Oracle® VM Server for SPARC 3.4 开发者指南

文件号码 **E71836**
2016 年 8 月

ORACLE

目录

使用本文档

- **概述**－提供了描述 Oracle VM Server for SPARC 模板、XML 接口及 Logical Domains Manager 发现和虚拟域信息 API 的详细信息和过程。
- **目标读者**－管理 SPARC 服务器上的虚拟化功能的系统管理员。
- **必需的知识**－这些服务器的系统管理员必须具有 UNIX 系统和 Oracle Solaris 操作系统 (Oracle Solaris operating system, Oracle Solaris OS) 的实际应用知识。

产品文档库

有关该产品及相关产品的文档和资源，可从以下网址获得：`http://www.oracle.com/technetwork/documentation/vm-sparc-194287.html`。

反馈

可以在 `http://www.oracle.com/goto/docfeedback` 上提供有关本文档的反馈。

♦ ♦ ♦ 第 1 章

开发 Oracle VM Server for SPARC 模板、XML 文件和程序

开发 Oracle VM Server for SPARC 模板、XML 文件和程序

本指南提供了有关以下 Oracle VM Server for SPARC 功能的信息：

- **Oracle VM Server for SPARC 模板实用程序。**可以使用这些实用程序创建您可以用来部署域配置的模板。请参见第 2 章 使用 Oracle VM Server for SPARC 模板。
- **Oracle VM Server for SPARC XML 接口。**可以使用此 XML 接口描述逻辑域的配置。请参见第 3 章 将 XML 接口与 Logical Domains Manager 结合使用。
- **Logical Domains Manager 发现 API。**可以使用此 API 编写能够发现哪些域在运行 Logical Domains Manager 的程序。请参见第 4 章 Logical Domains Manager 发现。
- **虚拟域信息 API。**可以使用此 API 编写用于提供有关虚拟域的信息的程序。请参见第 5 章 使用虚拟域信息命令和 API。

注 - 本书中介绍的功能可与《Oracle VM Server for SPARC 3.4 安装指南》中列出的所有受支持系统软件和硬件平台结合使用。但是，有些功能只适用于一部分受支持的系统软件和硬件平台。有关这些例外的信息，请参见《Oracle VM Server for SPARC 3.4发行说明》 中的 “本发行版新增功能”和 What's New in Oracle VM Server for SPARC Software (`http://www.oracle.com/technetwork/server-storage/vm/documentation/sparc-whatsnew-330281.html`)（Oracle VM Server for SPARC 软件中的新增功能）。

♦ ♦ ♦ 第 2 章

使用 Oracle VM Server for SPARC 模板

本章包括以下主题：

关于 Oracle VM Server for SPARC 模板

使用 Oracle VM Server for SPARC 模板命令可以为 SPARC 系统创建、部署和配置 Oracle VM Server for SPARC 模板。这些命令均基于 OVF 模板规范，此规范包括磁盘映像文件和包含在归档文件 (`.ova`) 中的 XML 描述符。

您可以在 Oracle VM Server for SPARC 系统的控制域中运行这些命令。除了在命令行上运行这些命令外，还可以在较大的工作流中采用完全自动化的方式或者从其他程序运行它们。

只能使用这些模板部署和配置来宾域、I/O 域和根域。但是，不能使用这些模板部署和配置同时还是 `primary` 域的 I/O 域或根域，也不能部署和配置服务域。

Oracle VM Server for SPARC 模板命令包括：

- `ovmtadm`－允许您通过启动 `ovmtcreate`、`ovmtconfig` 和 `ovmtdeploy` 命令来创建、配置、部署和删除模板。请参见 `ovmtadm(1M)` 手册页。
- `ovmtconfig`－通过将属性名称-值对传输到目标域中的应用程序和进程（例如 Oracle Solaris OS 配置机制和首次引导脚本），对域执行配置操作。属性名称-值对是使用 `ovmtprop` 命令设置的。

  您还可以使用此命令在后台挂载域的 ZFS 文件系统，以便控制域直接在这些文件系统上运行命令。此方法包括了复制文件、安装和升级 Oracle Solaris OS 软件包及执行配置操作等功能。请参见 `ovmtconfig(1M)` 手册页。
- `ovmtcreate`－从现有 Oracle VM Server for SPARC 域创建模板。请参见 `ovmtcreate(1M)` 手册页。

- `ovmtdeploy`－从 Oracle VM Server for SPARC 模板创建域。请参见 `ovmtdeploy(1M)` 手册页。
- `ovmtlibrary`－通过组织文件以及存储、检索和编辑数据库中的信息来管理 Oracle VM Server for SPARC 模板的数据库和基于文件系统的系统信息库。请参见 `ovmtlibrary(1M)` 手册页。
- `ovmtprop`－允许您查看和设置通过模板部署的域中的 Oracle Solaris OS 属性。将属性指定为名称-值对。此命令从其他脚本和程序调用，用以执行配置操作和可以由属性驱动的操作。请参见 `ovmtprop(1M)` 手册页。

有关 Oracle VM Server for SPARC 模板的入门信息，请参见以下博客：

- Deploying a Template (`https://blogs.oracle.com/cmt/en/entry/getting_started_with_ovm_templates`)（部署模板）
- Creating a Template (`https://blogs.oracle.com/cmt/en/entry/getting_started_with_ovm_templates1`)（创建模板）
- Using Templates on Oracle SuperCluster (`https://blogs.oracle.com/cmt/en/entry/getting_started_with_ovm_templates2`)（在 Oracle SuperCluster 上使用模板）

安装 Oracle VM Server for SPARC 模板实用程序

要在控制域安装 Oracle VM Server for SPARC 模板实用程序软件包，必须首先在控制域中下载并安装 Oracle VM Server for SPARC 3.4 软件。请参见《Oracle VM Server for SPARC 3.4 安装指南》 中的 第 2 章, “安装软件”。

然后安装 Oracle VM Server for SPARC 模板实用程序软件包 `ovmtutils`。

```
# pkg install -v pkg:/system/ldoms/ovmtutils
```

Oracle VM Server for SPARC 模板生命周期

本节介绍模板创建过程的各个阶段、执行的操作以及如何使用 Oracle VM Server for SPARC 模板实用程序在此过程中提供帮助：

注 - 使用应用程序和首次引导脚本创建和开发模板是一个重复过程。通过使用源代码管理系统管理脚本和属性来同步配置的所有方面时要谨慎。

下面介绍了模板创建过程的各个阶段、执行的操作以及如何使用 Oracle VM Server for SPARC 模板实用程序在此过程中提供帮助：

1. **设计模板。**预先构建时，系统会提供一般模板，您可以从现有域创建定制模板。此域必须包含要完全安装的所有操作系统组件、应用程序软件以及其他实用程序。

通常，尽可能将环境配置完整，使其只需少量操作就可以配置完毕。任何域设置（如内存、虚拟 CPU、虚拟网络和磁盘）应反映所需的部署。

在此阶段，您可以创建一个或多个“首次引导”脚本。将这些脚本包含在基于您提供的属性执行最终配置的环境中。确保在每个模板的自述文件中记录和描述这些属性。

注 - 如果任何首次引导脚本都能访问域变量，请确保在来宾域中安装 `ovmtprop` 实用程序。

---

2. **创建模板。**创建模板之前，请确保未配置源域环境，以便稍后能够按规定的操作（通常是首次引导脚本的一部分）配置该环境。

   例如，执行以下步骤：

   - 删除任何要稍后重建的特定于应用程序的配置。
   - 使用配置文件的默认值。
   - 导出除根文件系统以外的所有 zpool，以便这些 zpool 能够被新域识别。
   - 将操作系统恢复为未配置状态，以便它可以在通过模板部署后首次引导时接受配置属性。运行以下命令以删除所有站点级自定义、取消配置并停止操作系统：

     ```
     # sysconfig unconfigure -g system -s --include-site-profile --destructive
     ```

   执行这些步骤后，运行 `ovmtcreate` 命令以基于域创建模板。

注 - 如果您运行此命令并且在部署时未提供任何属性，则系统将在下次引导时运行 Oracle Solaris 交互式安装程序。

---

3. **指定模板名称。**使用一致的约定来标识模板，例如使用以下格式：

   technology`.`*OS*`.`*application*`.`*architecture*`.`*build*`.ova`

   例如，以下模板名称适用于在 SPARC 平台上运行内部版本为 2 的 Oracle Solaris 11.2 OS 并运行版本为 12.1.2 的 WebLogic Server 的域：`OVM_S11.2_WLS12.1.2_SPARC_B2.ova`

4. **分发模板。**模板是以 `.ova` 为扩展名的单个文件。该文件包含压缩的磁盘映像和部署所需的元数据。模板还包含有效负荷文件校验和的清单文件，可以将有效负荷文件校验和与整个归档文件校验和相结合，用于验证内容自分发以来是否已更改。

   您可以通过使用基于 Web 的服务或维护中央系统信息库分发模板，而不通过复制模板进行分发。

5. **部署模板。**因为模板仅捕获源域看到的系统部分，所以，您必须了解必须提供哪些服务才能支持模板部署。

   必要服务包括以下项：

   - 一个或多个虚拟交换机，用于提供模板中虚拟网络可能连接的相应接口
   - 虚拟磁盘服务

- 控制台服务
- 足以满足模板要求的虚拟 CPU 和内存

`ovmtdeploy` 实用程序会覆盖其中很多设置，模板提供的最小值代表基准要求。

您可以使用 `ovmtdeploy` 实用程序将虚拟磁盘自动提取、解压缩和复制到部署目录，构建模板描述的各种虚拟设备。

此时，您可以启动域，不过，在该域完全运行之前，您可能需要借助域控制台执行一些手动配置步骤。

6. **自动配置域。**由模板创建的域的配置过程包括多种类型的操作。例如，您可以通过指定属性名称-值对来提供含有要配置的信息的首次引导脚本。您还可以通过将虚拟磁盘向后挂载到控制域在该域文件系统上执行一些操作，例如，复制配置文件。

   借助 `ovmtconfig` 实用程序，这些域配置活动能够实现自动化，您也能够通过指定一个或多个命令脚本和属性文件来指定配置域时要执行的操作以及要使用的属性。

   要配置 Oracle Solaris OS，`ovmtconfig` 实用程序应向后挂载域的根文件系统并从提供的配置脚本和属性中创建 `sc_profile.xml` 文件。借助此配置文件，Oracle Solaris OS 能够在首次引导时对其自身进行配置。

7. **首次配置。**成功配置 Oracle Solaris OS 并首次引导后，必须配置所有安装的应用程序。在配置阶段，`ovmtconfig` 实用程序通过使用以下方法之一将配置信息传递到部署的域：

   - **域变量**－除了本地属性文件外，您还可以通过在控制域中运行 `ovmtconfig` 实用程序来设置域变量，之后，`ovmtprop` 实用程序将在来宾域中使用这些域变量。借助此方法，首次引导脚本能够直接访问属性并在配置完成后直接向来宾域提供配置信息。

     例如，您可以通过使用在来宾域运行 `ovmtprop` 的 supervisor 脚本并从控制域运行 `ovmtconfig -v` 来自动更改无网络访问权限的配置部分。

   - **直接操作**－`ovmtconfig` 实用程序将来宾域文件系统向后挂载到控制域并对文件和文件系统直接操作。这些操作可能包括创建配置文件或复制系统二进制文件。在您提供给 `ovmtconfig` 实用程序的脚本中列出了这些操作。您可以在 `/opt/ovmtutils/share/scripts` 目录中找到执行命令操作的脚本。

     注 - 这些操作通常不包括设计用于在来宾域运行的首次引导过程，因为此类操作可能会影响控制域。

     使用 `ovmtconfig -c` 命令指定要运行的命令。

注意 - 请勿使用未加密属性向域传送敏感信息，例如口令。用于配置 Oracle Solaris OS 以外的其他属性以明文格式作为 `ldm` 变量传送到域。这些属性值对控制域中授权执行 `ldm` 命令的用户以及登录部署的域的用户可见。

此时，域应已完全配置并能运行。

Oracle VM Server for SPARC 模板示例

本节介绍以下 Oracle VM Server for SPARC 模板任务示例：

- 例 1 “创建 Oracle VM Server for SPARC 模板”
- 例 2 “配置 Oracle VM Server for SPARC 模板属性”
- 例 3 “部署 Oracle VM Server for SPARC 模板”
- 例 4 “管理 Oracle VM Server for SPARC 模板库”

例 1 创建 Oracle VM Server for SPARC 模板

以下 `ovmtcreate` 命令基于名为 `ovmtcreate_example` 的 `ldg1` 域创建模板。请注意，生成的模板名称具有 `.ova` 后缀。

```
# ovmtcreate -d ldg1 -o ovmtcreate_example
...

STAGE 1 - EXAMINING SYSTEM AND ENVIRONMENT
------------------------------------------
Performing platform & prerequisite checks
Checking user permissions
Checking for required packages
Checking for required services
Checking directory permissions

STAGE 2 - ANALYZING DOMAIN
--------------------------
Retrieving and processing attributes
Checking domain state
Getting domain resource settings
Discovering network topology
Discovering disk topology

STAGE 3 - ARCHIVE CREATION
--------------------------
Checking destination and current directory capacity
Compressing disk image
Creating XML configuration
Calculating manifest checksums
Creating archive file
Checking archive

PROCESS COMPLETED
-----------------
Started: Tue Aug 18 15:29:14 PDT 2015
Completed: Tue Aug 18 15:41:25 PDT 2015
Elapsed time: 0:12:11
```

例 2 配置 Oracle VM Server for SPARC 模板属性

您可以使用 ovmtconfig 和 ovmtprop 实用程序分别指定 Oracle VM Server for SPARC 模板属性值和 Oracle Solaris OS 属性值。

- 以下 ovmtconfig 命令直接在 ldg1 域的文件系统上执行配置操作。

  -c 选项指定 /opt/ovmtutils/share/scripts/ovmt_s11_scprofile.sh 脚本来设置属性值。-p 选项为 com.oracle.solaris.network.ipaddr 和 com.oracle.solaris.system.computer-name 属性指定特定值。

```
# ovmtconfig -v -d ldg1 -f -s \
-c /opt/ovmtutils/share/scripts/ovmt_s11_scprofile.sh \
-p com.oracle.solaris.network.ipaddr.0=10.153.118.211,\
com.oracle.solaris.system.computer-name=system1
...

STAGE 1/7 - EXAMINING SYSTEM AND ENVIRONMENT
--------------------------------
Checking operating system
Checking platform type
Checking user permissions
Checking packages
Checking host domain name
Checking host domain type
Checking services

STAGE 2/7 - PROCESSING COMMAND LINE OPTIONS
--------------------------------
Parsing individual properties
Creating consolidated properties list

STAGE 3/7 - ANALYZING TARGET DOMAIN
-----------------------
Stopping domain ldg1
Analyzing domain disk topology for domain ldg1
Discovering 1 volumes for vDisks
Examining 1 backend devices
unbinding domain ldg1
Creating 1 virtual disks for back mount
Created virtual disk 0

STAGE 4/7 - PERFORMING BACKMOUNT
--------------------
Finding Solaris device for vdisks
Importing zpools for 1 Solaris devices
```

```
Detected conflicting zpool name, attempting rename
Getting boot file system for properties in 1 zpool
Setting properties in 1 zpools
Mounting ZFS file systems
Mounting ZFS found in zpool rpool_1

STAGE 5/7 - PERFORMING ACTIONS ON TARGET DOMAIN
---------------------------------

STAGE 6/7 - UNMOUNTING AND RESTORING DOMAIN STATE
-----------------------------------
Rolling back commands DEBUG [20150819-07:02:42]: Rolling back 8 /usr/sbin/zfs unmount -f
 rpool_1/ROOT/solaris/var
completed

STAGE 7/7 - SETTING TARGET DOMAIN ENVIRONMENT
--------------------------------
Checking 2 properties to set as domain variables
Process completed
```

- 以下 ovmtprop 命令指定 Oracle Solaris OS 属性值。

```
primary# ovmtprop set-prop com.oracle.solaris.system.computer-name=test ldg1
```

使用 ldm list -l 命令验证 com.oracle.solaris.system.computer-name 属性的值是否为 test。

```
primary# ldm list -l ldg1
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  NORM  UPTIME
ldg1             active     -n----  5000    8     8G       0.0%  0.0%  23h 23m
..
VARIABLES
    auto-boot?=true
    boot-file=-k
    pm_boot_policy=disabled=0;ttfc=2000;ttmr=0;

VMAPI TO GUEST
    com.oracle.solaris.fmri.count=0
    com.oracle.solaris.system.computer-name=test
```

例 3 部署 Oracle VM Server for SPARC 模板

以下 ovmtdeploy 命令通过在 /export/ovmtdeploy 目录中使用 ovmtcreate_example.ova Oracle VM Server for SPARC 模板创建名为 ldg1 的域。

```
# ovmtdeploy -d ldg1 -o /export/ovmtdeploy ovmtcreate_example.ova
...
```

```
STAGE 1 - EXAMINING SYSTEM AND ENVIRONMENT
------------------------------------------
Checking user privilege
Performing platform & prerequisite checks
Checking for required services
Named resourced available

STAGE 2 - ANALYZING ARCHIVE & RESOURCE REQUIREMENTS
---------------------------------------------------
Checking .ova format and contents
Validating archive configuration
Checking sufficient resources present
WARNING: Virtual switch primary-vsw0 already exists

STAGE 3 - EXTRACTING ARCHIVE
----------------------------
Extracting archive
Validating checksums
Decompressing disk image(s)

STAGE 4 - DEPLOYING DOMAIN
--------------------------
Creating domain and adding resources
Validating deployment
Domain created:
```

`ldm list` 输出显示您已创建名为 `ldg1` 的新域。

```
# ldm list
NAME    STATE  FLAGS  CONS VCPU MEMORY UTIL NORM UPTIME
primary active -n-cv- UART 8    40G    1.4% 1.1% 6d 2h 18m
ldg1    active -n---- 5000 8    8G     41%  38%  28s
```

例 4 管理 Oracle VM Server for SPARC 模板库

`ovmtlibrary` 通过组织文件以及存储、检索和编辑数据库中的信息来管理 Oracle VM Server for SPARC 模板的数据库和基于文件系统的系统信息库。

- 以下命令将在 `export/user1/ovmtlibrary_example` 中创建模板库：

  ```
  # ovmtlibrary -c init -l /export/user1/ovmtlibrary_example
  ...

  Init complete
  ```

- 以下命令将 `sol-11_2-ovm-2-sparc.ova` 模板存储在 `export/user1/ovmtlibrary_example` 库中：

  ```
  # ovmtlibrary -c store -d "ovmtlibrary example" -o http://
  system1.example.com/s11.2/templates/sol-11_2-ovm-2-sparc.ova -l /export/
  user1/ovmtlibrary_example
  ...

  Templates present in path "/export/user1/ovmtlibrary_example"

  event id is 2
  ```

```
********************************************************************************
converted 'http://system1.example.com/s11.2/templates/sol-11_2-ovm-2-sparc.ova' (646) ->
'http://system1.example.com/s11.2/templates/sol-11_2-ovm-2-sparc.ova' (UTF-8)
--2015-08-18 16:37:17-- http://system1.example.com/s11.2/templates/sol-11_2-ovm-2-
sparc.ova
Resolving system1.example.com (system1.example.com)... 10.134.127.18
Connecting to system1.example.com (system1.example.com)|10.134.127.18|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1018341888 (971M) [text/plain]
Saving to: '/export/user1/ovmtlibrary_example/repository/templates/1/1/sol-11_2-ovm-2-
sparc.ova'

/export/user1/ovmtlibrary_example/repo 100%
[================================================================================>]
 971.17M 6.05MB/s in 5m 37s
2015-08-18 16:42:55 (2.88 MB/s) - '/export/user1/ovmtlibrary_example/repository/
templates/1/1/sol-11_2-ovm-2-sparc.ova' saved
[1018341888/1018341888]
********************************************************************************
Download complete
Extracting the ova file...
Extract complete
Decompress file System.img.gz
Store complete
```

- 以下命令将列出 export/user1/ovmtlibrary_example 库的内容：

```
# ovmtlibrary -c list -l /export/user1/ovmtlibrary_example
...

Templates present in path "/export/user1/ovmtlibrary_example"

ID Name                         Version Description         Date
--------------------------------------------------------------------------------
1  sol-11_2-ovm-2-sparc         1       ovmtlibrary example 2015-08-18
```

- 以下命令将显示 export/user1/ovmtlibrary_example 库的详细列表：

```
# ovmtlibrary -c list -i 1 -o -l /export/user1/ovmtlibrary_example
...

Templates present in path "/export/user1/ovmtlibrary_example"

ID Name                            Type Path                        Size(bytes)
--------------------------------------------------------------------------------
```

```
1  sol-11_2-ovm-2-sparc.ova         ova /export/user1/ovmtlibrary_example/repository/
templates/1/1/sol-11_2-ovm-2-sparc.ova 1018341888
2  sol-11_2-ovm-sparc.ovf           ovf /export/user1/ovmtlibrary_example/repository/
templates/1/1/sol-11_2-ovm-sparc.ovf 3532
3  sol-11_2-ovm-sparc.mf            mf /export/user1/ovmtlibrary_example/repository/
templates/1/1/sol-11_2-ovm-sparc.mf 137
4  System.img                       img /export/user1/ovmtlibrary_example/repository/
templates/1/1/System.img 21474836480
```

- 以下命令从 `export/user1/ovmtlibrary_example` 库中删除模板 ID `1`：

  ```
  # ovmtlibrary -c delete -i 1 -l /export/user1/ovmtlibrary_example
  ```

♦ ♦ ♦ 第 3 章

3 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：

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。XML 接口仅支持传输层安全 (Transport Layer Security, TLS) 协议版本 1.2。

要禁用远程连接，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，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，以便该服务器可以为您授权以执行监视或管理操作。

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 Oracle VM Server for SPARC 定义的 XML 消息。XML 消息有两种常规类型：

- 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

到 Oracle VM Server for SPARC 的 XML 接口有两种不同的格式：

- 一种格式用于向 Logical Domains Manager 发送命令

- 一种格式用于使 Logical Domains Manager 响应传入消息的状态和在该消息中请求的操作。

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之间的差异，在此分别对它们进行讨论。

请求消息

向 Logical Domains Manager 发出的最基本传入 XML 请求包括对单个对象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。以下示例显示基本 XML 命令的结构。

例 5 在单个对象上执行的单个命令的格式

```
<LDM_interface version="1.3">
  <cmd>
    <action>Place command here</action>
    <options>Place options for certain commands here</options>
    <arguments>Place arguments for certain commands here</arguments>
    <data version="3.0">
      <Envelope>
        <References/>
        <!-- Note a <Section> section can be here instead of <Content>
        <Content xsi:type="ovf:VirtualSystem_Type" id="Domain name">
          <Section xsi:type="ovf:ResourceAllocationSection_type">
            <Item>
              <rasd:OtherResourceType>LDom Resource Type</rasd:OtherResourceType>
              <gprop:GenericProperty
              key="Property name">Property Value</gprop:GenericProperty>
            </Item>
          </Section>

        </Content>
      </Envelope>
    </data>

  </cmd>

</LDM_interface>
```

<LDM_interface> 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 version 属性，如例 5 “在单个对象上执行的单个命令的格式”中所示。

<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 <options> 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：

- `ldm remove-domain` 命令可使用 -a 选项。
- `ldm bind-domain` 命令可使用 -f 选项。
- `ldm add-vdsdev` 命令可使用 -f 选项。
- `ldm cancel-operation` 命令可使用 `migration` 或 `reconf` 选项。
- `ldm add-spconfig` 命令可使用 -r *autosave-name* 选项。
- `ldm remove-spconfig` 命令可使用 -r 选项。
- `ldm list-spconfig` 命令可使用 -r [*autosave-name*] 选项。
- `ldm stop-domain` 命令可以使用以下标记设置命令参数：
  - <force> 表示 -f 选项。
  - <halt> 表示 -h 选项。
  - <message> 表示 -m 选项。
  - <quick> 表示 -q 选项。
  - <reboot> 表示 -r 选项。
  - <timeout> 表示 -t 选项。

  请注意，标记不能包含任何内容值。但是，-t 和 -m 选项必须具有一个非 null 值，例如 <timeout>10</timeout> 或 <message>Shutting down now</message>。

以下 XML 示例片段说明如何通过重新引导消息向 `ldm stop-domain` 命令传递重新引导请求：

```
<action>stop-domain</action>
<arguments>
    <reboot/>
    <message>my reboot message</message>
</arguments>
```

<data> 标记

每个 <data> 段都包含与指定命令相关的对象的描述。<data> 段的格式以开放虚拟化格式 (Open Virtualization Format, OVF) 规范草案中的 XML 模式部分为基础。该模式定义了一个 <Envelope> 段（其中包含一个 <References> 标记，未被 Oracle VM Server for SPARC 使用）以及 <Content> 和 <Section> 段。

对于 Oracle VM Server for SPARC，<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。

如果您只需要标识一个域名，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，用于描述命令所需的资源，该段位于 <Content> 段之外，但仍在 <Envelope> 段之内。

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。

通过两种额外的 OVF 类型，可以使用 OVF 规范的模式来正确定义所有类型的对象：

- <gprop:GenericProperty> 标记
- <Binding> 标记

<gprop:GenericProperty> 标记用于处理任何对象中未在 OVF 规范中定义的属性。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> 标记会在 ldm list-bindings 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> 段可提供状态和消息信息。以下示例显示了基本 XML 请求的响应结构。

例 6 对单个对象上执行的单个命令的响应的格式

```
<LDM_interface version="1.3">
  <cmd>
    <action>Place command here</action>
    <data version="3.0">
      <Envelope>
        <References/>
        <!-- Note a <Section> section can be here instead of <Content>
        <Content xsi:type="ovf:VirtualSystem_Type" id="Domain name">
          <Section xsi:type="ovf:ResourceAllocationSection_type">
            <Item>
              <rasd:OtherResourceType>
                LDom Resource Type
              </rasd:OtherResourceType>
              <gprop:GenericProperty
              key="Property name">
                Property Value
            </gprop:GenericProperty>
            </Item>
          </Section>
          <!-- Note: More <Section>
        </Content>
      </Envelope>
```

```
      <response>
        <status>success or failure</status>
        <resp_msg>Reason for failure</resp_msg>
      </response>
    </data>

    <response>
      <status>success or failure</status>
      <resp_msg>Reason for failure</resp_msg>
    </response>
  </cmd>

  <response>
    <status>success or failure</status>
    <resp_msg>Reason for failure</resp_msg>
  </response>
</LDM_interface>
```

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> 标记用于显示命令成功还是失败。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，失败状态表示执行该命令的某个对象导致发生故障。

对象响应

最后，<cmd> 段中的每个 <data> 段还包含一个 <response> 段。此段可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。如果状态为 FAILURE，则 <response> 字段会显示一个或多个 <resp_msg> 标记，具体取决于对该对象运行该命令时遇到的错误。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“<data> 标记” [22]。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。可以单独订阅或集体订阅的事件类型有三种。有关完整的详细信息，请参见“事件类型” [26]。

注册和注销

使用 <LDM_interface> 消息可注册事件。请参见“<LDM_interface> 标记” [21]。<action> 标记详细说明了要注册或注销的事件类型，<data> 段保留为空。

例 7 事件注册请求消息示例

```
<LDM_interface version="1.3">
  <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。

例 8 事件注册响应消息示例

```
<LDM_interface version="1.3">
  <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 <action> 标记是为触发事件而执行的操作。消息的 <data> 段用于描述与事件关联的对象；其详细信息取决于所发生事件的类型。

例 9 <LDM_event> 通知示例

```
<LDM_event version='1.1'>
  <cmd>
    <action>Event command here</action>
    <data version='3.0'>
      <Envelope
        <References/>
        <Content xsi:type='ovf:VirtualSystem_Type' ovf:id='ldg1'/>
          <Section xsi:type="ovf:ResourceAllocationSection_type">
            <Item>
              <rasd:OtherResourceType>LDom Resource Type</rasd:OtherResourceType>
              <gprop:GenericProperty
              key="Property name">Property Value</gprop:GenericProperty>
            </Item>
          </Section>
      </Envelope>
    </data>
  </cmd>
</LDM_event>
```

事件类型

您可以订阅以下事件类型：

- 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

域事件用于描述可直接对域执行的操作。可以在 <LDM_event> 消息的 <action> 标记中指定以下域事件：

- add-domain
- bind-domain
- domain-reset

- migrate-domain
- panic-domain
- remove-domain
- start-domain
- stop-domain
- unbind-domain

这些事件在 OVF <data> 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，请发送 <LDM_interface> 消息，其中，<action> 标记设置为 reg-domain-events。要取消注册这些事件，请发送 <LDM_interface> 消息，其中，<action> 标记设置为 unreg-domain-events。

硬件事件

硬件事件与更改物理系统硬件相关。如果使用 Oracle VM Server for SPARC 软件，则唯一的硬件更改是，在添加、删除或设置服务处理器 (service processor, SP) 配置时对 SP 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 <data> 段中始终仅包含一个 <Section> 标记，用于描述发生事件的 SP 配置。要注册这些事件，请发送 <LDM_interface> 消息，其中，<action> 标记设置为 reg-hardware-events。要取消注册这些事件，请发送 <LDM_interface> 消息，其中，<action> 标记设置为 unreg-hardware-events。

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 <data> 段中始终仅包含一个 <Section> 标记，用于描述受事件影响的 SP 配置。要注册这些事件，请发送 <LDM_interface> 消息，其中，<action> 标记设置为 reg-hardware-events。要取消注册这些事件，请发送 <LDM_interface> 消息，其中，<action> 标记设置为 unreg-hardware-events。

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。其中某些事件的 <data> 段包含 <Content> 标记，该标记下的 <Section> 标记用于在 OVF <data> 段提供服务名称。

可以在 <LDM_event> 消息的 <action> 标记中指定以下事件：

- add-vdiskserverdevice
- remove-vdiskserverdevice
- set-vdiskserverdevice
- remove-vdiskserver
- set-vconscon
- remove-vconscon
- set-vswitch
- remove-vswitch

以下资源事件在 OVF <data> 段中始终仅包含 <Content> 标记，用于描述发生事件的域。

- add-vcpu
- add-crypto
- add-memory
- add-io
- add-variable
- add-vconscon
- add-vdisk
- add-vdiskserver
- add-vnet
- add-vsan
- add-vswitch
- remove-crypto
- remove-io
- remove-memory
- remove-variable
- remove-vcpu
- remove-vdisk
- remove-vnet
- set-crypto
- set-memory
- set-variable

- set-vconsole
- set-vcpu
- set-vdisk
- set-vnet

要注册资源事件，请发送 <LDM_interface> 消息，其中，<action> 标记设置为 reg-resource-events。要注销这些事件，需要使用 <LDM_interface> 消息，其中，<action> 标记设置为 unreg-resource-events。

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。要同时注册所有这三种事件，需要发送 <LDM_interface> 消息，其中，<action> 标记设置为 reg-all-events。要注销这些事件，需要使用 <LDM_interface> 消息，其中，<action> 标记设置为 unreg-all-events。

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别名。

下面是 <action> 标记中支持的字符串：

- add-domain
- add-io
- add-mau
- add-memory
- add-spconfig
- add-variable
- add-vconscon
- add-vcpu
- add-vdisk
- add-vdiskserver
- add-vdiskserverdevice
- add-vhba

- add-vnet
- add-vsan
- add-vswitch
- bind-domain
- cancel-operation
- list-bindings
- list-constraints
- list-dependencies
- list-devices
- list-domain
- list-hba
- list-rsrc-group
- list-services
- list-spconfig
- list-variable
- list-vmapi
- migrate-domain
- reg-all-events
- reg-domain-events
- reg-hardware-events
- reg-resource-events
- remove-domain
- remove-io
- remove-mau
- remove-memory
- remove-reconf
- remove-spconfig
- remove-variable
- remove-vconscon
- remove-vcpu
- remove-vdisk
- remove-vdiskserver
- remove-vdiskserverdevice
- remove-vhba
- remove-vmapi
- remove-vnet
- remove-vsan
- remove-vswitch

- rescan-vhba
- set-domain
- set-mau
- set-memory
- set-spconfig
- set-variable
- set-vconscon
- set-vconsole
- set-vcpu
- set-vhba
- set-vmapi
- set-vnet
- set-vswitch
- start-domain
- stop-domain
- unbind-domain
- unreg-all-events
- unreg-domain-events
- unreg-hardware-events
- unreg-resource-events

Logical Domains Manager 资源和属性

本节提供 Logical Domains Manager 资源以及可为每种资源定义的属性的示例。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。可以使用约束输出为 Logical Domains Manager 操作创建输入，但域迁移输出除外。请参见“域迁移” [45]。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

以下示例显示 ldom_info 资源的可选属性：

例 10 ldom_info XML 输出示例

以下示例显示为多个 ldom_info 属性指定的值，例如，uuid、hostid 和 Address。

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="primary">
    <Section xsi:type="ovf:ResourceAllocationSection_type">
      <Item>
        <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
        <uuid>c2c3d93b-a3f9-60f6-a45e-f35d55c05fb6</uuid>
        <rasd:Address>00:03:ba:d8:ba:f6</rasd:Address>
        <gprop:GenericProperty key="hostid">83d8baf6</gprop:GenericProperty>
        <gprop:GenericProperty key="master">plum</gprop:GenericProperty>
        <gprop:GenericProperty key="failure-policy">reset</gprop:GenericProperty>
        <gprop:GenericProperty key="extended-mapin-space">on</gprop:GenericProperty>
        <gprop:GenericProperty key="progress">45%</gprop:GenericProperty>
        <gprop:GenericProperty key="status">ongoing</gprop:GenericProperty>
        <gprop:GenericProperty key="source">system1</gprop:GenericProperty>
        <gprop:GenericProperty key="rc-add-policy"></gprop:GenericProperty>
        <gprop:GenericProperty key="perf-counters">global</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

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：

- <uuid> 标记（指定域的 UUID）。
- <rasd:Address> 标记，指定要分配给域的 MAC 地址。
- <gprop:GenericProperty key="extended-mapin-space"> 标记，用于指定为域启用 (on) 还是禁用 (off) 扩展 mapin 空间。默认值为 off。
- <gprop:GenericProperty key="failure-policy"> 标记，用于指定从属域在主域发生故障时应表现出的行为。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 在主域发生故障时使所有从属域都发生紧急情况。
  - reset 在主域发生故障时重置所有从属域。
  - stop 在主域发生故障时停止所有从属域。
- <gprop:GenericProperty key="hostid"> 标记，用于指定要分配给域的主机 ID。
- <gprop:GenericProperty key="master"> 标记，用于最多指定四个以逗号分隔的主域名称。
- <gprop:GenericProperty key="progress"> 标记，用于指定命令执行的进度百分比。
- <gprop:GenericProperty key="source"> 标记，用于指定报告命令进度的计算机。
- <gprop:GenericProperty key="status"> 标记，用于指定命令的状态（done、failed 或 ongoing）。
- <gprop:GenericProperty key="rc-add-policy"> 标记，用于指定对于可能添加到指定域的根联合体是启用还是禁用直接 I/O 和 SR-IOV I/O 虚拟化操作。有效值为 iov 和空值 (rc-add-policy=)。
- <gprop:GenericProperty key="perf-counters"> 标记，用于指定要访问的性能寄存器集（global、htstrand、strand）。

如果平台没有性能访问功能，将忽略 perf-counters 属性值。

CPU (cpu) 资源

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。

cpu 资源始终包含在 <Content> 段中。

例 11 ldm list-bindings 命令的 cpu XML 部分输出

以下示例使用 ldm list-bindings 命令显示 <cpu> 段的 XML 输出。

```
<?xml version="1.0"?>
<LDM_interface
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ovf="./schemas/envelope"
xmlns:rasd="./schemas/CIM_ResourceAllocationSettingData"
xmlns:vssd="./schemas/CIM_VirtualSystemSettingData"
xmlns:gprop="./schemas/GenericProperty"
xmlns:bind="./schemas/Binding"
version="1.3"
xsi:noNamespaceSchemaLocation="./schemas/combined-v3.xsd">
  <cmd>
    <action>list-bindings</action>
    <data version="3.0">
      <Envelope>
        <References/>
        <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="primary">
          <Section xsi:type="ovf:ResourceAllocationSection_Type">
            <Item>
              <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
              <uuid>1e04cdbd-472a-e8b9-ba4c-d3eee86e7725</uuid>
              <rasd:Address>00:21:28:f5:11:6a</rasd:Address>
              <gprop:GenericProperty key="hostid">0x8486632a</gprop:GenericProperty>
              <failure-policy>fff</failure-policy>
              <wcore>0</wcore>
              <extended-mapin-space>0</extended-mapin-space>
              <cpu-arch>native</cpu-arch>
              <rc-add-policy/>
              <gprop:GenericProperty key="state">active</gprop:GenericProperty>
            </Item>
          </Section>
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>cpu</rasd:OtherResourceType>
              <rasd:AllocationUnits>8</rasd:AllocationUnits>
              <bind:Binding>
                <Item>
                  <rasd:OtherResourceType>cpu</rasd:OtherResourceType>
                  <gprop:GenericProperty key="vid">0</gprop:GenericProperty>
                  <gprop:GenericProperty key="pid">0</gprop:GenericProperty>
                  <gprop:GenericProperty key="cid">0</gprop:GenericProperty>
                  <gprop:GenericProperty key="strand_percent">100</gprop:GenericProperty>
                  <gprop:GenericProperty key="util_percent">1.1%</gprop:GenericProperty>
```

```
                <gprop:GenericProperty key="normalized_utilization">0.1%</gprop:
GenericProperty>
              </Item>
          </Section>
        </Content>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

例 12 `ldm list-domain` 命令的 cpu XML 部分输出

以下示例使用 `ldm list-domain` 命令显示 <cpu> 段的 XML 输出。

```
<?xml version="1.0"?>
<LDM_interface
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ovf="./schemas/envelope"
xmlns:rasd="./schemas/CIM_ResourceAllocationSettingData"
xmlns:vssd="./schemas/CIM_VirtualSystemSettingData"
xmlns:gprop="./schemas/GenericProperty"
xmlns:bind="./schemas/Binding"
version="1.3"
xsi:noNamespaceSchemaLocation="./schemas/combined-v3.xsd">
  <cmd>
    <action>list-domain</action>
    <data version="3.0">
      <Envelope>
        <References/>
        <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="primary">
          <Section xsi:type="ovf:ResourceAllocationSection_Type">
            <Item>
              <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
              <gprop:GenericProperty key="state">active</gprop:GenericProperty>
              <gprop:GenericProperty key="flags">-n-cv-</gprop:GenericProperty>
              <gprop:GenericProperty key="utilization">0.7%</gprop:GenericProperty>
              <gprop:GenericProperty key="uptime">3h</gprop:GenericProperty>
              <gprop:GenericProperty key="normalized_utilization">0.1%</gprop:
GenericProperty>
            </Item>
          </Section>
        </Content>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

MAU (`mau`) 资源

`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。

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)。

例 13 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>
```

内存 (memory) 资源

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。

例 14 memory XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>memory</rasd:OtherResourceType>
        <rasd:AllocationUnits>4G</rasd:AllocationUnits>
      </Item>
    </Section>
  </Content>
</Envelope>
```

虚拟 SAN (vsan) 资源

虚拟 SAN (vsan) 资源可以位于 <Content> 段中。它必须使用带有以下项的 <gprop:GenericProperty> 标记：

- service_name－虚拟 SAN 的名称
- vsan_iport－虚拟 SAN 的启动器端口

必须具有在 vsan_volume 段中包含单个 vol_name 属性的项。此 vol_name 属性必须设置为 *。

例 15 vsan XML 示例

```
<Envelope>
    <References/>
```

```
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
        <Item>
            <rasd:OtherResourceType>vsan</rasd:OtherResourceType>
            <gprop:GenericProperty key="service_name">vs1</gprop:GenericProperty>
            <gprop:GenericProperty key="vsan_iport">
            /pci@400/pci@1/pci@0/pci@8/SUNW,emlxs@0/fp@0,0</gprop:GenericProperty>
            <bind:Binding>
                <Item>
                    <rasd:OtherResourceType>vsan_volume</rasd:OtherResourceType>
                    <gprop:GenericProperty key="vol_name">*</gprop:GenericProperty>
                </Item>
            </bind:Binding>
        </Item>
    </Section>
</Envelope>
```

虚拟磁盘服务器 (vds) 资源

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，该标记具有 service_name 键，其中包含所描述的 vds 资源的名称。

例 16 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>
```

虚拟磁盘服务器卷 (vds_volume) 资源

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）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

例 17 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>
        <gprop:GenericProperty key="vol_opts">ro</gprop:GenericProperty>
        <gprop:GenericProperty key="mpgroup">mpgroup-name</gprop:GenericProperty>
      </Item>
    </Section>
</Envelope>
```

磁盘 (disk) 资源

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）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（秒）。如果具有多个虚拟磁盘 (vdisk) 路径，则 vdc 可以尝试连接到其他 vds。超时设置可以确保在指定时间段内与任何 vds 建立连接。

例 18 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>
```

虚拟交换机 (vsw) 资源

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。它必须具有带 service_name 键的 <gprop:GenericProperty> 标记，该键是要分配给虚拟交换机的名称。

（可选）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
- default-vlan-id－指定在标记模式下，需要包含虚拟网络设备或虚拟交换机的默认虚拟局域网 (virtual local area network, VLAN)。第一个 VLAN ID (*vid1*) 是为 default-vlan-id 预留的。
- dev_path－要与此虚拟交换机关联的网络设备的路径
- id－指定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。
- inter_vnet_link－指定是否为 Inter-Vnet 通信分配 LDC 通道。有效值为 on、off 和 auto。缺省值为 auto。
- linkprop－指定虚拟设备获取物理链路状态更新（缺省值为 phys-state）。当值为空时，虚拟设备不会获取物理链路状态更新。
- mode－Oracle Solaris Cluster 心跳支持的 sc。
- 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。
- mtu－指定虚拟交换机或/和绑定到虚拟交换机的虚拟网络设备的最大传输单元 (maximum transmission unit, MTU)。有效值是 1500 到 16000。如果指定了无效值，ldm 命令会发出错误。
- 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。

例 19 vsw XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg2">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vsw</rasd:OtherResourceType>
        <rasd:Address>00:14:4f:fb:ec:00</rasd:Address>
        <gprop:GenericProperty key="service_name">test-vsw1</gprop:GenericProperty>
        <gprop:GenericProperty key="inter_vnet_link">auto</gprop:GenericProperty>
        <gprop:GenericProperty key="default-vlan-id">1</gprop:GenericProperty>
        <gprop:GenericProperty key="pvid">1</gprop:GenericProperty>
        <gprop:GenericProperty key="mtu">1500</gprop:GenericProperty>
        <gprop:GenericProperty key="dev_path">switch@0</gprop:GenericProperty>
        <gprop:GenericProperty key="id">0</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
```

```
</Envelope>
```

网络 (network) 资源

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－指定虚拟设备获取物理链路状态更新（缺省值为 phys-state）。当值为空时，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
- custom－指定对于可以从受信任主机分配给虚拟网络设备的最大 VLAN 和 MAC 地址数，应当启用还是禁用这些项的定制设置。缺省值为 disable。
- custom/max-mac-addrs－指定可以从受信任主机分配给虚拟网络设备的最大 MAC 地址数。缺省值为 4096。
- custom/max-vlans－指定可以从受信任主机分配给虚拟网络设备的最大 VLAN 数。缺省值为 4096。

（可选）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。

例 20 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="custom">enable</gprop:GenericProperty>
        <gprop:GenericProperty key="custom/max-mac-addrs">4096</gprop:GenericProperty>
        <gprop:GenericProperty key="custom/max-vlans">12</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>
```

虚拟控制台集中器 (vcc) 资源

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

例 21 vcc XML 示例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vcc</rasd:OtherResourceType>
        <gprop:GenericProperty key="service_name">vcc1</gprop:GenericProperty>
        <gprop:GenericProperty key="min_port">6000</gprop:GenericProperty>
        <gprop:GenericProperty key="max_port">6100</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

变量 (var) 资源

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

例 22 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>
```

物理 I/O 设备 (physio_device) 资源

physio_device 资源始终包含在 <Content> 段中。可以使用 add-io、set-io、remove-io、create-vf、destroy-vf 和 set-domain 子命令来修改此资源。

例 23 physio_device XML 示例

以下示例说明如何对虚拟功能、物理功能和根联合体执行操作。

- 以下 XML 示例片段说明如何使用 ldm add-io 命令将 /SYS/MB/NET0/IOVNET.PF0.VF0 虚拟功能添加到 ldg1 域。

```
<LDM_interface version="1.3">
  <cmd>
    <action>add-io</action>
    <data version="3.0">
      <Envelope>
        <References/>
        <Content xsi:type="ovf:VirtualSystem_Type" ovf:id="ldg1">
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
              <gprop:GenericProperty key="name">
              /SYS/MB/NET0/IOVNET.PF0.VF0</gprop:GenericProperty>
            </Item>
          </Section>
        </Content>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

- 以下 XML 示例片段说明如何使用 ldm set-io 命令将 pci_1 根联合体的 iov_bus_enable_iov 属性值设置为 on。

```
<LDM_interface version="1.3">
  <cmd>
    <action>set-io</action>
    <data version="3.0">
      <Envelope>
        <References/>
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
              <gprop:GenericProperty key="name">pci_1</gprop:GenericProperty>
```

```
            <gprop:GenericProperty key="iov_bus_enable_iov">
            on</gprop:GenericProperty>
          </Item>
        </Section>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

- 以下 XML 示例片段说明如何使用 `ldm set-io` 命令将 `/SYS/MB/NET0/IOVNET.PF1` 物理功能的 `unicast-slots` 属性值设置为 `6`。

```
<LDM_interface version="1.3">
  <cmd>
    <action>set-io</action>
    <data version="3.0">
      <Envelope>
        <References/>
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
              <gprop:GenericProperty key="name">
              /SYS/MB/NET0/IOVNET.PF1</gprop:GenericProperty>
              <gprop:GenericProperty key="unicast-slots">6</gprop:GenericProperty>
            </Item>
          </Section>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

- 以下 XML 示例片段说明如何使用 `ldm create-vf` 命令创建具有以下属性值的 `/SYS/MB/NET0/IOVNET.PF1.VF0` 虚拟功能。
  - `unicast-slots=6`
  - `pvid=3`
  - `mtu=1600`

```
<LDM_interface version="1.3">
  <cmd>
    <action>create-vf</action>
    <data version="3.0">
      <Envelope>
        <References/>
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>vf_device</rasd:OtherResourceType>
```

```
            <gprop:GenericProperty key="iov_pf_name">
            /SYS/MB/NET0/IOVNET.PF1</gprop:GenericProperty>
            <gprop:GenericProperty key="unicast-slots">6</gprop:GenericProperty>
            <gprop:GenericProperty key="pvid">3</gprop:GenericProperty>
            <gprop:GenericProperty key="mtu">1600</gprop:GenericProperty>
          </Item>
        </Section>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

- 以下 XML 示例片段说明如何使用 `ldm create-vf` 命令创建 `iov_pf_repeat_count_str` 值 (3) 指定的虚拟功能数以及 `/SYS/MB/NET0/IOVNET.PF1` 物理功能。使用 `iov_pf_repeat_count_str` 属性创建多个虚拟功能时，不能指定其他属性值。

```
<LDM_interface version="1.3">
  <cmd>
    <action>create-vf</action>
    <data version="3.0">
      <Envelope>
        <References/>
          <Section xsi:type="ovf:VirtualHardwareSection_Type">
            <Item>
              <rasd:OtherResourceType>vf_device</rasd:OtherResourceType>
              <gprop:GenericProperty key="iov_pf_name">
              /SYS/MB/NET0/IOVNET.PF1</gprop:GenericProperty>
              <gprop:GenericProperty key="iov_pf_repeat_count_str">
              3</gprop:GenericProperty>
            </Item>
          </Section>
      </Envelope>
    </data>
  </cmd>
</LDM_interface>
```

SP 配置 (spconfig) 资源

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。

例 24 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>
          key="spconfig_aux_status">degraded</gprop:GenericProperty>
      </Item>
    </Section>
</Envelope>
```

DRM 策略配置 (policy) 资源

DRM 策略 (policy) 资源显示在 <Envelope> 段，可以包含带以下键的 <gprop:GenericProperty> 标记：

- policy_name－DRM 策略的名称
- policy_enable－指定 DRM 策略启用还是禁用
- policy_priority－DRM 策略的优先级
- policy_vcpu_min－域的虚拟 CPU 资源的最小数量
- policy_vcpu_max－域的虚拟 CPU 资源的最大数量
- policy_util_lower－触发策略分析的最低利用率级别
- policy_util_upper－触发策略分析的最高利用率级别
- policy_tod_begin－DRM 策略的有效启动时间
- policy_tod_end－DRM 策略的有效停止时间
- policy_sample_rate－抽样率，这是以秒为单位的周期时间
- policy_elastic_margin－介于 CPU 利用率上下限之间的缓冲量
- policy_attack－在任一资源控制周期内可添加的最大资源量
- policy_decay－在任一资源控制周期内可删除的最大资源量

例 25 policy XML 示例

```
<Envelope>
   <Section xsi:type="ovf:VirtualHardwareSection_Type">
     <Item>
       <rasd:OtherResourceType>policy</rasd:OtherResourceType>
       <gprop:GenericProperty key="policy_name">test-policy</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_enable">on</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_priority">1</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_vcpu_min">12</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_vcpu_max">13</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_util_lower">8</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_util_upper">9</gprop:GenericProperty>
       <gprop:GenericProperty key="policy_tod_begin">07:08:09</gprop:GenericProperty>
```

```
      <gprop:GenericProperty key="policy_tod_end">09:08:07</gprop:GenericProperty>
      <gprop:GenericProperty key="policy_sample_rate">1</gprop:GenericProperty>
      <gprop:GenericProperty key="policy_elastic_margin">8</gprop:GenericProperty>
      <gprop:GenericProperty key="policy_attack">8</gprop:GenericProperty>
      <gprop:GenericProperty key="policy_decay">9</gprop:GenericProperty>
    </Item>
  </Section>
</Envelope>
```

控制台 (console) 资源

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
- enable-log－对此控制台启用或禁用虚拟控制台日志记录

例 26 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>
        <gprop:GenericProperty key="service_name">vcc2</gprop:GenericProperty>
        <gprop:GenericProperty key="group">group-name</gprop:GenericProperty>
        <gprop:GenericProperty key="enable-log">on</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

域迁移

以下示例显示了 ldm migrate-domain 命令的 <data> 段中包含的内容。

- 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，该用户名必须采用 SASL 64 位编码。
- `password` 是目标计算机的登录密码，该密码必须采用 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。

例 27 `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>
```

XML 模式

由 Logical Domains Manager 使用的 XML 模式位于 `/opt/SUNWldm/bin/schemas` 目录中。文件名如下：

- `cim-common.xsd`－`cim-common.xsd` 模式
- `cim-rasd.xsd`－`cim-rasd.xsd` 模式
- `cim-vssd.xsd`－`cim-vssd.xsd` 模式
- `cli-list-constraint-v3.xsd`－`cli-list-constraint-v3.xsd` 模式
- `combined-v3.xsd`－`LDM_interface` XML 模式
- `event-v3.xsd`－`LDM_Event` XML 模式
- `ldmd-binding.xsd`－`Binding_Type` XML 模式
- `ldmd-property.xsd`－`GenericProperty` XML 模式
- `ovf-core.xsd`－`ovf-core.xsd` 模式
- `ovf-envelope.xsd`－`ovf-envelope.xsd` 模式
- `ovf-section.xsd`－`ovf-section.xsd` 模式
- `ovf-strings.xsd`－`ovf-strings.xsd` 模式
- `ovfenv-core.xsd`－`ovfenv-core.xsd` 模式
- `ovfenv-section.xsd`－`ovfenv-section.xsd` 模式

♦ ♦ ♦ 第 4 章

Logical Domains Manager 发现

本章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

通过使用多播消息，可在子网上发现 Logical Domains Managers。`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`。

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;
```

▼ 如何发现在子网上运行的 Logical Domains Managers

1. **打开多播套接字。**
   确保您使用的是“多播通信” [47]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 Managers 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 Managers 的响应时，请确保已放弃任何自动分配 MAC 冲突检测包。

♦ ♦ ♦ 第 5 章

5 使用虚拟域信息命令和 API

本章介绍了虚拟域信息命令和 API。

使用虚拟域信息命令

使用 `virtinfo` 命令可以收集有关运行的虚拟域的信息。
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。

使用虚拟域信息 API

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。请参见 `libv12n`(3LIB) 和 `v12n`(3EXT) 手册页。

索引