器卷用作虚拟磁盘的域，以便完成重新同步。然后，可使用 metasync 命令重新同步 Solaris 卷管理器卷。

```
# metasync d0
```

在安装了 VxVM 的情况下使用虚拟磁盘

在系统上安装 Veritas 卷管理器 (Veritas Volume Manager, VxVM) 后，如果已针对要作为虚拟磁盘导出的物理磁盘或分区启用了 Veritas 动态多路径 (Dynamic Multipathing, DMP)，则在导出该磁盘或分区时不得设置 excl 选项（非默认选项）。否则，在绑定使用此类磁盘的域时，则会在 /var/adm/messages 中收到错误。

```
vd_setup_vd():  ldi_open_by_name(/dev/dsk/c4t12d0s2) = errno 16
vds_add_vd():  Failed to add vdisk ID 0
```

您可以通过查看命令 vxdisk list 的输出中的多路径信息，检查是否启用了 Veritas DMP；例如：

```
# vxdisk list Disk_3
Device:    Disk_3
devicetag: Disk_3
type:      auto
```

```
info:       format=none
flags:      online ready private autoconfig invalid
pubpaths:   block=/dev/vx/dmp/Disk_3s2 char=/dev/vx/rdmp/Disk_3s2
guid:       -
udid:       SEAGATE%5FST336753LSUN36G%5FDISKS%5F30323339483031444304E0000
site:       -
Multipathing information:
numpaths:   1
c4t12d0s2   state=enabled
```

或者，如果您针对要在设置了 `excl` 选项的情况下作为虚拟磁盘导出的磁盘或分片启用了 Veritas DMP，则可以使用 `vxdmpadm` 命令禁用 DMP。例如：

```
# vxdmpadm -f disable path=/dev/dsk/c4t12d0s2
```

在虚拟磁盘之上使用卷管理器

本节介绍在虚拟磁盘之上使用卷管理器的情况。

在虚拟磁盘之上使用 ZFS

任何虚拟磁盘都可以与 ZFS 结合使用。在任何域中都可以导入 ZFS 存储池 (`zpool`)，该域可以看到属于该 `zpool` 的所有存储设备，而不管该域将所有这些设备视为虚拟设备还是实际设备。

在虚拟磁盘之上使用 Solaris 卷管理器

在 Solaris 卷管理器本地磁盘组中可以使用任何虚拟磁盘。例如，虚拟磁盘可用于存储本地磁盘组的 Solaris 卷管理器元设备状态数据库 `metadb(1M)`，或者用于在本地磁盘组中创建 Solaris 卷管理器卷。

其后端为 SCSI 磁盘的任何虚拟磁盘都可以在 Solaris 卷管理器共享磁盘组 `metaset(1M)` 中使用。其后端不是 SCSI 磁盘的虚拟磁盘不能添加到 Solaris 卷管理器共享磁盘组。如果尝试将其后端不是 SCSI 磁盘的虚拟磁盘添加到 Solaris 卷管理器共享磁盘组，则会失败并显示类似下面内容的错误。

```
# metaset -s test -a c2d2
metaset: domain1: test: failed to reserve any drives
```

在虚拟磁盘之上使用 VxVM

有关来宾域中的 VxVM 支持的信息，请参见 Symantec 的 VxVM 文档。

◆ ◆ ◆ 第 8 章

使用虚拟网络

本章介绍如何通过 Oracle VM Server for SPARC 软件使用虚拟网络，其中涵盖下列主题：

- 第 97 页中的 “虚拟网络简介”
- 第 98 页中的 “虚拟交换机”
- 第 98 页中的 “虚拟网络设备”
- 第 99 页中的 “管理虚拟交换机”
- 第 101 页中的 “管理虚拟网络设备”
- 第 103 页中的 “虚拟设备标识符和网络接口名称”
- 第 105 页中的 “自动或手动分配 MAC 地址”
- 第 107 页中的 “将网络适配器和 Logical Domains 结合使用”
- 第 108 页中的 “针对 NAT 和路由配置虚拟交换机和服务域”
- 第 110 页中的 “在 Logical Domains 环境中配置 IPMP”
- 第 116 页中的 “使用 VLAN 标记”
- 第 118 页中的 “使用 NIU 混合 I/O”
- 第 121 页中的 “将链路聚合和虚拟交换机结合使用”
- 第 122 页中的 “配置巨型帧”

虚拟网络简介

虚拟网络允许域在不使用任何外部物理网络的情况下进行相互通信。虚拟网络也可以允许域使用同一物理网络接口访问物理网络并与远程系统通信。配置虚拟交换机可创建虚拟网络，您可以将虚拟网络设备连接到此虚拟交换机。

虚拟交换机

虚拟交换机 (vsw) 是在服务域中运行的组件，由虚拟交换机驱动程序管理。可以将虚拟交换机连接到某些来宾域，以在这些域之间进行网络通信。此外，如果虚拟交换机还与物理网络接口相关联，则允许通过该物理网络接口在来宾域和物理网络间进行网络通信。虚拟交换机还具有网络接口 vsw*n*，该接口允许服务域与连接到该虚拟交换机的其他域通信。可以像使用任何常规网络接口一样使用该接口，并使用 `ifconfig(1M)` 命令对其进行配置。

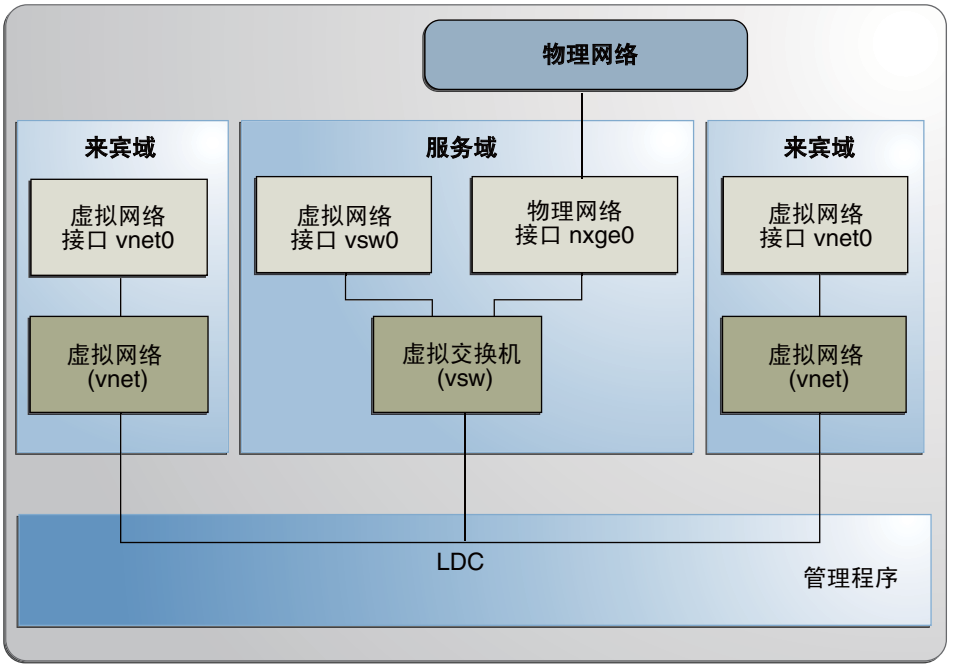
注 – 将虚拟交换机添加到服务域时，并未激活 (plumb) 其网络接口。因此，默认情况下，服务域无法与连接到其虚拟交换机的来宾域通信。要在来宾域和服务域之间进行网络通信，必须在服务域中激活 (plumb) 并配置关联的虚拟交换机的网络接口。有关说明，请参见第 46 页中的 “启用控制/服务域与其他域之间的联网”。

虚拟网络设备

虚拟网络 (vnet) 设备是在连接到虚拟交换机的域中定义的虚拟设备。虚拟网络设备由虚拟网络驱动程序管理，并且使用逻辑域通道 (logical domain channel, LDC) 通过虚拟机管理程序连接到虚拟网络。

虚拟网络设备可用作名为 vnet*n* 的网络接口，可以像使用任何常规网络接口一样使用该接口，并使用 `ifconfig(1M)` 命令对其进行配置。

图 8–1 设置虚拟网络



下面是图 8–1 中示例的说明。

- 服务域中的虚拟交换机连接到来宾域。这样，来宾域便可以相互通信。
- 虚拟交换机也连接到物理网络接口 `nxge0`。这样，来宾域便可以与物理网络通信。
- 虚拟交换机网络接口 `vsw0` 已在服务域中激活 (plumb)，因此，两个来宾域可以与服务域通信。
- 可以使用 `ifconfig`(1M) 命令配置服务域中的虚拟交换机网络接口 `vsw0`。
- 可以使用 `ifconfig`(1M) 命令配置来宾域中的虚拟网络接口 `vnet0`。

基本上，虚拟交换机的行为与常规物理网络交换机相同，可在不同的系统（例如它连接到的来宾域、服务域和物理网络）之间交换网络包。

管理虚拟交换机

本节介绍将虚拟交换机添加到域、设置虚拟交换机选项以及移除虚拟交换机。

▼ 添加虚拟交换机

- **使用以下命令语法添加虚拟交换机。**

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

其中：

- `default-vlan-id=`*vlan-id* 指定虚拟交换机及其关联的虚拟网络设备隐式属于的默认虚拟局域网 (virtual local area network, VLAN)（以无标记模式）。它用作虚拟交换机和虚拟网络设备的默认端口 VLAN ID (port VLAN id, pvid)。如果不使用此选项，此属性的默认值为 1。通常，您不需要使用此选项。仅提供其作为更改默认值 1 的方式。有关更多信息，请参见第 116 页中的 “使用 VLAN 标记”。
- `pvid=`*port-vlan-id* 指定虚拟交换机需要是其成员的 VLAN（以无标记模式）。有关更多信息，请参见第 116 页中的 “使用 VLAN 标记”。
- `vid=`*vlan-id* 指定虚拟交换机需要是其成员的一个或多个 VLAN（以标记模式）。有关更多信息，请参见第 116 页中的 “使用 VLAN 标记”。
- `linkprop=phys-state` 指定虚拟设备是否根据底层物理网络设备报告其链路状态。在命令行上指定 `linkprop=phys-state` 时，虚拟设备链路状态可反映物理链路状态。默认情况下，虚拟设备链路状态不反映物理链路状态。

  指定此选项以使用基于链路的 IPMP。请参见第 112 页中的 “在 Logical Domains 虚拟网络中使用基于链路的 IPMP”。
- `mac-addr=`*num* 是此交换机要使用的 MAC 地址。数字必须使用标准八位组表示法，例如 80:00:33:55:22:66。如果不指定 MAC 地址，将自动从分配给 Logical Domains Manager 的公共 MAC 地址范围为交换机分配一个地址。有关更多信息，请参见第 105 页中的 “自动或手动分配 MAC 地址”。
- `net-dev=`*device* 是到网络设备的路径，此交换机通过该设备运行。
- `mode=sc` 启用在 Logical Domains 环境中优先处理 Oracle Solaris Cluster 心跳包的虚拟网络支持。像 Oracle Solaris Cluster 这样的应用程序需要确保高优先级心跳包不被拥塞的虚拟网络和交换机设备丢弃。此选项排定 Oracle Solaris Cluster 心跳帧的优先次序并确保它们以可靠的方式传送。

  在 Logical Domains 环境中运行 Oracle Solaris Cluster 并且将来宾域用作 Oracle Solaris Cluster 节点时，必须设置此选项。没有在来宾域中运行 Oracle Solaris Cluster 软件时，请**不要**设置此选项，因为可能会影响虚拟网络性能。
- `mtu=`*size* 指定虚拟交换机设备的最大传输单元 (maximum transmission unit, MTU)。有效值是 1500 到 16000。
- `id=`*switch-id* 是新虚拟交换机设备的 ID。默认情况下将自动生成 ID 值，如果需要匹配 OS 中的现有设备名称，则设置此属性。请参见第 103 页中的 “虚拟设备标识符和网络接口名称”。
- *vswitch-name* 是要作为服务导出的交换机的唯一名称。客户机（网络）可以连接到此服务。

- *ldom* 指定要在其中添加虚拟交换机的逻辑域。

▼ 设置现有虚拟交换机的选项

- **使用以下命令语法设置已经存在的虚拟交换机的选项。**

  # ldm set-vsw [pvid=[*port-vlan-id***]] [vid=[***vlan-id1,vlan-id2,...***]] [mac-addr=***num***] [linkprop=[phys-state]] [net-dev=[***device***]] [mode=[sc]] [mtu=[***size***]]** *vswitch-name*

  其中：

  - `mode=`（留空）停止 Oracle Solaris Cluster 心跳包的特殊处理。
  - 否则，命令参数与第 100 页中的 “添加虚拟交换机”中所述相同。

▼ 移除虚拟交换机

- **使用以下命令语法移除虚拟交换机。**

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

  其中：

  - `-f` 尝试强制移除虚拟交换机。移除可能失败。
  - *vswitch-name* 是要作为服务移除的交换机的名称。

管理虚拟网络设备

本节介绍将虚拟网络设备添加到域、设置现有虚拟网络设备的选项以及移除虚拟网络设备。

▼ 添加虚拟网络设备

- **使用以下命令语法添加虚拟网络设备。**

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

其中：

- `mac-addr=`*num* 是此网络设备的 MAC 地址。数字必须使用标准八位组表示法，例如 `80:00:33:55:22:66`。有关更多信息，请参见第 105 页中的 “自动或手动分配 MAC 地址”。
- `mode=hybrid` 请求系统在此 `vnet` 上使用 NIU 混合 I/O（如有可能）。如果不可能，则系统恢复为虚拟 I/O。如果在活动 `vnet` 上进行设置，则此混合模式将被视为延迟重新配置。有关更多信息，请参见第 118 页中的 “使用 NIU 混合 I/O”。
- `pvid=`*port-vlan-id* 指定虚拟网络设备需要是其成员的 VLAN（以无标记模式）。有关更多信息，请参见第 116 页中的 “使用 VLAN 标记”。
- `linkprop=phys-state` 指定虚拟网络设备是否根据底层物理网络设备报告其链路状态。在命令行上指定 `linkprop=phys-state` 时，虚拟网络设备链路状态可反映物理链路状态。默认情况下，虚拟网络设备链路状态不反映物理链路状态。

  指定此选项以使用基于链路的 IPMP。请参见第 112 页中的 “在 Logical Domains 虚拟网络中使用基于链路的 IPMP”。
- `vid=`*vlan-id* 指定虚拟网络设备需要是其成员的一个或多个 VLAN（以标记模式）。有关更多信息，请参见第 116 页中的 “使用 VLAN 标记”。
- `mtu=`*size* 指定虚拟网络设备的最大传输单元 (maximum transmission unit, MTU)。有效值是 1500 到 16000。
- `id=`*network-id* 是新虚拟网络设备的 ID。默认情况下将自动生成 ID 值，如果需要匹配 OS 中的现有设备名称，则设置此属性。请参见第 103 页中的 “虚拟设备标识符和网络接口名称”。
- *if-name* 是逻辑域的唯一接口名称，它被分配到此虚拟网络设备实例，供在后续 `ldm set-vnet` 或 `ldm rm-vnet` 命令上引用。
- *vswitch-name* 是要连接到的现有网络服务（虚拟交换机）的名称。
- *ldom* 指定要向其添加虚拟网络设备的逻辑域。

▼ 设置现有虚拟网络设备的选项

- **使用以下命令语法设置已经存在的虚拟网络设备的选项。**

```
# 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
```

其中：

- `mode=`（留空）禁用 NIU 混合 I/O。
- *if-name* 为接口名称，它是分配给要设置的虚拟网络设备的唯一名称。
- *ldom* 指定要从中移除虚拟网络设备的逻辑域。
- 否则，命令参数与第 101 页中的 “添加虚拟网络设备”中所述相同。

▼ 移除虚拟网络设备

- **使用以下命令语法移除虚拟网络设备。**

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

  其中：

  - `-f` 尝试强制从逻辑域中移除虚拟网络设备。移除可能失败。
  - *if-name* 为接口名称，它是分配给要移除的虚拟网络设备的唯一名称。
  - *ldom* 指定要从中移除虚拟网络设备的逻辑域。

虚拟设备标识符和网络接口名称

将虚拟交换机或虚拟网络设备添加到域时，可以通过设置 `id` 属性来指定其设备编号。

 # ldm add-vsw [id=*switch-id***]** *vswitch-name ldom*
 # ldm add-vnet [id=*network-id***]** *if-name vswitch-name ldom*

域的每个虚拟交换机和虚拟网络设备都具有唯一的设备编号，该编号在绑定域时分配。如果使用显式设备编号（通过设置 `id` 属性）添加虚拟交换机或虚拟网络设备，将使用指定的设备编号。否则，系统将自动指定最低的可用设备编号。在这种情况下，分配的设备编号取决于将虚拟交换机或虚拟网络设备添加到系统的方式。当域被绑定时，在 `ldm list-bindings` 命令的输出中可以看到最终分配给虚拟交换机或虚拟网络设备的设备编号。

以下示例显示 `primary` 域有一个虚拟交换机 `primary-vsw0`。此虚拟交换机的设备编号是 `0`（`switch@0`）。

```
primary# ldm list-bindings primary
...
VSW
    NAME         MAC               NET-DEV DEVICE   DEFAULT-VLAN-ID PVID VID MTU MODE
    primary-vsw0 00:14:4f:fb:54:f2 nxge0   switch@0 1                 1    5,6 1500
...
```

以下示例显示 `ldg1` 域有两个虚拟网络设备：`vnet` 和 `vnet1`。`vnet` 设备的设备编号是 `0`（`network@0`），`vnet1` 设备的设备编号是 `1`（`network@1`）。

```
primary# ldm list-bindings ldg1
...
NETWORK
    NAME  SERVICE              DEVICE    MAC               MODE   PVID VID MTU
    vnet  primary-vsw0@primary network@0 00:14:4f:fb:e0:4b hybrid 1        1500
    ...
    vnet1 primary-vsw0@primary network@1 00:14:4f:f8:e1:ea        1        1500
...
```

当含交换机的域正在运行 Oracle Solaris OS 时，虚拟交换机具有网络接口 `vsw`*N*。但是，虚拟交换机的网络接口编号 *N* 不必与虚拟交换机的设备编号 *n* 相同。

同样，当含虚拟网络设备的域正在运行 Oracle Solaris OS 时，虚拟网络设备具有网络接口 vnet*N*。但是，虚拟网络设备的网络接口编号 *N* 不必与虚拟网络设备的设备编号 *n* 相同。

注意 – Oracle Solaris OS 根据设备编号保留网络接口名称和虚拟交换机或虚拟网络之间的映射。如果未将设备编号显式分配给虚拟交换机或虚拟网络设备，当域被取消绑定并且之后再次绑定时，其设备编号可以更改。在这种情况下，由域中正在运行的 OS 分配的网络接口名称也可以更改并改变系统的现有配置。可能会出现这种情况，例如，从域配置中移除虚拟交换机或虚拟网络接口时。

您不能使用 `ldm list-*` 命令直接确定与虚拟交换机或虚拟网络设备相对应的 Oracle Solaris OS 网络接口名称。但是，可以使用 `ldm list -l` 命令输出和 Oracle Solaris OS 的 `/devices` 下的条目的组合来获取此信息。

▼ 查找 Oracle Solaris OS 网络接口名称

在此示例过程中，来宾域 `ldg1` 包含两个虚拟网络设备：`net-a` 和 `net-c`。要查找与 `net-c` 对应的 `ldg1` 中的 Oracle Solaris OS 网络接口名称，请执行以下操作：此示例还显示了查找虚拟交换机而不是虚拟网络设备的网络接口名称的不同之处。

1 使用 `ldm` 命令查找 `net-c` 的虚拟网络设备编号。

```
# ldm list -l ldg1
...
NETWORK
NAME         SERVICE                     DEVICE       MAC
net-a        primary-vsw0@primary        network@0    00:14:4f:f8:91:4f
net-c        primary-vsw0@primary        network@2    00:14:4f:f8:dd:68
...
```

`net-c` 的虚拟网络设备编号是 2 (`network@2`)。

要确定虚拟交换机的网络接口名称，请查找虚拟交换机设备编号，*n* 以 `switch@`*n* 表示。

2 要查找 `ldg1` 上的相应网络接口，请登录 `ldg1` 并在 `/devices` 下查找此设备编号的条目。

```
# uname -n
ldg1
# find /devices/virtual-devices@100 -type c -name network@2\*
/devices/virtual-devices@100/channel-devices@200/network@2:vnet1
```

网络接口名称是冒号后面的条目部分，即 `vnet1`。

要确定虚拟交换机的网络接口名称，请将 `-name` 选项的参数替换为 `virtual-network-switch@`*n*`\*`。然后，查找具有名称 `vsw`*N* 的网络接口。

3 **激活 (plumb) `vnet1` 以确定其是否具有 MAC 地址 `00:14:4f:f8:dd:68`，如步骤 1 中 `net-c` 的 `ldm list -l` 输出中所示。**

```
# ifconfig vnet1
vnet1: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
          inet 0.0.0.0 netmask 0
          ether 0:14:4f:f8:dd:68
```

自动或手动分配 MAC 地址

您必须有足够的介质访问控制 (media access control, MAC) 地址，以便分配给计划使用的一定数目的逻辑域、虚拟交换机和虚拟网络。可以让 Logical Domains Manager 自动为逻辑域、虚拟网络 (`vnet`) 和虚拟交换机 (`vsw`) 分配 MAC 地址，也可以从您自己的已分配 MAC 地址池中手动分配 MAC 地址。设置 MAC 地址的 `ldm` 子命令是 `add-domain`、`add-vsw`、`set-vsw`、`add-vnet` 和 `set-vnet`。如果您未在这些子命令中指定 MAC 地址，Logical Domains Manager 会自动分配一个地址。

让 Logical Domains Manager 分配 MAC 地址的优势是，它能够利用专用于逻辑域的 MAC 地址块。此外，Logical Domains Manager 可以检测并防止 MAC 地址与同一子网中其他 Logical Domains Manager 实例发生冲突。这样，您就不必手动管理您的 MAC 地址池。

创建逻辑域或将网络设备配置到域中时，即会分配 MAC 地址。此外，除非删除了设备或逻辑域本身，否则此地址分配是持久性的。

分配给 Logical Domains 的 MAC 地址范围

已经为 Logical Domains 分配了以下 512K MAC 地址块：

`00:14:4F:F8:00:00 ~ 00:14:4F:FF:FF:FF`

较低的 256K 地址由 Logical Domains Manager 用于**自动** *MAC* **地址分配**，并且您**不能**手动请求此范围中的地址：

`00:14:4F:F8:00:00 - 00:14:4F:FB:FF:FF`

您可以使用此范围内的上一半地址执行**手动** *MAC* **地址分配**：

`00:14:4F:FC:00:00 - 00:14:4F:FF:FF:FF`

自动分配算法

如果您未在创建逻辑域或网络设备时指定 MAC 地址，Logical Domains Manager 会自动为该逻辑域或网络设备分配 MAC 地址。为了获得此 MAC 地址，Logical Domains Manager 会重复尝试选择地址，然后检查是否存在潜在冲突。

在选择可能的地址之前，Logical Domains Manager 先确定专用于此目的的数据库中是否保存有最近释放的、自动分配的地址（请参见第 107 页中的 “释放的 MAC 地址”）。如果有，Logical Domains Manager 会从数据库中选择该候选地址。

如果没有最近释放的地址，则会从专为此目的留出的 256K 地址范围中随机选择 MAC 地址。随机选择 MAC 地址可减少重复的 MAC 地址被选作候选地址的几率。

随后将对照其他系统上的其他 Logical Domains Manager 检查所选的地址，以防止实际分配重复的 MAC 地址。第 106 页中的 “检测重复的 MAC 地址”中介绍了采用的算法。如果该地址已经被分配，Logical Domains Manager 将重复操作，选择另一个地址，再检查是否存在冲突。此过程将一直持续下去，直至找到没有分配的 MAC 地址，或超出了 30 秒的时间限制。如果达到了时间限制，则创建设备失败，并显示类似以下内容的错误消息。

```
Automatic MAC allocation failed.  Please set the vnet MAC address manually.
```

检测重复的 MAC 地址

为避免将同一个 MAC 地址分配给不同的设备，一个 Logical Domains Manager 将与其他系统上的其他 Logical Domains Manager 进行核实，方法是通过控制域的默认网络接口发送多址广播消息，其中包括 Logical Domains Manager 希望分配给设备的地址。尝试分配 MAC 地址的 Logical Domains Manager 会等待一秒钟的时间，以便返回响应。如果该 MAC 地址已经分配给另一个启用 Logical Domains 的系统上的不同设备，则该系统上的 Logical Domains Manager 会发送回包含相关 MAC 地址的响应。如果发出请求的 Logical Domains Manager 收到响应，它便知道所选的 MAC 地址已被分配，会选择其他地址，并重复上述操作。

默认情况下，这些多址广播消息仅发送到同一子网中的其他 Logical Domains Manager；默认生存时间 (time-to-live, TTL) 为 1。可以使用服务管理工具 (Service Management Facility, SMF) 属性 `ldmd/hops` 配置 TTL。

每个 Logical Domains Manager 负责：

- 侦听多址广播消息
- 跟踪分配给域的 MAC 地址
- 查找重复项
- 为避免产生重复项而作出响应

如果系统上的 Logical Domains Manager 由于某种原因关闭，则在 Logical Domains Manager 关闭期间可能会产生重复的 MAC 地址。

创建逻辑域或网络设备时会执行自动 MAC 分配，而且自动 MAC 分配会一直保持到该设备或逻辑域被删除。

注 – 创建逻辑域或网络设备及启动逻辑域时，会执行检查重复 MAC 地址的检测。

释放的 MAC 地址

与自动 MAC 地址相关联的逻辑域或设备被删除后，相应的 MAC 地址将保存到最近释放的 MAC 地址的数据库中，供以后在该系统上使用。之所以保存这些 MAC 地址，是为了防止动态主机配置协议 (Dynamic Host Configuration Protocol, DHCP) 服务器的 Internet 协议 (Internet Protocol, IP) 地址耗尽。如果 DHCP 服务器分配 IP 地址，这些服务器会在一段时间（租用时间）内都执行 IP 地址的分配与释放。租用期限通常配置为较长的一段时间，一般是几小时或几天。如果创建和删除网络设备的频率相当高，而 Logical Domains Manager 不自动重用已分配的 MAC 地址，则已分配的 MAC 地址的数量会很快耗尽一个典型配置的 DHCP 服务器所拥有的 MAC 地址数量。

当要求 Logical Domains Manager 自动为逻辑域或网络设备获取 MAC 地址时，它将先查找已释放的 MAC 地址的数据库，以确定是否有先前分配的 MAC 地址可供重用。如果此数据库中具有可用的 MAC 地址，则运行重复 MAC 地址检测算法。如果该 MAC 地址自先前释放后没有分配给任何设备，则将重用该地址，并将其从数据库中删除。如果检测到冲突，则只是从数据库中删除该地址。Logical Domains Manager 随后将尝试数据库中的下一个地址，如果没有可用的地址，将随机选取一个新的 MAC 地址。

将网络适配器和 Logical Domains 结合使用

在逻辑域环境中，服务域中运行的虚拟交换机服务可以直接和与 GLDv3 兼容的网络适配器进行交互。虽然可以在这些系统中使用与 GLDv3 不兼容的网络适配器，但是虚拟交换机不能与这些网络适配器直接进行交互。有关如何使用与 GLDv3 不兼容的网络适配器的信息，请参见第 108 页中的 “针对 NAT 和路由配置虚拟交换机和服务域”。

有关使用链路聚合的信息，请参见第 121 页中的 “将链路聚合和虚拟交换机结合使用”。

▼ 确定网络适配器是否与 GLDv3 兼容

1. **使用 Oracle Solaris OS `dladm`(1M) 命令，其中，例如 `bge0` 是网络设备名称。**

```
# dladm show-link bge0
bge0            type: non-vlan   mtu: 1500      device: bge0
```

2. **在输出中查看 `type:`：**

   - 与 GLDv3 兼容的驱动程序的类型将是 `non-vlan` 或 `vlan`。
   - 与 GLDv3 不兼容的驱动程序的类型将是 `legacy`。

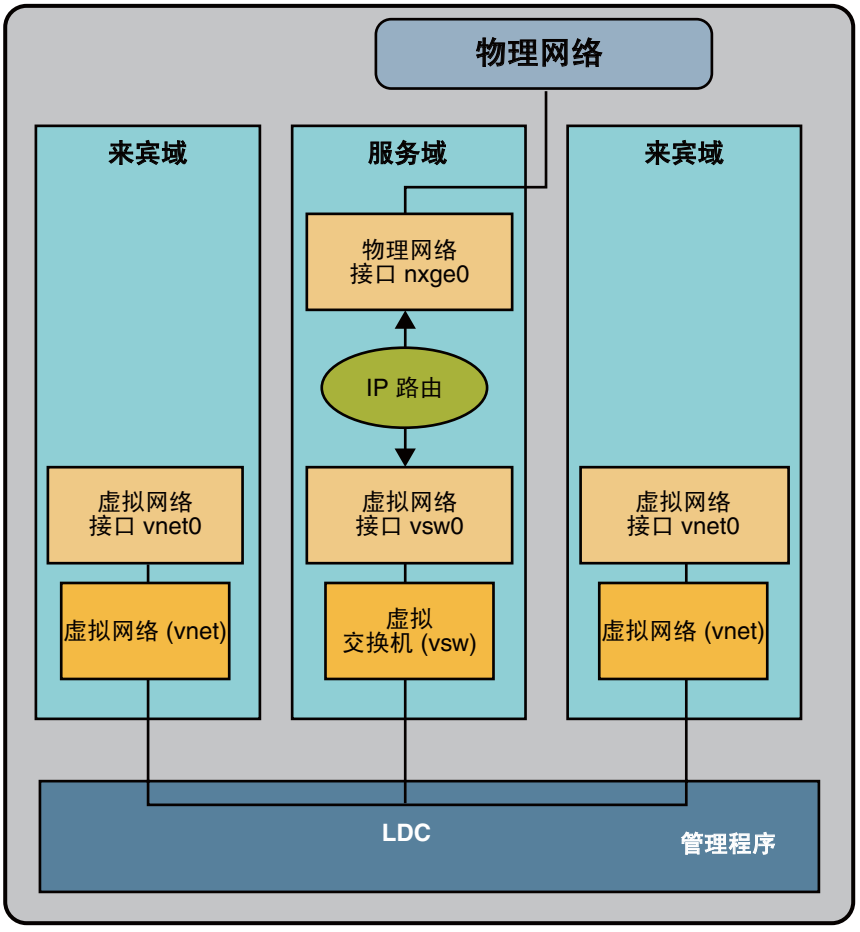
针对 NAT 和路由配置虚拟交换机和服务域

虚拟交换机 (vsw) 是第 2 层交换机，也可用作服务域中的网络设备。可以将虚拟交换机配置为仅充当不同逻辑域中的虚拟网络 (vnet) 设备之间的交换机，但不能通过物理设备与外界网络进行连接。在这种模式下，如果将 vsw 作为网络设备激活 (plumb) 并启用服务域中的 IP 路由，则虚拟网络可以通过将服务域用作路由器来与外界进行通信。当物理网络适配器不是兼容 GLDv3 的网络适配器时，要实现与域的外部连接，这种操作模式是非常重要的。

此配置的优势如下：

- 虚拟交换机不需要直接使用物理设备，即使底层设备不是兼容 GLDv3 的设备，也能提供外部连接。
- 此配置可以利用 Oracle Solaris OS 的 IP 路由和过滤功能。

图 8–2 虚拟网络路由



▼ 设置虚拟交换机以实现与域的外部连接

1 **创建与物理设备毫不相关的虚拟交换机。**

如果要分配地址，请确保虚拟交换机具有唯一的 MAC 地址。

```
primary# ldm add-vsw [mac-addr=xx:xx:xx:xx:xx:xx] primary-vsw0 primary
```

2 **将虚拟交换机作为域正在使用的物理网络设备之外的网络设备激活 (plumb)。**

有关激活 (plumb) 虚拟交换机的更多信息，请参见第 47 页中的 “将虚拟交换机配置为主接口”。

3 **如果需要，请为虚拟交换机设备配置 DHCP。**

有关为虚拟交换机设备配置 DHCP 的更多信息，请参见第 47 页中的 “将虚拟交换机配置为主接口”。

4 **如果需要，请创建 `/etc/dhcp.vsw` 文件。**

5 **在服务域中配置 IP 路由，并在所有域中设置所需的路由表。**

有关如何执行此操作的信息，请参阅《系统管理指南：IP 服务》中的“IPv4 网络上的包转发和路由”。

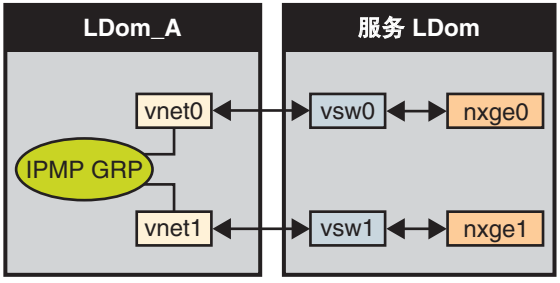
在 Logical Domains 环境中配置 IPMP

Logical Domains 1.3 发行版引入了对具有虚拟网络设备且基于链路的 IPMP 的支持。配置具有虚拟网络设备的 IPMP 组时，请将该组配置为使用基于链路的检测。如果使用 Oracle VM Server for SPARC (Logical Domains) 软件的较旧版本，则只能对虚拟网络设备配置基于探测的检测。

在域中将虚拟网络设备配置到 IPMP 组中

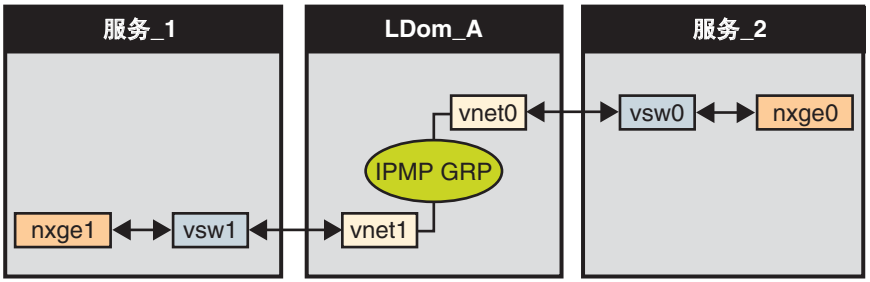
下图显示了两个虚拟网络（`vnet1` 和 `vnet2`），它们分别连接到服务域中的不同虚拟交换机实例（`vsw0` 和 `vsw1`），而这两个虚拟交换机又使用两个不同的物理接口（`nxge0` 和 `nxge1`）。如果服务域中发生物理链路故障，则绑定到此物理设备的虚拟交换机设备会检测到链路故障。然后，虚拟交换机设备将故障传播到绑定到此虚拟交换机的相应虚拟网络设备。虚拟网络设备将此链路事件的通知发送到来宾 `LDom_A` 中的 IP 层，从而导致故障转移到 IPMP 组中的其他虚拟网络设备。

图 8–3 两个连接到不同虚拟交换机实例的虚拟网络



在逻辑域中，通过将每个虚拟网络设备（`vnet0` 和 `vnet1`）连接到不同服务域中的虚拟交换机实例（如下图所示），可获得更高的可靠性。这种情况下，除了物理网络故障外，`LDom_A` 可以检测到虚拟网络故障，并在服务域崩溃或关闭后触发故障转移。

图 8–4 连接到不同服务域的各个虚拟网络设备

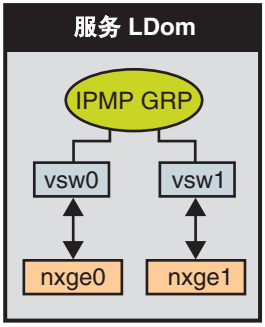


有关如何配置并使用 IPMP 组的更多信息，请参阅 Oracle Solaris 10 《系统管理指南：IP 服务》。

在服务域中配置并使用 IPMP

可以在服务域中配置 IPMP，方法是将虚拟交换机接口配置到组中。下图显示了两个绑定到两个不同物理设备的虚拟交换机实例（`vsw0` 和 `vsw1`）。随后可以激活 (plumb) 这两个虚拟交换机接口并将其配置到 IPMP 组中。如果物理链路出现故障，则绑定到此物理设备的虚拟交换机设备会检测到链路故障。然后，虚拟交换机设备将此链路事件的通知发送到服务域中的 IP 层，从而导致故障转移到 IPMP 组中的其他虚拟交换机设备。

图 8–5 两个配置为属于 IPMP 组的虚拟交换机接口



在 Logical Domains 虚拟网络中使用基于链路的 IPMP

自 Logical Domains 1.3 起，虚拟网络设备和虚拟交换机设备支持网络堆栈的链路状态更新。默认情况下，虚拟网络设备会报告其虚拟链路（到虚拟交换机的 LDC）的状态。默认情况下将启用此设置，不需要您执行其他配置步骤。

有时可能需要检测物理网络链路状态更改。例如，如果已将物理设备分配给虚拟交换机，即使从虚拟网络设备到其虚拟交换机设备的链路是连通的，从服务域到外部网络的物理网络链路也可能断开。在这种情况下，可能需要获取物理链路状态并将其报告给虚拟网络设备及其堆栈。

`linkprop=phys-state` 选项可用于配置虚拟网络设备以及虚拟交换机设备的物理链路状态跟踪。如果启用此选项，则当虚拟设备（虚拟网络或虚拟交换机）作为域中的接口激活 (plumb) 时，它会根据物理链路状态报告其链路状态。您可以使用标准 Oracle Solaris 网络管理命令（例如 `dladm` 和 `ifconfig`）来检查链路状态。请参见 `dladm(1M)` 和 `ifconfig(1M)` 手册页。此外，链路状态也记录在 `/var/adm/messages` 文件中。

注 – 您可以在 Logical Domains 系统上同时运行链路状态无感知和链路状态感知 `vnet` 和 `vsw` 驱动程序。但是，如果打算配置基于链路的 IPMP，则必须安装链路状态感知驱动程序。如果打算启用物理链路状态更新，则将 `vnet` 和 `vsw` 驱动程序都升级到 Oracle Solaris 10 9/10 OS，并且至少运行版本 1.3 的 Logical Domains Manager。

▼ 配置物理链路状态更新

此过程显示如何启用虚拟网络设备的物理链路状态更新。

也可以按照类似的步骤操作并指定 `ldm add-vsw` 和 `ldm set-vsw` 命令的 `linkprop=phys-state` 选项来启用虚拟交换机设备的物理链路状态更新。

注 – 仅当虚拟交换机设备本身作为接口激活 (plumb) 时，才需要使用 `linkprop=phys-state` 选项。如果指定了 `linkprop=phys-state` 且物理链路断开，则虚拟网络设备会将其链路状态报告为断开，即使与虚拟交换机的连接是连通的也是如此。由于 Oracle Solaris OS 当前未提供接口来报告两种不同的链路状态，例如虚拟链路状态和物理链路状态，因此会发生这种情况。

---

1. **成为超级用户或承担等效角色。**

   角色包含授权和具有一定权限的命令。有关角色的更多信息，请参见《系统管理指南：安全性服务》中的“配置 RBAC（任务列表）”。

2. **启用虚拟设备的物理链路状态更新。**

您可以通过以下方式来启用虚拟网络设备的物理链路状态更新：

- 运行 `ldm add-vnet` 命令时，通过指定 `linkprop=phys-state` 创建虚拟网络设备。

  指定 `linkprop=phys-state` 选项可配置虚拟网络设备以获取物理链路状态更新并将其报告给堆栈。

  注 – 如果指定了 `linkprop=phys-state` 且物理链路断开（即使与虚拟交换机的连接是连通的），则虚拟网络设备会将其链路状态报告为**断开**。由于 Oracle Solaris OS 当前未提供接口来报告两种不同的链路状态，例如虚拟链路状态和物理链路状态，因此会发生这种情况。

  ```
  # ldm add-vnet linkprop=phys-state if-name vswitch-name ldom
  ```

  以下示例启用在逻辑域 `ldom1` 上连接到 `primary-vsw0` 的 `vnet0` 的物理链路状态更新。

  ```
  # ldm add-vnet linkprop=phys-state vnet0 primary-vsw0 ldom1
  ```

- 运行 `ldm set-vnet` 命令时，通过指定 `linkprop=phys-state` 修改现有虚拟网络设备。

  ```
  # ldm set-vnet linkprop=phys-state if-name ldom
  ```

  以下示例启用逻辑域 `ldom1` 上 `vnet0` 的物理链路状态更新：

  ```
  # ldm set-vnet linkprop=phys-state vnet0 ldom1
  ```

要禁用物理链路状态更新，请运行 `ldm set-vnet` 命令指定 `linkprop=`。

以下示例禁用逻辑域 `ldom1` 上 `vnet0` 的物理链路状态更新：

```
# ldm set-vnet linkprop= vnet0 ldom1
```

示例 8–1 配置基于链路的 IPMP

以下示例显示了如何在启用和不启用物理链路状态更新的情况下配置基于链路的 IPMP：

- 以下示例在域中配置两个虚拟网络设备。每个虚拟网络设备连接到服务域中的独立虚拟交换机设备，以使用基于链路的 IPMP。

  注 – 未在这些虚拟网络设备上配置测试地址。此外，使用 `ldm add-vnet` 命令创建这些虚拟网络设备时，您不需要执行其他配置操作。

  下列命令将虚拟网络设备添加到域。请注意，由于未指定 `linkprop=phys-state`，因此只监视与虚拟交换机链路的状态更改。

  ```
  # ldm add-vnet vnet0 primary-vsw0 ldom1
  # ldm add-vnet vnet1 primary-vsw1 ldom1
  ```

以下命令在来宾域中配置虚拟网络设备并将其分配给 IPMP 组。请注意，未在这些虚拟网络设备上配置测试地址，原因是正在使用基于链路的故障检测。

```
# ifconfig vnet0 plumb
# ifconfig vnet1 plumb
# ifconfig vnet0 192.168.1.1/24 up
# ifconfig vnet1 192.168.1.2/24 up
# ifconfig vnet0 group ipmp0
# ifconfig vnet1 group ipmp0
```

- 以下示例在域中配置两个虚拟网络设备。每个域连接到服务域中的独立虚拟交换机设备，以使用基于链路的 IPMP。虚拟网络设备也配置为获取物理链路状态更新。

```
# ldm add-vnet linkprop=phys-state vnet0 primary-vsw0 ldom1
# ldm add-vnet linkprop=phys-state vnet1 primary-vsw1 ldom1
```

注 – 虚拟交换机必须分配有物理网络设备，域才能成功绑定。如果域已绑定而没有为虚拟交换机分配物理网络设备，则 `ldm add-vnet` 命令将失败。

以下命令激活 (plumb) 虚拟网络设备并将其分配给 IPMP 组：

```
# ifconfig vnet0 plumb
# ifconfig vnet1 plumb
# ifconfig vnet0 192.168.1.1/24 up
# ifconfig vnet1 192.168.1.2/24 up
# ifconfig vnet0 group ipmp0
# ifconfig vnet1 group ipmp0
```

在 Logical Domains 1.3 之前的发行版中配置并使用 IPMP

在早于 1.3 的 Logical Domains 发行版中，虚拟交换机和虚拟网络设备不能执行链路故障检测。在这些发行版中，可以使用基于探测的 IPMP 设置网络故障检测和恢复。

在来宾域中配置 IPMP

如图 8–3 和图 8–4 中所示，可以将来宾域中的虚拟网络设备配置到 IPMP 组中。唯一的区别是，通过在虚拟网络设备上配置测试地址而使用基于探测的故障检测。有关配置基于探测的 IPMP 的更多信息，请参见《系统管理指南：IP 服务》。

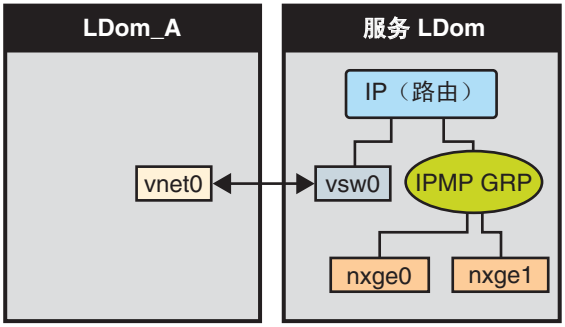
在服务域中配置 IPMP

在早于 1.3 的 Logical Domains 发行版中，虚拟交换机设备不能执行物理链路故障检测。在这种情况下，可以通过将服务域中的物理接口配置到 IPMP 组中来设置网络故障检测和恢复。要执行此操作，请在服务域中配置虚拟交换机，而不为其分配物理网络设备。即使用 `ldm add-vswitch` 命令创建虚拟交换机时，不要指定 `net-dev` (`net-dev=`) 属性的值。在服务域中激活 (Plumb) 虚拟交换机接口并将服务域本身配置为充当 IP 路由器。有关设置 IP 路由的信息，请参阅 Oracle Solaris 10 《系统管理指南：IP 服务》。

进行配置后，虚拟交换机将所有来自虚拟网络（目标是外部机器）的包发送到其 IP 层，而不是直接通过物理设备发送包。如果物理接口出现故障，IP 层会检测故障，并自动通过辅助接口对包进行重新路由。

由于物理接口被直接配置到 IPMP 组中，因此可以将该组设置为使用基于链路的检测或基于探测的检测。下图显示了两个配置为属于 IPMP 组的网络接口（`nxge0` 和 `nxge1`）。虚拟交换机实例 (`vsw0`) 已作为网络设备激活 (plumb)，以便向其 IP 层发送包。

图 8–6 两个配置为属于 IPMP 组的网络接口



▼ 为基于探测的 IPMP 配置主机路由

注 – 此过程仅适用于来宾域和早于 1.3 的发行版，其中仅支持基于探测的 IPMP。

如果没有为与 IPMP 接口对应的网络中的路由器配置显式路由，则需要配置一个或多个到目标系统的显式主机路由，以便 IPMP 基于探测的检测按预期方式工作。否则，探测检测可能无法检测网络故障。

- **配置主机路由。**

  ```
  # route add -host destination-IP gateway-IP -static
  ```

  例如：

  ```
  # route add -host 192.168.102.1 192.168.102.1 -static
  ```

  有关更多信息，请参阅《系统管理指南：IP 服务》中的“配置目标系统”。

使用 VLAN 标记

自 Oracle Solaris 10 10/08 OS 和 Logical Domains 1.1 软件发行版起，Logical Domains 网络基础结构中提供了 802.1Q VLAN 标记支持。

注 – Logical Domains 网络组件的所有早期发行版中均不支持标记的 VLAN。

虚拟交换机 (`vsw`) 和虚拟网络 (`vnet`) 设备支持根据虚拟局域网 (virtual local area network, VLAN) 标识符 (identifier, ID) 交换以太网包并处理必要的以太网帧标记或取消标记操作。

您可以在来宾域中的 `vnet` 设备上创建多个 VLAN 接口。可以使用 Oracle Solaris OS `ifconfig(1M)` 命令在虚拟网络设备上创建 VLAN 接口，用于在任何其他物理网络设备上配置 VLAN 接口的方法与之相同。Logical Domains 环境中的其他要求是，您必须使用 Logical Domains Manager CLI 命令将 `vnet` 分配给相应的 VLAN。有关 Logical Domains Manager CLI 命令的完整信息，请参阅 `ldm(1M)`。

同样，您可以在服务域中的虚拟交换机设备上配置 VLAN 接口。从 2 到 4094 的 VLAN ID 有效；VLAN ID 1 保留为 `default-vlan-id`。

在来宾域中创建 `vnet` 设备时，必须将其分配给所需的 VLAN，方法是使用 `ldm add-vnet` 命令的 `pvid=` 和 `vid=` 参数为此 `vnet` 指定一个端口 VLAN ID 和零或多个 VLAN ID。此操作可配置虚拟交换机在 Logical Domains 网络中支持多个 VLAN 并在网络中使用 MAC 地址和 VLAN ID 交换包。

同样，`vsw` 设备本身应属于的所有 VLAN 作为网络接口激活 (plumb) 时，必须使用 `ldm add-vsw` 命令的 `pvid=` 和 `vid=` 参数在 `vsw` 设备中进行配置。

您可以使用 `ldm set-vnet` 或 `ldm set-vsw` 命令更改设备所属的 VLAN。

端口 VLAN ID (PVID)

PVID 指示虚拟网络设备需要是其成员的 VLAN（以无标记模式）。在这种情况下，`vsw` 设备通过 PVID 指定的 VLAN 为 `vnet` 设备提供所需的帧标记或取消标记。虚拟网络中无标记的所有出站帧由虚拟交换机使用其 PVID 进行标记。使用此 PVID 标记的入站帧由虚拟交换机取消标记，然后再将其发送给 `vnet` 设备。因此，将 PVID 分配给 `vnet` 有以下暗示：对于 PVID 指定的 VLAN，虚拟交换机上的相应虚拟网络端口被标记为无标记。一个 `vnet` 设备只能有一个 PVID。

在不使用 VLAN ID 的情况下使用 `ifconfig(1M)` 命令配置相应的虚拟网络接口，并且仅使用其设备实例时，接口会被隐式分配给虚拟网络的 PVID 指定的 VLAN。

例如，如果您使用以下命令激活 (plumb) `vnet` 实例 `0`，并且如果已将 `vnet` 的 `pvid=` 参数指定为 `10`，则 `vnet0` 接口将被隐式分配为属于 VLAN 10。

```
# ifconfig vnet0 plumb
```

VLAN ID (VID)

VID 指示虚拟网络设备或虚拟交换机需要是其成员的 VLAN（以无标记模式）。虚拟网络设备通过其 VID 指定的 VLAN 发送和接收标记的帧。虚拟交换机在虚拟网络设备和外部网络之间传递使用指定的 VID 标记的所有帧。

▼ 为虚拟交换机和虚拟网络设备分配 VLAN

1 将虚拟交换机 (vsw) 分配给两个 VLAN。

例如，将 VLAN 21 配置为无标记并将 VLAN 20 配置为标记。将虚拟网络 (vnet) 分配给三个 VLAN。将 VLAN 20 配置为无标记并将 VLAN 21 和 22 配置为标记。

```
# ldm add-vsw net-dev=nxge0 pvid=21 vid=20 primary-vsw0 primary
# ldm add-vnet pvid=20 vid=21,22 vnet01 primary-vsw0 ldom1
```

2 激活 (Plumb) VLAN 接口。

本示例假定在域中这些设备的实例编号是 `0`，并且 VLAN 映射到这些子网：

VLAN	子网
20	192.168.1.0（网络掩码：255.255.255.0）
21	192.168.2.0（网络掩码：255.255.255.0）
22	192.168.3.0（网络掩码：255.255.255.0）

a. 在服务 (primary) 域中激活 (Plumb) VLAN 接口。

```
primary# ifconfig vsw0 plumb
primary# ifconfig vsw0 192.168.2.100 netmask 0xffffff00 broadcast + up
primary# ifconfig vsw20000 plumb
primary# ifconfig vsw20000 192.168.1.100 netmask 0xffffff00 broadcast + up
```

b. 在来宾 (ldom1) 域中激活 (Plumb) VLAN 接口。

```
ldom1# ifconfig vnet0 plumb
ldom1# ifconfig vnet0 192.168.1.101 netmask 0xffffff00 broadcast + up
ldom1# ifconfig vnet21000 plumb
ldom1# ifconfig vnet21000 192.168.2.101 netmask 0xffffff00 broadcast + up
ldom1# ifconfig vnet22000 plumb
ldom1# ifconfig vnet22000 192.168.3.101 netmask 0xffffff00 broadcast + up
```

有关如何在 Oracle Solaris OS 中配置 VLAN 接口的更多信息，请参阅《系统管理指南：IP 服务》中的“管理虚拟局域网”。

▼ 安装服务器位于 VLAN 中时安装来宾域

通过网络 (JumpStart) 安装来宾域且安装服务器位于 VLAN 中时要小心。将与安装服务器关联的 VLAN ID 指定为虚拟网络设备的 PVID，并且不要为此虚拟网络设备配置任何标记的 VLAN (`vid`)。您必须执行此操作，因为 OBP 不识别 VLAN 且不能处理 VLAN 标记的网络包。网络安装期间，虚拟交换机处理发往和发自来宾域的包的取消标记和标记。网络安装完成且 Oracle Solaris OS 引导后，您可以配置在此 VLAN 中标记虚拟网络设备。然后，可以在标记模式下将虚拟网络设备添加到其他 VLAN。

有关使用 JumpStart 安装来宾域的信息，请参见第 54 页中的 “在来宾域上执行 JumpStart 操作”。

1. **在无标记模式下初始配置网络设备。**

   例如，如果安装服务器位于 VLAN 21 中，则按如下所示初始配置虚拟网络：

   ```
   primary# ldm add-vnet pvid=21 vnet01 primary-vsw0 ldom1
   ```

2. **安装完成且 Oracle Solaris OS 引导后，在标记模式下配置虚拟网络。**

   ```
   primary# ldm set-vnet pvid= vid=21, 22, 23 vnet01 primary-vsw0 ldom1
   ```

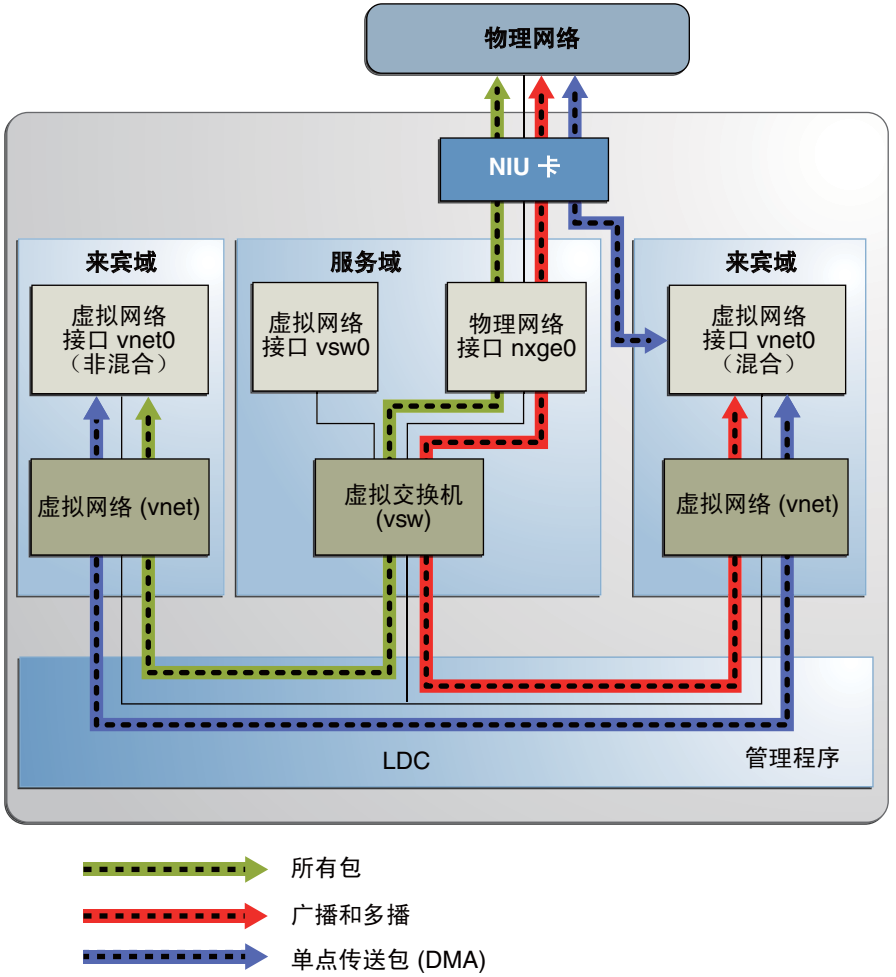
使用 NIU 混合 I/O

虚拟 I/O 框架可实施**混合** I/O 模型以改进功能和性能。混合 I/O 模型组合了直接 I/O 和虚拟化 I/O，以允许将 I/O 资源灵活部署到虚拟机。当直接 I/O 没有为虚拟机提供完整功能，或者不能为虚拟机持续提供直接 I/O 时，该模型尤其有用。这可能是资源可用性或虚拟机迁移引起的。混合 I/O 体系结构完全适用于 Sun UltraSPARC T2 和 SPARC T3 平台上的网络接口单元 (Network Interface Unit, NIU)。NIU 是在芯片上集成的网络 I/O 接口。使用此体系结构可以将直接内存访问 (Direct Memory Access, DMA) 资源动态分配到虚拟网络设备，从而为域中的应用程序提供稳定的性能。

NIU 混合 I/O 适用于 Sun UltraSPARC T2 和 SPARC T3 平台。此功能由可选混合模式启用，可提供给虚拟网络 (`vnet`) 设备，其中 DMA 硬件资源已借给来宾域中的 `vnet` 设备以改进性能。在混合模式中，来宾域中的 vnet 设备可以使用 DMA 硬件资源直接在外部网络和来宾域之间发送和接收单播通信。到同一系统中其他来宾域的广播或多播通信和单播通信继续使用虚拟 I/O 通信机制进行发送。

注 – NIU 混合 I/O 不适用于 UltraSPARC T2 Plus 平台。

图 8–7 混合虚拟网络



混合模式仅适用于与配置为使用 NIU 网络设备的虚拟交换机 (vsw) 关联的 vnet 设备。由于可共享的 DMA 硬件资源有限，因此每个 vsw 最多只能有三个 vnet 设备可以在给定时间内分配 DMA 硬件资源。如果超过三个 vnet 设备启用了混合模式，则根据先到先得原则完成分配。由于系统中有两个 NIU 网络设备，因此两个不同的虚拟交换机上一共可以有六个分配了 DMA 硬件资源的 vnet 设备。

下面是使用此功能时需要注意的要点：

- 仅将 vnet 设备的混合模式选项视为建议。这意味着仅当 DMA 资源可用且设备能够使用它们时才对其进行分配。
- Logical Domains Manager CLI 命令不会验证混合模式选项；也就是说，可以在任何 vnet 或任何数量的 vnet 设备上设置混合模式。

- 来宾域和服务域至少需要运行 Oracle Solaris 10 10/08 OS。
- 每个 vsw 最多只能有三个 vnet 设备可以在给定时间内借用 DMA 硬件资源。由于有两个 NIU 网络设备，因此一共可以有六个借用 DMA 硬件资源的 vnet 设备。

注 – 仅为每个 vsw 的三个 vnet 设备设置混合模式，以便保证它们分配到 DMA 硬件资源。

- 默认情况下为 vnet 设备禁用混合模式。需要使用 Logical Domains Manager CLI 命令显式启用它。请参见第 120 页中的 “启用混合模式”。

  （有关更多详细信息，请参阅 ldm(1M) 手册页。）
- 来宾域处于活动状态时不能动态更改混合模式选项。
- 仅当 vnet 设备处于活动状态，即在来宾域中激活 (plumb) 时，才分配 DMA 硬件资源。
- NIU 10 千兆位以太网驱动程序 (nxge) 用于 NIU 卡。同一驱动程序也用于其他 10 千兆位网卡。但是，NIU 混合 I/O 功能仅适用于 NIU 网络设备。

▼ 配置虚拟交换机和 NIU 网络设备

1 **确定 NIU 网络设备。**

以下示例显示了 UltraSPARC T2 服务器上的输出：

```
# grep nxge /etc/path_to_inst
"/niu@80/network@0" 0 "nxge"
"/niu@80/network@1" 1 "nxge"
```

以下示例显示了 SPARC T3-1 服务器上的输出：

```
# grep nxge /etc/path_to_inst
"/niu@480/network@0" 0 "nxge"
"/niu@480/network@1" 1 "nxge"
```

2 **配置虚拟交换机。**

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

▼ 启用混合模式

- **例如，创建 vnet 设备时为其启用混合模式。**

```
# ldm add-vnet mode=hybrid vnet01 primary-vsw0 ldom01
```

▼ 禁用混合模式

- **例如，为 vnet 设备禁用混合模式。**

```
# ldm set-vnet mode= vnet01 ldom01
```

将链路聚合和虚拟交换机结合使用

自 Oracle Solaris 10 10/08 OS 和 Logical Domains 1.1 软件发行版起，可以配置虚拟交换机以使用链路聚合。链路聚合用作虚拟交换机的网络设备，以连接到物理网络。使用此配置，虚拟交换机可以利用 EEE 802.3ad 链路聚合标准提供的功能。此类功能包括增加带宽、负载平衡和故障转移。有关如何配置链路聚合的信息，请参见《系统管理指南：IP 服务》。

创建链路聚合后，您可以将其分配给虚拟交换机。进行此分配类似于将物理网络设备分配给虚拟交换机。使用 `ldm add-vswitch` 或 `ldm set-vswitch` 命令设置 `net-dev` 属性。

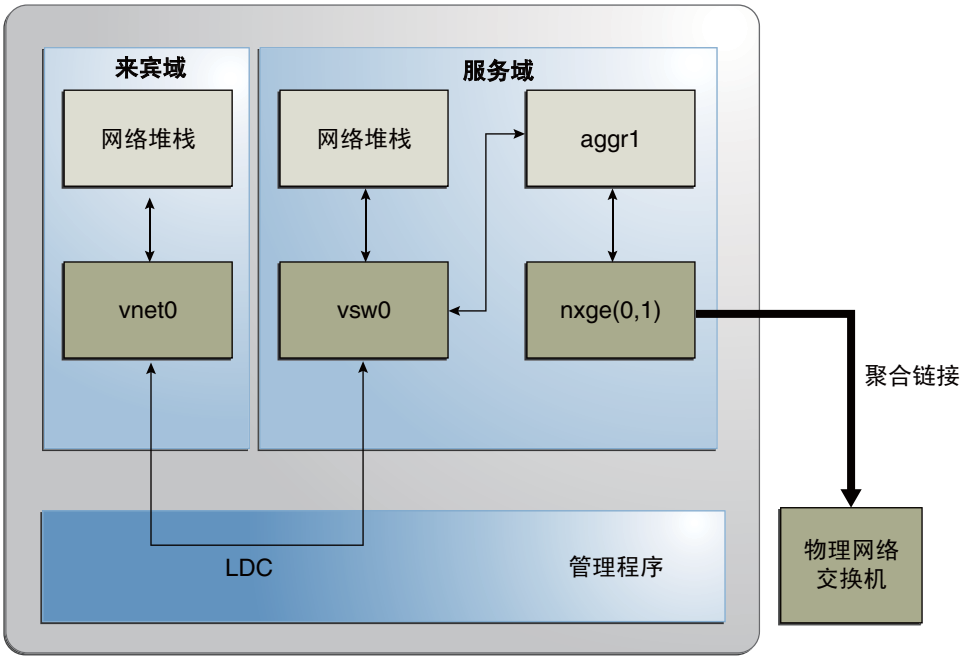
将链路聚合分配给虚拟交换机时，进出物理网络的通信会流经聚合。任何所需的负载平衡或故障转移由底层聚合框架透明处理。链路聚合对来宾域上以及绑定到使用聚合的虚拟交换机的虚拟网络 (`vnet`) 设备完全透明。

注 – 您不能将虚拟网络设备（`vnet` 和 `vsw`）组合到链路聚合中。

您可以在服务域中激活 (plumb) 并使用配置为使用链路聚合的虚拟交换机。请参见第 47 页中的 “将虚拟交换机配置为主接口”。

下图显示了配置为通过物理接口 `nxge0` 和 `nxge1` 使用聚合 `aggr1` 的虚拟交换机。

图 8–8 配置虚拟交换机以使用链路聚合



配置巨型帧

Logical Domains 虚拟交换机 (vsw) 和虚拟网络 (vnet) 设备现在可以支持有效负荷大小大于 1500 字节的以太网帧。此更改使这些驱动程序能够增加网络吞吐量。

▼ 配置虚拟网络和虚拟交换机设备以使用巨型帧

可通过指定虚拟交换机设备的最大传输单元 (maximum transmission unit, MTU) 启用巨型帧。在这种情况下，虚拟交换机设备和绑定到虚拟交换机设备的所有虚拟网络设备都使用指定的 MTU 值。

在某些情况下，您可以直接在虚拟网络设备上指定 MTU 值。如果虚拟网络设备所需的 MTU 值应小于虚拟交换机支持的值，则可以执行此操作。

注 – 在 Oracle Solaris 10 5/09 OS 上，必须将物理设备的 MTU 配置为与虚拟交换机的 MTU 相匹配。有关配置特定驱动程序的更多信息，请参见 Oracle Solaris 参考手册的 7D 部分中与此驱动程序对应的手册页。例如，要获取有关 nxge 驱动程序的信息，请参见 nxge(7D) 手册页。

---

1 **登录到控制域。**

2 **成为超级用户或承担等效角色。**

角色包含授权和具有一定权限的命令。有关角色的更多信息，请参见《系统管理指南：安全性服务》中的“配置 RBAC（任务列表）”。

3 **确定要用于虚拟网络的 MTU 值。**

您可以指定从 1500 到 16000 字节的 MTU 值。指定的 MTU 必须匹配分配给虚拟交换机的物理网络设备的 MTU。

4 **指定虚拟交换机设备或虚拟网络设备的 MTU 值。**

执行以下操作之一：

- 通过将 MTU 指定为 `mtu` 属性的值，在服务域中的新虚拟交换机设备上启用巨型帧。

  ```
  # ldm add-vsw mtu=value vswitch-name ldom
  ```

  除配置虚拟交换机外，此命令也会更新将要绑定到此虚拟交换机的每个虚拟网络设备的 MTU 值。

- 通过将 MTU 指定为 `mtu` 属性的值，在服务域中的现有虚拟交换机设备上启用巨型帧。

  ```
  # ldm set-vsw mtu=value vswitch-name
  ```

  除配置虚拟交换机外，此命令也会更新将要绑定到此虚拟交换机的每个虚拟网络设备的 MTU 值。

在少数情况下，您可以能需要使用 `ldm add-vnet` 或 `ldm set-vnet` 命令为虚拟网络设备指定与虚拟交换机的 MTU 值不同的 MTU 值。例如，如果您通过虚拟网络设备配置 VLAN 且最大的 VLAN MTU 小于虚拟交换机上的 MTU 值，则可以更改虚拟网络设备的 MTU 值。对于仅使用默认 MTU 值的域，可能不需要支持巨型帧的 `vnet` 驱动程序。但是，如果在域中虚拟网络设备已绑定到使用巨型帧的虚拟交换机，请确保 `vnet` 驱动程序支持巨型帧。

如果您在虚拟网络设备上使用 `ldm set-vnet` 命令指定 `mtu` 值，则不会将对虚拟交换机设备的 MTU 值的后续更新传播到此虚拟网络设备。要重新使虚拟网络设备从虚拟交换机设备获取 MTU 值，请运行以下命令：

```
# ldm set-vnet mtu= vnet-name ldom
```

请注意，为虚拟网络设备启用巨型帧会自动为分配给此虚拟网络设备的任何混合 IO 资源启用巨型帧。

在控制域中，Logical Domains Manager 将由 `ldm set-vsw` 和 `ldm set-vnet` 命令启动的 MTU 值作为延迟重新配置操作更新。要在控制域以外的域中更新 MTU，您必须先停止域，然后再运行 `ldm set-vsw` 或 `ldm set-vnet` 命令修改 MTU 值。

示例 8–2 在虚拟交换机和虚拟网络设备上配置巨型帧

- 以下示例显示了如何添加使用 MTU 值 9000 的新虚拟交换机设备。此 MTU 值将从虚拟交换机设备传播到所有客户机虚拟网络设备。

  首先，`ldm add-vsw` 命令创建 MTU 值为 9000 的虚拟交换机设备 `primary-vsw0`。请注意，网络设备 `nxge0` 的实例 0 被指定为 `net-dev` 属性的值。

  ```
  # ldm add-vsw net-dev=nxge0 mtu=9000 primary-vsw0 primary
  ```

  下一步，`ldm add-vnet` 命令将客户机虚拟网络设备添加到此虚拟交换机 `primary-vsw0`。请注意，虚拟网络设备的 MTU 是从此设备绑定到的虚拟交换机中隐式分配的。因此，`ldm add-vnet` 命令不需要您指定 `mtu` 属性的值。

  ```
  # ldm add-vnet vnet01 primary-vsw0 ldom1
  ```

  `ifconfig` 命令激活 (plumb) 服务域 `primary` 中的虚拟交换机接口。`ifconfig vsw0` 命令输出显示 `mtu` 属性的值是 9000。

  ```
  # ifconfig vsw0 plumb
  # ifconfig vsw0 192.168.1.100/24 up
  # ifconfig vsw0
  vsw0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 9000 index 5
          inet 192.168.1.100 netmask ffffff00 broadcast 192.168.1.255
          ether 0:14:4f:fa:0:99
  ```

  `ifconfig` 命令激活 (plumb) 来宾域 `ldom1` 中的虚拟网络接口。`ifconfig vnet0` 命令输出显示 `mtu` 属性的值是 9000。

  ```
  # ifconfig vnet0 plumb
  # ifconfig vnet0 192.168.1.101/24 up
  # ifconfig vnet0
  vnet0: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 9000 index 4
          inet 192.168.1.101 netmask ffffff00 broadcast 192.168.1.255
          ether 0:14:4f:f9:c4:13
  ```

- 以下示例显示如何使用 `ifconfig` 命令将接口的 MTU 更改为 4000。

  请注意，只能将接口的 MTU 更改为小于由 Logical Domains Manager 分配的设备的 MTU。已配置 VLAN 且每个 VLAN 接口需要不同的 MTU 时，此方法非常有用。

  ```
  # ifconfig vnet0 mtu 4000
  # ifconfig vnet0
  vnet0: flags=1201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS,FIXEDMTU>
  mtu 4000 index 4
          inet 192.168.1.101 netmask ffffff00 broadcast 192.168.1.255
          ether 0:14:4f:f9:c4:13
  ```

与 vnet 和 vsw 驱动程序的早期（巨型帧无感知）版本的兼容性

支持巨型帧的驱动程序可以与同一系统上不支持巨型帧的驱动程序交互操作。只要创建虚拟交换机时不启用巨型帧支持，就可以实现该互操作性。

注 – 如果与虚拟交换机关联的任何来宾域或服务域不使用支持巨型帧的 Logical Domains 驱动程序，请不要设置 `mtu` 属性。

可通过更改虚拟交换机的 `mtu` 属性的默认值 1500 来启用巨型帧。在这种情况下，早期驱动程序版本会忽略 `mtu` 设置并继续使用默认值。请注意，`ldm list` 输出将显示您指定的 MTU 值，而不是默认值。任何大于默认 MTU 的帧均不会被发送到这些设备，并被新的驱动程序丢弃。这种情况可能会导致仍然使用早期驱动程序版本的这些来宾域的网络行为不一致。这适用于客户机来宾域和服务域。

因此，启用巨型帧时，请确保已将 Logical Domains 网络中的所有虚拟设备升级为使用支持巨型帧的新驱动程序。此外，请确保至少升级到 Logical Domains 1.2，以便可以配置巨型帧。

◆ ◆ ◆ 第 9 章

9 迁移域

本章介绍如何将域从一台主机迁移到另一台主机。

本章包括以下内容：

- 第 127 页中的 “域迁移介绍”
- 第 128 页中的 “迁移操作概述”
- 第 128 页中的 “软件兼容性”
- 第 128 页中的 “对迁移操作进行验证”
- 第 129 页中的 “迁移域”
- 第 129 页中的 “迁移活动域”
- 第 133 页中的 “迁移绑定域或非活动域”
- 第 129 页中的 “执行模拟运行”
- 第 134 页中的 “监视正在进行的迁移”
- 第 135 页中的 “取消正在进行的迁移”
- 第 135 页中的 “从失败的迁移中恢复”
- 第 129 页中的 “执行非交互式迁移”
- 第 135 页中的 “迁移示例”

域迁移介绍

通过域迁移，可以将逻辑域从一台主机迁移到另一台主机。启动迁移的主机称为源计算机，要将域迁移到其中的主机称为目标计算机。同样，启动迁移后，在迁移进行过程中，要迁移的域称为源域，目标计算机上创建的域 shell 称为目标域。

迁移操作概述

源计算机上的 Logical Domains Manager 接受迁移域的请求，并建立与目标计算机上运行的 Logical Domains Manager 的安全网络连接。建立此连接后，即会发生迁移。迁移本身可分为几个不同的阶段。

阶段 1：与目标主机中运行的 Logical Domains Manager 建立连接后，有关源计算机和域的信息将传输到目标主机。此信息用于执行一系列检查，以确定迁移是否可行。这些检查会由于源域的状态不同而有所不同。例如，如果源域处于活动状态，将会执行一组与域处于绑定或非活动状态时不同的检查。

阶段 2：通过阶段 1 中的所有检查后，源计算机和目标计算机会为迁移做准备，源域将被暂停。在目标计算机上，将创建一个域来接收源域。

阶段 3：对于活动域，下一阶段是将域的所有运行时状态信息传输到目标。可从虚拟机管理程序检索此信息。在目标上，状态信息安装在虚拟机管理程序中。

阶段 4：移交。传输所有状态信息后，目标域恢复执行（如果源域在当时处于活动状态）时将发生移交，源域会被销毁。从此时起，目标域将成为正在运行的域的唯一版本。

软件兼容性

要使迁移能够发生，源计算机和目标计算机都必须运行如下兼容软件：

- 源计算机和目标计算机上的虚拟机管理程序都必须安装有兼容版本的固件。请参见《Oracle VM Server for SPARC 2.0 Release Notes》中的“Required Software to Enable Oracle VM Server for SPARC 2.0 Features”。
- 两台计算机上都必须运行兼容版本的 Logical Domains Manager。

注 – 迁移功能是随 Logical Domains 1.1 软件和相应固件首次发布的。有关平台的最新固件的信息，请参见《Oracle VM Server for SPARC 2.0 Release Notes》。

对迁移操作进行验证

由于迁移操作在两台计算机上执行，因此必须在源主机和目标主机上对用户进行验证。需特别指出的是，除超级用户以外的用户必须具有 `solaris.ldoms.read` 和 `solaris.ldoms.write` 授权。

通过适用于迁移的 `ldm` 命令行界面，用户可以指定可选的备用用户名以便在目标主机上进行验证。如果没有指定，将会使用执行迁移命令的用户的用户名。在这两种情况下，都会提示用户输入目标系统的密码，除非已使用 `-p` 选项启动非交互式迁移。

迁移域

可以使用 `ldm migrate-domain` 命令启动从一个系统到另一个系统的域迁移。

有关迁移选项和操作数的信息，请参见 `ldm(1M)` 手册页。

执行模拟运行

将 `-n` 选项提供给 `migrate-domain` 子命令时将执行迁移检查，但是不会迁移源域。任何未满足的要求都会报告为错误。这样，您可以在尝试执行真正迁移之前改正所有配置错误。

注 – 由于逻辑域的动态本性，可能会出现模拟运行成功而迁移失败的情况，反之亦然。

执行非交互式迁移

在 Logical Domains 1.3 软件发行版之前的版本中，迁移都是交互式操作。启动迁移时，系统会提示您输入用于目标计算机的密码。从 Logical Domains 1.3 开始，可以使用 `ldm migrate-domain -p` *filename* 命令启动非交互式迁移操作。

指定为 `-p` 选项的参数的文件名必须具有以下属性：

- 文件的第一行必须包含密码
- 密码必须是纯文本
- 密码的长度不得超过 256 个字符

密码结尾的换行符和第一行之后的所有行都会被忽略。

存储目标计算机密码的文件必须得到适当保护。如果打算以这种方式存储密码，请确保已设置文件权限，以便只有超级用户所有者或者特权用户可以读取或写入文件（400 或 600）。

迁移活动域

对于 Oracle VM Server for SPARC 2.0 软件，要使活动域的迁移能够发生，对源逻辑域、源计算机和目标计算机强加了一组特定要求和限制。以下几节针对各资源类型介绍这些要求和限制。

注 – 为源系统和目标系统上的 `primary` 域分配加密单元后，迁移操作的速度会加快。从 Logical Domains 1.3 开始，可以通过向源系统和目标系统的 `primary` 域添加更多虚拟 CPU 来加快迁移速度。

迁移活动域中的 CPU

下面是执行迁移时对 CPU 的要求和限制：

- 源计算机和目标计算机必须以相同频率运行相同的处理器类型。
- 目标计算机必须具有足够的空闲导线束，以适应正在由域使用的导线束数目。

在下列任何条件下都适用的其他要求和限制：

- 目标系统运行的是低于 Logical Domains Manager 版本 2.0 的版本。在这种情况下，可能会在迁移过程中看到以下消息：

  ```
  The target machine is running an older version of the domain
  manager that does not support the latest migration functionality.
  ```

- 源系统运行的是低于 Logical Domains Manager 版本 2.0 的版本。由于源域中的旧版 Logical Domains Manager 无法检测到软件不匹配情况，因此迁移将继续，而不会发出消息。
- 源域运行的是低于 Oracle Solaris 10 9/10 OS 的 Oracle Solaris OS 版本。在这种情况下，可能会在迁移过程中看到以下消息：

  ```
  Domain ldom is not running an operating system that is
  compatible with the latest migration functionality.
  ```

如果符合任何这些条件，则下列对 CPU 的要求和限制会适用：

- 必须为已迁移的域分配完整核心。如果源域中的导线束数目少于完整核心，则只有在已迁移的域重新引导后，附加的导线束才可用于任何域。
- 迁移后，将会对目标域禁用 CPU 动态重新配置 (dynamic reconfiguration, DR)，直到该域重新引导。重新引导后，CPU DR 会对该域可用。
- 目标系统必须具有足够的完整核心，可以完全自由地为已迁移的域提供所需数目的导线束。迁移后，如果已迁移的域仅使用了部分完整核心，则只有在已迁移的域重新引导后，附加的导线束才可用于任何域。

迁移活动域中的内存

目标计算机上必须具有足够的空闲内存，以适应源域的迁移。此外，迁移过程中必须维护下面一些属性：

- 必须可以创建相同数量的大小相同的内存块。
- 内存块的物理地址不需要匹配，但是迁移过程中必须维护相同的实际地址。

目标计算机必须具有足够的空闲内存，以适应源域的迁移。此外，目标计算机上可用内存的布局必须与源域的内存布局兼容，否则迁移将失败。

需特别指出的是，如果目标计算机上的内存已分为多个小的地址范围，但是源域需要一个大的地址范围，则迁移将失败。下面的示例对这种情况进行了说明。目标域在两个内存块中共有 2 GB 空闲内存：

```
# ldm list-devices memory
MEMORY
    PA                  SIZE
    0x108000000         1G
    0x188000000         1G
```

源域 `ldg-src` 也有 2 GB 空闲内存，但是这些内存分布在一个内存块中：

```
# ldm list -o memory ldg-src
NAME
ldg-src

MEMORY
    RA              PA              SIZE
    0x8000000       0x208000000     2G
```

在这种内存布局情况下，迁移将失败：

```
# ldm migrate-domain ldg-src dt212-239
Target Password:
Unable to bind 2G memory region at real address 0x8000000
Domain Migration of LDom ldg-src failed
```

注 – 迁移后，将会对目标域禁用内存动态重新配置 (dynamic reconfiguration, DR)，直到该域重新引导。重新引导完成后，将对该域重新启用内存 DR。

迁移活动域中的物理 I/O 设备

可以迁移与物理设备关联的虚拟设备。但是，无法迁移可以直接访问物理设备的域。例如，您无法迁移 I/O 域。

迁移活动域中的虚拟 I/O 设备

源域使用的所有虚拟 I/O (virtual I/O, VIO) 服务必须在目标计算机上可用。换而言之，必须满足下列条件：

- 源逻辑域中使用的每个逻辑卷也都必须在目标主机上可用，并且必须指向相同存储。

注意 – 如果目标上存在被源用作引导设备的逻辑卷，但是并没有指向相同存储，则迁移看起来好像会成功，但是该计算机并不可用，因为它无法访问其引导设备。必须停止域，改正配置问题，然后重新启动域。否则，该域可能会处于不一致的状态。

- 对于源域中的每个虚拟网络设备，目标主机上必须存在虚拟网络交换机，该交换机的名称与设备在源主机上连接到的虚拟网络交换机的名称相同。

  例如，如果源域中的 `vnet0` 连接到虚拟交换机服务名称 `switch-y`，则提供名为 `switch-y` 的虚拟交换机服务的目标主机上必须具有一个逻辑域。

  注 – 交换机不必连接到相同的网络也可发生迁移，但是，如果交换机没有连接到相同的网络，已迁移的域可能会出现联网问题。

  源域使用的自动分配范围内的 MAC 地址必须可供在目标主机上使用。
- 虚拟控制台集中器 (`vcc`) 服务必须在目标主机上存在，并至少具有一个空闲端口。迁移过程中会忽略显式控制台约束。目标域的控制台是通过使用目标域名称作为控制台组并使用控制域中第一个 `vcc` 设备上的任意可用端口创建的。如果与默认组名有冲突，迁移将失败。

迁移活动域中的 NIU 混合输入/输出

可以迁移使用 NIU 混合 I/O 资源的域。指定 NIU 混合 I/O 资源的约束不是逻辑域的硬性要求。如果将这样的域迁移到不具有可用 NIU 资源的计算机，该约束将得到保留，但是不会得到满足。

迁移活动域中的加密单元

从 Logical Domains 1.3 开始，如果某个具有绑定加密单元的来宾域运行的是支持加密单元动态重新配置 (dynamic reconfiguration, DR) 的操作系统，则可以迁移该来宾域。

下列 Oracle Solaris OS 版本支持加密单元 DR：

- Solaris 10 10/09 OS 或更高版本
- Oracle Solaris 10 5/08 OS 加修补程序 ID 142245-01 或更高版本

开始进行迁移时，Logical Domains Manager 会确定源域是否支持加密单元 DR。如果支持，Logical Domains Manager 会尝试从该域中删除所有加密单元。迁移完成后，会将加密单元重新添加到已迁移的域。

注 – 如果目标计算机上无法满足对加密单元的约束，迁移操作可能仍可成功完成。在这种情况下，该域最终可能会具有比迁移操作前更少的加密单元。

活动域中的延迟重新配置

源主机或目标主机上的任何活动的延迟重新配置操作都会阻止迁移启动。迁移正在进行时，延迟重新配置操作会受到阻止。

在活动域处于弹性模式时进行迁移

对于处于弹性模式的源计算机或目标计算机，不支持域迁移。如果当域处于性能模式时正在进行迁移，并且电源管理 (power management, PM) 策略已设置为弹性模式，则策略切换将延迟，直到迁移完成。如果在源计算机或目标计算机处于弹性模式时尝试域迁移，迁移命令会返回错误。

对其他域的操作

计算机上正在进行迁移时，可能会导致修改正迁移的域的状态或配置的任何操作都会被阻止。对该域本身的所有操作以及对计算机上其他域的操作（例如绑定或停止）都会被阻止。

迁移绑定域或非活动域

由于绑定域或非活动域不在迁移时执行，因此与迁移活动域时相比具有较少的限制。

绑定域的迁移要求目标可满足源域的 CPU、内存和 I/O 约束。否则，该迁移将失败。非活动域的迁移没有这样的要求。但是，稍后尝试进行绑定时，目标必须满足域的约束。否则，该域绑定将失败。

迁移绑定域或非活动域中的 CPU

可以在运行不同处理器类型的计算机和以不同频率运行的计算机之间迁移绑定域或非活动域。

来宾域中的 Oracle Solaris OS 映像必须支持目标计算机上的处理器类型。

迁移绑定域或非活动域中的虚拟输入/输出

对于非活动域，不会针对虚拟输入/输出 (virtual input/output, VIO) 约束执行检查。因此，不需要存在 VIO 服务器，迁移就可成功。与任何非活动域相同，绑定域时，VIO 服务器需要存在并处于可用状态。

迁移绑定域或非活动域中的 PCIe 端点设备

无法对配置有 PCIe 端点设备的 I/O 域执行域迁移。

有关直接 I/O (direct I/O, DIO) 功能的信息，请参见第 62 页中的 “分配 PCIe 端点设备”。

监视正在进行的迁移

迁移正在进行时，源域和目标域在状态输出中的显示有所不同。`ldm list` 命令的输出指示所迁移域的状态。

`FLAGS` 字段中的第六列显示以下值之一：

- 源域显示 `s` 以指示该域是迁移的源。
- 目标域显示 `t` 以指示该域是迁移的目标。
- 如果出现需要用户干预的错误，将显示 `e`。

以下内容显示 `ldg-src` 是迁移的源域：

```
# ldm list ldg-src
NAME       STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg-src    suspended  -n---s          1     1G       0.0%  2h 7m
```

以下内容显示 `ldg-tgt` 是迁移的目标域：

```
# ldm list ldg-tgt
NAME       STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg-tgt    bound      -----t  5000    1     1G
```

在完整格式的状态输出中，会显示有关迁移的其他信息。在源上，会显示操作完成的百分比以及目标主机和目标域的名称。同样，在目标上，会显示操作完成的百分比以及源主机和源域的名称。

示例 9–1 监视正在进行的迁移

```
# ldm list -o status ldg-src
NAME
ldg-src

STATUS
    OPERATION    PROGRESS    TARGET
```

示例 9–1 监视正在进行的迁移 （续）

```
    migration    17%         t5440-sys-2
```

取消正在进行的迁移

开始迁移后，如果通过 KILL 信号中断 `ldm` 命令，迁移将终止。目标域会被销毁，如果源域在当时处于活动状态，则会恢复。如果 `ldm` 命令的控制 shell 丢失，迁移将在后台继续进行。

还可以使用 `ldm cancel-operation` 命令从外部取消迁移操作。这将终止正在进行的迁移，源域会恢复为活动域。应从源系统启动 `ldm cancel-operation` 命令。在给定系统上，任何与迁移相关的命令都会影响从该系统启动的迁移操作。系统成为目标系统时，将无法控制迁移操作。

注 – 启动迁移后，暂停 `ldm(1M)` 进程不会暂停操作，这是因为目前影响迁移的是源计算机和目标计算机上的 Logical Domains Manager 守护进程 (`ldmd`)。`ldm` 进程等待来自 `ldmd` 的指示迁移已在返回前完成的信号。

从失败的迁移中恢复

如果在源完成将所有运行时状态信息发送到目标之后，但在目标可以确认域已恢复之前，失去网络连接，则迁移操作将终止，源将被置于错误状态。这表示需要通过用户交互来确定迁移是否已成功完成。在这种情况下，请执行以下步骤。

- 确定目标域是否已成功恢复。目标域将处于以下两种状态之一：
  - 如果迁移成功完成，目标域将处于正常状态。
  - 如果迁移失败，目标会清除并销毁目标域。
- 如果目标已恢复，可以放心地销毁处于错误状态的源域。如果目标不存在，则源域仍是域的主版本，必须对其进行恢复。为此，请在源计算机上执行取消命令。这将清除错误状态，并将源域恢复回到其初始状态。

迁移示例

示例 9–2 说明如何将名为 `ldg1` 的域迁移到名为 `t5440-sys-2` 的计算机。

示例 9–2 迁移来宾域

```
# ldm migrate-domain ldg1 t5440-sys-2
Target Password:
```

要在不提示输入目标密码的情况下执行此迁移，请使用以下命令：

示例 9–2 迁移来宾域 （续）

```
# ldm migrate-domain -p pfile ldg1 t5440-sys-2
```

-p 选项将文件名作为参数。指定的文件包含目标的超级用户密码。在此示例中，pfile 包含目标 t5440-sys-2 的密码。

示例 9–3 说明可以在迁移过程中重命名域。在此示例中，ldg-src 是源域，在迁移过程中，它在目标计算机 (t5440-sys-2) 上被重命名为 ldg-tgt。此外，还显式指定了目标计算机上的用户名 (root)。

示例 9–3 迁移和重命名来宾域

```
# ldm migrate ldg-src root@t5440-sys-2:ldg-tgt
Target Password:
```

示例 9–4 说明如果目标域不具有迁移支持，换而言之，如果运行的是早于版本 1.1 的 Logical Domains 版本，将出现故障消息样例。

示例 9–4 迁移故障消息

```
# ldm migrate ldg1 t5440-sys-2
Target Password:
Failed to establish connection with ldmd(1m) on target: t5440-sys-2
Check that the 'ldmd' service is enabled on the target machine and
that the version supports Domain Migration. Check that the 'xmpp_enabled'
and 'incoming_migration_enabled' properties of the 'ldmd' service on
the target machine are set to 'true' using svccfg(1M).
```

示例 9–5 说明如何在迁移正在进行时获取有关目标域的状态。在此示例中，源计算机为 t5440-sys-1。

示例 9–5 获取目标域状态

```
# ldm list -o status ldg-tgt
NAME
ldg-tgt

STATUS
    OPERATION    PROGRESS    SOURCE
    migration    55%         t5440-sys-1
```

示例 9–6 说明如何在迁移正在进行时获取有关源域的可解析状态。在此示例中，目标计算机为 t5440-sys-2。

示例 9–6 获取源域的可解析状态

```
# ldm list -o status -p ldg-src
VERSION 1.3
DOMAIN|name=ldg-src|
```

示例 9–6 获取源域的可解析状态 （续）

```
STATUS
|op=migration|progress=42|error=no|target=t5440-sys-2
```

◆ ◆ ◆ 第 10 章

10 管理资源

本章包含有关在 Oracle VM Server for SPARC 系统上执行资源管理的信息。

本章包括以下内容：

- 第 139 页中的 “资源重新配置”
- 第 140 页中的 “资源分配”
- 第 141 页中的 “CPU 分配”
- 第 144 页中的 “使用内存动态重新配置”
- 第 151 页中的 “使用电源管理”
- 第 154 页中的 “使用动态资源管理”
- 第 156 页中的 “列出域资源”

资源重新配置

运行 Oracle VM Server for SPARC 软件的系统可以配置资源（例如虚拟 CPU、虚拟 I/O 设备、加密单元和内存）。某些资源可以在正在运行的域上动态地进行配置，而其他一些资源则必须在停止的域上进行配置。如果无法在控制域上动态地配置资源，必须首先启动延迟重新配置。延迟重新配置会将配置活动推迟到控制域进行重新引导后。

动态重新配置

通过动态重新配置 (dynamic reconfiguration, DR)，可以在操作系统 (operating system, OS) 运行时添加或删除资源。对特定资源类型执行 DR 的功能取决于逻辑域中运行的 OS 是否具备相应支持。

支持对以下资源进行动态重新配置：

- **虚拟 CPU** – 在 Oracle Solaris 10 OS 的所有版本中均支持
- **虚拟 I/O 设备** – 在 Solaris 10 10/08 OS 及更高版本中支持
- **加密单元** – 在 Oracle Solaris 10 9/10 OS 及更高版本中支持

- **内存** – 从 Oracle VM Server for SPARC 2.0 发行版开始支持（请参见第 144 页中的 “使用内存动态重新配置”）
- **物理 I/O 设备** – 不支持

要使用 DR 功能，Logical Domains DR 守护进程 `drd` 必须在要更改的域中运行。请参见 `drd(1M)` 手册页。

延迟重新配置

与可以立即发生的 DR 操作不同，延迟重新配置操作在以下情况下生效：

- 下次重新引导 OS 之后
- 停止然后启动逻辑域之后

从 Logical Domains 1.2 软件开始，延迟重新配置操作仅限定于控制域。对于所有其他域，除非可以动态重新配置资源，否则必须停止域来修改配置。

从 Oracle VM Server for SPARC 2.0 软件开始，在执行资源配置操作之前，必须先在控制域上启动延迟重新配置。可以使用 `ldm start-reconf primary` 命令启动延迟重新配置。

如果控制域上正在执行延迟重新配置，则对控制域的其他重新配置请求将被延迟，直到重新引导控制域或停止然后启动控制域。此外，如果控制域中存在暂挂的延迟重新配置，则对其他逻辑域的重新配置请求会受到严格限制，而且会失败并显示相应的错误消息。

`ldm cancel-operation reconf` 命令可取消控制域上的延迟重新配置操作。可以使用 `ldm list-domain` 命令列出延迟重新配置操作。有关如何使用延迟重新配置功能的更多信息，请参见 `ldm(1M)` 手册页。

注 – 如果任何其他 `ldm remove-*` 命令已对虚拟 I/O 设备执行延迟重新配置操作，则无法使用 `ldm cancel-operation reconf` 命令。在这些情况下 `ldm cancel-operation reconf` 命令将失败。

资源分配

从 Oracle VM Server for SPARC 2.0 发行版开始，资源分配机制使用资源分配约束和提示在绑定时将资源分配到域。

资源分配约束是系统在向域分配资源时**必须**满足的硬性要求。如果不能满足该约束，资源分配和域绑定均将失败。

资源分配提示是系统在向域分配资源时**尝试**满足的软性要求。即使不能完全满足该提示，资源分配仍可成功，并且可以绑定域。如果系统不必满足要求也可以分配资源，则可能会发生这种情况。

CPU 分配

CPU 分配机制针对 CPU 资源使用以下约束和提示：

- **整体核心约束。**此约束指定基于指定的 CPU 核心数将虚拟 CPU 分配给域。系统必须能够分配指定数目的核心，还必须能够将这些分配了核心的所有虚拟 CPU 分配给域。如果系统无法分配指定数目的核心，则无法绑定域。
- **最大核心数约束。**此约束指定可以分配给绑定域或活动域的最大核心数。在域上设置整体核心约束后会自动启用此约束。在这种情况下，系统会将最大核心数自动设置为在域处于非活动状态时所配置的核心数。目前，无法独立于整体核心约束启用此约束，而且无法手动设置最大核心数。
- **核心关联性提示。**此提示请求基于相同的 CPU 核心或最少数目的 CPU 核心为域分配虚拟 CPU。系统会尽最大可能来实现此请求。**仅**当系统上没有足够的可用虚拟 CPU 时，才无法绑定域。

  默认情况下启用核心关联性提示，并且无法禁用它。

注 – 整体核心约束和核心关联性提示仅对虚拟 CPU 在核心上的位置进行寻址。它们不对核心在芯片上的位置或芯片在插槽上的位置进行寻址。

启用整体核心约束

当您指定要分配给域的核心数时，将会自动启用整体核心约束。默认情况下，需指定要分配给域的虚拟 CPU。只能在非活动域上而不能在绑定域或活动域上启用整体核心约束。在控制域上启用整体核心约束之前，必须先启动延迟重新配置。

使用 `ldm add-vcpu -c` *number*、`ldm set-vcpu -c` *number* 或 `ldm remove-vcpu -c` *number* 命令将 CPU 核心分配给域或从域中删除 CPU 核心。*number* 指定 CPU 核心的数量并启用整体核心约束。有关更多信息，请参见 `ldm(1M)` 手册页。

可以在之前已配置虚拟 CPU 的域上使用 `ldm add-vcpu -c` *number* 或 `ldm remove-vcpu -c` *number* 命令。在这种情况下，现有虚拟 CPU 数将自动转换为相应核心数。仅当现有虚拟 CPU 数是每个核心内虚拟 CPU 数的倍数时，才可能发生此转换操作。否则，不会执行转换操作，且命令将失败。

注 – 如果使用这些命令以延迟重新配置模式在非活动域或控制域上启用整体核心约束，则也会设置最大核心数。在绑定域或活动域上使用这些命令时，最大核心数不受影响。

例如，一个核心由八个虚拟 CPU 组成。如果为某域分配了七个虚拟 CPU，则 `ldm add-vcpu -c` 或 `ldm remove-vcpu -c` 命令可能无法满足整体核心约束。此时应使用 `set-vcpu -c` 命令指定核心数并启用整体核心约束。

以下示例在非活动域 ldg1 上启用整体核心约束。ldm list 命令可验证整体核心约束是否已启用。

```
primary# ldm add-vcpu -c 1 ldg1
primary# ldm list -o resmgmt ldg1
NAME
ldg1
CONSTRAINT
   whole-core
   max-cores=1
```

注 – 如果在域上启用整体核心约束，则增加核心时不会影响与这些核心关联的加密单元。因此，系统不会自动向域添加关联的加密单元或从域中删除关联的加密单元。而且，如果已将相应的加密单元分配给域，则无法删除核心。

禁用整体核心约束

如果为域分配虚拟 CPU 而不是核心，则会禁用整体核心约束。只能在非活动域上而不能在绑定域或活动域上禁用整体核心约束。在控制域上禁用整体核心约束之前，必须先启动延迟重新配置。

使用 ldm add-vcpu *number*、ldm set-vcpu *number* 或 ldm remove-vcpu *number* 命令将虚拟 CPU 分配给域或从域中删除虚拟 CPU。*number* 指定虚拟 CPU 的数量并禁用整体核心约束。有关更多信息，请参见 ldm(1M) 手册页。

可以在之前已配置 CPU 核心的域上使用 ldm add-vcpu *number* 或 ldm rm-vcpu *number* 命令。在这种情况下，现有 CPU 核心数将自动转换为相应的虚拟 CPU 数。

注 – 如果禁用整体核心约束，也会自动禁用最大核心约束。

以下示例在非活动域 ldg1 上禁用整体核心约束：

```
primary# ldm set-vcpu 1 ldg1
```

将 CPU 分配到控制域

要在控制域上启用整体核心约束，控制域必须处于延迟重新配置模式下。仅当可用 CPU 核心数足以满足请求的约束时，才能成功在控制域上启用整体核心约束。也就是说，必须具有未使用的核心、控制域已使用的核心或控制域部分使用的核心。否则，控制域上的 CPU 分配将保持不变。

注 – 如果控制域处于延迟重新配置模式下，整体核心约束和核心数设置也会指定最大核心数。

以下示例在控制域 (primary) 上启用整体核心约束。首先，在控制域上启动延迟重新配置。接下来向控制域分配一个整体核心，然后重新引导域，使更改生效。

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

primary# ldm add-vcpu -c 1 primary
primary# reboot
```

整体核心约束和其他域功能之间的交互作用

本节介绍整体核心约束和以下功能之间的交互作用：

- 第 143 页中的 “CPU 动态重新配置”
- 第 143 页中的 “动态资源管理”
- 第 144 页中的 “域迁移”
- 第 144 页中的 “电源管理”

CPU 动态重新配置

整体核心约束与 CPU 动态重新配置 (dynamic reconfiguration, DR) 完全兼容。为域定义整体核心约束后，可以使用 `ldm add-vcpu -c`、`ldm set-vcpu -c` 或 `remove-vcpu -c` 命令更改活动域上的核心数。

但是，如果绑定域或活动域未处于延迟重新配置模式下，则其核心数不能超过最大核心数。最大核心数随最大核心约束进行设置，在启用整体核心约束时会自动启用该约束。任何不满足最大核心约束的 CPU DR 操作都将失败。

动态资源管理

整体核心约束与动态资源管理 (dynamic resource management, DRM) 不兼容。如果在使用整体核心约束的域上启用 DRM 策略，系统会自动禁用该策略。整体核心约束会保持启用状态。

即使在使用整体核心约束时无法启用 DRM 策略，您仍可以为域定义 DRM 策略。请注意，自动禁用某策略后，它仍保持活动状态。如果重新启动域时没有启用整体核心约束，则会自动重新启用 DRM 策略。

以下为整体核心约束和 DRM 之间的预期交互作用关系：

- 如果在域上设置整体核心约束，当您尝试在该域上启用 DRM 策略时系统将发出警告消息。
- 如果是在非活动域上使用 DRM 策略，则允许您在该域上启用整体核心约束。当该域转为活动状态且 DRM 策略处于启用状态时，系统会为该域自动禁用 DRM 策略。
- 如果在活动域或绑定域上启用 DRM 策略，则不允许您启用整体核心约束。

域迁移

CPU 整体核心配置与域迁移不兼容。但是，您仍可以迁移已配置 CPU 整体核心的域。要在进行了这样的迁移后恢复整体核心约束，请停止域并针对整体核心分配对其进行重新配置。

电源管理

整体核心约束与电源管理 (power management, PM) 性能和弹性模式完全兼容。启用弹性模式后，PM 子系统可以向已配置整体核心约束的域添加 CPU 核心或从这些域中删除 CPU 核心。在这种情况下，整体核心约束继续保持启用状态，使用该约束的域仍保持仅配置整体核心。

使用内存动态重新配置

Oracle VM Server for SPARC 2.0 发行版引入了内存动态重新配置 (dynamic reconfiguration, DR)。此功能基于容量，通过此功能可以向活动的逻辑域中添加或从中删除任意数量的内存。

下面是使用内存 DR 功能的要求和限制：

- 您可以在任意域上执行内存 DR 操作。但是，给定时间内在一个域上只能执行一个内存 DR 操作。
- 内存 DR 功能可对给定操作中涉及的内存的地址和大小强制执行 256 MB 对齐。请参见第 146 页中的 “内存对齐”。
- 使用内存 DR 功能**无法**将可用内存池内未对齐的内存分配给域。请参见第 147 页中的 “添加未对齐的内存”。

如果使用内存 DR 操作无法重新配置域的内存，则必须停止域才能重新配置内存。如果域是控制域，则必须先启动延迟重新配置。

添加内存

如果域处于活动状态，可以使用 `ldm add-memory` 命令向域动态添加内存。如果指定内存大小大于域的当前内存大小，也可以使用 `ldm set-memory` 命令动态添加内存。

删除内存

如果域处于活动状态，可以使用 `ldm remove-memory` 命令从域中动态删除内存。如果指定内存大小小于域的当前内存大小，也可以使用 `ldm set-memory` 命令动态删除内存。

内存删除操作可能要运行很长时间。您可以跟踪操作的进度或者取消正在进行的内存 DR 请求。

跟踪内存 DR 请求的进度

通过对指定域运行 `ldm list -l` 命令，可以跟踪 `ldm remove-memory` 命令的进度。

取消内存 DR 请求

通过中断 `ldm remove-memory` 命令（按 Ctrl-C）或发出 `ldm cancel-operation memdr` 命令，可以取消正在进行的删除请求。如果取消内存删除请求，只会影响删除请求的未处理部分，即，仍要从域删除的内存量。

部分内存 DR 请求

如果可用内存不足以完成整个请求，将拒绝内存添加请求。但是，如果目标域无法添加 Logical Domains Manager 请求的任何内存，则可部分完成内存添加请求。

如果域中没有足够的内存来完成整个请求，将拒绝内存删除请求。但是，如果目标域无法删除 Logical Domains Manager 请求的任何内存，则可部分完成内存删除请求。

注 – 从域中删除内存后，在将其添加到其他域之前，会先进行清除。

重新配置控制域内存

可以使用内存 DR 功能重新配置控制域的内存。如果无法在控制域上执行内存 DR 请求，则必须先启动延迟重新配置。

从活动域中删除大量内存时可能不适合使用内存 DR，因为内存 DR 操作可能要运行很长时间。特别地，在系统初始配置期间，应该使用延迟重新配置来减少控制域中的内存。

减少控制域的内存

使用延迟重新配置而不是内存 DR 从默认出厂初始配置中减少控制域的内存。在该默认配置中，控制域拥有主机系统的所有内存。不太适合使用内存 DR 功能来实现此目的，因为不一定可以添加活动域，或者通常会放弃所有请求的内存。当然，该域中运行的 OS 会尽最大可能来完成请求。另外，内存删除操作可能要运行很长时间。涉及到大量内存操作时这些问题会扩大化，正如初始减少控制域内存的情况。

因此，请按照以下步骤使用延迟重新配置：

1. 使用 `ldm start-reconf primary` 命令将控制域置于延迟重新配置模式下。
2. 根据需要对控制域拥有的主机系统资源进行分区。
3. 如果需要，使用 `ldm cancel-reconf` 命令撤消步骤 2 中的操作，重新从头开始。
4. 重新引导控制域以使重新配置更改生效。

动态重新配置和延迟重新配置

如果控制域中存在暂挂的延迟重新配置，系统将拒绝任何其他域的内存重新配置请求。如果控制域中不存在暂挂的延迟重新配置，系统将拒绝任何不支持内存 DR 的域的内存重新配置请求。不支持内存 DR 的控制域上的内存重新配置请求将转换为延迟重新配置请求。

内存对齐

内存重新配置请求具有不同的对齐要求，具体取决于将应用请求的域的状态。

活动域的内存对齐

- **动态添加和删除。**内存块的地址和大小为对齐的 256 MB，以用于进行动态添加和动态删除。最小操作大小为 256 MB。

  将拒绝大于绑定大小的未对齐请求或删除请求。

  使用以下命令调整内存分配：

  - `ldm add-memory`。如果在此命令中指定 `--auto-adj` 选项，则要添加的内存量为对齐的 256 MB，这可能会增加实际添加到域的内存量。
  - `ldm remove-memory`。如果在此命令中指定 `--auto-adj` 选项，则要删除的内存量为对齐的 256 MB，这可能会减少实际从域中删除的内存量。
  - `ldm set-memory`。此命令被视为添加或删除操作。如果指定 `--auto-adj` 选项，则要添加或删除的内存量为对齐的 256 MB，如上所述。请注意，此对齐可能会增大域的最终内存大小。
- **延迟重新配置。**内存块的地址和大小为对齐的 4 MB。如果发出的未对齐的请求，系统会将该请求舍入为对齐的 4 MB。

绑定域的内存对齐

绑定域的内存块的地址和大小为对齐的 4 MB。如果发出的未对齐的请求，系统会将该请求舍入为对齐的 4 MB。这意味着域的最终内存大小可能会大于您指定的大小。

对于 `ldm add-memory`、`ldm set-memory` 和 `ldm remove-memory` 命令，`--auto-adj` 选项会将最终内存的大小舍入为对齐的 256 MB。这意味着最终内存可能会大于您指定的大小。

非活动域的内存对齐

对于 `ldm add-memory`、`ldm set-memory` 和 `ldm remove-memory` 命令，`--auto-adj` 选项会将最终内存的大小舍入为对齐的 256 MB。不存在对非活动域的对齐要求。第 146 页中的 “绑定域的内存对齐”中所述的限制在绑定非活动域后生效。

添加未对齐的内存

内存 DR 功能对动态添加到活动域或从活动域中删除的内存的地址和大小强制执行 256 MB 内存对齐。这意味着，使用内存 DR 无法删除活动域中任何未对齐的内存。

还意味着使用内存 DR 无法将可用内存池中任何未对齐的内存添加到活动域。

在所有对齐的内存都已分配后，可以使用 `ldm add-memory` 命令将剩余的未对齐内存添加到绑定域或非活动域。还可以使用此命令借助延迟重新配置操作将剩余的未对齐内存添加到控制域。

以下示例说明如何将两个剩余的 128 MB 内存块添加到 `primary` 和 `ldom1` 域中。`ldom1` 域处于绑定状态。以下命令可添加这两个剩余内存块。第一个命令可在控制域上启动延迟重新配置操作。第二个命令可将其中一个 128 MB 内存块添加到控制域。第五个命令可将另一个 128 MB 内存块添加到 `ldom1` 域。

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

# ldm add-memory 128M primary
--------------------------------------------------------------------------------
Notice: The primary domain is in the process of a delayed reconfiguration.
Any changes made to the primary domain will only take effect after it reboots.
--------------------------------------------------------------------------------

# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -ndcv-  SP      8     2688M    0.1%  23d 8h 8m

# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      8     2560M    0.5%  23d 8h 9m
ldom1            bound      ------  5000    1     524M

# ldm add-mem 128M ldom1
# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      8     2560M    0.1%  23d 8h 9m
ldom1            bound      ------  5000    1     652M
```

内存 DR 示例

以下示例说明如何执行内存 DR 操作。有关相关 CLI 命令的信息，请参见 ldm(1M) 手册页。

示例 10–1 活动域上的内存 DR 操作

本示例说明如何动态向活动域 ldom1 添加内存和从中删除内存。

ldm list 的输出显示了 Memory (内存) 字段中各个域的内存。第一个 ldm add-mem 命令将退出并显示错误，因为您必须将内存指定为 256 MB 的倍数。下一个 ldm add-mem 命令使用 --auto-adj 选项，这样，即使您将 200M 指定为要添加的内存量，系统也会将内存量舍入为 256 MB。

ldm rm-mem 命令将退出并显示错误，因为您必须将内存指定为 256 MB 的倍数。将 --auto-adj 选项添加到同一命令后，内存删除将成功，因为内存量会向下舍入到邻近的 256 MB 边界。

```
# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.4%  1d 22h 53m
ldom1            active     -n----  5000    2     2G       0.4%  1d 1h 23m
ldom2            bound      ------  5001    2     200M

# ldm add-mem 200M ldom1
The size of memory must be a multiple of 256MB.

# ldm add-mem --auto-adj 200M ldom1
Adjusting request size to 256M.
The ldom1 domain has been allocated 56M more memory
than requested because of memory alignment constraints.

# ldm list
NAME         STATE     FLAGS     CONS     VCPU     MEMORY     UTIL     UPTIME
primary     active     -n-cv-     SP     4     27392M     5.0%     8m
ldom1         active     -n----     5000     2     2304M     0.5%     1m
ldom2         bound     ------     5001     2     200M

# ldm rm-mem --auto-adj 300M ldom1
Adjusting requested size to 256M.
The ldom1 domain has been allocated 44M more memory
than requested because of memory alignment constraints.

# ldm list
NAME         STATE     FLAGS     CONS     VCPU     MEMORY     UTIL     UPTIME
primary     active     -n-cv-     SP     4     27392M     0.3%     8m
ldom1         active     -n----     5000     2     2G     0.2%     2m
ldom2         bound     ------     5001     2     200M
```

示例 10–2 绑定域上的内存 DR 操作

本示例说明如何向绑定域 ldom2 添加内存和从中删除内存。

示例 10–2 绑定域上的内存 DR 操作 （续）

ldm list 的输出显示了 Memory (内存) 字段中各个域的内存。第一个 ldm add-mem 命令可向 ldom2 域中添加 100 MB 的内存。下一个 ldm add-mem 命令中指定了 --auto-adj 选项，这可导致将额外的 112 MB 内存动态添加到 ldom2 中。

ldm rm-mem 命令可从 ldom2 域中动态删除 100 MB 的内存。如果在同一命令中指定 --auto-adj 选项以删除 300 MB 内存，内存量将向下舍入到邻近的 256 MB 边界。

```
# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.4%  1d 22h 53m
ldom1            active     -n----  5000    2     2G       0.4%  1d 1h 23m
ldom2            bound      ------  5001    2     200M

# ldm add-mem 100M ldom2

# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.5%  1d 22h 54m
ldom1            active     -n----  5000    2     2G       0.2%  1d 1h 25m
ldom2            bound      ------  5001    2     300M

# ldm add-mem --auto-adj 100M ldom2
Adjusting request size to 256M.
The ldom2 domain has been allocated 112M more memory
than requested because of memory alignment constraints.

# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.4%  1d 22h 55m
ldom1            active     -n----  5000    2     2G       0.5%  1d 1h 25m
ldom2            bound      ------  5001    2     512M

# ldm rm-mem 100M ldom2
# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   3.3%  1d 22h 55m
ldom1            active     -n----  5000    2     2G       0.2%  1d 1h 25m
ldom2            bound      ------  5001    2     412M

# ldm rm-mem --auto-adj 300M ldom2
Adjusting request size to 256M.
The ldom2 domain has been allocated 144M more memory
than requested because of memory alignment constraints.

# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.5%  1d 22h 55m
ldom1            active     -n----  5000    2     2G       0.2%  1d 1h 26m
ldom2            bound      ------  5001    2     256M
```

示例 10–3 设置域内存大小

本示例说明如何使用 ldm set-memory 命令向域中添加内存和从中删除内存。

示例 10–3 设置域内存大小 （续）

`ldm list` 的输出显示了 Memory (内存) 字段中各个域的内存。第一个 `ldm set-mem` 命令尝试将 `primary` 域的大小设置为 3400 MB。最终错误表明指定的值不在 256 MB 边界上。将 `--auto-adj` 选项添加到同一命令后，可以成功删除某些内存并保持在 256 MB 边界上。此命令还会发出警告，以指出并非所有请求的内存都可以删除，因为域可能正在使用某些内存。

下一个 `ldm set-mem` 命令会将处于绑定状态的 `ldom2` 域的内存大小设置为 690 MB。如果向同一命令中添加 `--auto-adj` 选项，将动态向 `ldom2` 中添加额外的 78 MB 内存以保持在 256 MB 边界上。

```
# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     27392M   0.5%  1d 22h 55m
ldom1            active     -n----  5000    2     2G       0.2%  1d 1h 26m
ldom2            bound      ------  5001    2     256M

# ldm set-mem 3400M primary
An ldm set-mem 3400M command would remove 23992MB, which is not a multiple
of 256MB. Instead, run ldm rm-mem 23808MB to ensure a 256MB alignment.

# ldm set-mem --auto-adj 3400M primary
Adjusting request size to 3.4G.
The primary domain has been allocated 184M more memory
than requested because of memory alignment constraints.
Only 9472M of memory could be removed from the primary domain
because the rest of the memory is in use.

# ldm set-mem 690M ldom2
# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     17920M   0.5%  1d 22h 56m
ldom1            active     -n----  5000    2     2G       0.6%  1d 1h 27m
ldom2            bound      ------  5001    2     690M

# ldm set-mem --auto-adj 690M ldom2
Adjusting request size to 256M.
The ldom2 domain has been allocated 78M more memory
than requested because of memory alignment constraints.

# ldm list
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active     -n-cv-  SP      4     17920M   2.1%  1d 22h 57m
ldom1            active     -n----  5000    2     2G       0.2%  1d 1h 27m
ldom2            bound      ------  5001    2     768M
```

使用电源管理

要使用电源管理 (Power Management, PM)，需要先在 Oracle Integrated Lights Out Manager (ILOM) 3.0 固件中设置 PM 模式。本节汇总了所需的信息，以便能够在 Oracle VM Server for SPARC 软件中使用 PM。

有关 ILOM 的更多信息，请参见以下内容：

- 《Sun Integrated Lights Out Manager (ILOM) 3.0 CLI 过程指南》 (http://dlc.sun.com/pdf/820-6412-12/820-6412-12.pdf) 中的“监视功耗”
- 《Oracle Integrated Lights Out Manager (ILOM) 3.0 Feature Updates and Release Notes》 (http://dlc.sun.com/pdf/820-7329-17/820-7329-17.pdf)（《Oracle Integrated Lights Out Manager (ILOM) 3.0 功能更新和发行说明》）

电源模式是可在任何时候控制系统用电的设置。从 Logical Domains 1.3 发行版开始，支持以下电源模式，假定底层平台已实现 PM 功能：

- **性能模式**。系统可以使用所有可用功率。
- **弹性模式**。根据当前利用率级别调节系统用电。例如，利用率降低，资源的电源状态也将降级。

下面介绍 PM 功能：

- **CPU 核心自动禁用**。PM 会在 CPU 核心上的所有导线束都已禁用时自动禁用该核心。
- **CPU 时钟周期跳步**。从 Oracle VM Server for SPARC 2.0 发行版开始，PM 可以在 SPARC T3 平台上自动调整 CPU 时钟周期跳步。通过调整，可以增加或减少要跳过的时钟周期数，以使所有域都保持在电源利用率阈值之内。PM 基于 CPU 利用率确定是否要进行此类调整。当系统进入性能模式后，要跳过的时钟周期数将自动调整为零。
- **深度空闲模式下的内存操作**。从 Oracle VM Server for SPARC 2.0 发行版开始，SPARC T3 平台在处于弹性模式下时，会自动将未充分利用的内存配置为在深度空闲模式下运行，以便节省电能。
- **功率极限**。从 Oracle VM Server for SPARC 2.0 发行版开始，可以在 SPARC T3 平台上设置**功率极限**以限制系统的功率消耗。如果功率消耗超过功率极限，将使用 PM 技术降低功率。可以使用 ILOM 服务处理器 (service processor, SP) 设置功率极限。

  请参见以下文档：

  - 《Sun Integrated Lights Out Manager (ILOM) 3.0 CLI 过程指南》 (http://dlc.sun.com/pdf/820-6412-12/820-6412-12.pdf)
  - 《Oracle Integrated Lights Out Manager (ILOM) 3.0 Feature Updates and Release Notes》 (http://dlc.sun.com/pdf/820-7329-17/820-7329-17.pdf)（《Oracle Integrated Lights Out Manager (ILOM) 3.0 功能更新和发行说明》）

可以使用 ILOM 界面设置功率极限、宽限期和违规操作。如果宽限期过后仍超出功率极限，将执行违规操作。

如果当前功率消耗超出功率极限，系统会尝试对可进行电源管理的资源的电源状态进行降级。如果功率消耗降到功率极限以下，则允许升级这些资源的电源状态。如果系统处于弹性模式下，则根据利用率级别来确定是否升级资源的电源状态。

当系统处于弹性模式下时，会先验证某些域配置修改，以确认未超出功率极限。如果超出功率极限，可能会按照要求仅修改或添加某些资源。如果稍后提高功率极限，则可以添加未成功修改的任何资源。

如果域的加载导致资源消耗更多功率，则只能成功打开其功率消耗保持在功率极限以下的资源的电源。

有关使用 ILOM 3.0 固件 CLI 配置电源模式的说明，请参见《Sun Integrated Lights Out Manager (ILOM) 3.0 CLI 过程指南》 (http://dlc.sun.com/pdf/820-6412-12/820-6412-12.pdf) 中的“监视功耗”。

列出受 CPU 电源管理的导线束

本节说明如何列出受电源管理的导线束和虚拟 CPU。

▼ 列出受 CPU 电源管理的导线束

- **使用下列某一命令列出受电源管理的导线束：**

  a. 使用 `list -l` 子命令。

  在输出中，CPU 的 `UTIL` 列中的短划线 (`---`) 表示导线束受电源管理。

```
# ldm list -l primary
NAME         STATE   FLAGS  CONS  VCPU MEMORY  UTIL  UPTIME
primary      active  -n-cv  SP    8    4G      4.3%  7d 19h 43m

SOFTSTATE
Solaris running

MAC
    00:14:4f:fa:ed:88

HOSTID
    0x84faed88

CONTROL
    failure-policy=ignore

DEPENDENCY
    master=

VCPU
```

```
    VID    PID    UTIL STRAND
    0      0      0.0%   100%
    1      1      ---    100%
    2      2      ---    100%
    3      3      ---    100%
    4      4      ---    100%
    5      5      ---    100%
    6      6      ---    100%
    7      7      ---    100%
....
```

b. 在 `list -l` 子命令中使用可解析选项 (`-p`)。

在输出中，`util=` 后留空表示导线束受电源管理。

```
# ldm list -l -p

VCPU
|vid=0|pid=0|util=0.7%|strand=100
|vid=1|pid=1|util=|strand=100
|vid=2|pid=2|util=|strand=100
|vid=3|pid=3|util=|strand=100
|vid=4|pid=4|util=0.7%|strand=100
|vid=5|pid=5|util=|strand=100
|vid=6|pid=6|util=|strand=100
|vid=7|pid=7|util=|strand=100
```

▼ 列出受电源管理的 CPU

- **使用下列某一命令列出受电源管理的 CPU：**

a. 使用 `list-devices -a cpu` 命令。

在输出的 `PM` 列中，`yes` 表示 CPU 受电源管理，`no` 表示 CPU 电源已打开。假定默认情况下 100% 空闲 CPU 受电源管理，因此其 `PM` 列下均具有短划线 (`---`)。

```
# ldm list-devices -a cpu
VCPU
   PID     %FREE      PM
   0       0          no
   1       0          yes
   2       0          yes
   3       0          yes
   4       100        ---
   5       100        ---
   6       100        ---
   7       100        ---
```

b. 在 `list-devices -a cpu` 子命令中使用可解析选项 (`-p`)。

在输出的 `pm=` 字段中，`yes` 表示 CPU 受电源管理，`no` 表示 CPU 电源已打开。假定默认情况下 100% 空闲 CPU 受电源管理，因此其 pm= 字段后留空。

```
# ldm list-devices -a -p cpu
VERSION 1.4
VCPU
|pid=0|free=0|pm=no
```

```
|pid=1|free=0|pm=yes
|pid=2|free=0|pm=yes
|pid=3|free=0|pm=yes
|pid=4|free=0|pm=no
|pid=5|free=0|pm=yes
|pid=6|free=0|pm=yes
|pid=7|free=0|pm=yes
|pid=8|free=100|pm=
|pid=9|free=100|pm=
|pid=10|free=100|pm=
```

使用动态资源管理

从 Logical Domains 1.3 软件开始，可以使用策略来确定自动执行 DR 活动的方式。目前，仅可以创建相应策略来控制虚拟 CPU 的动态资源管理。

注意 – 以下问题会影响 CPU 动态资源管理 (dynamic resource management, DRM)：

- 当 PM 处于弹性模式下时，无法启用 DRM。
- 启用 DRM 后，从性能模式到弹性模式的任何更改均将延迟。
- 在执行域迁移操作之前，确保禁用 CPU DRM。
- DRM 策略不适用于配置有**整体核心**约束的域。

资源管理策略可指定在何种条件下可以自动向逻辑域中添加虚拟 CPU 或从中删除虚拟 CPU。使用 `ldm add-policy`、`ldm set-policy` 和 `ldm remove-policy` 命令管理策略：

```
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
```

有关这些命令以及创建资源管理策略的信息，请参见 `ldm(1M)` 手册页。

策略在 `tod-begin` 和 `tod-end` 属性指定的时间内有效。策略使用 `priority` 属性值来确定在同时存在多个有效策略的情况下要使用哪个策略。

策略使用 `util-high` 和 `util-low` 属性值指定 CPU 利用率的阈值上限和下限。如果利用率超出 `util-high` 的值，将向域中添加虚拟 CPU，直到利用率的值位于 `vcpu-min` 和 `vcpu-max` 值之间。如果利用率降到 `util-low` 值以下，将从域中删除虚拟 CPU，直到利用率的值位于 `vcpu-min` 和 `vcpu-max` 值之间。如果达到 `vcpu-min` 值，将无法再动态删除任何虚拟 CPU。如果达到 `vcpu-max` 值，将无法再动态添加任何虚拟 CPU。

示例 10–4 添加资源管理策略

例如，观察系统的典型利用率数周后，您可以设置策略，以优化资源使用情况。资源占用高峰期是每天上午 9:00 至下午 6:00（太平洋时间），资源占用低谷期是每天下午 6:00 至次日上午 9:00（太平洋时间）。

根据对系统利用率的观察，您决定基于系统总体利用率创建以下高峰期和低谷期策略：

- **高峰期**：每天上午 9:00 至下午 6:00（太平洋时间）
- **低谷期**：每天下午 6:00 至次日上午 9:00（太平洋时间）

以下 `ldm add-policy` 命令可在 `ldom1` 域上创建利用率高峰期内要使用的 `high-usage` 策略。

以下 `high-usage` 策略执行以下操作：

- 通过设置 `tod-begin` 和 `tod-end` 属性，将开始和结束时间分别指定为上午 9:00 和下午 6:00。
- 通过设置 `util-lower` 和 `util-upper` 属性，将执行策略分析的下限和上限分别指定为 25% 和 75%。
- 通过设置 `vcpu-min` 和 `vcpu-max` 属性，将最小和最大虚拟 CPU 数分别指定为 2 和 16。
- 通过设置 `attack` 属性，将任一资源控制周期内要添加的最大虚拟 CPU 数指定为 1。
- 通过设置 `decay` 属性，将任一资源控制周期内要删除的最大虚拟 CPU 数指定为 1。
- 通过设置 `priority` 属性，将此策略的优先级指定为 1。优先级为 1 表示即使其他策略可以生效，也强制执行此策略。
- 通过设置 `name` 属性，将策略文件的名称指定为 `high-usage`。
- 对于未指定的属性（例如 `enable` 和 `sample-rate`），使用其默认值。请参见 `ldm(1M)` 手册页。

```
# ldm add-policy tod-begin=09:00 tod-end=18:00 util-lower=25 util-upper=75 \
vcpu-min=2 vcpu-max=16 attack=1 decay=1 priority=1 name=high-usage ldom1
```

以下 `ldm add-policy` 命令可在 `ldom1` 域上创建利用率低谷期内要使用的 `med-usage` 策略。

以下 `med-usage` 策略执行以下操作：

- 通过设置 `tod-begin` 和 `tod-end` 属性，将开始和结束时间分别指定为下午 6:00 和上午 9:00。
- 通过设置 `util-lower` 和 `util-upper` 属性，将执行策略分析的下限和上限分别指定为 10% 和 50%。
- 通过设置 `vcpu-min` 和 `vcpu-max` 属性，将最小和最大虚拟 CPU 数分别指定为 2 和 16。

示例 10–4 添加资源管理策略 （续）

- 通过设置 attack 属性，将任一资源控制周期内要添加的最大虚拟 CPU 数指定为 1。
- 通过设置 decay 属性，将任一资源控制周期内要删除的最大虚拟 CPU 数指定为 1。
- 通过设置 priority 属性，将此策略的优先级指定为 1。优先级为 1 表示即使其他策略可以生效，也强制执行此策略。
- 通过设置 name 属性，将策略文件的名称指定为 high-usage。
- 对于未指定的属性（例如 enable 和 sample-rate），使用其默认值。请参见 ldm(1M) 手册页。

```
# ldm add-policy tod-begin=18:00 tod-end=09:00 util-lower=10 util-upper=50 \
 vcpu-min=2 vcpu-max=16 attack=1 decay=1 priority=1 name=med-usage ldom1
```

列出域资源

本节显示了 ldm 子命令的语法用法，定义了一些输出项（例如，标志和使用率统计信息），并提供了类似于实际显示内容的输出示例。

计算机可读的输出

如果要创建使用 ldm list 命令输出的脚本，请**始终**使用 -p 选项生成计算机可读形式的输出。有关更多信息，请参见第 158 页中的 “生成可解析的、计算机可读的列表 (-p)”。

▼ 显示 ldm 子命令的语法用法

- **查看所有 ldm 子命令的语法用法。**

```
primary# ldm --help
```

有关 ldm 子命令的更多信息，请参见 ldm(1M) 手册页。

标志定义

在域的输出中 (ldm list) 可以显示以下标志。如果使用命令的可解析长选项 (-l -p)，则会拼写出标志，例如，flags=normal,control,vio-service。否则，将显示字母缩写，例如 -n-cv-。列表标志值与位置相关。在从左至右排列的六列中的每一列中可以出现以下值：

第 1 列

- s 启动或停止
- - 占位符

第 2 列

- n 正常
- t 转换

第 3 列

- d 延迟重新配置
- r 内存动态重新配置 (dynamic reconfiguration, DR)
- - 占位符

第 4 列

- c 控制域
- - 占位符

第 5 列

- v 虚拟 I/O 服务域
- - 占位符

第 6 列

- s 迁移操作中的源域
- t 迁移操作中的目标域
- e 迁移过程中发生错误
- - 占位符

使用率统计信息定义

在 `ldm list` 命令中使用长 (`-l`) 选项，可以显示每个虚拟 CPU 的使用率统计信息 (`UTIL`)。统计信息是虚拟 CPU 代表客操作系统执行操作所用的时间的百分比。除非虚拟 CPU 被移交给虚拟机管理程序，否则将其视为代表客操作系统执行。如果客操作系统没有将虚拟 CPU 移交给虚拟机管理程序，则客操作系统中的 CPU 使用率始终显示为 `100%`。

为逻辑域报告的使用率统计信息是域中虚拟 CPU 的虚拟 CPU 使用率平均值。`UTIL` 列中的短划线 (`---`) 表示导线束受电源管理。

查看各种列表

▼ 显示软件版本 (-V)

- **查看当前安装的软件版本。**

```
primary# ldm -V
```

▼ 生成短列表

- **生成所有域的短列表。**

  ```
  primary# ldm list
  ```

▼ 生成长列表 (-l)

- **生成所有域的长列表。**

  ```
  primary# ldm list -l
  ```

▼ 生成扩展列表 (-e)

- **生成所有域的扩展列表。**

  ```
  primary# ldm list -e
  ```

▼ 生成可解析的、计算机可读的列表 (-p)

- **生成所有域的可解析的、计算机可读的列表。**

  ```
  primary# ldm list -p
  ```

▼ 生成长列表的子集 (-o *format*)

- **通过输入以下一个或多个 *format* 选项将输出生成为资源的子集。如果指定多种格式，请使用逗号分隔各项，其间不留空格。**

  ```
  primary# ldm list -o resource[,resource...] ldom
  ```

  - `console`－输出包含虚拟控制台 (`vcons`) 和虚拟控制台集中器 (`vcc`) 服务
  - `core`－输出包含有关已分配整体核心的域的信息
  - `cpu`－输出包含有关虚拟 CPU (`vcpu`)、物理 CPU (`pcpu`) 和核心 ID 的信息
  - `crypto`－加密单元输出包含模块化算术单元 (`mau`) 和任何其他支持 LDoms 的加密单元，例如控制字队列 (Control Word Queue, CWQ)
  - `disk`－输出包含虚拟磁盘 (`vdisk`) 和虚拟磁盘服务器 (`vds`)
  - `domain`－输出包含变量 (`var`)、主机 ID (`hostid`)、域状态、标志、UUID 和软件状态
  - `memory`－输出包含 `memory`
  - `network`－输出包含介质访问控制 (`mac`) 地址、虚拟网络交换机 (`vsw`) 和虚拟网络 (`vnet`) 设备
  - `physio`－物理输入/输出包含外设部件互连 (`pci`) 和网络接口单元 (`niu`)
  - `resmgmt`－输出包含动态资源管理 (DRM) 策略信息，指出当前运行的策略并列出与整体核心配置相关的约束

- `serial`－输出包含虚拟逻辑域通道 (`vldc`) 服务、虚拟逻辑域通道客户机 (`vldcc`)、虚拟数据平面通道客户机 (`vdpcc`) 和虚拟数据平面通道服务 (`vdpcs`)
- `stats`－输出包含与资源管理策略相关的统计信息
- `status`－输出包含有关正在执行的域迁移的状态

以下示例显示了您可以指定的输出的各种子集：

- 列出控制域的 CPU 信息

  ```
  # ldm list -o cpu primary
  ```
- 列出来宾域的域信息

  ```
  # ldm list -o domain ldm2
  ```
- 列出来宾域的内存和网络信息

  ```
  # ldm list -o network,memory ldm1
  ```
- 列出来宾域的 DRM 策略信息

  ```
  # ldm list -o resmgmt,stats ldm1
  ```

▼ 列出变量

- **显示域的变量及变量值。**

  `primary# ldm list-variable` *variable-name ldom*

  例如，以下命令可显示 `ldg1` 域上 `boot-device` 变量的值：

  ```
  primary# ldm list-variable boot-device ldg1
  boot-device=/virtual-devices@100/channel-devices@200/disk@0:a
  ```

▼ 列出绑定

- **列出绑定到域的资源。**

  `primary# ldm list-bindings` *ldom*

▼ 列出配置

- **列出 SP 上已存储的逻辑域配置。**

示例 10–5 配置列表

`ldm list-config` 命令可列出服务处理器上已存储的逻辑域配置。将此命令与 `-r` 选项一起使用，可列出允许控制域上存在自动保存文件的配置。

有关配置的更多信息，请参见第 165 页中的 “管理 Logical Domains 配置”。有关更多示例，请参见 `ldm(1M)` 手册页。

```
primary# ldm list-config
factory-default
3guests
foo [next poweron]
primary
reconfig-primary
```

更多信息 **标签的含义**

配置名称右侧的标签具有以下含义：

- `[current]`－最后引导的配置，仅当符合当前运行的配置时；也就是说，直到您启动重新配置。重新配置后，注释将更改为 `[next poweron]`。
- `[next poweron]`－下次关开机循环时将使用的配置。

▼ 列出设备

- **列出所有服务器资源（绑定资源和非绑定资源）。**

```
primary# ldm list-devices -a
```

▼ 列出可用内存

- **列出可进行分配的内存量。**

```
primary# ldm list-devices mem
MEMORY
    PA                   SIZE
    0x14e000000          2848M
```

▼ 列出服务

- **列出可用的服务。**

```
primary# ldm list-services
```

列出约束

对于 Logical Domains Manager，约束是您希望分配给特定域的一个或多个资源。您可能会接收到要求添加到域中的所有资源，也可能会得不到任何资源，这取决于可用资源。`list-constraints` 子命令可列出您要求分配给域的那些资源。

▼ 列出一个域的约束

- **列出一个域的约束。**

```
primary# ldm list-constraints ldom
```

▼ 以 XML 格式列出约束

- 以 XML 格式列出特定域的约束。

  ```
  primary# ldm list-constraints -x ldom
  ```

▼ 以计算机可读格式列出约束

- 以可解析格式列出所有域的约束。

  ```
  primary# ldm list-constraints -p
  ```

◆ ◆ ◆ 第 11 章

11 管理配置

本章包含有关管理域配置的信息。

本章包括以下内容：

- 第 163 页中的 “保存域配置用于将来重建”
- 第 165 页中的 “管理 Logical Domains 配置”

保存域配置用于将来重建

基本过程是将每个域的资源约束信息保存到 XML 文件中，之后可向 Logical Domains Manager 重新发出这些约束信息，例如，在硬件故障后重建所需配置。

第 164 页中的 “从 XML 文件恢复域配置 (`ldm add-domain`)”对来宾域有效，但对控制 (`primary`) 域无效。您可以将 `primary` 域的约束保存到一个 XML 文件中，但无法将该文件应用于 `ldm add-domain -i` 命令中。不过，您可以使用 `ldm init-system` 命令和 XML 文件中的资源约束重新配置 `primary` 域。还可以使用 `ldm init-system` 命令重新配置 XML 文件中描述的其他域，但是当配置完成时，这些域将保留为非活动状态。

以下方法不保留实际绑定，只保留用于创建这些绑定的约束。这意味着，在完成该过程后，各域会具有相同的虚拟资源，但不一定绑定到相同的物理资源。

▼ 保存域配置

本过程显示了如何保存系统上的单个域或所有域的域配置。

- **保存一个或多个域的域配置。**
  - **要保存单个域的配置，请创建一个包含该域的约束的 XML 文件。**

    ```
    # ldm list-constraints -x ldom >ldom.xml
    ```

以下示例显示如何创建 XML 文件 ldg1.xml，该文件包含 ldg1 域的约束：

```
# ldm list-constraints -x ldg1 >ldg1.xml
```

- **要保存系统上所有域的配置，请创建一个包含所有域的约束的 XML 文件。**

  ```
  # ldm list-constraints -x >file.xml
  ```

  以下示例显示如何创建 XML 文件 config.xml，该文件包含系统上所有域的约束：

  ```
  # ldm list-constraints -x >config.xml
  ```

▼ 从 XML 文件恢复域配置 (ldm add-domain)

除此过程外，您还可以使用 ldm init-system 命令从 XML 文件恢复域配置。请参见第 164 页中的 “从 XML 文件恢复域配置 (ldm init-system)”。

1. **通过使用创建的 XML 文件作为输入创建域。**

   ```
   # ldm add-domain -i ldom.xml
   ```

2. **绑定域。**

   ```
   # ldm bind-domain ldom
   ```

3. **启动域。**

   ```
   # ldm start-domain ldom
   ```

示例 11–1 从 XML 文件恢复单个域

以下示例显示如何恢复单个域。首先，从 XML 文件恢复 ldg1 域。然后，绑定并重新启动所恢复的 ldg1 域。

```
# ldm add-domain -i ldg1.xml
# ldm bind ldg1
# ldm start ldg1
```

▼ 从 XML 文件恢复域配置 (ldm init-system)

本过程说明如何使用 ldm init-system 命令和 XML 文件重新创建以前保存的配置。该 XML 文件描述一个或多个域配置。可通过运行 ldm ls-constraints -x 命令创建该 XML 文件。ldm init-system 命令预计在**出厂默认**配置中运行，但是它可以从 XML 文件恢复任何配置。将根据文件中的指定对 primary 域进行重新配置，也会对在 XML 文件中具有配置的所有非 primary 域进行重新配置，但这些域会保留为非活动状态。

除此过程外，您还可以使用 ldm add-domain 命令从 XML 文件恢复单个域配置。请参见第 164 页中的 “从 XML 文件恢复域配置 (ldm add-domain)”。

1. **登录到 primary 域。**

2. **检验系统是否为出厂默认配置。**

   ```
   primary# ldm list-config | grep "factory-default"
   factory-default [current]
   ```

   如果系统不是**出厂默认**配置，请参见第 34 页中的 “恢复出厂默认配置”。

3. **成为超级用户或承担等效角色。**

   角色包含授权和具有一定权限的命令。有关角色的更多信息，请参见《系统管理指南：安全性服务》中的“配置 RBAC（任务列表）”。

4. **从 XML 文件还原一个或多个域配置。**

   ```
   # ldm init-system [-rs] -i filename.xml
   ```

   -r 选项将在配置后重新引导 primary 域。如果不指定 -r 选项，则必须手动执行重新引导。-s 选项仅恢复虚拟服务配置（vds、vcc 和 vsw），执行时可能不必重新引导。

示例 11–2 从 XML 配置文件恢复域

以下示例显示如何使用 ldm init-system 命令从**出厂默认**配置恢复系统上的 primary 域和所有域。

- **恢复 primary 域。** -r 选项用于在配置完成后重新引导 primary 域。primary.xml 文件包含先前保存的 XML 域配置。

  ```
  primary# ldm init-system -r -i primary.xml
  ```

- **恢复系统上的所有域。**将系统上的域恢复到 config.xml XML 文件中的配置。config.xml 文件包含先前保存的 XML 域配置。primary 域将由 ldm init-system 命令自动重新启动。将恢复所有其他域，但不会绑定和重新启动这些域。

  ```
  # ldm init-system -r -i config.xml
  ```

  系统重新引导之后，以下命令将绑定和重新启动 ldg1 和 ldg2 域：

  ```
  # ldm bind ldg1
  # ldm start ldg1
  # ldm bind ldg2
  # ldm start ldg2
  ```

管理 Logical Domains 配置

Logical Domains **配置**是单个系统中所有域和其资源分配的完整说明。可以在服务处理器 (service processor, SP) 上保存和存储配置，以供以后使用。

启动系统时，SP 将引导选定的配置。通过引导配置，系统将运行配置中指定的同一组域，并使用配置中指定的同一虚拟化和分区资源分配。默认配置是最近保存的配置。

从 Logical Domains 1.2 发行版开始，每次更改 Logical Domains 配置时，都会在控制域上自动保存当前配置的副本。

自动保存操作会立即发生，即使处于以下情况也是如此：

- 当新配置未明确保存在 SP 上时
- 当直到受影响的域重新引导才会进行实际配置更改时

当保存在 SP 上的配置丢失时，通过此自动保存操作，可以恢复配置。当系统执行关开机循环时未将当前配置明确保存到 SP 时，也可以通过此操作恢复配置。这些情况下，如果该配置比为下次引导标记的配置新，Logical Domains Manager 可以在重新启动时恢复该配置。

注 – 电源管理、FMA、ASR 以及 PRI 更新事件不会导致对自动保存文件进行更新。

可以自动或手动将自动保存文件恢复到新的或现有的配置。默认情况下，当自动保存配置比相应的运行中配置新时，会将一条消息写入 Logical Domains 日志。因此，必须使用 `ldm add-spconfig -r` 命令手动更新现有配置或根据自动保存数据创建新配置。

注 – 当延迟的重新配置处于暂挂状态时，将立即自动保存配置更改。因此，如果运行 `ldm list-config -r` 命令，自动保存配置将显示为比当前配置新。

有关如何使用 `ldm *-spconfig` 命令管理配置和手动恢复自动保存文件的信息，请参见 `ldm(1M)` 手册页。

有关如何选择要引导的配置的信息，请参见第 173 页中的 “将 Logical Domains Manager 与服务处理器结合使用”。

▼ 修改自动恢复策略

自动恢复策略指定当自动保存在控制域上的一个配置比相应的运行中配置新时如何处理配置的恢复。自动恢复策略是通过设置 `ldmd` SMF 服务的 `autorecovery_policy` 属性指定的。`autorecovery_policy` 属性可以具有下列值：

- `autorecovery_policy=1` — 自动保存配置比相应的运行中配置新时，记录警告消息。这些消息记录在 `ldmd` SMF 日志文件中。用户必须手动执行所有配置恢复。这是默认策略。
- `autorecovery_policy=2` — 如果自动保存配置比相应的运行中配置新，则显示通知消息。每次重新启动 Logical Domains Manager 之后，首次发出 `ldm` 命令时，此通知消息将显示在所有 `ldm` 命令的输出中。用户必须手动执行所有配置恢复。

- `autorecovery_policy=3` — 如果自动保存配置比相应的运行中配置新，将自动更新该配置。此操作会覆写将在下次关开机循环期间使用的 SP 配置。将使用保存在控制域上的较新配置更新此配置。此操作不会影响当前运行的配置。它只会影响将在下次关开机循环期间使用的配置。还会记录一条消息，声明 SP 上保存了较新配置，将在系统下次执行关开机循环时对其进行引导。这些消息记录在 `ldmd` SMF 日志文件中。

1. **登录到控制域。**

2. **成为超级用户或承担等效角色。**

   角色包含授权和具有一定权限的命令。有关角色的更多信息，请参见《系统管理指南：安全性服务》中的“配置 RBAC（任务列表）”。

3. **查看 `autorecovery_policy` 属性值。**

   ```
   # svccfg -s ldmd listprop ldmd/autorecovery_policy
   ```

4. **停止 `ldmd` 服务。**

   ```
   # svcadm disable ldmd
   ```

5. **更改 `autorecovery_policy` 属性值。**

   ```
   # svccfg -s ldmd setprop ldmd/autorecovery_policy=value
   ```

   例如，要将策略设置为执行自动恢复，则将属性值设置为 `3`：

   ```
   # svccfg -s ldmd setprop ldmd/autorecovery_policy=3
   ```

6. **刷新并重新启动 `ldmd` 服务。**

   ```
   # svcadm refresh ldmd
   # svcadm enable ldmd
   ```

示例 11–3 从日志修改自动恢复策略以自动恢复

以下示例显示如何查看 `autorecovery_policy` 属性的当前值并将其更改为新值。此属性的原始值为 `1`，这意味着会记录自动保存更改。`svcadm` 命令用于停止并重新启动 `ldmd` 服务，`svccfg` 命令用于查看和设置属性值。

```
# svccfg -s ldmd listprop ldmd/autorecovery_policy
ldmd/autorecovery_policy integer 1
# svcadm disable ldmd
# svccfg -s ldmd setprop ldmd/autorecovery_policy=3
# svcadm refresh ldmd
# svcadm enable ldmd
```

◆ ◆ ◆ 第 12 章

12 执行其他管理任务

本章包含前面章节中未介绍的有关使用 Oracle VM Server for SPARC 软件的信息和任务。

本章包括以下内容：

- 第 169 页中的 “在 CLI 中输入名称”
- 第 170 页中的 “通过网络连接到来宾控制台”
- 第 170 页中的 “使用控制台组”
- 第 171 页中的 “停止高负载的域会超时”
- 第 172 页中的 “操作具有 Oracle VM Server for SPARC 的 Oracle Solaris OS”
- 第 173 页中的 “将 Logical Domains Manager 与服务处理器结合使用”
- 第 174 页中的 “配置域依赖关系”
- 第 177 页中的 “通过映射 CPU 和内存地址来确定出错位置”
- 第 179 页中的 “使用通用唯一标识符”
- 第 180 页中的 “虚拟域信息命令和 API”

在 CLI 中输入名称

以下各节介绍了在 Logical Domains Manager CLI 中输入名称的限制。

文件名 (*file*) 和变量名 (*var-name*)

- 第一个字符必须是字母、数字或正斜杠 (/)。
- 后面的字符必须是字母、数字或标点。

虚拟磁盘服务器 *backend* 和虚拟交换机设备名称

名称必须包含字母、数字或标点。

配置名称 (*config-name*)

为存储在服务处理器 (service processor, SP) 上的配置指定的逻辑域配置名称 (*config-name*) 不得超过 64 个字符。

所有其他名称

其余名称，例如逻辑域名称 (*ldom*)、服务名称（*vswitch-name*、*service-name*、*vdpcs-service-name* 和 *vcc-name*）、虚拟网络名称 (*if-name*) 以及虚拟磁盘名称 (*disk-name*)，必须采用以下格式：

- 第一个字符必须是字母或数字。
- 后面的字符必须是字母、数字或以下任一字符：`-_+#.:;~()`。

通过网络连接到来宾控制台

如果在 vntsd(1M) SMF 清单中将 `listen_addr` 属性设置为控制域的 IP 地址，则可以通过网络连接到来宾控制台。例如：

```
$ telnet host-name 5001
```

注 – 启用对控制台的网络访问会有安全隐患。任何用户都可以连接到控制台，因此默认情况下它处于禁用状态。

服务管理工具清单是一个描述服务的 XML 文件。有关创建 SMF 清单的更多信息，请参阅 Oracle Solaris 10 System Administrator Collection - Simplified Chinese (`http://docs.sun.com/app/docs/coll/47.16`)。

注 – 要通过控制台访问来宾域中的非英语 OS，控制台终端必须采用该 OS 所需的语言环境。

使用控制台组

通过虚拟网络终端服务器守护进程 vntsd(1M)，您可以使用一个 TCP 端口访问多个域控制台。创建域时，Logical Domains Manager 通过为该域的控制台创建新的默认组，来为每个控制台分配唯一的 TCP 端口。该 TCP 端口随后被分配给控制台组，而非控制台本身。可以使用 `set-vcons` 子命令将控制台绑定到现有的组。

▼ 将多个控制台组成一个组

1 **将域的控制台绑定到一个组。**

以下示例说明将三个不同域（ldg1、ldg2 和 ldg3）的控制台绑定到同一控制台组 (group1)。

```
primary# ldm set-vcons group=group1 service=primary-vcc0 ldg1
primary# ldm set-vcons group=group1 service=primary-vcc0 ldg2
primary# ldm set-vcons group=group1 service=primary-vcc0 ldg3
```

2 **连接到关联的 TCP 端口（此示例中为 localhost 的端口 5000）。**

```
# telnet localhost 5000
primary-vnts-group1: h, l, c{id}, n{name}, q:
```

系统会提示您选择一个域控制台。

3 **通过选择 l（列表）列出组中的域。**

```
primary-vnts-group1: h, l, c{id}, n{name}, q: l
DOMAIN ID           DOMAIN NAME                   DOMAIN STATE
0                   ldg1                          online
1                   ldg2                          online
2                   ldg3                          online
```

注 – 要将控制台重新分配到不同的组或 vcc 实例，必须将域取消绑定；即，域必须处于非活动状态。有关配置和使用 SMF 来管理 vntsd 以及使用控制台组的更多信息，请参阅 Oracle Solaris 10 OS vntsd(1M) 手册页。

停止高负载的域会超时

ldm stop-domain 命令会在域完成关闭操作之前超时。如果发生这种情况，Logical Domains Manager 会返回一个类似以下内容的错误。

```
LDom ldg8 stop notification failed
```

但是，域可能仍在处理关闭请求。使用 ldm list-domain 命令可以检验域的状态。例如：

```
# ldm list-domain ldg8
NAME         STATE   FLAGS  CONS   VCPU MEMORY  UTIL UPTIME
ldg8         active  s----  5000   22   3328M   0.3% 1d 14h 31m
```

上面的列表显示域处于活动状态，但 s 标志指示域正在停止过程中。此状态应该是短暂状态。

以下示例说明域现在已停止。

```
# ldm list-domain ldg8
NAME         STATE   FLAGS  CONS   VCPU MEMORY  UTIL UPTIME
```

```
ldg8            bound    -----  5000   22   3328M
```

操作具有 Oracle VM Server for SPARC 的 Oracle Solaris OS

本节介绍实例化由 Logical Domains Manager 创建的配置之后，在使用 Oracle Solaris OS 方面所发生的行为变化。

OpenBoot 固件在 Oracle Solaris OS 启动之后不可用

OpenBoot 固件在 Oracle Solaris OS 启动之后不可用，因为它已从内存中删除。

要从 Oracle Solaris OS 进入 `ok` 提示符，您必须停止域。可以使用 Oracle Solaris OS `halt` 命令来停止域。

对服务器执行关开机循环

每当在运行 Oracle VM Server for SPARC 软件的系统上执行需要对服务器进行关开机循环的任何维护操作时，您必须先将当前逻辑域配置保存到 SP。

▼ 将当前域配置保存到 SP

- **使用以下命令。**

  `# ldm add-config` *config-name*

请勿对电源管理域中的活动 CPU 使用 `psradm(1M)` 命令

请勿尝试通过使用 `psradm(1M)` 命令来更改电源管理域中活动 CPU 的操作状态。

在 Oracle Solaris OS 中发出中断的结果

执行以下操作时会出现本节中所述的行为：

1. 输入设备设置为 `keyboard` 时按 L1-A 键序。
2. 虚拟控制台处于 `telnet` 提示符下时输入 `send break` 命令。

在发出这些类型的中断之后，会收到以下提示：

```
c)ontinue, s)ync, r)eset, h)alt?
```

键入代表发出这些类型的中断后希望系统执行的操作的字母。

停止或重新引导控制域的结果

下表列出了停止或重新引导控制 (primary) 域的预期行为。

表 12–1 停止或重新引导控制 (primary) 域的预期行为

命令	是否配置其他域？	行为
halt	未配置	主机断电并保持关机状态，直到 SP 通电。
	已配置	如果变量 auto-boot?=true，则软复位并引导。如果变量 auto-boot?=false，则软复位并在 ok 提示符处停止。
reboot	未配置	重新引导主机，不关闭电源。
	已配置	重新引导主机，不关闭电源。
shutdown -i 5	未配置	主机断电并保持关机状态，直到 SP 通电。
	已配置	软复位并重新引导。

有关重新引导具有根域角色的控制域的后果信息，请参见第 66 页中的 “重新引导 primary 域”。

将 Logical Domains Manager 与服务处理器结合使用

本节介绍在将 Integrated Lights Out Manager (ILOM) 服务处理器 (service processor, SP) 与 Logical Domains Manager 配合使用时要注意的信息。有关使用 ILOM 软件的更多信息，请参见特定平台的相应文档，如《Sun SPARC Enterprise T5120 and T5220 Servers Topic Set》（对于 Sun SPARC Enterprise T5120 和 T5220 服务器）。

现有的 ILOM 命令可以使用一个附加选项。

```
-> set /HOST/bootmode config=config-name
```

使用 config=*config-name* 选项可以在下次打开电源时将配置设置为其他配置，包括 factory-default 出厂配置。

无论主机的电源是打开还是关闭，您都可以调用此命令。它将在下次主机复位或打开电源时生效。

▼ 将域配置重置为默认配置或其他配置

- **通过执行以下命令，将下次打开电源时所用的逻辑域配置重置为出厂默认配置：**

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

此外，也可以选择已创建（在 Logical Domains Manager 中使用 `ldm add-config` 命令）且存储在服务处理器 (service processor, SP) 上的其他配置。执行 ILOM `bootmode` 命令时，可以使用在 Logical Domains Manager `ldm add-config` 命令中指定的名称来选择配置。例如，假设您使用名称 `ldm-config1` 存储了配置。

```
-> set /HOST/bootmode config=ldm-config1
```

现在，您必须对系统执行关开机循环以加载新配置。

有关 `ldm add-config` 命令的更多信息，请参见 `ldm(1M)` 手册页。

配置域依赖关系

可以使用 Logical Domains Manager 建立域之间的依赖关系。如果域具有一个或多个依赖于它的域，则该域称为**主域**。如果域依赖于其他域，该域称为**从属域**。

通过设置 `master` 属性，每个从属域最多可以指定四个主域。例如，`pine` 从属域在以下用逗号分隔的列表中指定其四个主域：

```
# ldm add-domain master=apple,lemon,orange,peach pine
```

每个主域都可以指定在主域失败时对其从属域产生何种影响。例如，如果主域失败，它可能会要求其从属域发生紧急情况。如果从属域具有多个主域，第一个失败的主域会触发其所有从属域上的已定义失败策略。

注 – 如果同时有多个主域失败，仅会对所有受影响的从属域强制执行一种指定的失败策略。例如，如果失败的主域具有两个失败策略，`stop` 和 `panic`，所有从属域都将被停止或发生紧急情况。

主域的失败策略是通过将以下任一值设置为 `failure-policy` 属性来控制的：

- `ignore` 在主域失败时忽略所有从属域。
- `panic` 在主域失败时使所有从属域都发生紧急情况。
- `reset` 在主域失败时重置所有从属域。
- `stop` 在主域失败时停止所有从属域。

在此示例中，主域按如下所示指定其失败策略：

```
# ldm set-domain failure-policy=ignore apple
# ldm set-domain failure-policy=panic lemon
```

```
# ldm set-domain failure-policy=reset orange
# ldm set-domain failure-policy=stop peach
```

可以使用此机制创建域之间的显式依赖关系。例如，来宾域隐式依赖于服务域以提供其虚拟设备。当来宾域所依赖的服务域未启动并运行时，来宾域的 I/O 将受到阻止。通过将来宾域定义为其服务域的从属域，可以指定来宾域在其服务域关闭时的行为。如果未建立此类依赖关系，来宾域只会等待其服务域返回到服务状态。

注 – Logical Domains Manager 不允许创建会产生依赖关系循环的域关系。有关更多信息，请参见第 176 页中的 “依赖关系循环”。

有关域依赖关系 XML 示例，请参见示例 D–6。

域依赖关系示例

以下示例说明如何配置域依赖关系。

- 第一个命令创建名为 `twizzle` 的主域。此命令使用 `failure-policy=reset` 来指定如果 `twizzle` 域失败从属域将重置。第二个命令修改名为 `primary` 的主域。此命令使用 `failure-policy=panic` 来指定如果 `primary` 域失败从属域将发生紧急情况。第三个命令创建名为 `chocktaw` 的从属域，该从属域依赖于 `twizzle` 和 `primary` 两个主域。从属域使用 `master=twizzle,primary` 来指定其主域。如果 `twizzle` 域或 `primary` 域失败，`chocktaw` 域将重置或发生紧急情况。第一个失败的主域是负责确定从属域行为的主域。

  ```
  # ldm add-domain failure-policy=reset twizzle
  # ldm set-domain failure-policy=panic primary
  # ldm add-domain master=twizzle,primary chocktaw
  ```

- 本示例说明如何使用 `ldm set-domain` 命令修改 `orange` 域，以将 `primary` 分配为主域。第二个命令使用 `ldm set-domain` 命令将 `orange` 和 `primary` 分配为 `tangerine` 域的主域。第三个命令列出有关所有这些域的信息。

  ```
  # ldm set-domain master=primary orange
  # ldm set-domain master=orange,primary tangerine
  # ldm list -o domain
  NAME             STATE      FLAGS   UTIL
  primary          active     -n-cv-  0.2%

  SOFTSTATE
  Solaris running

  HOSTID
      0x83d8b31c

  CONTROL
      failure-policy=ignore

  DEPENDENCY
  ```

```
    master=

------------------------------------------------------------------------------
NAME             STATE      FLAGS   UTIL
orange           bound      ------

HOSTID
    0x84fb28ef

CONTROL
    failure-policy=stop

DEPENDENCY
    master=primary

VARIABLES
    test_var=Aloha

------------------------------------------------------------------------------
NAME             STATE      FLAGS   UTIL
tangerine        bound      ------

HOSTID
    0x84f948e9

CONTROL
    failure-policy=ignore

DEPENDENCY
    master=orange,primary

VARIABLES
    test_var=A hui hou
```

- 以下显示了一个列有可解析输出的示例：

```
# ldm list -o domain -p
```

依赖关系循环

Logical Domains Manager 不允许创建会产生依赖关系循环的域关系。**依赖关系循环**是指会导致从属域依赖于其本身或主域依赖于其某个从属域这一情况的两个或多个域之间的关系。

Logical Domains Manager 在添加依赖关系之前确定是否存在依赖关系循环。Logical Domains Manager 自从属域开始沿着主阵列指定的所有路径进行搜索，直到到达路径的终点。沿途所发现的任何依赖关系循环都将被报告为错误。

以下示例说明了依赖关系循环可能的创建方式。第一个命令创建名为 `mohawk` 的从属域，该从属域将其主域指定为 `primary`。因此，在以下依赖关系链中 `mohawk` 依赖于 `primary`：

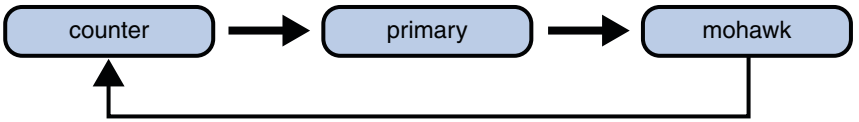
图 12–1 单个域依赖关系

第二个命令创建名为 primary 的从属域，该从属域将其主域指定为 counter。因此，在以下依赖关系链中，mohawk 依赖于 primary，而后者又依赖于 counter：

图 12–2 多个域依赖关系

第三个命令尝试创建 counter 和 mohawk 域之间的依赖关系，这种依赖关系将产生以下依赖关系循环：

图 12–3 域依赖关系循环



ldm set-domain 命令将失败，并显示以下错误消息：

```
# ldm add-domain master=primary mohawk
# ldm set-domain master=counter primary
# ldm set-domain master=mohawk counter
Dependency cycle detected: LDom "counter" indicates "primary" as its master
```

通过映射 CPU 和内存地址来确定出错位置

本节介绍如何能够将 Oracle Solaris 故障管理体系结构 (Fault Management Architecture, FMA) 报告的信息与标记为出现故障的逻辑域资源相关联。

FMA 以物理 CPU 编号形式报告 CPU 错误，以物理内存地址形式报告内存错误。

如果您想要确定发生错误的逻辑域，以及该域中相应的虚拟 CPU 编号或实际内存地址，则必须执行映射。

CPU 映射

可以使用以下过程确定域以及该域中的虚拟 CPU 编号（与给定物理 CPU 编号对应）。

▼ 确定 CPU 编号

1. 生成所有域的可解析长列表。

```
primary# ldm list -l -p
```

2. 在列表的 VCPU 部分中查找 pid 字段等于物理 CPU 编号的条目。

   - 如果找到了这样的条目，则 CPU 所在的域就是其下方列出此条目的域，该域中的虚拟 CPU 编号由此条目的 vid 字段指定。
   - 如果未找到这样的条目，则 CPU 不在任何域中。

内存映射

可以按以下方式确定域以及该域中的实际内存地址（与给定物理内存地址 (PA) 对应）。

▼ 确定实际内存地址

1. 生成所有域的可解析长列表。

```
primary# ldm list -l -p
```

2. 在列表的 MEMORY 部分查找满足以下条件的行：PA 落在 *pa* 到 (*pa* + *size* - 1) 之间（包括 pa），即，*pa* <= PA < (*pa* + *size* - 1)。

   此处 *pa* 和 *size* 指的是该行中对应字段的值。

   - 如果找到了这样的条目，则 PA 所在的域就是其下方列出此条目的域，该域中的相应实际地址由 *ra* + (PA - *pa*) 指定。
   - 如果未找到这样的条目，则 PA 不在任何域中。

CPU 和内存映射示例

假设您具有如示例 12–1 中所示的逻辑域配置，并想要确定与物理 CPU 编号 5 对应的域和虚拟 CPU，以及与物理地址 0x7e816000 对应的域和实际地址。

在列表中浏览 VCPU 条目以查找 pid 字段等于 5 的条目，可以发现以下条目位于逻辑域 ldg1 之下。

```
|vid=1|pid=5|util=29|strand=100
```

因此，物理 CPU 编号 5 位于域 ldg1 中，在该域中对应的虚拟 CPU 编号为 1。

在列表中浏览 MEMORY 条目，可以发现以下条目位于域 ldg2 之下。

```
ra=0x8000000|pa=0x78000000|size=1073741824
```

其中 0x78000000 <= 0x7e816000 <= (0x78000000 + 1073741824 - 1)；即，*pa* <= PA <= (*pa* + *size* - 1)。因此，PA 位于域 ldg2 中，相应的实际地址为 0x8000000 + (0x7e816000 - 0x78000000) = 0xe816000。

示例 12–1 Logical Domains 配置的可解析长列表

```
primary# ldm list -l -p
VERSION 1.0
DOMAIN|name=primary|state=active|flags=normal,control,vio-service|cons=SP|ncpu=4|mem=1073741824|util=0.6|
uptime=64801|softstate=Solaris running
VCPU
|vid=0|pid=0|util=0.9|strand=100
|vid=1|pid=1|util=0.5|strand=100
|vid=2|pid=2|util=0.6|strand=100
|vid=3|pid=3|util=0.6|strand=100
MEMORY
|ra=0x8000000|pa=0x8000000|size=1073741824
IO
|dev=pci@780|alias=bus_a
|dev=pci@7c0|alias=bus_b
...
DOMAIN|name=ldg1|state=active|flags=normal|cons=5000|ncpu=2|mem=805306368|util=29|uptime=903|
softstate=Solaris running
VCPU
|vid=0|pid=4|util=29|strand=100
|vid=1|pid=5|util=29|strand=100
MEMORY
|ra=0x8000000|pa=0x48000000|size=805306368
...
DOMAIN|name=ldg2|state=active|flags=normal|cons=5001|ncpu=3|mem=1073741824|util=35|uptime=775|
softstate=Solaris running
VCPU
|vid=0|pid=6|util=35|strand=100
|vid=1|pid=7|util=34|strand=100
|vid=2|pid=8|util=35|strand=100
MEMORY
|ra=0x8000000|pa=0x78000000|size=1073741824
...
```

使用通用唯一标识符

从 Oracle VM Server for SPARC 2.0 发行版开始，会为每个域分配一个通用唯一标识符 (universally unique identifier, UUID)。UUID 在创建域时分配。对于传统域，UUID 是在 ldmd 守护进程初始化时分配的。

注 – 如果使用 `ldm migrate-domain -f` 命令将域迁移到运行较早版本 Logical Domains Manager 的目标计算机，UUID 会丢失。从运行较早版本 Logical Domains Manager 的源计算机中迁移域时，会在迁移过程中为域分配新的 UUID。否则，会迁移 UUID。

可以通过运行 `ldm list -l`、`ldm list-bindings` 或 `ldm list -o domain` 命令来获取域的 UUID。以下示例显示了 `ldg1` 域的 UUID：

```
primary# ldm create ldg1
primary# ldm ls -l ldg1
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg1             inactive   ------

UUID
    6c908858-12ef-e520-9eb3-f1cd3dbc3a59

primary# ldm ls -l -p ldg1
VERSION 1.4
DOMAIN|name=ldg1|state=inactive|flags=|cons=|ncpu=|mem=|util=|uptime=
UUID|uuid=6c908858-12ef-e520-9eb3-f1cd3dbc3a59
```

虚拟域信息命令和 API

使用 `virtinfo` 命令可以收集有关运行的虚拟域的信息。还可以使用虚拟域信息 API 创建程序以收集与虚拟域相关的信息。

以下列表显示了一些可以通过使用命令或 API 收集的有关虚拟域的信息。

- 域类型（实施、控制、来宾、I/O、服务、根）
- 由虚拟域管理器确定的域名
- 域的通用唯一标识符 (universally unique identifier, UUID)
- 域的控制域的网络节点名称
- 运行域的机箱序列号

有关 `virtinfo` 命令的信息，请参见 `virtinfo(1M)` 手册页。有关 API 的信息，请参见 `libv12n(3LIB)` 和 `v12n(3EXT)` 手册页。

◆ ◆ ◆ 附 录 A

Oracle VM Server for SPARC 物理机到虚拟机转换工具

本附录包括以下内容：

- 第 181 页中的 “Oracle VM Server for SPARC P2V 工具概述”
- 第 183 页中的 “后端设备”
- 第 184 页中的 “安装 Oracle VM Server for SPARC P2V 工具”
- 第 186 页中的 “使用 `ldmp2v` 命令”

Oracle VM Server for SPARC P2V 工具概述

Oracle VM Server for SPARC P2V 工具可自动将现有物理系统转换为在芯片多线程 (chip multithreading, CMT) 系统上的逻辑域中运行的虚拟系统。源系统可以是以下任一种：

- 运行 Solaris 8 OS 或更高版本的任何基于 `sun4u` SPARC 的系统
- 运行 Oracle Solaris 10 OS，但不在逻辑域中运行的任何 `sun4v` 系统

可在以下各阶段执行从物理系统到虚拟系统的转换：

- **收集阶段**。在物理源系统上运行。在 `collect` 阶段，会基于收集到的有关源系统的配置信息创建源系统的文件系统映像。
- **准备阶段**。在目标系统的控制域上运行。在 `prepare` 阶段，会基于 `collect` 阶段收集到的配置信息在目标系统上创建逻辑域。文件系统映像将恢复到一个或多个虚拟磁盘。可以使用 P2V 工具在纯文本文件或 ZFS 卷上创建虚拟磁盘。还可以在物理磁盘、LUN 或您创建的卷管理器卷上创建虚拟磁盘。映像会被修改，以允许它作为逻辑域运行。
- **转换阶段**。在目标系统的控制域上运行。在 `convert` 阶段，将通过标准 Solaris 升级过程将创建的逻辑域转换为运行 Oracle Solaris 10 OS 的逻辑域。

有关 P2V 工具的信息，请参见 `ldmp2v(1M)` 手册页。

以下几节介绍如何在各阶段执行从物理系统到虚拟系统的转换。

收集阶段

收集阶段在要转换的系统上运行。要创建一致的文件系统映像，请确保系统尽可能以静默方式运行，且所有应用程序都已停止。`ldmp2v` 命令会创建所有已挂载 UFS 文件系统的备份，因此，请确保要移动到逻辑域的所有文件系统都已挂载。可以排除不需要移动的已挂载文件系统，例如 SAN 存储上的文件系统或将以其他方式移动的文件系统。可使用 `-x` 选项排除此类文件系统。使用 `-x` 选项排除的文件系统不会在来宾域上重新创建。可以使用 `-O` 选项排除文件和目录。

无需在源系统上进行任何更改。唯一要求是 `ldmp2v` 脚本已安装在控制域上。确保源系统上存在 `flarcreate` 实用程序。

准备阶段

准备阶段使用收集阶段收集到的数据创建与源系统相当的逻辑域。

可通过以下方式之一使用 `ldmp2v prepare` 命令：

- **自动模式。**此模式会自动创建虚拟磁盘并恢复文件系统数据。
  - 创建逻辑域以及与源系统上大小相同的所需虚拟磁盘。
  - 对磁盘进行分区并恢复文件系统。

    如果 `/`、`/usr` 和 `/var` 文件的总大小小于 10 GB，考虑到 Oracle Solaris 10 OS 的更大磁盘空间要求，将自动调整这些文件系统的大小。可以使用 `-x no-auto-adjust-fs` 选项禁用自动调整大小，或者使用 `-m` 选项手动调整文件系统的大小。
  - 修改逻辑域的 OS 映像，以使用适用于逻辑域的版本替换对物理硬件的所有引用。这样就可以通过常规 Solaris 升级过程将系统升级到 Oracle Solaris 10 OS。所做修改包括更新 `/etc/vfstab` 文件以包含新的磁盘名称。任何 Solaris 卷管理器或 Veritas 卷管理器 (Veritas Volume Manager, VxVM) 封装的引导磁盘都会在此过程中自动取消封装。磁盘会在取消封装时转换为普通磁盘分片。如果 VxVM 安装在源系统上，则 P2V 进程会在创建的来宾域上禁用 VxVM。
- **非自动模式。**必须手动创建虚拟磁盘并恢复文件系统数据。通过此模式，您能够更改磁盘的大小和数量、分区和文件系统布局。此模式中的准备阶段仅在文件系统上运行逻辑域创建和 OS 映像修改步骤。
- **清除模式。**删除逻辑域和由 `ldmp2v` 创建的所有底层后端设备。

转换阶段

在转换阶段中，逻辑域使用 Solaris 升级过程升级到 Oracle Solaris 10 OS。升级操作会删除所有现有软件包并安装 Oracle Solaris 10 `sun4v` 软件包，这会自动执行 `sun4u` 到 `sun4v` 的转换。`convert` 阶段可使用 Oracle Solaris DVD iso 映像或网络安装映像。还可以使用自定义 JumpStart 执行完全脱离手动干预的自动升级操作。

后端设备

可以基于以下多种后端类型为来宾域创建虚拟磁盘：文件 (`file`)、ZFS 卷 (`zvol`)、物理磁盘或 LUN (`disk`) 或卷管理器卷 (`disk`)。如果通过以下方式之一指定 `file` 或 `zvol` 作为后端类型，`ldmp2v` 命令将自动创建相应大小的文件或 ZFS 卷：

- 通过使用 `-b` 选项
- 通过在 `/etc/ldmp2v.conf` 文件中指定 `BACKEND_TYPE` 参数的值

通过 `disk` 后端类型，您可以将物理磁盘、LUN 或卷管理器卷（Solaris 卷管理器和 Veritas 卷管理器 (Veritas Volume Manager, VxVM)）用作虚拟磁盘的后端设备。必须在开始 `prepare` 阶段之前创建相应大小的磁盘或卷。对于物理磁盘或 LUN，请指定后端设备作为块的分片 2 或磁盘的字符设备，例如 `/dev/dsk/c0t3d0s2`。对于卷管理器卷，请为该卷指定块或字符设备，例如为 Solaris 卷管理器指定 `/dev/md/dsk/d100`，或者为 VxVM 指定 `/dev/vx/dsk/ldomdg/vol1`。

除非使用 `-B` *backend*:*volume*:*vdisk* 选项指定卷和虚拟磁盘名称，否则为来宾域创建的卷和虚拟磁盘会采用给定的默认名称。

- *backend* 用于指定要使用的后端的名称。必须为 `disk` 后端类型指定 *backend*。对于 `file` 和 `zvol` 后端类型，*backend* 是可选的，可以用于为 `ldmp2v` 创建的文件或 ZFS 卷设置非默认名称。默认名称为 `$BACKEND_PREFIX/`*guest-name*`/disk`*N*。
- *volume* 对于所有后端类型都是可选的，可指定为来宾域创建的虚拟磁盘服务器卷的名称。如果没有指定，则 *volume* 为 *guest-name*`-vol`*N*。
- *vdisk* 对于所有后端类型都是可选的，可指定来宾域中卷的名称。如果没有指定，则 *vdisk* 为 `disk`*N*。

注 – 在转换过程中，虚拟磁盘临时命名为 *guest-name*`-disk`*N* 以确保该名称在控制域中是唯一的。

要为 *backend*、*volume* 或 *vdisk* 指定一个空值，请仅包括冒号分隔符。例如，指定 `-B ::vdisk001` 会将虚拟磁盘名称设置为 `vdisk001`，并使用后端和卷的默认名称。如果没有指定 *vdisk*，可省略结尾冒号分隔符。例如，`-B /ldoms/ldom1/vol001:vol001` 将后端文件的名称指定为 `/ldoms/ldom1/vol001`，将卷名称指定为 `vol001`。默认的虚拟磁盘名称为 `disk0`。

安装 Oracle VM Server for SPARC P2V 工具

只能在目标系统的控制域上安装和配置 Oracle VM Server for SPARC P2V 工具软件包。不需要在源系统上安装该软件包。您可以直接将 `/usr/sbin/ldmp2v` 脚本从目标系统复制到源系统。

先决条件

运行 Oracle VM Server for SPARC P2V 工具之前，请确保符合以下条件：

- 源系统上已安装有下列 Flash 实用程序修补程序：
  - **对于 Solaris 8 OS**：修补程序 ID 109318-34 或更高版本
  - **对于 Solaris 9 OS**：修补程序 ID 113343-06 或更高版本
- 目标系统在以下操作系统上运行 Logical Domains 1.1 或更高版本：
  - Oracle Solaris 10 10/08 OS
  - 带有相应 Logical Domains 1.1 修补程序的 Oracle Solaris 10 5/08 OS
- 来宾域至少运行 Oracle Solaris 10 5/08 OS
- 源系统至少运行 Solaris 8 OS

除了这些先决条件外，还请配置一个要由源系统和目标系统共享的 NFS 文件系统。此文件系统应可由 `root` 写入。但是，如果共享文件系统不可用，请使用足够大的本地文件系统来存放源系统在源系统和目标系统上的文件系统转储。

限制

Oracle VM Server for SPARC P2V 工具的版本 2.0 具有以下限制：

- 仅支持 UFS 文件系统。
- 源系统上仅支持普通磁盘 (`/dev/dsk/c0t0d0s0`)、Solaris 卷管理器元设备 (`/dev/md/dsk/d`*NNN*) 和 VxVM 封装的引导磁盘。
- 在 P2V 进程执行期间，每个来宾域都只可具有一个虚拟交换机和虚拟磁盘服务器。P2V 转换后，可以将多个虚拟交换机和虚拟磁盘服务器添加到域。
- 在已封装的引导磁盘上，对 VxVM 卷的支持仅限于以下卷：`rootvol`、`swapvol`、`usr`、`var`、`opt` 和 `home`。这些卷的原始分片必须仍在引导磁盘上存在。在 Solaris 10 OS 上，P2V 工具支持 Veritas Volume Manager 5.x。不过，您也可以使用 P2V 工具转换使用 VxVM 的 Solaris 8 和 Solaris 9 操作系统。
- 无法转换配置有区域的 Solaris 10 系统。

▼ 安装 Oracle VM Server for SPARC P2V 工具

1. **转到 Oracle VM Server for SPARC 下载页面，网址为：`http://www.sun.com/servers/coolthreads/ldoms/get.jsp`。**

2. **下载 P2V 软件包 `SUNWldmp2v`。**

   从 Logical Domains 1.2 发行版开始，`SUNWldmp2v` 软件包会包含在 Oracle VM Server for SPARC zip 文件中。

3. **成为超级用户或承担等效角色。**

   角色包含授权和具有一定权限的命令。有关角色的更多信息，请参见《系统管理指南：安全性服务》中的“配置 RBAC（任务列表）”。

4. **使用 `pkgadd` 命令安装 `SUNWldmp2v` 软件包。**

   ```
   # pkgadd -d . SUNWldmp2v
   ```

5. **创建 `/etc/ldmp2v.conf` 文件并配置下列默认属性：**

   - `VDS`－虚拟磁盘服务的名称，例如 `VDS="primary-vds0"`
   - `VSW`－虚拟交换机的名称，例如 `VSW="primary-vsw0"`
   - `VCC`－虚拟控制台集中器的名称，例如 `VCC="primary-vcc0"`
   - `BACKEND_TYPE`－`zvol`、`file` 或 `disk` 的后端类型
   - `BACKEND_SPARSE`－创建后端设备用作稀疏卷或文件 (`BACKEND_SPARSE="yes"`) 还是非稀疏卷或文件 (`BACKEND_SPARSE="no"`)
   - `BACKEND_PREFIX`－用于创建虚拟磁盘后端设备的位置

     `BACKEND_TYPE="zvol"` 时，将 `BACKEND_PREFIX` 值指定为 ZFS 数据集名称。`BACKEND_TYPE="files"` 时，`BACKEND_PREFIX` 值将解释为相对于 `/` 的目录的路径名称。

     例如，`BACKEND_PREFIX="tank/ldoms"` 将导致在 `tank/ldoms/`*domain-name* 数据集中创建 ZVOL，在 `/tank/ldoms/`*domain-name* 子目录中创建文件。

     `BACKEND_PREFIX` 属性不适用于 `disk` 后端。
   - `BOOT_TIMEOUT`－Oracle Solaris OS 引导的超时时间（以秒为单位）

   有关更多信息，请参见 `ldmp2v.conf.sample` 配置文件，该配置文件包含在可下载软件包中。

使用 ldmp2v 命令

本节包含三个阶段的示例。

示例 A–1 收集阶段示例

以下示例说明如何使用 ldmp2v collect 命令。

- **共享已挂载 NFS 的文件系统。**以下示例说明了执行 collect 步骤的最简单方法，其中源系统和目标系统共享一个已挂载 NFS 的文件系统。

  作为超级用户，确保所有所需的 UFS 文件系统都已挂载。

```
volumia# df -k
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c1t1d0s0    16516485  463289 15888032     3%    /
/proc                       0       0       0     0%    /proc
fd                          0       0       0     0%    /dev/fd
mnttab                      0       0       0     0%    /etc/mnttab
/dev/dsk/c1t1d0s3    8258597    4304 8171708     1%    /var
swap                 4487448      16 4487432     1%    /var/run
swap                 4487448      16 4487432     1%    /tmp
/dev/dsk/c1t0d0s0    1016122       9  955146     1%    /u01
vandikhout:/u1/home/dana
                     6230996752 1051158977 5179837775    17%    /home/dana
```

  以下内容说明了如何在源系统和目标系统共享已挂载 NFS 的文件系统时运行收集工具：

```
volumia# ldmp2v collect -d home/dana/volumia
Collecting system configuration ...
Archiving file systems ...
Determining which filesystems will be included in the archive...
Creating the archive...
895080 blocks
Archive creation complete.
```

- **没有共享已挂载 NFS 的文件系统。**源系统和目标系统没有共享已挂载 NFS 的文件系统时，可将文件系统映像写入本地存储，稍后将其复制到控制域。Flash 实用程序会自动排除它所创建的归档文件。

```
volumia# ldmp2v collect -d /var/tmp/volumia
Collecting system configuration ...
Archiving file systems ...
Determining which filesystems will be included in the archive...
Creating the archive...
895080 blocks
Archive creation complete.
```

  将 Flash 归档文件和 manifest 文件从 /var/tmp/volumia 目录复制到目标系统。

- **跳过文件系统备份步骤。**如果系统的备份已经可以通过第三方备份工具（例如 NetBackup）使用，您可以使用 none 归档方法跳过文件系统备份步骤。使用此选项时，将仅创建系统配置清单。

```
volumia# ldmp2v collect -d /home/dana/p2v/volumia -a none
Collecting system configuration ...
```

示例 A–1 收集阶段示例 （续）

```
The following file system(s) must be archived manually: / /u01 /var
```

请注意，如果源系统和目标系统没有共享由 -d 指定的目录，您必须将该目录的内容复制到控制域。必须在准备阶段之前将目录内容复制到控制域。

示例 A–2 准备阶段示例

以下示例说明如何使用 ldmp2v prepare 命令。

- 以下示例使用 /etc/ldmp2v.conf 中配置的默认值创建名为 volumia 的逻辑域，同时保留物理系统的 MAC 地址：

```
# ldmp2v prepare -d /home/dana/p2v/volumia -o keep-mac volumia
Creating vdisks ...
Creating file systems ...
Populating file systems ...
Modifying guest domain OS image ...
Removing SVM configuration ...
Unmounting guest file systems ...
Creating domain volumia ...
Attaching vdisks to domain volumia ...
```

- 以下命令显示了有关 volumia 逻辑域的信息：

```
# ldm list -l volumia
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
volumia          inactive   ------          2     4G

NETWORK
    NAME    SERVICE                 DEVICE     MAC               MODE   PVID VID
    vnet0   primary-vsw0                       00:03:ba:1d:7a:5a        1

DISK
    NAME    DEVICE  TOUT MPGROUP        VOLUME                       SERVER
    disk0                                       volumia-vol0@primary-vds0
    disk1                                       volumia-vol1@primary-vds0
```

- 以下内容显示可以使用 -C 选项完全删除域及其后端设备：

```
# ldmp2v prepare -C volumia
Cleaning up domain volumia ...
Removing vdisk disk0 ...
Removing vdisk disk1 ...
Removing domain volumia ...
Removing volume volumia-vol0@primary-vds0 ...
Removing ZFS volume tank/ldoms/volumia/disk0 ...
Removing volume volumia-vol1@primary-vds0 ...
Removing ZFS volume tank/ldoms/volumia/disk1 ...
```

- 以下内容显示可以通过使用 -m 选项指定挂载点和新大小，在 P2V 过程中调整一个或多个文件系统的大小。

```
# ldmp2v prepare -d /home/dana/p2v/normaal -m /:8g normaal
Resizing file systems ...
Creating vdisks ...
Creating file systems ...
```

示例 A–2 准备阶段示例 （续）

```
Populating file systems ...
Modifying guest domain OS image ...
Removing SVM configuration ...
Modifying file systems on SVM devices ...
Unmounting guest file systems ...
Creating domain normaal ...
Attaching vdisks to domain normaal ...
```

示例 A–3 转换阶段示例

以下示例说明如何使用 ldmp2v convert 命令。

- **使用网络安装服务器。**ldmp2v convert 命令可使用指定的虚拟网络接口通过网络引导域。必须在安装服务器上运行 setup_install_server 和 add_install_client 脚本。

  可以使用自定义 JumpStart 功能执行完全脱离手动干预的转换。此功能需要您在 JumpStart 服务器上为客户机创建和配置相应的 sysidcfg 文件和配置文件。该配置文件应包含以下行：

  ```
  install_type    upgrade
  root_device     c0d0s0
  ```

  sysidcfg 文件仅用于升级操作，因此如下配置应已足够：

  ```
  name_service=NONE
  root_password=uQkoXlMLCsZhI
  system_locale=C
  timeserver=localhost
  timezone=Europe/Amsterdam
  terminal=vt100
  security_policy=NONE
  nfs4_domain=dynamic
  network_interface=PRIMARY {netmask=255.255.255.192
          default_route=none protocol_ipv6=no}
  ```

  有关使用自定义 JumpStart 的更多信息，请参见《Oracle Solaris 10 9/10 安装指南：自定义 JumpStart 和高级安装》。

  ```
  # ldmp2v convert -j -n vnet0 -d /p2v/volumia volumia
  LDom volumia started
  Waiting for Solaris to come up ...
  Using Custom JumpStart
  Trying 0.0.0.0...
  Connected to 0.
  Escape character is '^]'.

  Connecting to console "volumia" in group "volumia" ....
  Press ~? for control options ..
  SunOS Release 5.10 Version Generic_137137-09 64-bit
  Copyright (c) 1983-2010, Oracle and/or its affiliates.  All rights reserved.
  Configuring devices.
  Using RPC Bootparams for network configuration information.
  Attempting to configure interface vnet0...
  ```

示例 A–3 转换阶段示例 （续）

```
Configured interface vnet0
Reading ZFS config: done.
Setting up Java. Please wait...
Serial console, reverting to text install
Beginning system identification...
Searching for configuration file(s)...
Using sysid configuration file
  129.159.206.54:/opt/SUNWjet/Clients/volumia/sysidcfg
Search complete.
Discovering additional network configuration...
Completing system identification...
Starting remote procedure call (RPC) services: done.
System identification complete.
Starting Solaris installation program...
Searching for JumpStart directory...
Using rules.ok from 129.159.206.54:/opt/SUNWjet.
Checking rules.ok file...
Using begin script: Clients/volumia/begin
Using profile: Clients/volumia/profile
Using finish script: Clients/volumia/finish
Executing JumpStart preinstall phase...
Executing begin script "Clients/volumia/begin"...
Begin script Clients/volumia/begin execution completed.
Searching for SolStart directory...
Checking rules.ok file...
Using begin script: install_begin
Using finish script: patch_finish
Executing SolStart preinstall phase...
Executing begin script "install_begin"...
Begin script install_begin execution completed.
WARNING: Backup media not specified.  A backup media (backup_media)
  keyword must be specified if an upgrade with disk space reallocation
  is required

Processing profile

Loading local environment and services

Generating upgrade actions
Checking file system space: 100% completed
Space check complete.

Building upgrade script

Preparing system for Solaris upgrade

Upgrading Solaris: 10% completed
[...]
```

- **使用 ISO 映像**。`ldmp2v convert` 命令可将 Oracle Solaris DVD ISO 映像附加到逻辑域并从中进行引导。要进行升级，请回答所有 `sysid` 提示，然后选择“升级”。

示例 A–3 转换阶段示例 （续）

注 – 对 sysid 问题的回答仅用于升级过程进行期间。此数据不会应用至磁盘上的现有 OS 映像。执行转换的最快速、最简单的方法是选择“非联网”。指定的 root 密码不需要与源系统的 root 密码匹配。升级操作会保留系统的原始标识，该标识将在升级后的重新引导后生效。执行升级所需的时间取决于原始系统上安装的 Oracle Solaris Cluster。

```
# ldmp2v convert -i /tank/iso/s10s_u5.iso -d /home/dana/p2v/volumia volumia
Testing original system status ...
LDom volumia started
Waiting for Solaris to come up ...

        Select 'Upgrade' (F2) when prompted for the installation type.
        Disconnect from the console after the Upgrade has finished.

Trying 0.0.0.0...
Connected to 0.
Escape character is '^]'.

Connecting to console "volumia" in group "volumia" ....
Press ~? for control options ..
Configuring devices.
Using RPC Bootparams for network configuration information.
Attempting to configure interface vnet0...
Extracting windowing system. Please wait...
Beginning system identification...
Searching for configuration file(s)...
Search complete.
Discovering additional network configuration...
Configured interface vnet0
Setting up Java. Please wait...

Select a Language

   0. English
   1. French
   2. German
   3. Italian
   4. Japanese
   5. Korean
   6. Simplified Chinese
   7. Spanish
   8. Swedish
   9. Traditional Chinese

Please make a choice (0 - 9), or press h or ? for help:
[...]
- Solaris Interactive Installation ----------------------------------------------

  This system is upgradable, so there are two ways to install the Solaris
  software.

  The Upgrade option updates the Solaris software to the new release, saving
  as many modifications to the previous version of Solaris software as
```

示例 A–3 转换阶段示例 （续）

```
possible.  Back up the system before using the Upgrade option.

The Initial option overwrites the system disks with the new version of
Solaris software.  This option allows you to preserve any existing file
systems.  Back up any modifications made to the previous version of Solaris
software before starting the Initial option.

After you select an option and complete the tasks that follow, a summary of
your actions will be displayed.

--------------------------------------------------------------------------------
    F2_Upgrade    F3_Go Back    F4_Initial    F5_Exit    F6_Help
```

◆ ◆ ◆ 附 录 B

Oracle VM Server for SPARC Configuration Assistant

Oracle VM Server for SPARC Configuration Assistant 将引导您完成通过设置基本属性配置逻辑域的过程。它在基于芯片多线程 (chip multithreading, CMT) 的系统上运行。

收集配置数据之后，Configuration Assistant 将创建适合于作为逻辑域引导的配置。还可以使用 Configuration Assistant 选定的默认值创建可用的系统配置。

Configuration Assistant 可同时作为图形用户界面 (graphical user interface, GUI) 和基于终端的工具 `ldmconfig` 提供。

有关基于终端的工具的信息，请参见第 194 页中的 “使用 Configuration Assistant (ldmconfig)”和 `ldmconfig(1M)` 手册页。

有关启动 GUI 工具的信息，请参见第 193 页中的 “使用 Configuration Assistant (GUI)”。

使用 Configuration Assistant (GUI)

Configuration Assistant GUI 作为 Oracle VM Server for SPARC zip 包的一部分提供。

确保目标系统至少运行 Logical Domains 1.2 软件，并且您的系统至少运行 Java SE Runtime Environment 1.6 版。

要从命令行运行 Configuration Assistant GUI，请键入以下命令：

```
$ java -jar "Configurator.jar"
```

此 GUI 工具包括屏幕上的文档，可帮助您创建适用于您系统的配置。

使用 Configuration Assistant (ldmconfig)

基于终端的 Configuration Assistant (ldmconfig) 通过与用户界面屏幕相对应的一系列操作工作。最终结果是创建可部署到逻辑域的配置。

以下各节说明了如何安装 ldmconfig 命令以及 Configuration Assistant 工具的一些功能。

安装 Configuration Assistant

Configuration Assistant 作为 SUNWldm 软件包的一部分提供。

安装 SUNWldm 软件包之后，可以在 /usr/sbin 目录中找到 ldmconfig 命令。出于传统目的，该命令同时安装于 /opt/SUNWldm/bin 目录。

先决条件

安装并运行 Configuration Assistant 之前，请确保满足以下条件：

- 目标系统必须至少运行 Logical Domains 1.2 软件。
- 您的终端窗口必须至少 80 个字符宽、24 行长。

限制和已知问题

Configuration Assistant 有以下限制：

- 使用 ldmconfig 时调整终端大小可能会导致输出乱码
- 仅支持 UFS 磁盘文件作为虚拟磁盘
- 仅可在不存在现有逻辑域配置的系统上工作
- 虚拟控制台集中器端口从 5000 到 5100
- 无法更改用于来宾域、服务和设备的默认名称

ldmconfig 功能

基于终端的 Configuration Assistant (ldmconfig) 通过与用户界面屏幕相对应的一系列操作工作。可以在这些屏幕中向后（上一步）和向前（下一步）导航，直到到达最后一步。最后一步将生成配置。任何时候都可以退出 Configuration Assistant 或重置配置以使用默认值。在最终屏幕中，可以将配置部署到逻辑域。

首先，Configuration Assistant 自动检查系统以根据最佳实践确定最合适的默认属性值，然后显示控制部署所需的这些属性。请注意，这不是详尽的列表。可以设置其他属性以进一步自定义配置。

有关使用 ldmconfig 工具的信息，请参见 ldmconfig(1M) 手册页。

您可以调整以下属性：

- **来宾域数量。**指定应用程序可创建的来宾域数量。最少为一个来宾域。最大值由 VCPU 资源的可用性确定。例如，在 64 线程的 CMT 系统上，最多可以创建 60 个来宾域（每个域一个线程，保留四个线程用于控制域）。如果选择了最佳实践，则每个来宾域的最小 VCPU 资源数是单个内核。因此，在已选择最佳实践的 8 内核、每内核 8 线程的系统上，最多可以创建七个来宾域，每个域一个内核。此外，将一个内核分配给控制域。

  Configuration Assistant 显示可为此系统配置的域的最大数目。

  Configuration Assistant 执行以下任务来创建域：

  - **对于所有域。**
    - 在端口 5000 到 5100 上创建虚拟终端服务
    - 创建虚拟磁盘服务
    - 在指定的网络适配器上创建虚拟网络交换机
    - 启用虚拟终端服务器守护进程
  - **对于每个域。**
    - 创建逻辑域
    - 配置分配给域的 VCPU
    - 配置分配给域的内存
    - 创建用作虚拟磁盘的 UFS 磁盘文件
    - 为磁盘文件创建虚拟磁盘服务器设备 (`vdsdev`)
    - 将磁盘文件指定为域的虚拟磁盘 `vdisk0`
    - 添加连接到指定的网络适配器上的虚拟交换机的虚拟网络适配器
    - 设置 OBP 属性 `auto-boot?=true`
    - 设置 OBP 属性 `boot-device=vdisk0`
    - 绑定域
    - 启动域
- **默认网络。**指定新域将用于虚拟网络的网络适配器。该适配器必须存在于系统中。Configuration Assistant 将突出显示系统当前用作默认适配器的适配器，以及具有活动链路状态的适配器（已连线的适配器）。
- **虚拟磁盘大小。**为每个新域创建虚拟磁盘。这些虚拟磁盘是根据位于本地文件系统中的磁盘文件创建的。此属性控制每个虚拟磁盘的大小（以 GB 为单位）。最小大小 8 GB 基于包含 Oracle Solaris 10 OS 所需的近似大小，最大大小为 100 GB。

  如果 Configuration Assistant 找不到具有足够空间来包含所有域的磁盘文件的文件系统，将显示错误屏幕。这种情况下，可能需要在重新运行应用程序之前执行以下操作：
  - 减小虚拟磁盘的大小
  - 减少域数
  - 添加更多更大容量的文件系统

- **虚拟磁盘目录。**指定具有足够容量的文件系统，用于存储将作为新域的虚拟磁盘创建的文件。该目录基于选定的域数和虚拟磁盘的大小。每次更改这些属性值时，必须重新计算该值并选择目标目录。Configuration Assistant 提供具有足够空间的文件系统列表。指定文件系统名称之后，Configuration Assistant 在此文件系统中创建名为 `/ldoms/disks` 的目录，在该目录中创建磁盘映像。
- **最佳实践。**指定是否对属性值使用最佳实践。
  - 值为 `yes` 时，Configuration Assistant 将为几个配置属性值使用最佳实践。它将强制使用每个域一个内核的最小值，包括系统域。因此，这会将来宾域的最大数限制为系统中存在的内核总数减去用于系统域的一个内核。例如，对于每路八个内核的双路 SPARC Enterprise T5140，来宾域的最大数是 15（加上系统域）。
  - 值为 `no` 时，Configuration Assistant 允许创建最少有一个线程的域，但至少会保留四个线程用于系统域。

接下来，Configuration Assistant 将概述要创建的部署配置，其中包括以下信息：

- 域数量
- 分配给每个来宾域的 CPU
- 分配给每个来宾域的内存
- 虚拟磁盘的大小和位置
- 将用于来宾域的虚拟网络服务的网络适配器
- 系统将用于服务的 CPU 和内存量
- 如果识别出有效的 Oracle Solaris OS DVD，将使用它来创建共享虚拟 CD-ROM 设备，以允许来宾域安装 Oracle Solaris OS

最后，Configuration Assistant 将配置系统，以创建指定的 Logical Domains 部署。它还会说明要采取的操作并显示要运行的命令，以配置系统。此信息可帮助您了解如何使用配置系统所需的 `ldm` 命令。

注意 – 不要与此配置步骤进行交互且**不要**中断此过程，否则可能会导致部分配置的系统。

命令成功完成后，重新引导系统以使更改生效。

◆ ◆ ◆ 附录 C

Logical Domains Manager 发现

通过使用多播消息，可在子网上发现 &Manager。ldmd 守护进程能够在网络上侦听特定的多播包。如果此多播消息为某种特定类型，ldmd 将回复调用方。这样，就可以在运行 Oracle VM Server for SPARC 的系统上发现 ldmd。

本附录提供了有关发现在子网的系统中运行的 Logical Domains Manager 的信息。

发现运行 Logical Domains Manager 的系统

多播通信

此发现机制使用 ldmd 守护进程所使用的多播网络，以检测自动分配 MAC 地址时所产生的冲突。要配置多播套接字，必须提供以下信息：

```
#define    MAC_MULTI_PORT        64535
#define    MAC_MULTI_GROUP        "239.129.9.27"
```

默认情况下，在计算机已连接到的子网上**只能**发送多播包。可以通过设置 ldmd 守护进程的 ldmd/hops SMF 属性来更改以上行为。

消息格式

发现消息必须清晰标记，才能与其他消息区分开。以下多播消息格式确保发现侦听进程可以区分发现消息：

```
#include <netdb.h> /* Used for MAXHOSTNAMELEN definition */
#define    MAC_MULTI_MAGIC_NO    92792004
#define    MAC_MULTI_VERSION    1

enum {
```

```
    SEND_MSG = 0,
    RESPONSE_MSG,
    LDMD_DISC_SEND,
    LDMD_DISC_RESP,
};

typedef struct {
    uint32_t    version_no;
    uint32_t    magic_no;
    uint32_t    msg_type;
    uint32_t    resv;
    union {
        mac_lookup_t        Mac_lookup;
        ldmd_discovery_t    Ldmd_discovery;
    } payload;
#define    lookup        payload.Mac_lookup
#define    discovery    payload.Ldmd_discovery
} multicast_msg_t;

#define    LDMD_VERSION_LEN    32

typedef struct {
    uint64_t mac_addr;
    char        source_ip[INET_ADDRSTRLEN];
} mac_lookup_t;

typedef struct {
    char        ldmd_version[LDMD_VERSION_LEN];
    char        hostname[MAXHOSTNAMELEN];
    struct in_addr    ip_address;
    int        port_no;
} ldmd_discovery_t;
```

▼ 发现在子网上运行的 Logical Domains Manager

1. **打开多播套接字。**
   确保您使用的是第 197 页中的“多播通信”中指定的端口和组信息。

2. **通过套接字发送 `multicast_msg_t` 消息。**
   此消息应包含以下内容：
   - `version_no` 的有效值，由 `MAC_MULTI_VERSION` 定义，其值为 `1`。
   - `magic_no` 的有效值，由 `MAC_MULTI_MAGIC_NO` 定义，其值为 `92792004`。
   - `LDMD_DISC_SEND` 的 `msg_type`

3. **侦听多播套接字，以获取来自 Logical Domains Manager 的响应。**
   这些响应必须是包含以下内容的 `multicast_msg_t` 消息：
   - `version_no` 的有效值
   - `magic_no` 的有效值
   - 已设置为 `LDMD_DISC_RESP` 的 `msg_type`

- 有效荷载包含 `ldmd_discovery_t` 结构，其中包含以下信息：
  - `ldmd_version` — 在系统上运行的 Logical Domains Manager 的版本
  - `hostname` — 系统的主机名称
  - `ip_address` — 系统的 IP 地址
  - `port_no` — Logical Domains Manager 用于通信的端口号，应该是 XMPP 端口 `6482`

当侦听来自 Logical Domains Manager 的响应时，请确保已放弃任何自动分配 MAC 冲突检测包。

◆ ◆ ◆ 附 录 D

将 XML 接口与 Logical Domains Manager 结合使用

本章介绍可扩展标记语言 (Extensible Markup Language, XML) 通信机制，通过该机制，可连接外部用户程序和 Oracle VM Server for SPARC 软件。本章包含下列基本主题：

- 第 201 页中的 “XML 传输”
- 第 202 页中的 “XML 协议”
- 第 206 页中的 “事件消息”
- 第 210 页中的 “Logical Domains Manager 操作”
- 第 211 页中的 “Logical Domains Manager 资源和属性”

有关与 Logical Domains Manager 结合使用的各种模式，请参见附录 E，Logical Domains Manager XML 模式。

XML 传输

外部程序可利用可扩展消息处理现场协议 (Extensible Messaging and Presence Protocol, XMPP – RFC 3920) 与 Logical Domains Manager 进行通信。XMPP 受本地和远程连接支持，且默认情况下处于启用状态。要关闭远程连接，请将 `ldmd/xmpp_enabled` SMF 属性设置为 `false` 并重新启动 Logical Domains Manager。

```
# svccfg -s ldom/ldmd setprop ldmd/xmpp_enabled=false
# svcadm refresh ldmd
# svcadm restart ldmd
```

注 – 禁用 XMPP 服务器还会阻止域迁移和内存动态重新配置。

XMPP 服务器

Logical Domains Manager 实现了一个能够与许多可用 XMPP 客户机应用程序和库进行通信的 XMPP 服务器。Logical Domains Manager 使用下列安全机制：

- 传输层安全 (Transport Layer Security, TLS)，用于保护客户机与其自身之间的通信通道。
- 简单身份验证和安全层 (Simple Authentication and Security Layer, SASL)，用于身份验证。`PLAIN` 是唯一受支持的 SASL 机制。您必须将用户名和密码发送到服务器，以便服务器可在允许监视或管理操作之前为您授权。

本地连接

Logical Domains Manager 会检测用户客户机是否与其在同一域中运行，如果是，则与该客户机进行最小 XMPP 握手。具体地说，通过 TLS 设置安全通道后的 SASL 身份验证步骤将被跳过。身份验证和授权的完成都将基于实现客户机接口的进程的凭据。

客户机可以选择实现完全 XMPP 客户机，也可以选择仅运行进行流化处理的 XML 解析器，例如 `libxml2` XML 简单 API (Simple API for XML, SAX) 解析器。无论采用哪种方式，客户机在进行 TLS 协商时都必须处理 XMPP 握手。有关所需次序，请参阅 XMPP 规范。

XML 协议

完成通信初始化后，接下来会发送 Logical Domains 定义的 XML 消息。有以下两种常规 XML 消息类型：

- 请求和响应消息使用 `<LDM_interface>` 标记。此类型的 XML 消息用于传送命令并从 Logical Domains Manager 获取返回的结果，类似于使用命令行界面 (command-line interface, CLI) 执行命令。此标记还用于事件的注册和注销。
- 事件消息使用 `<LDM_event>` 标记。此类型的 XML 消息用于异步报告 Logical Domains Manager 发布的事件。

请求和响应消息

结合到 Logical Domains 中的 XML 接口有两种不同的格式：

- 一种格式用于向 Logical Domains Manager 中发送命令
- 另一种格式用于 Logical Domains Manager 响应传入消息的状态以及该消息内请求的操作。

这两种格式共享许多共同的 XML 结构，但为了更好地理解它们的差异，在此分别对它们进行讨论。本文档还包含一个 XML 模式，其中详述了传入和传出组合 XML（请参见第 225 页中的 “LDM_Event XML 模式”）。

请求消息

对 Logical Domains Manager 的最基本的传入 XML 请求包括在单个对象上执行的单个命令的描述。更加复杂的请求可以处理多个命令以及每个命令对应的多个对象。下面是 XML 基本命令的结构。

示例 D–1 在单个对象上执行的单个命令的格式

```
<LDM_interface version="1.0">
  <cmd>
    <action>Place command here</action>
    <option>Place options for certain commands here</option>
    <data version="3.0">
      <Envelope>
        <References/>
        <!-- Note a <Section> section can be here instead of <Content> -->
        <Content xsi:type="ovf:VirtualSystem_Type" id="Domain name">
          <Section xsi:type="ovf:ResourceAllocationSection_type">
            <Item>
              <rasd:OtherResourceType>LDom Resource Type</rasd:OtherResourceType>
              <gprop:GenericProperty
              key="Property name">Property Value</gprop:GenericProperty>
            </Item>
          </Section>
          <!-- Note: More Sections sections can be placed here -->
        </Content>
      </Envelope>
    </data>
    <!-- Note: More Data sections can be placed here -->
  </cmd>
  <!-- Note: More Commands sections can be placed here -->
</LDM_interface>
```

<LDM_interface> 标记

发送给 Logical Domains Manager 的所有命令都必须以 <LDM_interface> 标记开头。发送到 Logical Domains Manager 的任何文档内必须仅包含一个 <LDM_interface> 标记。<LDM_interface> 标记必须包含如示例 D–1 所示的版本属性。

<cmd> 标记

在 <LDM_interface> 标记内，文档必须至少包含一个 <cmd> 标记。每个 <cmd> 段必须仅包含一个 <action> 标记。<action> 标记用于描述要运行的命令。每个 <cmd> 标记必须至少包含一个 <data> 标记，用于描述要在其上执行命令的对象。

<cmd> 标记还可以包含一个 <option> 标记，用于与某些命令关联的选项和标志。下列命令使用选项：

- remove-domain 命令可使用 -a 选项。
- stop-domain 命令可使用 -f 选项。
- cancel-operation 命令可使用 migration 或 reconf 选项。
- add-spconfig 命令可使用 -r *autosave-name* 选项。
- remove-spconfig 命令可使用 -r 选项。
- list-spconfig 命令可使用 -r [*autosave-name*] 选项。

<data> 标记

每个 <data> 段都包含与指定命令相关的对象的描述。数据段的格式基于开放虚拟化格式 (Open Virtualization Format, OVF) 规范草案中的 XML 模式部分。该模式定义了一个 <Envelope> 段（其中包含一个 <References> 标记，未被 Logical Domains 使用）以及 <Content> 和 <Section> 段。

对于 Logical Domains，<Content> 段用于标识和描述特定域。<Content> 节点的 id= 属性中的域名用于标识域。<Content> 段中有一个或多个 <Section> 段，它们根据特定命令需要来描述域的资源。

如果您仅需要标识一个域名，则无需使用任何 <Section> 标记。相反，如果命令不需要任何域标识符，则需要提供一个 <Section> 段（在 <Content> 段之外，但仍在 <Envelope> 段之内）来描述命令所需的资源。

在可以推断出对象信息的情况下，<data> 段不需要包含 <Envelope> 标记。此情况主要适用于对监控适用于操作的所有对象的请求以及事件注册和注销请求。

为允许使用 OVF 规范的模式来正确定义所有类型的对象，定义了两种附加的 OVF 类型：

- <gprop:GenericProperty> 标记（请参见 第 242 页中的 “GenericProperty XML 模式”）。
- <Binding> 标记（请参见 第 242 页中的 “Binding_Type XML 模式”）。

<gprop:GenericProperty> 标记被定义用于处理 OVF 规范未包含其定义的任意对象的属性。属性名称在节点的 key= 属性中定义，属性的值是节点的内容。<binding> 标记在 list-bindings 子命令输出使用，用于定义绑定到其他资源的资源。

响应消息

从包含的命令和对象上来讲，传出 XML 响应的结构与传入请求的结构很相似，只是添加了针对指定的每个对象和命令的 <Response> 段以及针对请求的总体 <Response> 段。<Response> 段提供如示例 D–2 所述的状态和消息信息。下面是对基本 XML 请求的响应的结构。

示例 D–2 对单个对象上执行的单个命令的响应的格式

```
<LDM_interface version="1.0">
  <cmd>
    <action>Place command here</action>
    <data version="3.0">
      <Envelope>
        <References/>
        <!-- Note a <Section> section can be here instead of <Content> -->
        <Content xsi:type="ovf:VirtualSystem_Type" id="Domain name">
          <Section xsi:type="ovf:ResourceAllocationSection_type">
            <Item>
              <rasd:OtherResourceType>
                LDom Resource Type
              </rasd:OtherResourceType>
              <gprop:GenericProperty
              key="Property name">
                Property Value
            </gprop:GenericProperty>
            </Item>
          </Section>
          <!-- Note: More <Section> sections can be placed here -->
        </Content>
      </Envelope>
      <response>
        <status>success or failure</status>
        <resp_msg>Reason for failure</resp_msg>
      </response>
    </data>
    <!-- Note: More Data sections can be placed here -->
    <response>
      <status>success or failure</status>
      <resp_msg>Reason for failure</resp_msg>
    </response>
  </cmd>
  <!-- Note: More Command sections can be placed here -->
  <response>
    <status>success or failure</status>
    <resp_msg>Reason for failure</resp_msg>
  </response>
</LDM_interface>
```

总体响应

此 <response> 段是 <LDM_interface> 段的直接子级，用于指示整个请求的整体成功或失败。除非传入 XML 文档格式错误，否则，<response> 段仅包含一个 <status> 标记。如果此响应状态指示成功，则说明所有对象上的所有命令都已成功。如果此响应状态为失败且没有 <resp_msg> 标记，则说明原始请求中的其中一个命令失败。<resp_msg> 标记仅用于描述 XML 文档自身的一些问题。

命令响应

<cmd> 段下的 <response> 段通知用户特定命令是成功还是失败。<status> 标记显示命令是成功还是失败。与总响应一样，如果命令失败，<response> 段仅包含一个 <resp_msg> 标记（如果请求的 <cmd> 段的内容格式错误）。否则，失败状态表示针对其运行该命令的其中一个对象导致了故障。

对象响应

最后，<cmd> 段中的每个 <data> 段还包含一个 <response> 段。它显示在此特定对象上运行的命令是成功还是失败。如果响应的状态是 SUCCESS，则 <response> 段中不会出现 <resp_msg> 标记。如果状态是 FAILURE，则 <response> 字段中会出现一个或多个 <resp_msg> 标记，具体取决于针对该对象运行命令时遇到的错误。运行命令时出现问题、对象格式错误或未知均可导致对象错误。

除 <response> 段外，<data> 段还可以包含其他信息。此信息的格式与传入 <data> 字段的格式相同，用于描述导致失败的对象。请参见 第 204 页中的 “<data> 标记”。此附加信息在以下情况下非常有用：

- 当某命令针对特定 <data> 段失败而针对所有其他 <data> 段成功时
- 当空的 <data> 段传入命令并对一些域失败，而对其他域成功时

事件消息

您可以订阅以接收发生的某些状态更改的事件通知，而不使用轮询。有三种可单独或集体订阅的事件类型。有关完整的详细信息，请参见第 208 页中的 “事件类型”。

注册和注销

使用 <LDM_interface> 消息可注册事件。请参见 第 203 页中的 “<LDM_interface> 标记”。操作标记详述了要注册或注销的事件的类型，<data> 段保留为空。

示例 D–3 事件注册请求消息示例

```
<LDM_interface version="1.0">
  <cmd>
    <action>reg-domain-events</action>
    <data version="3.0"/>
  </cmd>
</LDM_interface>
```

Logical Domains Manager 通过表示注册或注销是否成功的 <LDM_interface> 响应消息进行响应。

示例 D–4 事件注册响应消息示例

```
<LDM_interface version="1.0">
  <cmd>
    <action>reg-domain-events</action>
    <data version="3.0"/>
      <response>
        <status>success</status>
      </response>
    </data>
    <response>
      <status>success</status>
    </response>
  </cmd>
  <response>
    <status>success</status>
  </response>
</LDM_interface>
```

在事件子段中会列出每个类型的事件的操作字符串。

<LDM_event> 消息

事件消息除其开始标记是 <LDM_event> 之外，其格式与传入 <LDM_interface> 消息相同。消息的操作标记是为触发事件而执行的操作。消息的数据段描述与事件关联的对象，其详细信息取决于发生的事件的类型。

示例 D–5 <LDM_event> 通知示例

```
<LDM_event version='1.0'>
  <cmd>
    <action>Event command here</action>
    <data version='3.0'>
      <Envelope
        <References/>
        <Content xsi:type='ovf:VirtualSystem_Type' ovf:id='ldg1'/>
          <Section xsi:type="ovf:ResourceAllocationSection_type">
            <Item>
              <rasd:OtherResourceType>LDom Resource Type</rasd:OtherResourceType>
              <gprop:GenericProperty
              key="Property name">Property Value</gprop:GenericProperty>
            </Item>
          </Section>
      </Envelope>
    </data>
  </cmd>
</LDM_event>
```

事件类型

下面是可订阅的事件类型：

- 域事件
- 硬件事件
- 进度事件
- 资源事件

所有事件都与 `ldm` 子命令相对应。

域事件

域事件描述可对域直接执行的操作。下表显示可在 `<LDM_event>` 消息中的 `<action>` 标记中列出的域事件。

域事件	域事件	域事件
`add-domain`	`remove-domain`	`bind-domain`
`unbind-domain`	`start-domain`	`stop-domain`
`domain-reset`	`panic-domain`	`migrate-domain`

这些事件在 OVF 数据段中始终**仅**包含一个 `<Content>` 标记，用于描述发生事件的域。要注册域事件，需要发送 `<action>` 标记设置为 **`reg-domain-events`** 的 `<LDM_interface>` 消息。要注销这些事件，需要使用操作标记设置为 **`unreg-domain-events`** 的 `<LDM_interface>` 消息。

硬件事件

硬件事件与更改物理系统硬件相关。对于 Oracle VM Server for SPARC 软件，唯一可执行的硬件更改是用户添加、删除或设置服务处理器 (service processor, SP) 配置时对其进行的更改。当前，适用于此情况的仅有三种事件：

- `add-spconfig`
- `set-spconfig`
- `remove-spconfig`

硬件事件在 OVF 数据段中始终**仅**包含一个 `<Section>` 标记，用于描述发生此事件的 SP 配置。要注册这些事件，需要发送 `<action>` 标记设置为 **`reg-hardware-events`** 的 `<LDM_interface>` 消息。要注销这些事件，需要使用 `<action>` 标记设置为 **`unreg-hardware-events`** 的 `<LDM_interface>` 消息。

进度事件

进度事件是针对长时间运行的命令（例如，域迁移）发布的。这些事件可报告在运行命令期间完成的进展程度。目前，仅报告 `migration-process` 事件。

进度事件在 OVF 数据段中始终**仅**包含一个 <Section> 标记，用于描述受此事件影响的 SP 配置。要注册这些事件，需要发送 <action> 标记设置为 reg-hardware-events 的 <LDM_interface> 消息。要注销这些事件，需要使用 <action> 标记设置为 unreg-hardware-events 的 <LDM_interface> 消息。

进度事件的 <data> 段包含一个 <content> 段，用于描述受影响的域。此 <content> 段使用 ldom_info <Section> 标记来更新进度。下列常规属性显示于 ldom_info 段中：

- --progress－命令执行的进度百分比
- --status－命令状态，可以是 ongoing、failed 或 done 中的一种。
- --source－报告进度的计算机

资源事件

在任意域中添加、删除或更改资源时，会发生资源事件。其中一些事件的数据段在 OVF 数据段中包含带有 <Section> 标记（可提供服务名称）的 <Content> 标记。下表显示可在 <LDM_event> 消息中的 <action> 标记中列出的事件。

资源事件	资源事件
add-vdiskserverdevice	remove-vdiskserverdevice
set-vdiskserverdevice	remove-vdiskserver
set-vconscon	remove-vconscon
set-vswitch	remove-vswitch
remove-vdpcs	

其余资源事件在 OVF 数据段中始终**仅**包含 <Content> 标记，用于描述发生事件的域。

资源事件	资源事件	资源事件
add-vcpu	add-crypto	add-memory
add-io	add-variable	add-vconscon
add-vdisk	add-vdiskserver	add-vnet
add-vswitch	add-vdpcs	add-vdpcc
set-vcpu	set-crypto	set-memory
set-variable	set-vnet	set-vconsole
set-vdisk	remove-vcpu	remove-crypto
remove-memory	remove-io	remove-variable

资源事件	资源事件	资源事件
remove-vdisk	remove-vnet	remove-vdpcc

要注册资源事件，需要发送 <action> 标记设置为 **reg-resource-events** 的 <LDM_interface> 消息。要注销这些事件，需要使用 <action> 标记设置为 **unreg-resource-events** 的 <LDM_interface> 消息。

所有事件

您还可以注册以侦听所有三种类型的事件，而无需单独注册每个事件。要同时注册所有三种类型的事件，需要发送 <action> 标记设置为 **reg-all-events** 的 <LDM_interface> 消息。要注销这些事件，需要使用 <action> 标记设置为 **unreg-all-events** 的 <LDM_interface> 消息。

Logical Domains Manager 操作

<action> 标记中指定的命令（*-*-events 命令除外）对应于 ldm 命令行界面中的那些命令。有关 ldm 子命令的详细信息，请参见 ldm(1M) 手册页。

注 – XML 接口不支持 Logical Domains Manager CLI 所支持的动词或命令 *aliases*。

下面是 <action> 标记中支持的字符串：

Logical Domains Manager 操作	Logical Domains Manager 操作	Logical Domains Manager 操作
list-bindings	list-services	list-constraints
list-devices	add-domain	remove-domain
list-domain	start-domain	stop-domain
bind-domain	unbind-domain	add-io
remove-io	add-mau	set-mau
remove-mau	add-memory	set-memory
remove-memory	remove-reconf	add-spconfig
set-spconfig	remove-spconfig	list-spconfig
add-variable	set-variable	remove-variable
list-variable	add-vconscon	set-vconscon
remove-vconscon	set-vconsole	add-vcpu

Logical Domains Manager 操作	Logical Domains Manager 操作	Logical Domains Manager 操作
set-vcpu	remove-vcpu	add-vdisk
remove-vdisk	add-vdiskserver	remove-vdiskserver
add-vdpcc	remove-vdpcc	add-vdpcs
remove-vdpcs	add-vdiskserverdevice	remove-vdiskserverdevice
add-vnet	set-vnet	remove-vnet
add-vswitch	set-vswitch	remove-vswitch
reg-domain-events	unreg-domain-events	reg-resource-events
unreg-resource-events	reg-hardware-events	unreg-hardware-events
reg-all-events	unreg-all-events	migrate-domain
cancel-operation	set-domain	

Logical Domains Manager 资源和属性

下面是 Logical Domains Manager 资源以及可以为每个资源定义的属性。这些资源和属性在 XML 示例中以**粗体**显示。下列示例显示了未绑定输出的资源。约束输出可用于为 Logical Domains Manager 操作创建输入。但域迁移输出除外。请参见第 221 页中的 “域迁移”。每个资源都定义在 <Section> OVF 段中并由 <rasd:OtherResourceType> 标记指定。

域信息 (ldom_info) 资源

示例 D–6 ldom_info XML 输出示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="primary">
    <Section xsi:type="ovf:ResourceAllocationSection_type">
      <Item>
        <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
        <rasd:Address>00:03:ba:d8:ba:f6</rasd:Address>
        <gprop:GenericPropertykey="hostid">83d8baf6</gprop:GenericProperty>
        <gprop:GenericProperty key="master">plum</gprop:GenericProperty>
        <gprop:GenericProperty key="failure-policy">reset</gprop:GenericProperty>
        <gprop:GenericProperty key="progress">45%</gprop:GenericProperty>
        <gprop:GenericProperty key="status">ongoing</gprop:GenericProperty>
        <gprop:GenericProperty key="source">dt90-319</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

ldom_info 资源始终包含在 <Content> 段中。ldom_info 资源中的下列属性是可选属性：

- <rasd:Address> 标记，指定要分配给域的 MAC 地址。
- <gprop:GenericPropertykey="failure-policy"> 标记，指定从属域在主域失败时应表现出的行为。默认值为 ignore。下面是有效的属性值：
  - ignore 忽略主域故障（从属域不受影响）。
  - panic 在主域失败时使所有从属域都发生紧急情况。
  - reset 在主域失败时重置所有从属域。
  - stop 在主域失败时停止所有从属域。
- <gprop:GenericPropertykey="hostid"> 标记，指定要分配给域的主机 ID。
- <gprop:GenericPropertykey="master"> 标记，指定最多四个用逗号分隔的主域名称。
- <gprop:GenericPropertykey="progress"> 标记，指定命令执行的进度百分比。
- <gprop:GenericPropertykey="source"> 标记，指定报告命令进度的计算机。
- <gprop:GenericPropertykey="status"> 标记，指定命令的状态（done、failed 或 ongoing）。

CPU (cpu) 资源

add-vcpu、set-vcpu 和 remove-vcpu XML 请求操作的等效操作是按照如下方式设置 <gpropGenericProperty key="wcore"> 标记的值：

- 如果使用 -c 选项，请将 wcore 属性设置为指定的全体核心数。
- 如果**未**使用 -c 选项，请将 wcore 属性设置为 0。

请注意，cpu 资源的分配单位属性 <rasd:AllocationUnits> 始终指定虚拟 CPU 数，而非核心数。

示例 D–7 cpu XML 示例

以下示例显示 ldm add-vcpu -c 1 ldg1 命令的等效 XML 请求：

```
<?xml version="1.0"?>
<LDM_interface version="1.2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="./schemas/combined-v3.xsd"
xmlns:ovf="./schemas/envelope"
xmlns:rasd="./schemas/CIM_ResourceAllocationSettingData"
xmlns:vssd="./schemas/CIM_VirtualSystemSettingData"
xmlns:gprop="./schemas/GenericProperty"
xmlns:bind="./schemas/Binding">
  <cmd>
    <action>add-vcpu</action>
    <data version="3.0">
      <Envelope>
        <References/>
```

示例 D–7 cpu XML 示例 （续）

```
        <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="ldg1">
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>cpu</rasd:OtherResourceType>
              <rasd:AllocationUnits>8</rasd:AllocationUnits>
              <gprop:GenericProperty key="wcore">1</gprop:GenericProperty>
            </Item>
          </Section>
        </Content>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

cpu 资源始终包含在 <Content> 段中。

MAU (mau) 资源

注 – mau 资源是受支持服务器上的任意受支持的加密单元。当前，存在两个受支持的加密单元：模运算单元 (Modular Arithmetic Unit, MAU) 和控制字队列 (Control Word Queue, CWQ)。

示例 D–8 mau XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>mau</rasd:OtherResourceType>
        <rasd:AllocationUnits>1</rasd:AllocationUnits>
      </Item>
    </Section>
  </Content>
</Envelope>
```

mau 资源始终包含在 <Content> 段中。唯一属性为 <rasd:AllocationUnits> 标记，用于指出 MAU 数或其他加密单元数。

内存 (memory) 资源

示例 D–9 memory XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
```

示例 D–9 memory XML 示例 （续）

```
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>memory</rasd:OtherResourceType>
        <rasd:AllocationUnits>4G</rasd:AllocationUnits>
      </Item>
    </Section>
  </Content>
</Envelope>
```

内存资源始终包含在 <Content> 段中。唯一属性为 <rasd:AllocationUnits> 标记，用于指出内存大小。

虚拟磁盘服务器 (vds) 资源

示例 D–10 vds XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vds</rasd:OtherResourceType>
        <gprop:GenericProperty
          key="service_name">vdstmp</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

虚拟磁盘服务器 (vds) 资源可以在 <Content> 段中作为域描述的一部分，也可以独自出现于 <Envelope> 段中。唯一属性为 <gprop:GenericProperty> 标记，其中含有 service_name 项，该项包含所描述的 vds 资源的名称。

虚拟磁盘服务器卷 (vds_volume) 资源

示例 D–11 vds_volume XML 示例

```
<Envelope>
  <References/>
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vds_volume</rasd:OtherResourceType>
        <gprop:GenericProperty key="vol_name">vdsdev0</gprop:GenericProperty>
        <gprop:GenericProperty key="service_name">primary-vds0</gprop:GenericProperty>
        <gprop:GenericProperty key="block_dev">
          opt/SUNWldm/domain_disks/testdisk1</gprop:GenericProperty>
```

示例 D–11 vds_volume XML 示例 （续）

```
        <gprop:GenericProperty key="vol_opts">ro</gprop:GenericProperty>
        <gprop:GenericProperty key="mpgroup">mpgroup-name</gprop:GenericProperty>
      </Item>
    </Section>
</Envelope>
```

vds_volume 资源可以在 <Content> 段中作为域描述的一部分，也可以独自出现于 <Envelope> 段中。它必须含有带有以下项的 <gprop:GenericProperty> 标记：

- vol_name－卷名
- service_name－此卷要绑定到的虚拟磁盘服务器的名称
- block_dev－要与此卷关联的文件或设备名称

可选地，vds_volume 资源还可以具有以下属性：

- vol_opts－下列一项或多项，用逗号分隔，在一个字符串内：{ro,slice,excl}
- mpgroup－多路径（故障转移）组的名称

磁盘 (disk) 资源

示例 D–12 disk XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>disk</rasd:OtherResourceType>
        <gprop:GenericProperty key="vdisk_name">vdisk0</gprop:GenericProperty>
        <gprop:GenericProperty
          key="service_name">primary-vds0</gprop:GenericProperty>
        <gprop:GenericProperty key="vol_name">vdsdev0</gprop:GenericProperty>
        <gprop:GenericProperty key="timeout">60</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

disk 资源始终包含在 <Content> 段中。它必须含有带有以下项的 <gprop:GenericProperty> 标记：

- vdisk_name－虚拟磁盘的名称
- service_name－此虚拟磁盘要绑定到的虚拟磁盘服务器的名称
- vol_name－此虚拟磁盘要关联到的虚拟磁盘服务设备

可选地，disk 资源还可以具有 timeout 属性，它是在虚拟磁盘客户机 (vdc) 与虚拟磁盘服务器 (vds) 之间建立连接时的超时值（以秒计）。如果存在多个虚拟磁盘 (vdisk) 路径，则 vdc 可以尝试连接到不同的 vds，该超时可确保在指定的时间内建立到任意 vds 的连接。

虚拟交换机 (vsw) 资源

示例 D–13 vsw XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vsw</rasd:OtherResourceType>
        <gprop:GenericProperty key="service_name">vsw1-ldg1</gprop:GenericProperty>
        <gprop:GenericProperty key="dev_path">bge0</gprop:GenericProperty>
        <gprop:GenericProperty key="linkprop">phys-state</gprop:GenericProperty>
        <rasd:Address>00:14:4f:fc:00:01</rasd:Address>
        <gprop:GenericProperty key="mode">sc</gprop:GenericProperty>
        <gprop:GenericProperty key="pvid">12345678</gprop:GenericProperty>
        <gprop:GenericProperty key="vid">87654321</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

vsw 资源既可以在 <Content> 段中作为域描述的一部分，也可以独自出现于 <Envelope> 段中。它必须含有带有以下项的 <gprop:GenericProperty> 标记：

- service_name－要分配给虚拟交换机的名称。
- linkprop－指定虚拟设备是否应获取物理链路状态更新。当值为 phys-state 时，虚拟设备可获取物理链路状态更新。当值为空时，虚拟设备不会获取物理链路状态更新。默认情况下，虚拟设备不会获取物理链路状态更新。
- dev_path－要与此虚拟交换机关联的网络设备的路径

可选地，vsw 资源还可以具有以下属性：

- <rasd:Address>－向虚拟交换机分配 MAC 地址
- pvid－端口的虚拟局域网 (virtual local area network, VLAN) 标识符 (identifier, ID)，表示虚拟网络需要以无标记模式成为其成员的 VLAN。
- vid－虚拟局域网 (Virtual local area network, VLAN) 标识符 (identifier, ID)，表示虚拟网络和虚拟交换机需要以标记模式成为其成员的 VLAN。
- mode－Oracle Solaris Cluster 心跳支持的 sc。

网络 (network) 资源

示例 D–14 network XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>network</rasd:OtherResourceType>
        <gprop:GenericProperty key="linkprop">phys-state</gprop:GenericProperty>
        <gprop:GenericProperty key="vnet_name">ldg1-vnet0</gprop:GenericProperty>
        <gprop:GenericProperty
          key="service_name">primary-vsw0</gprop:GenericProperty>
        <rasd:Address>00:14:4f:fc:00:01</rasd:Address>
      </Item>
    </Section>
  </Content>
</Envelope>
```

network 资源始终包含在 <Content> 段中。它必须含有带有以下项的 <gprop:GenericProperty> 标记：

- linkprop－指定虚拟设备是否应获取物理链路状态更新。当值为 phys-state 时，虚拟设备可获取物理链路状态更新。当值为空时，虚拟设备不会获取物理链路状态更新。默认情况下，虚拟设备不会获取物理链路状态更新。
- vnet_name－虚拟网络 (vnet) 的名称
- service_name－此虚拟网络要绑定到的虚拟交换机 (vswitch) 的名称

可选地，network 资源还可以具有以下属性：

- <rasd:Address>－向虚拟交换机分配 MAC 地址
- pvid－端口的虚拟局域网 (virtual local area network, VLAN) 标识符 (identifier, ID)，表示虚拟网络需要以无标记模式成为其成员的 VLAN。
- vid－虚拟局域网 (Virtual local area network, VLAN) 标识符 (identifier, ID)，表示虚拟网络和虚拟交换机需要以标记模式成为其成员的 VLAN。
- mode－用于为该虚拟网络启用混合 I/O 的 hybrid。

虚拟控制台集中器 (vcc) 资源

示例 D–15 vcc XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
```

示例 D–15 vcc XML 示例 （续）

```
        <rasd:OtherResourceType>vcc</rasd:OtherResourceType>
        <gprop:GenericProperty key="service_name">vcc1</gprop:GenericProperty>
        <gprop:GenericProperty key="min_port">6000</gprop:GenericProperty>
        <gprop:GenericProperty key="max_port">6100</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

vcc 资源既可以在 <Content> 段中作为域描述的一部分，也可以独自出现于 <Envelope> 段中。它可以含有带有以下项的 <gprop:GenericProperty> 标记：

- service_name－要分配给虚拟控制台集中器服务的名称
- min_port－要与此 vcc 关联的最小端口号
- max_port－要与此 vcc 关联的最大端口号

变量 (var) 资源

示例 D–16 var XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>var</rasd:OtherResourceType>
        <gprop:GenericProperty key="name">test_var</gprop:GenericProperty>
        <gprop:GenericProperty key="value">test1</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

var 资源始终包含在 <Content> 段中。它可以含有带有以下项的 <gprop:GenericProperty> 标记：

- name－变量的名称
- value－变量的值

物理 I/O 设备 (physio_device) 资源

示例 D–17 physio_device XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
```

示例 D–17 physio_device XML 示例 （续）

```
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
        <gprop:GenericProperty key="name">pci@780</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

physio_device 资源始终包含在 <Content> 段中。唯一属性为 <gprop:GenericProperty> 标记，其中含有 name 属性值项，该项为所描述的 I/O 设备的名称。

SP 配置 (spconfig) 资源

示例 D–18 spconfig XML 示例

```
<Envelope>
    <Section xsi:type="ovf:ResourceAllocationSection_type">
      <Item>
        <rasd:OtherResourceType>spconfig</rasd:OtherResourceType>
        <gprop:GenericProperty
          key="spconfig_name">primary</gprop:GenericProperty>
        <gprop:GenericProperty
          key="spconfig_status">current</gprop:GenericProperty>
      </Item>
    </Section>
</Envelope>
```

服务处理器 (service processor, SP) 配置 (spconfig) 资源始终独自出现于 <Envelope> 段中。它可以含有带有以下项的 <gprop:GenericProperty> 标记：

- spconfig_name－要在 SP 上存储的配置的名称
- spconfig_status－特定 SP 配置的当前状态在 ldm list-spconfig 命令的输出中会使用此属性。

虚拟数据平面通道服务 (vdpcs) 资源

示例 D–19 vdpcs XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vdpcs</rasd:OtherResourceType>
```

示例 D–19 vdpcs XML 示例 （续）

```
        <gprop:GenericProperty key="service_name">dg1-vdpcs</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

仅 Netra DPS 环境中会涉及此资源。vdpcs 资源既可以在 <Content> 段中作为域描述的一部分，也可以独自出现于 <Envelope> 段中。唯一属性为 <gprop:GenericProperty> 标记，其中含有 service_name 属性值项，该项为所描述的虚拟数据平面通道服务 (vdpcs) 资源的名称。

虚拟数据平面通道客户机 (vdpcc) 资源

示例 D–20 vdpcc XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vdpcc</rasd:OtherResourceType>
        <gprop:GenericProperty key="vdpcc_name">vdpcc</gprop:GenericProperty>
        <gprop:GenericProperty
          key="service_name">ldg1-vdpcs</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

仅 Netra DPS 环境中会涉及此资源。虚拟数据平面通道客户机资源始终包含在 <Content> 段中。它可以含有带有以下项的 <gprop:GenericProperty> 标记：

- vdpcc_name－虚拟数据平面通道客户机 (vdpcc) 的名称
- service_name－此 vdpcc 要绑定到的虚拟数据平面通道服务 vdpcs 的名称

控制台 (console) 资源

示例 D–21 console XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>console</rasd:OtherResourceType>
        <gprop:GenericProperty key="port">6000</gprop:GenericProperty>
```

示例 D–21 console XML 示例 （续）

```
        <gprop:GenericProperty key="service_name">vcc2</gprop:GenericProperty>
        <gprop:GenericProperty key="group">group-name</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

console 资源始终包含在 <Content> 段中。它可以含有带有以下项的 <gprop:GenericProperty> 标记：

- port－此虚拟控制台 (console) 要更改为的端口
- service_name－此 console 要绑定到的虚拟控制台集中器 (vcc) 服务
- group－此 console 要绑定到的组的名称

域迁移

以下示例将介绍 migrate-domain 子命令的 <data> 段中包含的内容。

示例 D–22 migrate-domain <data> 段示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="ldg1"/>
  <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="ldg1"/>
    <Section xsi:type="ovf:ResourceAllocationSection_Type">
      <Item>
        <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
        <gprop:GenericProperty key="target">target-host</gprop:GenericProperty>
        <gprop:GenericProperty key="username">user-name</gprop:GenericProperty>
        <gprop:GenericProperty key="password">password</gprop:GenericProperty>
      <Item>
    </Section>
  </Content>
</Envelope>
```

其中：

- 第一个 <Content> 节点（不含 <ldom_info> 段）是要迁移的源域。
- 第二个 <Content> 节点（含有 <ldom_info> 段）是要迁移到的目标域。源域和目标域的名称可以相同。
- 目标域的 <ldom_info> 段描述要迁移到的计算机以及迁移到该计算机所需的详细信息：
  - target-host 是要迁移到的目标计算机。
  - user-name 是目标计算机的登录用户名。必须采用 SASL 64 位编码。
  - password 是登录到目标计算机时使用的密码。必须采用 SASL 64 位编码。

注 – Logical Domains Manager 使用 `sasl_decode64()` 对目标用户名和密码进行解码，使用 `sasl_encode64()` 对这些值进行编码。SASL 64 编码等同于 `base64` 编码。

◆ ◆ ◆ 附 录 E

Logical Domains Manager XML 模式

此附录为您提供了与 Logical Domains Manager 配合使用的各种 XML 模式。

本章包括以下内容：

- 第 223 页中的 “LDM_interface XML 模式”
- 第 225 页中的 “LDM_Event XML 模式”
- 第 226 页中的 “ovf-envelope.xsd 模式”
- 第 228 页中的 “ovf-section.xsd 模式”
- 第 228 页中的 “ovf-core.xsd 模式”
- 第 233 页中的 “ovf-virtualhardware.xsc 模式”
- 第 234 页中的 “cim-rasd.xsd 模式”
- 第 238 页中的 “cim-vssd.xsd 模式”
- 第 238 页中的 “cim-common.xsd 模式”
- 第 242 页中的 “GenericProperty XML 模式”
- 第 242 页中的 “Binding_Type XML 模式”

LDM_interface XML 模式

此模式是开放式虚拟机格式 (Open Virtualization Format, OVF) 草稿规范 0.98 版

示例 E–1 LDM_interface XML 模式

```
<?xml version="1.0"?>
xs:schema
     xmlns:ovf="/var/opt/SUNWldom/envelope"
     xmlns:xs="http://www.w3.org/2001/XMLSchema">
     <xs:import namespace="/var/opt/SUNWldom/envelope" schemaLocation="ovf-envelope.xsd"/>

     <xs:annotation>
          <xs:documentation>
               Copyright (c) 2007, 2010, Oracle and/or its affiliates. All rights reserved.
          </xs:documentation>
     </xs:annotation>
```

示例 E–1 LDM_interface XML 模式 （续）

```
<!--
==================
Type Definitions
==================
-->
  <xs:simpleType name="statusStringType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="success"/>
      <xs:enumeration value="failure"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="responseType">
    <xs:sequence>
      <xs:element name="status" type="statusStringType"/>
      <xs:element name="resp_msg" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- LDM interface document -->
  <xs:element name="LDM_interface">
    <xs:complexType>
      <xs:sequence>

        <!-- START cmd -->
        <xs:element name="cmd" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="action" type="xs:string" minOccurs="0"/>

              <!-- START data -->
              <xs:element name="data" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:choice minOccurs="1" maxOccurs="unbounded">

                    <!--OVF Evelope Version 0.9 -->
                    <xs:element name="Envelope" type="ovf:Envelope_Type"/>
                    <!-- DATA response -->
                    <xs:element name="response" type="responseType" minOccurs="0" maxOccurs="1"/>
                  </xs:choice>
                  <xs:attribute name="version" type="xs:string" use="required"/>
                </xs:complexType>
              </xs:element> <!-- END data -->

              <!-- CMD response -->
              <xs:element name="response" type="responseType" minOccurs="0" maxOccurs="1"/>

            </xs:sequence>
          </xs:complexType>
        </xs:element> <!-- END cmd -->

        <!-- DOCUMENT response -->
        <xs:element name="response" type="responseType" minOccurs="0" maxOccurs="1"/>

      </xs:sequence>
      <xs:attribute name="version" type="xs:string" use="required"/>
```

示例 E–1 LDM_interface XML 模式 （续）

```
    </xs:complexType>
  </xs:element> <!-- LDM interface document -->

</xs:schema>
```

LDM_Event XML 模式

示例 E–2 LDM_Event XML 模式

```
<?xml version="1.0"?>
<xs:schema
    xmlns:ovf="/var/opt/SUNWldom/envelope"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">

    <xs:import namespace="/var/opt/SUNWldom/envelope" schemaLocation="ovf-envelope.xsd"/>

    <xs:annotation>
      <xs:documentation>
        Copyright (c) 2007, 2010, Oracle and/or its affiliates.  All rights reserved.
      </xs:documentation>
    </xs:annotation>

  <!-- LDM interface document -->
  <xs:element name="LDM_event">
    <xs:complexType>
      <xs:sequence>

       <!-- START cmd -->
       <xs:element name="cmd" minOccurs="1" maxOccurs="unbounded">
         <xs:complexType>
           <xs:sequence>
             <xs:element name="action" type="xs:string" minOccurs="0"/>

             <!-- START data -->
             <xs:element name="data" minOccurs="0" maxOccurs="unbounded">
               <xs:complexType>
                 <xs:choice minOccurs="1" maxOccurs="unbounded">

                 <!--OVF Evelope Version 0.9 -->
                 <xs:element name="Envelope" type="ovf:Envelope_Type"/>

                 </xs:choice>
                 <xs:attribute name="version" type="xs:string" use="required"/>
               </xs:complexType>
             </xs:element> <!-- END data -->

           </xs:sequence>
         </xs:complexType>
       </xs:element> <!-- END cmd -->

      </xs:sequence>
      <xs:attribute name="version" type="xs:string" use="required"/>
    </xs:complexType>
```

示例 E–2 LDM_Event XML 模式 （续）

```
  </xs:element> <!-- LDM interface document -->

</xs:schema>
```

ovf-envelope.xsd 模式

示例 E–3 ovf-envelope.xsd 模式

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/envelope"
  xmlns:ovf="/var/opt/SUNWldom/envelope"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- Include virtual hardware schema -->
  <xs:include schemaLocation="./ovf-section.xsd"/>
  <xs:include schemaLocation="./cim-virtualhardware.xsd"/>
  <xs:include schemaLocation="./ovf-core.xsd"/>

  <!-- Root element of a OVF package-->
  <xs:element name="Envelope" type="ovf:Envelope_Type"/>

  <xs:complexType name="Envelope_Type">
    <xs:sequence>
      <!--- References to all external files -->
      <xs:element name="References" type="ovf:References_Type"/>

      <!-- Package level meta-data -->
      <xs:element name="Section" type="ovf:Section_Type" minOccurs="0" maxOccurs="unbounded"/>

      <!-- Content. A virtual machine or a vService -->
      <xs:element name="Content" type="ovf:Entity_Type" minOccurs="0" maxOccurs="unbounded"/>

      <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="signed" type="xs:boolean" use="optional"/>
    <xs:attribute name="manifest" type="xs:boolean" use="optional"/>
    <xs:anyAttribute namespace="##any"/>
  </xs:complexType>

  <xs:complexType name="References_Type">
    <xs:sequence>
      <xs:element name="File" type="ovf:File_Type" minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
<xs:anyAttribute namespace="##any"/>
  </xs:complexType>

  <!--Type for an external reference to a resource -->
  <xs:complexType name="File_Type">
    <xs:sequence>
```

示例 E–3 ovf-envelope.xsd 模式 （续）

```
      <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>

    <!-- Reference key used in other parts of the package -->
    <xs:attribute name="id" type="xs:string" use="required"/>
    <!-- Same as using a single part element -->
    <xs:attribute name="href" type="xs:string" use="required"/>
    <!-- Size in bytes of the files (if known) -->
    <xs:attribute name="size" type="xs:integer" use="optional"/>
    <!-- Estimated size in bytes of the files (if a good guess is known) -->
    <xs:attribute name="estSize" type="xs:integer" use="optional"/>
    <!-- Compression type (gzip or bzip2) -->
    <xs:attribute name="compression" type="xs:string" use="optional"/>
    <!-- Chunk size (except of last chunk) -->
    <xs:attribute name="chunkSize" type="xs:long" use="optional"/>

    <xs:anyAttribute namespace="##any"/>
  </xs:complexType>

  <!-- Base class for an entity -->
  <xs:complexType name="Entity_Type" abstract="true">
    <xs:sequence>
      <xs:element name="Info" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Section" type="ovf:Section_Type" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
  </xs:complexType>

  <!-- A Virtual Machine Entity -->
  <xs:complexType name="VirtualSystem_Type">
<xs:complexContent>
      <xs:extension base="ovf:Entity_Type"> </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- A Composite Service -->
  <xs:complexType name="VirtualSystemCollection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Entity_Type">
       <xs:sequence>
        <xs:element name="Content" type="ovf:Entity_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
       </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

ovf-section.xsd 模式

示例 E–4 ovf-section.xsd 模式

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/envelope"
  xmlns:ovf="/var/opt/SUNWldom/envelope"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <!-- The base class for a section. Subclassing this is the most common form of extensibility -->
  <xs:complexType name="Section_Type" abstract="true">
    <xs:sequence>
      <!-- The info element specifies the meaning of the section. This is typically shown
        if the section is not understood by the importer -->
      <xs:element name="Info" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  <!-- Whether the import should fail or not, if the section is not understood -->
    <xs:attribute name="required" type="xs:boolean" use="optional"/>
    <xs:anyAttribute namespace="##any"/>
    <!-- Subtypes defines more specific elements -->
  </xs:complexType>

  <!-- A basic type for a localizable string -->
  <xs:complexType name="Info_Type">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute ref="xml:lang"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
```

ovf-core.xsd 模式

示例 E–5 ovf-core.xsd 模式

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/envelope"
  xmlns:ovf="/var/opt/SUNWldom/envelope"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:include schemaLocation="ovf-section.xsd"/>
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <!-- A user defined annotation on an entity -->
  <xs:complexType name="AnnotationSection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Section_Type">
        <xs:sequence>
```

示例 E–5 ovf-core.xsd 模式 （续）

```
        <!-- Several localized annotations can be included -->
        <xs:element name="Annotation" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
          maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Product information about a virtual appliance -->
<xs:complexType name="ProductSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Product" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Vendor" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Version" type="xs:string" minOccurs="0"/>
        <xs:element name="Full-version" type="xs:string" minOccurs="0"/>
        <xs:element name="ProductUrl" type="xs:string" minOccurs="0"/>
        <xs:element name="VendorUrl" type="xs:string" minOccurs="0"/>
        <xs:element name="AppUrl" type="xs:string" minOccurs="0"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Configuration parameters that can be passed to the virtual machine for
  application-level configuration -->
<xs:complexType name="PropertySection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Property" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Description" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
              <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
                maxOccurs="unbounded"/>
              <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="key" type="xs:string"/>
            <xs:attribute name="type" type="xs:string"/>
            <xs:attribute name="configurableByUser" type="xs:boolean" use="optional"/>
            <xs:attribute name="configurableAtRuntime" type="xs:boolean" use="optional"/>
            <xs:attribute name="defaultValue" type="xs:string" use="optional"/>
            <xs:anyAttribute namespace="##any"/>
          </xs:complexType>
        </xs:element>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
```

示例 E–5 ovf-core.xsd 模式 （续）

```
      <!-- A comma-separated list of transports that are supported by the virtual machine to
        access the OVF environment. -->
      <xs:attribute name="transport" type="xs:string" use="optional"/>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Provides descriptions for the logical networks used within the package. These descriptions are
  typically used as an aid when the package is deployed. -->
<xs:complexType name="NetworkSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Network" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Description" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
              <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
                maxOccurs="unbounded"/>
              <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required"/>
            <xs:anyAttribute namespace="##any"/>
          </xs:complexType>
        </xs:element>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Provides meta-information description of the virtual disks in the package -->
<xs:complexType name="DiskSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Disk" type="ovf:VirtualDiskDesc_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- Disk -->
<xs:complexType name="VirtualDiskDesc_Type">
  <!-- A logical ID for the virtual disk within this package -->
  <xs:attribute name="diskId" type="xs:string" use="required"/>
  <!-- A file reference to the virtual disk file. If this is not specified a blank virtual disk is
    created of the given size -->
  <xs:attribute name="fileRef" type="xs:string" use="optional"/>
  <!-- Capacity in bytes. The capacity can be specified as either a size or as a reference to a property
    using $(property_name) -->
```

示例 E–5 ovf-core.xsd 模式 （续）

```
  <xs:attribute name="capacity" type="xs:string" use="required"/>
  <!-- Format of the disk. The format is an URL that identifies the disk type,
    e.g., http://www.vmware.com/format/vmdk.html#sparse -->
  <xs:attribute name="format" type="xs:string" use="required"/>
  <!-- Populated size of disk. This is an estimation of how much storage the disk needs if backed by
    a non pre-allocated (aka. sparse) disk. This size does not take the meta-data into
    account used by a sparse disk. -->
  <xs:attribute name="populatedSize" type="xs:long" use="optional"/>
  <!-- Reference to a potential parent disk -->
  <xs:attribute name="parentRef" type="xs:string" use="optional"/>
</xs:complexType>

<!-- CPU Architecture requirements for the guest software. -->
<xs:complexType name="CpuCompatibilitySection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Level" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="level" type="xs:int" use="optional"/>
            <xs:attribute name="eax" type="xs:string" use="optional"/>
            <xs:attribute name="ebx" type="xs:string" use="optional"/>
            <xs:attribute name="ecx" type="xs:string" use="optional"/>
            <xs:attribute name="edx" type="xs:string" use="optional"/>
          </xs:complexType>
        </xs:element>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="Vendor" type="xs:string"/>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Specification of the operating system installed in the guest -->
<xs:complexType name="OperatingSystemSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="Description" type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <!-- The IDs are the enumeration used in CIM_OperatingSystem_Type -->
      <xs:attribute name="id" type="xs:string"/>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- End-User License Agreement -->
<xs:complexType name="EulaSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
```

示例 E–5 ovf-core.xsd 模式 （续）

```
        <!-- Contains the license agreement in plain text. Several different locales can be
          specified -->
        <xs:element name="License" type="ovf:Info_Type" minOccurs="1" maxOccurs="unbounded"/>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- For a VirtualSystemCollection, this section is used to specify the order in which the
  contained entities are to be powered on. -->
<xs:complexType name="StartupSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <!-- Id of entity in collection -->
            <xs:attribute name="id" type="xs:string"/>
            <!-- Startup order. Entities are started up starting with lower-numbers first. Items with
              same order identifier may be started up concurrently or in any order.
              The order is reversed for shutdown. -->
            <xs:attribute name="order" type="xs:int"/>
            <!-- Delay in seconds to wait for the power on to complete -->
            <xs:attribute name="startDelay" type="xs:int"/>
            <!-- Whether to resume power-on sequence, once the guest reports ok. -->
            <xs:attribute name="waitingForGuest" type="xs:boolean"/>
            <!-- Delay in seconds to wait for the power on to complete -->
            <xs:attribute name="stopDelay" type="xs:int"/>
            <!-- Stop action to use. Valid values are: 'powerOn' (default), 'none'. -->
            <xs:attribute name="startAction" type="xs:string"/>
            <!-- Stop action to use. Valid values are: 'powerOff' (default), 'guestShutdown',
              'suspend'. -->
            <xs:attribute name="stopAction" type="xs:string"/>
            <xs:anyAttribute namespace="##any"/>
          </xs:complexType>
        </xs:element>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <!-- A comma-separated list of transports that the virtual machine supports to provide
        feedback. -->
      <xs:anyAttribute namespace="##any"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- If this section is present, it indicates that the virtual machine needs to be initially
  booted to install and configure the software. -->
<xs:complexType name="InstallSection_Type">
  <xs:complexContent>
    <xs:extension base="ovf:Section_Type">
      <xs:sequence>
        <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
```

示例 E–5 ovf-core.xsd 模式 （续）

```
            maxOccurs="unbounded"/>
          <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <!-- A comma-separated list of transports that the virtual machine supports to provide
          feedback. -->
        <xs:attribute name="transport" type="xs:string"/>
        <xs:anyAttribute namespace="##any"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

ovf-virtualhardware.xsc 模式

示例 E–6 ovf-virtualhardware.xsc 模式

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/envelope"
  xmlns:ovf="/var/opt/SUNWldom/envelope"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:vssd="/var/opt/SUNWldom/CIM_VirtualSystemSettingData
  xmlns:rasd="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData">

  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <xs:include schemaLocation="ovf-section.xsd"/>

  <xs:import namespace="/var/opt/SUNWldom/CIM_VirtualSystemSettingData" schemaLocation="cim-vssd.xsd"/>
  <xs:import namespace="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
    schemaLocation="cim-rasd.xsd"/>

  <!-- Specifies the virtual hardware for a virtual machine -->
  <xs:complexType name="VirtualHardwareSection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Section_Type">
        <xs:sequence>
          <xs:element name="System" type="vssd:CIM_VirtualSystemSettingData_Type" minOccurs="0"/>
          <xs:element name="Item" type="rasd:CIM_ResourceAllocationSettingData_Type"
            minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Specifies a section for resource constraints on a VirtualSystemCollection -->
  <xs:complexType name="ResourceAllocationSection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Section_Type">
        <xs:sequence>
          <xs:element name="Item" type="rasd:CIM_ResourceAllocationSettingData_Type"
            minOccurs="0" maxOccurs="unbounded"/>
```

示例 E–6 ovf-virtualhardware.xsc 模式 （续）

```
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

cim-rasd.xsd 模式

示例 E–7 cim-rasd.xsd 模式

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
  xmlns:class="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
  xmlns:cim="/var/opt/SUNWldom/common"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:import namespace="/var/opt/SUNWldom/common" schemaLocation="cim-common.xsd"/>

  <xs:element name="Caption" nillable="true" type="cim:cimString"/>

  <xs:element name="Description" nillable="true" type="cim:cimString"/>

  <xs:element name="InstanceId" nillable="true" type="cim:cimString"/>

  <xs:element name="ResourceType" nillable="true">
    <xs:complexType>
      <xs:simpleContent>
        <xs:restriction base="xs:anyType">
          <xs:simpleType>
            <xs:union>
              <xs:simpleType>
                <xs:restriction base="xs:unsignedShort">
                  <xs:enumeration value="1"/> <!-- Other -->
                  <xs:enumeration value="2"/> <!-- Computer System -->
                  <xs:enumeration value="3"/> <!-- Processor-->
                  <xs:enumeration value="4"/> <!-- Memory-->
                  <xs:enumeration value="5"/> <!-- IDE Controller -->
                  <xs:enumeration value="6"/> <!-- Parallel SCSI HBA -->
                  <xs:enumeration value="7"/> <!-- FC HBA -->
                  <xs:enumeration value="8"/> <!-- iSCSI HBA -->
                  <xs:enumeration value="9"/> <!-- IB HCA -->
                  <xs:enumeration value="10"/> <!-- Ethernet Adapter -->
                  <xs:enumeration value="11"/> <!-- Other Network Adapter -->
                  <xs:enumeration value="12"/> <!-- I/O Slot -->
                  <xs:enumeration value="13"/> <!-- I/O Device -->
                  <xs:enumeration value="14"/> <!-- Floppy Drive -->
                  <xs:enumeration value="15"/> <!-- CD Drive -->
                  <xs:enumeration value="16"/> <!-- DVD drive -->
                  <xs:enumeration value="17"/> <!-- Disk Drive -->
                  <xs:enumeration value="18"/> <!-- Tape Drive -->
                  <xs:enumeration value="19"/> <!-- Storage Extent -->
                  <xs:enumeration value="20"/> <!-- Other storage device -->
```

示例 E–7 cim-rasd.xsd 模式 （续）

```
            <xs:enumeration value="21"/> <!-- Serial port -->
            <xs:enumeration value="22"/> <!-- Parallel port -->
            <xs:enumeration value="23"/> <!-- USB Controller -->
            <xs:enumeration value="24"/> <!-- Graphics controller -->
            <xs:enumeration value="25"/> <!-- IEEE 1394 Controller -->
            <xs:enumeration value="26"/> <!-- Partitionable Unit -->
            <xs:enumeration value="27"/> <!-- Base Partitionable Unit -->
            <xs:enumeration value="28"/> <!-- Power Supply -->
            <xs:enumeration value="29"/> <!-- Cooling Device -->
            <xs:enumeration value="29"/> <!-- Cooling Device -->
            <xs:enumeration value="31"/> <!-- PS2 Controller -->
            <xs:enumeration value="32"/> <!-- SIO Controller -->
            <xs:enumeration value="33"/> <!-- Keyboard -->
            <xs:enumeration value="34"/> <!-- Pointing Device -->
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:unsignedShort">
            <xs:minInclusive value="30"/>
            <xs:maxInclusive value="32769"/>
          </xs:restriction>
        </xs:simpleType>
        <xs:simpleType>
          <xs:restriction base="xs:unsignedShort">
            <xs:minInclusive value="32768"/>
            <xs:maxInclusive value="65535"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:union>
    </xs:simpleType>
    <xs:anyAttribute namespace="##any"/>
  </xs:restriction>
 </xs:simpleContent>
</xs:complexType>
</xs:element>

<xs:element name="OtherResourceType" nillable="true" type="cim:cimString"/>

<xs:element name="ResourceSubType" nillable="true" type="cim:cimString"/>

<xs:element name="PoolID" nillable="true" type="cim:cimString"/>

<xs:element name="ConsumerVisibility" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="xs:anyType">
        <xs:simpleType>
          <xs:union>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:enumeration value="0"/>
                <xs:enumeration value="2"/>
                <xs:enumeration value="3"/>
                <xs:enumeration value="4"/>
              </xs:restriction>
            </xs:simpleType>
```

示例 E–7 cim-rasd.xsd 模式 （续）

```
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:minInclusive value="5"/>
                <xs:maxInclusive value="32768"/>
              </xs:restriction>
            </xs:simpleType>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:minInclusive value="32767"/>
                <xs:maxInclusive value="65535"/
              </xs:restriction>
            </xs:simpleType>
          </xs:union>
        </xs:simpleType>
        <xs:anyAttribute namespace="##any"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>

<xs:element name="HostResource" nillable="true" type="xs:anyType"/>
<xs:element name="AllocationUnits" nillable="true" type="cim:cimString"/>
<xs:element name="VirtualQuantity" nillable="true" type="cim:cimUnsignedLong"/>
<xs:element name="Reservation" nillable="true" type="cim:cimUnsignedLong"/>
<xs:element name="Limit" nillable="true" type="cim:cimUnsignedLong"/>
<xs:element name="Weight" nillable="true" type="cim:cimUnsignedInt"/>
<xs:element name="AutomaticAllocation" nillable="true" type="cim:cimBoolean"/>
<xs:element name="AutomaticDeallocation" nillable="true" type="cim:cimBoolean"/>
<xs:element name="Parent" nillable="true" type="cim:cimString"/>
<xs:element name="Connection" nillable="true" type="cim:cimString"/>
<xs:element name="Address" nillable="true" type="cim:cimString"/
<xs:element name="MappingBehavior" nillable="true">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="xs:anyType">
        <xs:simpleType>
          <xs:union>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:enumeration value="0"/>
                <xs:enumeration value="1"/>
                <xs:enumeration value="2"/>
                <xs:enumeration value="3"/>
                <xs:enumeration value="4"/>
              </xs:restriction>
            </xs:simpleType>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort">
                <xs:minInclusive value="5"/>
                <xs:maxInclusive value="32768"/>
              </xs:restriction>
            </xs:simpleType>
            <xs:simpleType>
              <xs:restriction base="xs:unsignedShort"
                <xs:minInclusive value="32767"/>
                <xs:maxInclusive value="65535"/>
```

示例 E–7 cim-rasd.xsd 模式 （续）

```
              </xs:restriction>
            </xs:simpleType>
          </xs:union>
        </xs:simpleType>
        <xs:anyAttribute namespace="##any"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
<xs:element name="AddressOnParent" nillable="true" type="cim:cimString"/>

<xs:element name="BusNumber" nillable="true" type="cim:cimUnsignedShort"/>

<xs:complexType name="CIM_ResourceAllocationSettingData_Type">
  <xs:sequence>
    <xs:element ref="class:Caption" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="class:Description" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="class:InstanceId" minOccurs="0"/>
    <xs:element ref="class:ResourceType" minOccurs="0"/>
    <xs:element ref="class:OtherResourceType" minOccurs="0"/>
    <xs:element ref="class:ResourceSubType" minOccurs="0"/>
    <xs:element ref="class:PoolID" minOccurs="0"/>
    <xs:element ref="class:ConsumerVisibility" minOccurs="0"/>
    <xs:element ref="class:HostResource" maxOccurs="unbounded" minOccurs="0"/>
    <xs:element ref="class:AllocationUnits" minOccurs="0"/>
    <xs:element ref="class:VirtualQuantity" minOccurs="0"/>
    <xs:element ref="class:Reservation" minOccurs="0"/>
    <xs:element ref="class:Limit" minOccurs="0"/>
    <xs:element ref="class:Weight" minOccurs="0"/>
    <xs:element ref="class:AutomaticAllocation" minOccurs="0"/>
    <xs:element ref="class:AutomaticDeallocation" minOccurs="0"/>
    <xs:element ref="class:Parent" minOccurs="0"/>
    <xs:element ref="class:Connection" maxOccurs="unbounded" minOccurs="0"/>
    <xs:element ref="class:Address" minOccurs="0"/>
    <xs:element ref="class:MappingBehavior" minOccurs="0"/>
    <xs:element ref="class:AddressOnParent" minOccurs="0"/>
    <xs:element ref="class:BusNumber" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##any"/>
</xs:complexType>

<xs:element name="CIM_ResourceAllocationSettingData"
  type="class:CIM_ResourceAllocationSettingData_Type"/>
</xs:schema>
```

cim-vssd.xsd 模式

示例 E–8 cim-vssd.xsd 模式

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/CIM_VirtualSystemSettingData"
  xmlns:class="/var/opt/SUNWldom/CIM_VirtualSystemSettingData"
  xmlns:cim="/var/opt/SUNWldom/common"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:import namespace="/var/opt/SUNWldom/common"
    schemaLocation="cim-common.xsd"/>

  <xs:element name="Caption" nillable="true" type="cim:cimString"/>

  <xs:element name="Description" nillable="true" type="cim:cimString"/>

  <xs:element name="InstanceId" nillable="true" type="cim:cimString"/>

  <xs:element name="VirtualSystemIdentifier" nillable="true" type="cim:cimString"/>

  <xs:element name="VirtualSystemType" nillable="true" type="cim:cimString"/>

  <xs:complexType name="CIM_VirtualSystemSettingData_Type">
    <xs:sequence>
      <xs:element ref="class:Caption" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="class:Description" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element ref="class:InstanceId" minOccurs="0"/>
      <xs:element ref="class:VirtualSystemIdentifier" minOccurs="0"/>
      <xs:element ref="class:VirtualSystemType" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute namespace="##any"/>
  </xs:complexType>

  <xs:element name="CIM_VirtualSystemSettingData" type="class:CIM_VirtualSystemSettingData_Type"/>

</xs:schema>
```

cim-common.xsd 模式

示例 E–9 cim-common.xsd 模式

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/common"
  xmlns:cim="/var/opt/SUNWldom/common"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">

  <!-- The following are runtime attribute definitions -->
  <xs:attribute name="Key" type="xs:boolean"/>

  <xs:attribute name="Version" type="xs:string"/>
```

示例 E–9 cim-common.xsd 模式 （续）

```
<!-- The following section defines the extended WS-CIM datatypes -->
<xs:complexType name="cimDateTime">
  <xs:choice>
    <xs:element name="CIM_DateTime" type="xs:string" nillable="true"/>
    <xs:element name="Interval" type="xs:duration"/>
    <xs:element name="Date" type="xs:date"/>
    <xs:element name="Time" type="xs:time"/>
    <xs:element name="Datetime" type="xs:dateTime"/>
  </xs:choice>
  <xs:anyAttribute namespace="##any" processContents="lax"/>
</xs:complexType>

<xs:complexType name="cimUnsignedByte">
  <xs:simpleContent>
    <xs:extension base="xs:unsignedByte">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimByte">
  <xs:simpleContent>
    <xs:extension base="xs:byte">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimUnsignedShort">
  <xs:simpleContent>
    <xs:extension base="xs:unsignedShort">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimShort">
  <xs:simpleContent>
    <xs:extension base="xs:short">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimUnsignedInt">
  <xs:simpleContent>
    <xs:extension base="xs:unsignedInt">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimInt">
  <xs:simpleContent>
    <xs:extension base="xs:int">
```

示例 E–9 cim-common.xsd 模式 （续）

```
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimUnsignedLong">
  <xs:simpleContent>
    <xs:extension base="xs:unsignedLong">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimLong">
  <xs:simpleContent>
    <xs:extension base="xs:long">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimString">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimBoolean">
  <xs:simpleContent>
    <xs:extension base="xs:boolean">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimFloat">
  <xs:simpleContent>
    <xs:extension base="xs:float">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimDouble">
  <xs:simpleContent>
    <xs:extension base="xs:double">
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="cimChar16">
  <xs:simpleContent>
      <xs:restriction base="cim:cimString">
```

示例 E–9 cim-common.xsd 模式 （续）

```
        <xs:maxLength value="1"/>
        <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="cimBase64Binary">
    <xs:simpleContent>
      <xs:extension base="xs:base64Binary">
        <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="cimHexBinary">
    <xs:simpleContent>
      <xs:extension base="xs:hexBinary">
        <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="cimReference">
    <xs:sequence>
      <xs:any namespace="##other" maxOccurs="unbounded"/>
    </xs:sequence>
    xs:anyAttribute namespace="##any" processContents="lax"/>
  </xs:complexType>

  <!-- The following datatypes are used exclusively to define metadata fragments -->
  <xs:attribute name="qualifier" type="xs:boolean"/>

  <xs:complexType name="qualifierString">
    <xs:simpleContent>
      <xs:extension base="cim:cimString">
        <xs:attribute ref="cim:qualifier" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="qualifierBoolean">
    <xs:simpleContent>
      <xs:extension base="cim:cimBoolean">
        <xs:attribute ref="cim:qualifier" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="qualifierUInt32">
    <xs:simpleContent>
      <xs:extension base="cim:cimUnsignedInt">
        <xs:attribute ref="cim:qualifier" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
```

示例 E–9 cim-common.xsd 模式 （续）

```
  <xs:complexType name="qualifierSInt64">
    <xs:simpleContent>
      <xs:extension base="cim:cimLong">
        <xs:attribute ref="cim:qualifier" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
<!--
  <xs:complexType name="qualifierSArray">
    <xs:complexContent>
      <xs:extension base="cim:qualifierString"/>
    </xs:complexContent>
  </xs:complexType>
-->
  <!-- The following element is to be used only for defining metadata -->
  <xs:element name=" DefaultValue" type="xs:anySimpleType"/>
</xs:schema>
```

GenericProperty XML 模式

此模式是对开放式虚拟机格式 (Open Virtualization Format, OVF) 模式的扩展。

示例 E–10 GenericProperty XML 模式

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/GenericProperty"
  xmlns:class="/var/opt/SUNWldom/GenericProperty"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:complexType name="GenericProperty_Type" type="xs:string">
  <xs:attribute name="key" type="xs:string" use="required"/>

  </xs:complexType>
  <xs:element name="GenericProperty" type="class:GenericProperty_Type"/>

</xs:schema>
```

Binding_Type XML 模式

此模式是对开放式虚拟机格式 (Open Virtualization Format, OVF) 模式的扩展。

示例 E–11 Binding_Type XML 模式

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/Binding"
  xmlns:class="/var/opt/SUNWldom/Binding"
  xmlns:rasd="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
```

示例 E–11 Binding_Type XML 模式 （续）

```
  xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:import namespace="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
      schemaLocation="cim-rasd.xsd"/>

  <xs:complexType name="Binding_Type">
    <xs:sequence>
      <xs:element name="Item"
        type="rasd:CIM_ResourceAllocationSettingData_Type"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

词汇表

本列表定义了 Oracle VM Server for SPARC 文档中的术语、缩写和首字母缩略词。

A

API Application programming interface（应用编程接口）

auditreduce 从审计跟踪文件中合并和选择审计记录（请参见 auditreduce(1M) 手册页）。

auditing（审计） 使用 Oracle Solaris OS 审计来识别安全更改源

authorization（授权） 使用 Oracle Solaris OS RBAC 设置授权

B

bge 用于 Broadcom BCM57xx 设备的 Broadcom 千兆位以太网驱动程序

BSM Basic Security Module（基本安全模块）

bsmconv 启用 BSM（请参见 bsmconv(1M) 手册页）。

bsmunconv 禁用 BSM（请参见 bsmunconv(1M) 手册页）。

C

CD Compact disc（光盘）

CLI Command-line interface（命令行界面）

compliance（法规遵从性） 确定系统配置是否符合预先定义的安全配置文件

configuration（配置） 保存在服务处理器上的逻辑域配置的名称

CMT Chip multithreading（芯片多线程）

constraints（约束） 对于 Logical Domains Manager，约束是您希望分配给特定域的一个或多个资源。您可能会接收到要求添加到域中的所有资源，也可能会得不到任何资源，这取决于可用资源。

control domain（控制域） 创建和管理其他逻辑域及服务的域

CPU Central processing unit（中央处理器）

CWQ Control Word Queue（控制字队列）；基于 Oracle Sun UltraSPARC T2 的平台的加密单元

D

DHCP Dynamic Host Configuration Protocol（动态主机配置协议）

DMA Direct Memory Access（直接内存访问），能够不通过 CPU 在内存和设备（例如，网卡）之间传输数据。

DMP Dynamic Multipathing（动态多路径）(Veritas)

Logical Domains Manager 用于创建和管理逻辑域并将资源分配给域的 CLI

DPS Data plane software（数据平面软件）

DR Dynamic reconfiguration（动态重新配置）

drd 用于 Logical Domains Manager 的 Oracle Solaris 10 OS 动态重新配置守护进程（请参见 `drd(1M)` 手册页）。

DS Domain Service（域服务）模块 (Oracle Solaris 10 OS)

DVD Digital versatile disc（数字多功能光盘）

E

EFI Extensible firmware interface（可扩展的固件接口）

ETM Encoding Table Management（编码表管理）模块 (Oracle Solaris 10 OS)

F

FC_AL Fiber Channel Arbitrated Loop（光纤通道仲裁环路）

FMA Fault Management Architecture（故障管理体系结构）

fmd Oracle Solaris 10 OS 故障管理器守护进程（请参见 fmd(1M) 手册页）。

format 磁盘分区和维护实用程序（请参见 format(1M) 手册页）。

fmthard 填充硬盘上的标签（请参见 fmthard(1M) 手册页）。

FTP File Transfer Protocol（文件传输协议）

G

Gb Gigabit（千兆位）

guest domain（来宾域） 它使用来自 I/O 域和服务域的服务，并由控制域进行管理。

GLDv3 Generic LAN Driver version 3（Generic LAN Driver 版本 3）。

H

hardening（强化） 修改 Oracle Solaris OS 配置以提高安全性

HDD Hard disk drive（硬盘驱动器）

hypervisor（虚拟机管理程序） 在操作系统和硬件层之间插入的固件层

I

I/O domain（I/O 域） 此域对物理 I/O 设备具有直接拥有权和直接访问权，而且它与其他逻辑域共享以虚拟设备形式出现的那些设备

IB Infiniband

IDE Integrated Drive Electronics（集成驱动器电子装置）

IDR Interim Diagnostics Release（临时诊断版）

ILOM Integrated Lights Out Manager

io I/O 设备，例如内部磁盘和 PCIe 控制器及其连接的适配器和设备

ioctl Input/output control call（输入/输出控制调用）

IP Internet Protocol（Internet 协议）

IPMP Internet Protocol Network Multipathing（Internet 协议网络多路径）

ISO International Organization for Standardization（国际标准化组织）

K

kaio Kernel asynchronous input/output（内核异步输入/输出）

KB Kilobyte（千字节）

KU Kernel update（内核更新）

L

LAN Local-area network（局域网）

LDAP Lightweight Directory Access Protocol（轻量目录访问协议）

LDC Logical domain channel（逻辑域通道）

ldm Logical Domains Manager 实用程序（请参见 `ldm(1M)` 手册页）。

ldmd Logical Domains Manager 守护进程

lofi Loopback file（回送文件）

logical domain（逻辑域） 一种由资源的离散逻辑分组构成的虚拟机，在一个计算机系统中有其自身的操作系统和标识

LUN Logical unit number（逻辑单元编号）

M

MAC 介质访问控制 (media access control, MAC) 地址；Logical Domains 可以自动分配 MAC 地址，您也可以手动分配 MAC 地址

MAU Modular Arithmetic Unit（模运算单元）

MB Megabyte（兆字节）

MD 服务器数据库中的机器描述 (machine description, MD)

mem, memory 内存单元默认大小（以字节为单位），也可以指定千兆字节 (G)，千字节 (K) 或兆字节 (M)。可分配给来宾域的服务器虚拟内存。

metadb 创建和删除 Solaris 卷管理器元设备状态数据库的副本（请参见 metadb(1M) 手册页）。

metaset 配置磁盘集（请参见 metaset(1M) 手册页）。

mhd 多主机磁盘控制操作（请参见 mhd(7i) 手册页）。

MIB Management Information Base（管理信息库）

minimizing（最小化） 安装所必需的最少数量的核心 Oracle Solaris OS 软件包

MMF Multimode fiber（多模光纤）

MMU Memory management unit（内存管理单元）

mpgroup 虚拟磁盘故障转移的多路径组 (multipathing group, mpgroup) 名称

mtu Maximum transmission unit（最大传输单位）

N

NAT Network Address Translation（网络地址转换）

ndpsldcc Netra DPS Logical Domain Channel Client（Netra DPS 逻辑域通道客户机）。**另请参见** vdpcc。

ndpsldcs Netra DPS Logical Domain Channel Service（Netra DPS 逻辑域通道服务）。**另请参见** vdpcs。

NFS Network file system（网络文件系统）

NIS Network Information Services（网络信息服务）

NIU Network Interface Unit（网络接口单元）（Oracle 的 Sun SPARC Enterprise T5120 和 T5220 服务器）

NTS Network terminal server（网络终端服务器）

NVRAM Non-volatile random-access memory（非易失性随机存取存储器）

nxge 用于 NIU 10Gb 以太网适配器的驱动程序

O

OS Operating system（操作系统）

OVF Open Virtualization Format（开放虚拟化格式）

P

P2V Logical Domains 物理机到虚拟机 (Physical-to-Virtual, P2V) 转换工具

PA Physical address（物理地址）

PCI 外设部件互连 (peripheral component interconnect, PCI) 总线

PCIe PCI EXPRESS 总线

PCI-X PCI Extended (PCI-X) 总线

pcpu Physical CPU（物理 CPU）

physio Physical input/output（物理输入/输出）

PICL Platform Information and Control Library（平台信息和控制库）

picld PICL 守护进程（请参见 picld(1M) 手册页）。

PM 虚拟 CPU 和内存的电源管理 (Power management, PM)

praudit 显示审计跟踪文件的内容（请参见 praudit(1M) 手册页）。

PRI Priority（优先级）

R

RA Real address（实际地址）

RAID Redundant Array of Inexpensive Disk（廉价磁盘冗余阵列）

RBAC Role-Based Access Control（基于角色的访问控制）

RPC Remote Procedure Call（远程过程调用）

S

SASL Simple Authentication and Security Layer（简单验证和安全层）

SAX XML 简单 API (Simple API for XML, SAX) 解析程序，用于遍历 XML 文档。SAX 解析程序以事件为基础，通常用于对数据进行流化处理。

系统控制器 (system controller, SC) 另请参见服务处理器

SCSI Small Computer System Interface（小型计算机系统接口）

service domain（服务域） 向其他逻辑域提供诸如虚拟交换机、虚拟控制台连接器和虚拟磁盘服务器等设备的逻辑域

SMA System Management Agent（系统管理代理）

SMF Service Management Facility（服务管理工具）

SNMP Simple Network Management Protocol（简单网络管理协议）

服务处理器 (service processor, SP) SP 也称为系统控制器 (system controller, SC)，用于监视和运行物理计算机。

SSH Secure Shell（安全 Shell）

ssh 安全 Shell (Secure Shell, ssh) 命令（请参见 ssh(1) 手册页）。

sshd 安全 Shell 守护进程 (Secure Shell daemon, sshd)（请参见 sshd(1M) 手册页）。

SunVTS Sun Validation Test Suite（Sun 验证测试套件）

svcadm 处理服务实例（请参见 svcadm(1M) 手册页）。

T

TCP Transmission Control Protocol（传输控制协议）

TLS Transport Layer Security（传输层安全）

U

UDP User Diagram Protocol（用户图表协议）

UFS UNIX File System（UNIX 文件系统）

unicast 在单个发送方和单个接收方之间发生的网络通信

USB Universal Serial Bus（通用串行总线）

uscsi User SCSI command interface（用户 SCSI 命令接口）（请参见 uscsi(7I) 手册页）。

UTP Unshielded twisted pair（非屏蔽双绞线）

V

var 变量

VBSC Virtual blade system controller（虚拟刀片系统控制器）

vcc, vconscon 虚拟控制台集中器服务，带有分配给来宾域的特定端口范围

vcons, vconsole 用于访问系统级消息的虚拟控制台。通过连接控制域中特定端口上的 vconscon 服务来实现连接。

vcpu Virtual central processing unit（虚拟中央处理器）。服务器中的每个内核都表示为一个虚拟 CPU。例如，Oracle 的一个 8 核 Sun Fire T2000 服务器具有可以在逻辑域之间分配的 32 个虚拟 CPU。

vdc Virtual disk client（虚拟磁盘客户机）

vdisk 虚拟磁盘是与各种类型的物理设备、卷或文件关联的通用块设备。

vdpcc Netra DPS 环境中的虚拟数据平面通道客户机 (Virtual data plane channel client, vdpcc)

vdpcs Netra DPS 环境中的虚拟数据平面通道服务 (Virtual data plane channel service, vdpcs)

vds, vdiskserver 通过虚拟磁盘服务器可以将虚拟磁盘导入到逻辑域中

vdsdev, vdiskserverdevice 虚拟磁盘服务器设备是由虚拟磁盘服务器导出的。该设备可以是整个磁盘、一个磁盘分片、一个文件或一个磁盘卷。

VLAN Virtual local area network（虚拟局域网）

vldc 虚拟逻辑域通道 (Virtual logical domain channel, vldc) 服务

vldcc Virtual logical domain channel client（虚拟逻辑域通道客户机）

vnet 虚拟网络设备可实现虚拟以太网设备，还可以通过使用虚拟网络交换机 (vswitch) 与系统中的其他 vnet 设备进行通信。

vntsd Logical Domains 控制台的 Oracle Solaris 10 OS 虚拟网络终端服务器守护进程（请参见 vntsd(1M) 手册页）。

volfs Volume Management file system（卷管理文件系统）（请参见 volfs(7FS) 手册页）。

vsw, vswitch 虚拟网络交换机，它可将虚拟网络设备连接到外部网络，还可以在虚拟网络设备和外部网络之间交换包

VTOC Volume table of content（卷目录）

VxDMP Veritas Dynamic Multipathing（Veritas 动态多路径）

VxVM Veritas Volume Manager（Veritas 卷管理器）

W

WAN Wide-area network（广域网）

X

XFP eXtreme Fast Path（极速路径）

XML Extensible Markup Language（可扩展标记语言）

XMPP Extensible Messaging and Presence Protocol（可扩展消息处理现场协议）

Z

ZFS Zettabyte File System（Zettabyte 文件系统）(Oracle Solaris 10 OS)

zpool ZFS 存储池（请参见 zpool(1M) 手册页）。

ZVOL ZFS Volume Emulation Driver（ZFS 卷模拟驱动程序）

索引

C

cancel-operation reconf 子命令，140
CLI, **请参见**命令行界面
configuration assistant GUI，193
CPU 核心禁用，151
CPU 时钟周期跳步，151

D

DR, **请参见**动态重新配置

I

I/O 域，57–58, 58–62, 62–71
  PCI EXPRESS (PCIe) 总线，57–58
  创建，59
  分配 PCIe 总线，58–62
  分配端点设备，62–71
  迁移限制，57

L

LDC, **请参见**逻辑域通道
ldm 子命令
  cancel-operation reconf，140
  ls-dom，140
  用户授权，37
ldm(1M) 命令，20
ldm(1M) 手册页，20
ldmconfig(1M) 命令，22, 193, 194
ldmd, Logical Domains Manager 守护进程，20
ldmp2v(1M) 命令，181
Logical Domains Manager，18, 19
  XML 模式配合使用，223
  发现机制，197
  结合使用 XML 模式，201
  守护进程 (ldmd)，20
ls-dom 子命令，140

P

PCI EXPRESS (PCIe) 总线，57–58
primary 域，19
  重新引导，66

S

SPARC T3 服务器，20
SUNWldm 软件包，20

U

UltraSPARC T2 Plus 服务器，20

X

XML 模式
  Logical Domains Manager 配合使用，223
  结合使用 Logical Domains Manager，201

创
创建 I/O 域, 完整 PCIe 总线，59

电
电源管理 (power management, PM)，151

动
动态重新配置 (dynamic reconfiguration, DR)，139
动态重新配置守护进程 (drd)，140

读
读, 授权，37
读写, 授权，37

多
多路径, 虚拟磁盘，84

非
非交互式域迁移，135

服
服务处理器 (service processor, SP)
  存储配置，140
  监视和运行物理机，19
服务域，19, 20

根
根域，19

功
功率极限，151

规
规划
  直接 I/O (DIO)，64

基
基于链路的 IPMP, 使用，112–114

将
将
  PCIe 总线分配到 I/O，58–62
  端点设备分配到 I/O 域，62–71

角
角色, 逻辑域，19

巨
巨型帧, 配置，122–125

控
控制域，19

来
来宾域，20

逻
逻辑域
  定义，17
  角色，19
逻辑域通道 (logical domain channel, LDC)，19

命
命令
  `ldm`(1M)，20
  `ldmconfig`(1M)，22, 193, 194
  `ldmp2v`(1M)，181
命令行界面，20

内
内存电源管理 (power management, PM)，151
内存动态重新配置 (dynamic reconfiguration, DR)，144

配
配置
  存储在服务处理器上，140
  巨型帧，122–125
  选择以引导，21

平
平台
  SPARC T3 服务器，20
  UltraSPARC T2 Plus 服务器，20

迁
迁移, 非交互式，135
迁移限制, I/O 域，57

软
软件包, `SUNWldm`，20

设
设置, 功率极限，151

使
使用基于链路的 IPMP，112–114

守
守护进程
  `drd`，140
  `ldmd`，20
  `vntsd`，21

授
授权
  `ldm` 子命令，37
  读，37
  读写，37
  级别，37

物
物理机，19
物理设备，19, 20

系
系统控制器, **请参见**服务处理器 (service processor, SP)

虚
虚拟磁盘多路径，84
虚拟机，19
虚拟机管理程序，定义，17
虚拟设备
  I/O，20
  虚拟磁盘服务 (vds)，21
  虚拟磁盘客户机 (vdc)，21
  虚拟交换机 (vsw)，20
  虚拟控制台集中器 (vcc)，21
  虚拟网络 (vnet)，20
虚拟网络终端服务器守护进程 (vntsd)，21

延
延迟重新配置，140

域
域
  服务，20
  类型，19，20
域迁移，非交互式，135

直
直接 I/O (DIO)，规划，64

重
重新引导 primary 域，66

资
资源
  另请参见虚拟设备
  定义，19