Logical Domains 1.3 管理ガイド

Part No: 821–1077–10
2010 年 1 月

100301@23474

目次

はじめに

『Logical Domains 1.3 管理ガイド』では、サポートされるサーバー、ブレード、およびサーバーモジュールでの Logical Domains Manager 1.3 ソフトウェアの概要、セキュリティー上の考慮事項、インストール、構成、変更、および一般的なタスクの実行に関する詳細な情報や手順について説明します。一覧については、『Logical Domains 1.3 リリースノート』の「サポートされるプラットフォーム」を参照してください。

このマニュアルは、UNIX® システムおよび Solaris™ オペレーティングシステム (Solaris OS) の実践的な知識がある、これらのサーバーのシステム管理者を対象としています。

関連マニュアル

次の表に、LogicalDomains1.3 リリースで利用できるマニュアルを示します。これらのマニュアルは、特に記載がないかぎり、HTML 形式と PDF 形式で利用できます。

表 P–1 関連マニュアル

用途	タイトル	Part No.
LogicalDomains1.3 ソフトウェア	『Logical Domains 1.3 管理ガイド』	821-1077
	『Logical Domains 1.3 リリースノート』	821-1079
	『Logical Domains 1.3 リファレンスマニュアル』 Solaris 10 Reference Manual Collection ■ `drd`(1M) マニュアルページ ■ `vntsd`(1M) マニュアルページ	821-1078
LDoms ソフトウェアの基本	『Beginners Guide to LDoms: Understanding and Deploying Logical Domains』(PDF)	820-0832
LDoms 管理情報ベース (MIB)	『Logical Domains (LDoms) MIB 1.0.1 Administration Guide』	820-3456-10
	『Logical Domains (LDoms) MIB 1.0.1 Release Notes』	820-3462-10

表 P–1 関連マニュアル (続き)

用途	タイトル	Part No.
Solaris OS: インストールと構成	Solaris 10 10/09 Release and Installation Collection	なし

使用しているサーバー、ソフトウェア、または Solaris OS に関連するマニュアルは、http://docs.sun.com で参照できます。必要なマニュアルや情報を検索するには、「Search」ボックスを使用します。

コメントをお寄せください

マニュアルの品質改善のため、お客様からのご意見およびご要望をお待ちしております。コメントは http://www.sun.com/secure/products-n-solutions/hardware/docs/feedback/ よりお送りください。

ご意見をお寄せいただく際には、下記のタイトルと Part No. を記載してください。『Logical Domains 1.3 管理ガイド』、Part No. 821-1077-10

マニュアル、サポート、およびトレーニング

Sun の Web サイトでは、次の追加リソースに関する情報を提供しています。

- マニュアル (http://jp.sun.com/documentation/)
- サポート (http://jp.sun.com/support/)
- トレーニング (http://jp.sun.com/training/)

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 P–2 表記上の規則

字体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力を示します。	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 machine_name% you have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	machine_name% **su** Password:

表 P–2 表記上の規則 *(続き)*

字体または記号	意味	例
aabbcc123	可変部分: 実際に使用する特定の名前または値で置き換えます。	ファイルを削除するコマンドは、`rm` *filename* です。
AaBbCc123	書名、新規単語、強調する単語を示します。	『ユーザーズガイド』の第 6 章を参照してください。 キャッシュは、ローカルに格納されたコピーです。 ファイルを保存しないでください。 注: 一部の強調された項目はオンラインでは太字で表示されます。

コマンド例のシェルプロンプト

次の表に、Solaris OS に含まれるシェルのデフォルトの UNIX システムプロンプトおよびスーパーユーザーのプロンプトを示します。コマンド例に示されるデフォルトのシステムプロンプトは Solaris のリリースに応じて異なることに注意してください。

表 P–3 シェルプロンプト

シェル	プロンプト
Bash シェル、Korn シェル、および Bourne シェル	`$`
スーパーユーザーの Bash シェル、Korn シェル、および Bourne シェル	`#`
C シェル	`machine_name%`
スーパーユーザーの C シェル	`machine_name#`

◆ ◆ ◆ 第 1 章

Logical Domains ソフトウェアの概要

この章では、Logical Domains ソフトウェアの概要について説明します。

Sun の Logical Domains ソフトウェアは、特定の Solaris OS バージョン、必須ソフトウェアパッチ、および特定バージョンのシステムファームウェアに依存しています。詳細は、『Logical Domains 1.3 リリースノート』の「必須および推奨される Solaris OS」を参照してください。

この章の内容は次のとおりです。

注 – Logical Domains 1.3 ソフトウェアは、OpenSolaris 2009.06 リリース以降の OpenSolaris OS でサポートされています。Logical Domains 1.3 のマニュアルでは、Solaris 10 OS での Logical Domains の使用法を中心に説明します。Logical Domains は、Solaris 10 OS と OpenSolaris OS の両方で同じ機能を使用できます。ただし、OpenSolaris OS で Logical Domains を使用する場合には、わずかに異なる点があることがあります。OpenSolaris OS については、OpenSolaris Information Center を参照してください。

ハイパーバイザと論理ドメイン

この節では、Logical Domains をサポートしている SPARC® ハイパーバイザの概要について説明します。

SPARC ハイパーバイザは、小さなファームウェア層で、オペレーティングシステムを記述できる安定した仮想化マシンアーキテクチャーを提供します。ハイパーバイザを使用する Sun サーバーでは、論理オペレーティングシステムの活動をハイパーバイザが制御できるようにするためのハードウェア機能が用意されています。

論理ドメインは、リソースの個別の論理グループで構成される仮想マシンです。論理ドメインは、単一のコンピュータシステム内で独自のオペレーティングシステムおよびIDを持っています。各論理ドメインは独立して作成、削除、再構成、および再起動することができ、そのときサーバーの電源の再投入は必要ありません。パフォーマンスおよびセキュリティー上の理由から、さまざまなアプリケーションソフトウェアを異なる論理ドメイン上で動作させて、アプリケーションの独立性を維持することができます。

各論理ドメインは、ハイパーバイザがそのドメインに対して利用可能にしたサーバーリソースに対してのみ、監視および対話が許可されています。Logical Domains Managerを使用すると、ハイパーバイザが制御ドメインを介して実行する処理を指定できます。つまり、ハイパーバイザは、サーバーのリソースをパーティションに分割し、限定的なサブセットを複数のオペレーティングシステム環境に提供します。このパーティションの分割と提供は、論理ドメインを作成する場合の基本的なメカニズムです。次の図に、2つの論理ドメインをサポートするハイパーバイザを示します。また、Logical Domainsの機能を構成する次の層についても示します。

- アプリケーションまたはユーザー/サービス
- カーネルまたはオペレーティングシステム
- ファームウェアまたはハイパーバイザ
- ハードウェア(CPU、メモリー、I/Oなど)

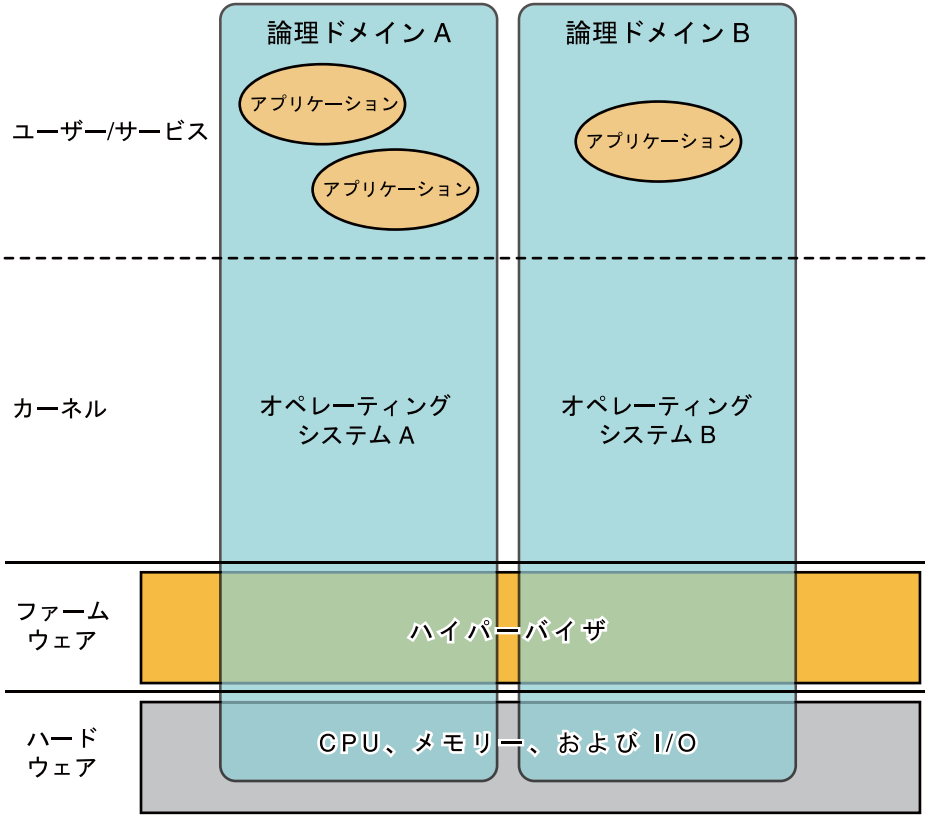

図 1–1　2つの論理ドメインをサポートするハイパーバイザ

特定の SPARC ハイパーバイザがサポートする各論理ドメインの数と機能は、サーバーによって異なります。ハイパーバイザは、サーバー全体の CPU、メモリー、および I/O リソースのサブセットを特定の論理ドメインに割り当てることができます。これにより、それぞれが独自の論理ドメイン内にある複数のオペレーティングシステムを同時にサポートすることができます。リソースは、任意に細分化して個々の論路ドメイン間で再配置できます。たとえば、メモリーは 8K バイトの単位で論理ドメインに割り当てることができます。

各論理ドメインは、次のような独自のリソースを持つ完全に独立したマシンとして管理できます。

- カーネル、パッチ、およびチューニングパラメータ
- ユーザーアカウントおよび管理者
- ディスク
- ネットワークインタフェース、MAC アドレス、および IP アドレス

各論理ドメインは、サーバーの電源の再投入を必要とすることなく、互いに独立して停止、起動、および再起動できます。

ハイパーバイザソフトウェアは、論理ドメイン間の分離を維持する役割を果たします。ハイパーバイザソフトウェアは、論理ドメインが相互に通信できるように論理ドメインチャネル (LDC) も提供します。LDC を使用することで、ドメインはネットワークサービスやディスクサービスなどのサービスを相互に提供できます。

サービスプロセッサ (SP) はシステムコントローラ (SC) とも呼ばれ、物理マシンを監視および実行しますが、論理ドメインは管理しません。論理ドメインは、Logical Domains Manager で管理します。

Logical Domains Manager

Logical Domains Manager は、論理ドメインの作成と管理、および物理リソースへの論理ドメインの割り当てを行うために使用します。Logical Domains Manager は、サーバーごとに 1 つだけ実行できます。

論理ドメインの役割

論理ドメインはすべて同じですが、論理ドメインに対して指定する役割に基づいてそれぞれ区別できます。論理ドメインが実行できる役割は、次のとおりです。

- 制御ドメイン。このドメインでは、Logical Domains Manager が実行されます。ほかの論理ドメインを作成および管理したり、ほかのドメインに仮想リソースを割り当てたりすることができます。制御ドメインは、サーバーごとに 1 つだけ存在できます。制御ドメインは、Logical Domains ソフトウェアをインストールすると最初に作成されるドメインです。制御ドメインの名前は `primary` です。
- サービスドメイン。サービスドメインは、仮想スイッチ、仮想コンソール端末集配信装置、仮想ディスクサーバーなどの仮想デバイスサービスをほかのドメインに提供します。どのドメインも、サービスドメインとして構成できます。
- **I/O** ドメイン。I/O ドメインは、PCI EXPRESS® コントローラのネットワークカードなど、物理 I/O デバイスに対する直接の所有権と直接のアクセス権を持っています。I/O ドメインは、仮想デバイスの形式で、ほかのドメインと物理デバイスを共有するためにサービスドメインとして使用されることがよくあります。制御ドメインは I/O ドメインであり、サービスドメインとしても使用できます。設定できる I/O ドメインの数は、プラットフォームによって異なります。たとえば、Sun SPARC Enterprise® Server T5440 を使用している場合は、最大 4 つの I/O ドメインを設定できます。
- ゲストドメイン。ゲストドメインは、I/O ドメイン以外のドメインで、1 つ以上のサービスドメインによって提供される仮想デバイスサービスを使用します。ゲストドメインは、物理 I/O デバイスを持っておらず、仮想ディスクや仮想ネットワークインタフェースなどの仮想 I/O デバイスのみを持ちます。

Logical Domains Manager は、Logical Domains でまだ構成されていない既存のシステムにインストールできます。この場合、OS の現在のインスタンスが制御ドメインになります。また、このシステムは、唯一のドメインとして制御ドメインを持つ Logical Domains システムとして構成されます。制御ドメインを構成したあと、システム全体をもっとも効率的に利用できるように、アプリケーションの負荷をほかのドメイン間で分散できます。これを行うには、ドメインを追加し、制御ドメインから新しいドメインにアプリケーションを移動します。

コマンド行インタフェース

Logical Domains Manager は、コマンド行インタフェース (CLI) を使用して、論理ドメインを作成および構成します。CLI には、単一のコマンド `ldm` があり、これは複数のサブコマンドを備えています。`ldm`(1M) マニュアルページを参照してください。

Logical Domains Manager CLIを使用するには、Logical Domains Manager デーモン (`ldmd`) が実行されている必要があります。

仮想入出力

Logical Domains 環境では、UltraSPARC® T2 Plus プロセッサ システム上に最大 128 のドメインをプロビジョニングすることができます。これらのシステムでは、I/O バスおよび物理 I/O スロットの数に制限があります。そのため、これらのシステムのすべてのドメインに対して、物理ディスクおよびネットワークデバイスへの排他的なアクセスを提供することはできません。PCI バスをドメインに割り当てて、物理デバイスへのアクセスを提供できます。この解決方法は、すべてのドメインにデバイスへの排他的なアクセスを提供するには不十分です。63 ページの「I/O ドメインと PCI EXPRESS バス」 を参照してください。このように物理 I/O デバイスへの直接アクセスが不足している状況は、仮想化 I/O モデルを実装することで対処されます。

物理 I/O アクセスを行わない論理ドメインは、サービスドメインと通信する仮想 I/O デバイスを使用して構成されます。サービスドメインは、仮想デバイスサービスを実行して、物理デバイスまたはその機能にアクセスを提供します。このようなクライアントサーバーモデルで、仮想 I/O デバイスは、論理ドメインチャネル (LDC) と呼ばれるドメイン間通信チャネルを使用して、相互に、またはサービスの対象と通信します。仮想化 I/O 機能には、仮想ネットワーク、ストレージ、およびコンソールのサポートが含まれています。

仮想ネットワーク

Logical Domains は、仮想ネットワークデバイスおよび仮想ネットワークスイッチデバイスを使用して、仮想ネットワークを実装します。仮想ネットワーク (`vnet`) デバイスは、Ethernet デバイスをエミュレートし、ポイントツーポイントチャネルを使用してシステム内のほかの `vnet` デバイスと通信します。仮想スイッチ (`vsw`) デバイス

は、主に仮想ネットワークのすべての受信パケットおよび送信パケットのマルチプレクサとして機能します。vsw デバイスは、サービスドメインの物理ネットワークアダプタに直接接続し、仮想ネットワークの代わりにパケットを送受信します。vsw デバイスは、単純なレイヤー 2 スイッチとしても機能し、システム内で vsw デバイスに接続された vnet デバイス間でパケットをスイッチします。

仮想ストレージ

仮想ストレージインフラストラクチャーは、クライアントサーバーモデルを使用して、論理ドメインに直接割り当てられていないブロックレベルのストレージに論理ドメインがアクセスできるようにします。このモデルは、次のコンポーネントを使用します。

- ブロック型デバイスインタフェースをエクスポートする仮想ディスククライアント (vdc)
- 仮想ディスククライアントの代わりにディスク要求を処理して、その要求をサービスドメイン上に存在するバックエンドストレージに送信する仮想ディスクサービス (vds)

クライアントドメインでは仮想ディスクは通常のディスクとして認識されますが、ほとんどのディスク操作は仮想ディスクサービスに転送され、サービスドメインで処理されます。

仮想コンソール

Logical Domains 環境では、primary ドメインからのコンソール I/O は、サービスプロセッサに転送されます。ほかのすべてのドメインからのコンソール I/O は、仮想コンソール端末集配信装置 (vcc) を実行しているサービスドメインにリダイレクトされます。通常、vcc を実行するドメインは、primary ドメインです。仮想コンソール端末集配信装置サービスは、すべてのドメインのコンソールトラフィックの端末集配信装置として機能します。また、仮想ネットワーク端末サーバーデーモン (vntsd) とのインタフェースを提供し、UNIX ソケットを使用して各コンソールへのアクセスを提供します。

動的再構成

動的再構成 (DR) は、オペレーティングシステムの動作中にリソースを追加または削除できる機能です。特定のリソースタイプの動的再構成が実行可能かどうかは、論理ドメインで動作している OS でのサポート状況によって異なります。

動的再構成は、次のリソースに対してサポートされています。

- 仮想 **CPU** – すべてのバージョンの Solaris 10 OS でサポート
- 仮想 **I/O** デバイス – Solaris 10 10/08 OS 以降でサポート
- 暗号化装置 – Solaris 10 10/09 OS 以上でサポート

- メモリー – サポートなし
- 物理 I/O デバイス – サポートなし

動的再構成機能を使用するには、変更するドメインで Logical Domains 動的再構成デーモン (drd) を実行する必要があります。drd(1M) マニュアルページを参照してください。

遅延再構成

即座に有効になる動的再構成処理とは対照的に、遅延再構成処理は、次の状況で有効になります。

- OS の次回の再起動後
- 論理ドメインの停止および起動後

Logical Domains Manager 1.2 ソフトウェア以降では、遅延再構成処理は制御ドメインに制限されます。ほかのすべてのドメインの場合、リソースの動的再構成が可能でないかぎり、構成を変更するにはドメインを停止する必要があります。

制御ドメインで遅延再構成が進行中の場合、その制御ドメインが再起動するまで、または停止して起動するまで、その制御ドメインに対するその他の再構成要求は延期されます。また、制御ドメインに対して未処理の遅延再構成がある場合、その他の論理ドメインに対する再構成要求は厳しく制限され、適切なエラーメッセージを表示して失敗します。

Logical Domains Manager の `ldm cancel-operation reconf` コマンドは、制御ドメインの遅延再構成処理を取り消します。遅延再構成処理は、`ldm list-domain` コマンドを使用して一覧表示できます。遅延再構成機能の使用法については、ldm(1M) マニュアルページを参照してください。

注 – その他の `ldm remove-*` コマンドが仮想 I/O デバイスで遅延再構成処理をすでに実行している場合、`ldm cancel-operation reconf` コマンドを使用できません。このような状況では、`ldm cancel-operation reconf` コマンドは失敗します。

持続的な構成

`ldm` コマンドを使用して、論理ドメインの現在の構成をサービスプロセッサに格納できます。構成の追加、使用する構成の指定、構成の削除、および構成の表示を行うことができます。ldm(1M) マニュアルページを参照してください。SP から起動する構成を指定することもできます。178 ページの「LDoms とサービスプロセッサの使用」 を参照してください。

構成の管理については、170 ページの「Logical Domains 構成の管理」 を参照してください。

Logical Domains Physical-to-Virtual 移行ツール

Logical Domains Physical-to-Virtual (P2V) 移行ツールは、既存の物理システムを、チップマルチスレッディング (CMT) システム上の論理ドメインで実行される仮想システムに自動的に変換します。ソースシステムは、次のいずれかにすることができます。

- Solaris 8 オペレーティングシステム以降で動作する `sun4u` SPARC システム
- Solaris 10 OS を実行していても、Logical Domains ソフトウェアを実行していない `sun4v` システム

ツールとそのインストールについては、付録 C 「Logical Domains Physical-to-Virtual 移行ツール」 を参照してください。`ldmp2v` コマンドについては、`ldmp2v(1M)` マニュアルページを参照してください。

Logical Domains Configuration Assistant

Logical Domains Configuration Assistant を使用すると、基本的なプロパティーを設定することによって論理ドメインの構成手順を実行できます。Logical Domains Configuration Assistant は、Sun CoolThreads サーバーと呼ばれる CMT ベースのシステム上で実行されます。このツールを使用すると、Logical Domains ソフトウェアがインストールされていて、まだその構成が行われていないシステムを構成できます。

Configuration Assistant は、構成データを収集したあと、論理ドメインとして起動するのに適した構成を作成します。Configuration Assistant によって選択されるデフォルト値を使用して、有効なシステム構成を作成することもできます。

Configuration Assistant は、グラフィカルユーザーインタフェース (GUI) ツールおよび端末ベースのツールの両方として使用できます。

詳細は、付録 D 「Logical Domains Configuration Assistant」 および `ldmconfig(1M)` マニュアルページを参照してください。

◆ ◆ ◆ 第 2 章

ソフトウェアのインストールおよび有効化

この章では、Logical Domains (LDoms) 1.3 ソフトウェアを有効にするために必要なさまざまなソフトウェアコンポーネントをインストールまたはアップグレードする方法について説明します。LDoms ソフトウェアを使用するには、次のコンポーネントが必要です。

- サポートされるプラットフォーム。サポートされるプラットフォームの一覧については、『Logical Domains 1.3 リリースノート』の「サポートされるプラットフォーム」を参照してください。
- 『Logical Domains 1.3 リリースノート』の「必須のソフトウェアとパッチ」で推奨されるすべてのパッチが適用された、Solaris 10 10/09 OS 以上のオペレーティングシステムが動作している制御ドメイン。33 ページの「Solaris OS のアップグレード」を参照してください。
- Sun UltraSPARC T2 または T2 Plus プラットフォーム用のシステムファームウェア Version 7.2.6 以上。27 ページの「システムファームウェアのアップグレード」を参照してください。
- 制御ドメインにインストールされて有効になっている LogicalDomains1.3 ソフトウェア。29 ページの「Logical Domains Manager のインストール」を参照してください。
- (省略可能) Logical Domains Management Information Base (MIB) ソフトウェアパッケージ。LDoms MIB の使用法の詳細は、『Logical Domains (LDoms) MIB 1.0.1 Administration Guide』を参照してください。

Logical Domains Manager をインストールまたはアップグレードする前に、Solaris OS およびシステムファームウェアが、使用しているサーバーでインストールまたはアップグレードされている必要があります。システムですでに Logical Domains ソフトウェアを使用している場合、33 ページの「Logical Domains をすでに使用しているシステムのアップグレード」を参照してください。そうでない場合は、26 ページの「新しいシステムへの Logical Domains ソフトウェアのインストール」を参照してください。

この章の内容は次のとおりです。

注 – Solaris Security Toolkit ソフトウェアは Logical Domains ソフトウェアと同梱されなくなりました。最新バージョンの Solaris Security Toolkit ソフトウェアを使用する場合、『Logical Domains 1.3 リリースノート』 を参照してください。

新しいシステムへの Logical Domains ソフトウェアのインストール

Logical Domains ソフトウェアをサポートする Sun プラットフォームは、Solaris 10 OS がプリインストールされた状態で出荷されます。初期状態では、プラットフォームは 1 つのオペレーティングシステムのみをホストする単一のシステムとして示されます。Solaris OS、システムファームウェア、および Logical Domains Manager をインストールすると、Solaris OS の元のシステムおよびインスタンスが制御ドメインになります。プラットフォームのこの最初のドメインには、`primary` という名前が付けられます。この名前を変更したり、このドメインを削除したりすることはできません。このドメインから、Solaris OS のさまざまなインスタンスをホストする複数のドメインを持つようにプラットフォームを再構成できます。

Solaris OS のアップグレード

新しいシステムでは、インストールポリシーに一致するように OS を再インストールする必要がある場合があります。この場合、『Logical Domains 1.3 リリースノート』の「必須および推奨される Solaris OS」 を参照して、このバージョンの Logical Domains ソフトウェアで使用する必要のある Solaris 10 OS を調べてください。Solaris OS をインストールする詳細な手順については、使用している Solaris 10 OS のインストールマニュアルを参照してください。インストール内容は、使用しているシステムの要件に合わせて調整できます。

システムがすでにインストールされている場合は、このバージョンの Logical Domains ソフトウェアを使用するために必要な適切な Solaris 10 OS にアップグレードする必要があります。このバージョンの Logical Domains ソフトウェアで使用する必要のある Solaris 10 OS、および必須パッチと推奨されるパッチを調べるには、『Logical Domains 1.3 リリースノート』の「必須のソフトウェアとパッチ」 を参照してください。Solaris OS をアップグレードするための手順全体は、「Solaris 10 10/09 Release and Installation Collection (`http://docs.sun.com/app/docs/coll/1236.11`)」を参照してください。

システムファームウェアのアップグレード

次のタスクでは、Advanced Lights Out Manager (ALOM) ソフトウェアを使用してシステムファームウェアを更新する方法を示します。

Integrated Lights Out Manager (ILOM) ソフトウェアを使用したシステムファームウェアの更新については、『Sun SPARC Enterprise T5120 and T5220 Servers Topic Set』の「Update the Firmware」を参照してください。

▼ システムファームウェアをアップグレードする

使用しているプラットフォームのシステムファームウェアは、SunSolve サイト (`http://sunsolve.sun.com`)から入手できます。

サポートされるサーバーで必要なシステムファームウェアについては、『Logical Domains 1.3 リリースノート』の「システムファームウェアの必須パッチ」を参照してください。

この手順では、サービスプロセッサで `flashupdate` コマンドを使用してシステムファームウェアをアップグレードする方法について説明します。

- ローカル FTP サーバーへアクセスできない場合は、28 ページの「FTP サーバーを使用せずに、システムファームウェアをアップグレードする」を参照してください。
- 制御ドメインからシステムファームウェアを更新する場合は、使用しているシステムファームウェアのリリースノートを参照してください。

サポートされるサーバーのシステムファームウェアのインストールおよび更新については、そのサーバーの管理マニュアルまたはプロダクトノートを参照してください。

1. サービスプロセッサに接続されたシリアルまたはネットワークのいずれかの管理ポートを使用して、ホストサーバーを停止して電源を切ります。

```
# shutdown -i5 -g0 -y
```

2. 使用しているサーバーに応じて、`flashupdate` コマンドを使用してシステムファームウェアをアップグレードします。

   ファームウェアをアップグレードする方法の詳細は、プラットフォームのマニュアルを参照してください。

   次に、`flashupdate` コマンドのサンプルを示します。

```
sc> flashupdate -s IP-address -f path/Sun_System_Firmware-
x_x_x_build_nn-server-name.bin
username: your-userid
password: your-password
```

各表記の意味は次のとおりです。

- *IP-address* は、使用している FTP サーバーの IP アドレスです。
- *path* は、システムファームウェアイメージを入手できる SunSolve[sm] 内の場所または独自のディレクトリです。
- *x_x_x* は、システムファームウェアのバージョン番号です。
- *nn* は、このリリースに適用されるビルド番号です。
- *server-name* は、使用しているサーバーの名前です。たとえば、SPARC® Enterprise T5440 サーバーの *server-name* は `SPARC_Enterprise_T5440` です。

3 サービスプロセッサをリセットします。

```
sc> resetsc -y
```

4 ホストサーバーの電源を入れて起動します。

```
sc> poweron -c
ok boot disk
```

▼ FTP サーバーを使用せずに、システムファームウェアをアップグレードする

サービスプロセッサにファームウェアをアップロードするためのローカル FTP サーバーにアクセスできない場合は、`sysfwdownload` ユーティリティーを使用できます。このユーティリティーは、システムファームウェアアップグレードパッケージとともに SunSolve サイトで提供されています。

`http://sunsolve.sun.com`

1 **Solaris OS** 内で次のコマンドを実行します。

```
# cd firmware_location
# sysfwdownload system_firmware_file
```

2 **Solaris OS** インスタンスを停止します。

```
# shutdown -i5 -g0 -y
```

3 システムの電源を切り、ファームウェアを更新します。

```
sc> poweroff -fy
sc> flashupdate -s 127.0.0.1
```

4 サービスプロセッサをリセットしてシステムの電源を入れます。

```
sc> resetsc -y
sc> poweron
```

Logical Domains Manager のダウンロード

▼ ソフトウェアをダウンロードする

1 **Sun** のソフトウェアダウンロードサイトから **zip** ファイル (`LDoms_Manager-1_3.zip`) をダウンロードします。

ソフトウェアは `http://www.sun.com/servers/coolthreads/ldoms/get.jsp` で入手できます。

2 **zip** ファイルを解凍します。

```
$ unzip LDoms_Manager-1_3.zip
```

ファイルの構造およびファイルの内容の詳細は、『Logical Domains 1.3 リリースノート』の「LDoms 1.3 ソフトウェアの場所」 を参照してください。

Logical Domains Manager のインストール

Logical Domains Manager ソフトウェアをインストールする方法は 3 つあります。

- インストールスクリプトを使用してパッケージおよびパッチをインストールします。この方法では Logical Domains Manager ソフトウェアは自動的にインストールされます。30 ページの「Logical Domains Manager ソフトウェアの自動的なインストール」 を参照してください。
- JumpStart を使用してパッケージをインストールします。31 ページの「JumpStart を使用した Logical Domains Manager 1.3 ソフトウェアのインストール」 を参照してください。
- パッケージを手動でインストールします。31 ページの「Logical Domains Manager ソフトウェアの手動によるインストール」 を参照してください。

注 – Logical Domains パッケージをインストールしたあとで、LDoms MIB ソフトウェアパッケージを手動でインストールする必要があります。これは、ほかのパッケージとともに自動的にはインストールされません。LDoms MIB のインストールおよび使用法の詳細は、『Logical Domains (LDoms) MIB 1.0.1 Administration Guide』を参照してください。

Logical Domains Manager ソフトウェアの自動的なインストール

install-ldm インストールスクリプトを使用する場合、スクリプトの実行方法を指定する選択肢がいくつかあります。それぞれの選択肢について、次の手順で説明します。

- オプションを指定せずに install-ldm スクリプトを使用すると、自動的に次の処理を行います。
  - Solaris OS リリースが Solaris 10 10/09 OS 以上であることを確認します。
  - パッケージのサブディレクトリである SUNWldm/ および SUNWldmp2v/ が存在することを確認します。
  - 前提条件となる Solaris Logical Domains ドライバパッケージの SUNWldomr および SUNWldomu が存在することを確認します。
  - SUNWldm および SUNWldmp2v パッケージがインストールされていないことを確認します。
  - Logical Domains Manager 1.3 ソフトウェアをインストールします。
  - すべてのパッケージがインストールされていることを確認します。
  - Solaris Security Toolkit (SUNWjass) がすでにインストールされている場合、制御ドメインの Solaris OS の強化を求めるプロンプトが表示されます。
  - Logical Domains Configuration Assistant (ldmconfig) を使用してインストールを実行するかどうかを判断します。
- -c オプションを指定して install-ldm スクリプトを使用すると、ソフトウェアのインストール後に自動的に **Logical Domains Configuration Assistant** を実行します。
- -s オプションを指定して install-ldm スクリプトを使用すると、**Logical Domains Configuration Assistant** の実行をスキップします。
- **Solaris Security Toolkit** ソフトウェアとともに install-ldm スクリプトおよび次のオプションを使用すると、次の操作を実行できます。
  - install-ldm -d。-secure.driver で終わるドライバ以外の Solaris Security Toolkit ドライバを指定できます。このオプションは、前述の選択肢で示したすべての機能を自動的に実行し、Solaris Security Toolkit のカスタマイズドライバ (たとえば server-secure-myname.driver) を指定して制御ドメインの Solaris OS を強化します。
  - install-ldm -d none。Solaris Security Toolkit を使用して制御ドメインで実行している Solaris OS を強化しないことを指定します。このオプションは、前述の選択肢で示した強化以外のすべての機能を自動的に実行します。Solaris Security Toolkit の使用を省略することはお勧めしません。別の処理を使用して制御ドメインを強化する場合にかぎり、この使用を省略するようにしてください。

- `install-ldm -p`。Logical Domains Manager デーモン (`ldmd`) の有効化および Solaris Security Toolkit の実行といったインストール後の処理のみを実行することを指定します。たとえば、`SUNWldm` および `SUNWjass` パッケージがサーバーにプリインストールされている場合に、このオプションを使用します。

JumpStart を使用した Logical Domains Manager 1.3 ソフトウェアのインストール

JumpStart の使用法の詳細は、『JumpStart Technology: Effective Use in the Solaris Operating Environment』を参照してください。

注意 – ネットワークインストール中は、仮想コンソールから接続を解除しないでください。

▼ JumpStart サーバーを設定する

この手順の詳細は、『Solaris 10 10/09 インストールガイド (カスタム JumpStart/ 上級編)』を参照してください。

1. 『Solaris 10 10/09 インストールガイド (カスタム JumpStart/ 上級編)』を参照してください。

   次の手順を実行します。

   a. 『Solaris 10 10/09 インストールガイド (カスタム JumpStart/ 上級編)』の「作業マップ: カスタム JumpStart インストールの準備」を参照してください。

   b. 「ネットワーク上のシステム用のプロファイルサーバーの作成」の手順に従って、ネットワークに接続されたシステムを設定します。

   c. 「`rules` ファイルの作成」の手順に従って、`rules` ファイルを作成します。

2. 「`rules` ファイルの妥当性を検査する」の手順に従って、`rules` ファイルの妥当性検査を行います。

Logical Domains Manager ソフトウェアの手動によるインストール

▼ Logical Domains Manager (LDoms) 1.3 ソフトウェアを手動でインストールする

始める前に　Sun のソフトウェアダウンロードサイトから、Logical Domains Manager 1.3 ソフトウェアの `SUNWldm` パッケージと `SUNWldmp2v` パッケージをダウンロードします。具体的な手順については、29 ページの「ソフトウェアをダウンロードする」を参照してください。

1 pkgadd コマンドを使用して、SUNWldm.v パッケージと SUNWldmp2v パッケージをインストールします。

pkgadd コマンドの詳細は、pkgadd(1M) マニュアルページを参照してください。

-G オプションはパッケージを大域ゾーンのみにインストールし、-d オプションは SUNWldm.v パッケージと SUNWldmp2v パッケージが含まれるディレクトリのパスを指定します。

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

2 対話型プロンプトのすべての質問に対して、y (はい) と答えます。

3 pkginfo コマンドを使用して、**Logical Domains Manager 1.3** パッケージの SUNWldm と SUNWldmp2v がインストールされていることを確認します。

pkginfo コマンドの詳細は、pkginfo(1) マニュアルページを参照してください。

バージョン (REV) 情報の例を次に示します。

```
# pkginfo -l SUNWldm | grep VERSION
VERSION=1.3,REV=2009.12.03.10.20
```

Logical Domains Manager デーモンの有効化

install-ldm インストールスクリプトを使用すると、Logical Domains Manager デーモン (ldmd) が自動的に有効になります。SUNWldm パッケージをインストールした場合も、ldmd デーモンは自動的に有効になります。このデーモンが有効になると、論理ドメインを作成、変更、および制御できます。

▼ Logical Domains Manager デーモンを有効にする

ldmd デーモンが無効になっている場合、次の手順に従ってこのデーモンを有効にします。

1 svcadm コマンドを使用して、**Logical Domains Manager** デーモンの ldmd を有効にします。

svcadm コマンドの詳細は、svcadm(1M) マニュアルページを参照してください。

```
# svcadm enable ldmd
```

2 `ldm list` コマンドを使用して、**Logical Domains Manager** デーモンが実行中であることを確認します。

`ldm list` コマンドを実行すると、システム上で現在定義されているすべてのドメインが一覧表示されます。特に、`primary` ドメインが表示され、状態が `active` になっているはずです。次のサンプル出力は、システム上に `primary` ドメインのみが定義されていることを示します。

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

Logical Domains をすでに使用しているシステムのアップグレード

この節では、Logical Domains ソフトウェアをすでに使用しているシステムで Solaris OS、ファームウェア、および Logical Domains Manager コンポーネントをアップグレードするプロセスについて説明します。

使用しているシステムですでに Logical Domains ソフトウェアが構成されている場合は、その制御ドメインをアップグレードする必要があります。LogicalDomains1.3 ソフトウェアのすべての機能を使用可能にする場合は、その他の既存のドメインもアップグレードする必要があります。

Solaris OS のアップグレード

このバージョンの Logical Domains ソフトウェアで使用する必要のある Solaris 10 OS、および各種ドメインに必須および推奨されるパッチを調べるには、『Logical Domains 1.3 リリースノート』の「必須のソフトウェアとパッチ」 を参照してください。Solaris OS をアップグレードする詳細な手順については、Solaris 10 のインストールマニュアルを参照してください。

制御ドメインで Solaris OS を再インストールする場合、この節に示すとおり、Logical Domains の自動保存構成データおよび制約データベースファイルを保存および復元する必要があります。

自動保存構成ディレクトリの保存および復元

Logical Domains 1.2 リリース以降では、制御ドメインでオペレーティングシステムを再インストールする前に、自動保存構成ディレクトリを保存および復元できます。制御ドメインでオペレーティングシステムを再インストールするたびに、Logical Domains の自動保存構成データを保存および復元する必要があります。このデータは、`/var/opt/SUNWldm/autosave-`*autosave-name* ディレクトリに格納されています。

tar または cpio コマンドを使用して、ディレクトリのすべての内容を保存および復元できます。

注 – 各自動保存ディレクトリには、関連する構成の前回の SP 構成更新のタイムスタンプが含まれています。自動保存ファイルを復元すると、タイムスタンプが同期しなくなることがあります。この場合、復元された自動保存構成は、以前の状態 ([newer] または最新) で表示されます。

自動保存構成の詳細は、170 ページの「Logical Domains 構成の管理」 を参照してください。

▼ 自動保存ディレクトリを保存および復元する

この手順は、自動保存ディレクトリを保存および復元する方法を示します。

1 自動保存ディレクトリを保存します。

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

2 (省略可能) クリーンな復元操作を行えるように、既存の自動保存ディレクトリを削除します。

自動保存ディレクトリには、以前の構成によって残されたファイルなどの不要なファイルが含まれていることがあります。このようなファイルは、SP にダウンロードされた構成を破壊することがあります。このような場合、この例に示すとおり、復元操作の前に自動保存ディレクトリを削除します。

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

3 自動保存ディレクトリを復元します。

これらのコマンドは、/var/opt/SUNWldm ディレクトリ内のファイルおよびディレクトリを復元します。

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

Logical Domains の制約データベースファイルの保存および復元

制御ドメインでオペレーティングシステムをアップグレードするたびに、/var/opt/SUNWldm/ldom-db.xml で参照できる Logical Domains の制約データベースファイルを保存および復元する必要があります。

注 – また、ディスクスワップなど、制御ドメインのファイルデータを破損するその他の操作を行うときは、/var/opt/SUNWldm/ldom-db.xml ファイルも保存および復元します。

Live Upgrade を使用する場合の Logical Domains の制約データベースファイルの保持

制御ドメインで Live Upgrade を使用する場合は、/etc/lu/synclist ファイルに次の行を追加することを検討してください。

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

これによって、データベースがアクティブなブート環境から新しいブート環境に自動的にコピーされます。/etc/lu/synclist と、ブート環境間でのファイルの同期については、『Solaris 10 10/09 インストールガイド (Solaris Live Upgrade とアップグレードの計画)』の「ブート環境間でのファイルの同期」 を参照してください。

Solaris 10 5/08 OS より前の Solaris 10 OS からのアップグレード

制御ドメインで Solaris 10 5/08 OS より前のバージョンの Solaris 10 OS (またはパッチ 127127-11 が適用されていない Solaris 10 OS) からのアップグレードを行う場合、およびボリュームマネージャーのボリュームが仮想ディスクとしてエクスポートされている場合は、Logical Domain Manager をアップグレードしたあとに、options=slice を指定して仮想ディスクバックエンドを再エクスポートする必要があります。詳細は、82 ページの「ボリュームのエクスポートおよび下位互換性」 を参照してください。

Logical Domains Manager およびシステムファームウェアのアップグレード

この節では、LogicalDomains1.3 ソフトウェアにアップグレードする方法について説明します。

まず、Logical Domains Manager を制御ドメインにダウンロードします。29 ページの「Logical Domains Manager のダウンロード」 を参照してください。

次に、プラットフォーム上で動作している制御ドメイン以外のすべてのドメインを停止します。

▼ プラットフォーム上で動作している制御ドメイン以外のすべてのドメインを停止する

1. 各ドメインで ok プロンプトに移行します。
2. 制御ドメインから各ドメインに対して stop-domain サブコマンドを実行します。

   ```
   primary# ldm stop-domain ldom
   ```

3. 制御ドメインから各ドメインに対して unbind-domain サブコマンドを実行します。

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

LogicalDomains1.3 ソフトウェアへのアップグレード

この節では、LogicalDomains1.3 ソフトウェアにアップグレードする方法について説明します。

既存の LDoms 1.0 の設定を LogicalDomains1.3 ソフトウェアで使用する場合は、『Logical Domains 1.3 リリースノート』の「LDoms 1.0 ソフトウェアからのみアップグレードする」に示す手順を実行してください。既存の LDoms 1.0 の設定は、LogicalDomains1.3 ソフトウェアでは機能しません。

より新しいバージョンの Logical Domains ソフトウェアからアップグレードする場合は、36 ページの「Logical Domains 1.3 ソフトウェアにアップグレードする」に示す手順を実行してください。このような既存の LDoms の設定は、LogicalDomains1.3 ソフトウェアでも機能します。

▼ Logical Domains 1.3 ソフトウェアにアップグレードする

1. システムのファームウェアをフラッシュ更新します。

   手順全体については、27 ページの「システムファームウェアをアップグレードする」または28 ページの「FTP サーバーを使用せずに、システムファームウェアをアップグレードする」を参照してください。

2. Logical Domains Manager デーモン (ldmd) を無効にします。

   ```
   # svcadm disable ldmd
   ```

3. 古い SUNWldm パッケージを削除します。

   ```
   # pkgrm SUNWldm
   ```

4 新しい SUNWldm パッケージを追加します。

-d オプションの指定は、パッケージが現在のディレクトリに存在することを前提としています。

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

5 ldm list コマンドを使用して、**Logical Domains Manager** デーモンが実行中であることを確認します。

ldm list コマンドを実行すると、システム上で現在定義されているすべてのドメインが一覧表示されます。特に、primary ドメインが表示され、状態が active になっているはずです。次のサンプル出力は、システム上に primary ドメインのみが定義されていることを示します。

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

出荷時デフォルト構成と Logical Domains の無効化

プラットフォームが 1 つのオペレーティングシステムのみをホストする単一のシステムとして表示される初期構成は、出荷時デフォルト構成と呼ばれます。論理ドメインを無効にする場合には、他のドメインに割り当てられている可能性のあるすべてのリソース (CPU、メモリー、I/O) にシステムが再びアクセスできるように、この構成の復元も必要になる場合があります。

この節では、すべてのゲストドメインを削除し、Logical Domains のすべての構成を削除し、構成を出荷時のデフォルトに戻す方法について説明します。

▼ すべてのゲスト論理ドメインを削除する

1 サービスプロセッサに格納されている論理ドメイン構成をすべて一覧表示します。

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

2 factory-default 構成を除き、以前にサービスプロセッサ (SP) に保存されたすべての構成 (*config-name*) を削除します。

各構成に対して次のコマンドを使用します。

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

以前に SP に保存されたすべての構成を削除すると、factory-default ドメインは、制御ドメイン (primary) が再起動されるときに使用される次のドメインになります。

3 --a オプションを使用して、すべてのドメインを停止します。

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

4 primary ドメインを除き、すべてのドメインのバインドを解除します。

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

注 – 分割 PCI 構成では、制御ドメインが必要とするサービスを I/O ドメインが提供している場合、その I/O ドメインのバインドを解除できないことがあります。この場合は、この手順をスキップします。

▼ 出荷時デフォルト構成を復元する

1 出荷時デフォルト構成を選択します。

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

2 制御ドメインを停止します。

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

3 factory-default 構成が読み込まれるように、システムの電源を切ってすぐに入れ直します。

```
sc> poweroff
sc> poweron
```

▼ Logical Domains Manager を無効にする

- 制御ドメインから **Logical Domains Manager** を無効にします。

```
primary# svcadm disable ldmd
```

注 – Logical Domains Manager を無効にしても動作中のドメインは停止しませんが、新しいドメインの作成、既存のドメインの構成の変更、またはドメインの状態の監視を行う機能は無効になります。

注意 – Logical Domains Manager を無効にすると、エラー報告、電源管理など、一部のサービスが無効になります。エラー報告については、factory-default 構成の場合は、単独のドメインを再起動してエラーの報告を復元することはできます。ただし、電源管理の場合にはこの方法は使用できません。また、一部のシステム管理または監視ツールは、Logical Domains Manager に依存しています。

▼ Logical Domains Manager を削除する

出荷時デフォルト構成を復元して Logical Domains Manager を無効にしたあとで、Logical Domains Manager ソフトウェアを削除できます。

- **Logical Domains Manager ソフトウェアを削除します。**

```
primary# pkgrm SUNWldm SUNWldmp2v
```

注 – 出荷時デフォルト構成を復元する前に Logical Domains Manager を削除する場合は、次の手順に示すように、サービスプロセッサから出荷時デフォルト構成を復元できます。

▼ サービスプロセッサから出荷時デフォルト構成を復元する

出荷時デフォルト構成を復元する前に Logical Domains Manager を削除する場合は、サービスプロセッサから出荷時デフォルト構成を復元できます。

1. サービスプロセッサから出荷時デフォルト構成を復元します。

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

2. システムの電源を切ってすぐに入れ直し、出荷時デフォルト構成を読み込みます。

◆ ◆ ◆ 第 3 章

3 セキュリティー

この章では、Logical Domains システムで有効にできるいくつかのセキュリティー機能について説明します。

この章の内容は次のとおりです。

LDoms Manager の承認

Logical Domains Manager の承認には、次の 2 つのレベルがあります。

- 読み取り – 構成を表示できますが、変更できません。
- 読み取りおよび書き込み – 構成を表示および変更できます。

変更は、Solaris OS に加えられるのではなく、Logical Domains Manager のインストール時にパッケージスクリプト `postinstall` を使用することで、承認ファイルに追加されます。同様に、承認エントリは、パッケージスクリプト `preremove` によって削除されます。

`ldm` サブコマンドと、そのコマンドの実行に必要な対応するユーザー承認を次の表に示します。

表 3–1　ldm サブコマンドおよびユーザー承認

ldm サブコマンド[1]	ユーザー承認
add-*	solaris.ldoms.write
bind-domain	solaris.ldoms.write
list	solaris.ldoms.read
list-*	solaris.ldoms.read
panic-domain	solaris.ldoms.write
remove-*	solaris.ldoms.write
set-*	solaris.ldoms.write
start-domain	solaris.ldoms.write
stop-domain	solaris.ldoms.write
unbind-domain	solaris.ldoms.write

[1] 追加、表示、削除、または設定できるすべてのリソースを指します。

ゲストコンソールアクセス用の RBAC の構成

vntsd デーモンでは、vntsd/authorization という SMF プロパティーを使用できます。このプロパティーを構成すると、ドメインコンソールまたはコンソールグループ用にユーザーおよび役割の承認チェックを有効にできます。承認チェックを有効にするには、svccfg コマンドを使用して、このプロパティーの値を true に設定します。このオプションが有効な場合、vntsd は、localhost のみで接続を待機して受け入れます。vntsd/authorization が有効な場合、listen_addr プロパティーに代替 IP アドレスを指定していても、vntsd は代替 IP アドレスを無視し、引き続き localhost のみで待機します。

デフォルトでは、vntsd サービスが有効な場合、すべてのゲストコンソールにアクセスするための承認は、auth_attr データベースに追加されます。

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

スーパーユーザーは、usermod コマンドを使用して、必要な承認をほかのユーザーまたは役割に割り当てることができます。これにより、特定のドメインコンソールまたはコンソールグループにアクセスするために必要な承認を持つユーザーまたは役割のみが許可されます。

次の例は、ユーザー terry に、すべてのドメインコンソールにアクセスするための承認を付与します。

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

次の例は、ldg1 という名前の特定のドメインコンソール用の新しい承認を追加し、この承認をユーザー sam に割り当てます。

1. 新しい承認エントリを、ドメイン ldg1 の auth_attr ファイルに追加します。

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

2. この承認をユーザー sam に割り当てます。

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

承認および RBAC の詳細は、『Solaris のシステム管理 (セキュリティサービス)』を参照してください。

ユーザーアカウントに対する承認およびプロファイルの作成と役割の割り当て

Logical Domains Manager 用に変更された Solaris OS の役割に基づくアクセス制御 (RBAC) を使用して、ユーザーアカウントに対する承認およびプロファイルを設定し、役割を割り当てます。RBAC の詳細は、「Solaris 10 System Administrator Collection (http://docs.sun.com/app/docs/coll/47.16)」を参照してください。

Logical Domains Manager の承認には、次の 2 つのレベルがあります。

- 読み取り - 構成を表示できますが、変更できません。
- 読み取りおよび書き込み - 構成を表示および変更できます。

Solaris OS の /etc/security/auth_attr ファイルには、次の Logical Domains エントリが自動的に追加されます。

- solaris.ldoms.:::LDoms Administration::
- Solaris.ldoms.grant:::Delegate Ldoms Configuration::
- Solaris.ldoms.read:::View Ldoms Configuration::
- Solaris.ldoms.write:::Manage Ldoms Configuration::

ユーザー承認の管理

▼ ユーザーの承認を追加する

必要に応じて次の手順を使用して、Logical Domains Manager ユーザーに対する承認を /etc/security/auth_attr ファイルに追加します。スーパーユーザーには solaris.* 承認がすでに設定されているため、スーパーユーザーは solaris.ldoms.* 承認の承認をすでに持っています。

1 ldm(1M) のサブコマンドを使用するために承認を必要とするユーザーごとに、ローカルユーザーアカウントを作成します。

注 – ユーザーの Logical Domains Manager 承認を追加するには、そのユーザーに対してローカル (非 LDAP) アカウントを作成する必要があります。詳細は、「Solaris 10 System Administrator Collection (http://docs.sun.com/app/docs/coll/47.16)」を参照してください。

---

2 ユーザーによるアクセスを可能にする **ldm(1M)** のサブコマンドに応じて、次のいずれかを実行します。

ldm(1M) コマンドとそれらのユーザー承認の一覧は、表 3–1 を参照してください。

- usermod(1M) コマンドを使用して、ユーザーの読み取り専用承認を追加します。

  # usermod -A solaris.ldoms.read *username*

- usermod(1M) コマンドを使用して、ユーザーの読み取りおよび書き込み承認を追加します。

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

▼ ユーザーのすべての承認を削除する

- ローカルユーザーアカウントのすべての承認を削除します (使用できる唯一のオプション)。

  # usermod -A ‘‘ *username*

ユーザープロファイルの管理

SUNWldm パッケージによって、/etc/security/prof_attr ファイルにシステムで定義された 2 つの RBAC プロファイルが追加されます。これらは、スーパーユーザー以外による Logical Domains Manager へのアクセスを承認するために使用されます。2 つの LDoms 固有のプロファイルは次のとおりです。

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

次の手順を使用して、前述のいずれかのプロファイルをユーザーアカウントに割り当てることができます。

▼ ユーザーのプロファイルを追加する

- ローカルユーザーアカウントに管理プロファイル (たとえば、**LDoms Management**) を追加します。

  # usermod -P “LDoms Management” *username*

▼ ユーザーのすべてのプロファイルを削除する

- ローカルユーザーアカウントのすべてのプロファイルを削除します (使用できる唯一のオプション)。

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

ユーザーへの役割の割り当て

この手順を使用する利点は、特定の役割が割り当てられたユーザーだけがその役割になることができることです。役割にパスワードが設定されている場合は、その役割になるときにパスワードが必要になります。これにより、2 層のセキュリティーが実現します。ユーザーに役割が割り当てられていない場合、ユーザーがその正しいパスワードを知っていたとしても、su *role-name* コマンドを実行してその役割になることはできません。

▼ 役割を作成し、ユーザーにその役割を割り当てる

1. 役割を作成します。

```
# roleadd -A solaris.ldoms.read ldm_read
```

2. 役割にパスワードを割り当てます。

```
# passwd ldm_read
```

3. ユーザー (たとえば user_1) に役割を割り当てます。

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

4. ユーザー (user_1) にパスワードを割り当てます。

```
# passwd user_1
```

5. ldm_read アカウントになるために、user_1 アカウントに対するアクセス権のみを割り当てます。

```
# su user_1
```

6. プロンプトが表示されたら、ユーザーのパスワードを入力します。

7. ユーザー ID を確認して、ldm_read 役割にアクセスします。

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

8 読み取り承認を持つ ldm サブコマンドに対して、ユーザーにアクセス権を提供します。

```
# su ldm_read
```

9 プロンプトが表示されたら、ユーザーのパスワードを入力します。

10 id コマンドを入力してユーザーを表示します。

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

ドメインの移行に必要な特権の追加

ドメインを別のシステムに移行するには、Logical Domains の承認 (solaris.ldoms.*) に加え、file_dac_read 特権および file_dac_search 特権を使用する必要があります。これらの特権を所有することで、ユーザーは Logical Domains Manager のキー /var/opt/SUNWldm/server.key を読み取ることができます。このキーはセキュリティー上の理由により、スーパーユーザーのみが読み取ることができます。

▼ ドメインの移行を可能にするためにほかの特権を追加する

1 スーパーユーザーになるか、同等の役割を取得します。

役割には、承認および特権付きコマンドが含まれます。役割の詳細は、『Solaris のシステム管理 (セキュリティサービス)』の「RBAC の構成 (作業マップ)」 を参照してください。

2 usermod コマンドを使用して、file_dac_read 特権と file_dac_search 特権をユーザーに追加します。

```
# usermod -K defaultpriv=basic,file_dac_read,file_dac_search username
```

usermod コマンドの詳細は、usermod(1M) マニュアルページを参照してください。

次のコマンドは、file_dac_read 特権と file_dac_search 特権を ldm_mig ユーザーに追加します。

```
# usermod -K defaultpriv=basic,file_dac_read,file_dac_search ldm_mig
```

▼ ローカルユーザーアカウントのすべての特権を削除する

1 スーパーユーザーになるか、同等の役割を取得します。

役割には、承認および特権付きコマンドが含まれます。役割の詳細は、『Solaris のシステム管理 (セキュリティサービス)』の「RBAC の構成 (作業マップ)」 を参照してください。

2 usermod コマンドを使用して、ユーザーのすべての特権を削除します。

```
# usermod -K defaultpriv=basic username
```

usermod コマンドの詳細は、usermod(1M) マニュアルページを参照してください。

次のコマンドは、ldm_mig ユーザーの特権を削除します。

```
# usermod -K defaultpriv=basic ldm_mig
```

BSM 監査の有効化と使用

Logical Domains Manager では、Solaris OS の基本セキュリティーモジュール (BSM) 監査機能を使用します。BSM 監査は、制御ドメインの処理およびイベントの履歴を調べて、何が発生したかを調べるための手段を提供します。履歴は、何が、いつ、誰によって行われ、どのような影響があるかを示すログに保持されます。

この監査機能を有効化および無効化するには、Solaris OS の bsmconv(1M) コマンドおよび bsmunconv(1M) コマンドを使用します。この節では、監査機能の確認、監査の出力の表示、および監査ログのローテーションを行う方法を示すタスクについても説明します。BSM 監査の詳細は、Solaris 10 の『Solaris のシステム管理 (セキュリティサービス)』で参照できます。

▼ BSM 監査を有効にする

1 /etc/security/audit_control ファイルの flags: 行に vs を追加します。

2 bsmconv(1M) コマンドを実行します。

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

このコマンドの詳細は、bsmconv(1M) マニュアルページを参照してください。

3 Solaris OS を再起動して、監査を有効にします。

▼ BSM 監査が有効であることを確認する

1 次のコマンドを入力します。

```
# auditconfig -getcond
```

2 出力に audit condition = auditing が表示されていることを確認します。

▼ BSM 監査を無効にする

1 bsmunconv コマンドを実行して、**BSM** 監査を無効にします。

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

このコマンドの詳細は、bsmunconv(1M) マニュアルページを参照してください。

2 **Solaris OS** を再起動して、監査を無効にします。

▼ 監査の出力を表示する

- **BSM** 監査の出力を表示するには、次のいずれかの方法を使用します。
  - auditreduce(1M) コマンドと praudit(1M) コマンドを使用して、監査の出力を表示します。

    ```
    # auditreduce -c vs | praudit
    # auditreduce -c vs -a 20060502000000 | praudit
    ```
  - praudit -x コマンドを使用して、XML 出力を表示します。

▼ 監査ログをローテーションする

- audit -n コマンドを使用して、監査ログをローテーションします。

◆ ◆ ◆ 第 4 章

サービスと論理ドメインの設定

この章では、デフォルトのサービス、制御ドメイン、およびゲストドメインの設定に必要な手順について説明します。

Logical Domains Configuration Assistant を使用して論理ドメインおよびサービスを構成することもできます。付録 D 「Logical Domains Configuration Assistant」 を参照してください。

この章の内容は次のとおりです。

出力メッセージ

primary ドメインのいずれかのデバイスまたはいずれかのサービスに対して動的に実行できない最初の操作のあと、次のメッセージを受け取ります。

```
Initiating delayed reconfigure operation on LDom primary. All
configuration changes for other LDoms are disabled until the
LDom reboots, at which time the new configuration for LDom
primary will also take effect.
```

primary ドメインを再起動するまで、その後の各操作のあとに次の通知を受け取ります。

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

デフォルトのサービスの作成

あとで使用できるように、次のデフォルトの仮想サービスを最初に作成する必要があります。

- vdiskserver - 仮想ディスクサーバー
- vswitch - 仮想スイッチサービス
- vconscon - 仮想コンソール端末集配信装置サービス

▼ デフォルトのサービスを作成する

1 論理ドメインに仮想ディスクをインポートできるように、仮想ディスクサーバー (vds) を作成します。

たとえば、次のコマンドを使用して、仮想ディスクサーバー (primary-vds0) を制御ドメイン (primary) に追加します。

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

2 仮想ネットワーク端末サーバーデーモン (vntsd) が使用する仮想コンソール端末集配信装置 (vcc) サービスを、すべての論理ドメインコンソールの端末集配信装置として作成します。

たとえば、次のコマンドを使用して、ポートの範囲が 5000 ～ 5100 の仮想コンソール端末集配信装置サービス (primary-vcc0) を、制御ドメイン (primary) に追加します。

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

3 論理ドメインの仮想ネットワーク (vnet) デバイス間でネットワークを有効にするには、仮想スイッチサービス (vsw) を作成します。

各論理ドメインが仮想スイッチを使用して外部と通信する必要がある場合は、GLDv3 準拠のネットワークアダプタを仮想スイッチに割り当てます。

たとえば、次のコマンドを使用して、ネットワークアダプタドライバ nxge0 の仮想スイッチサービス (primary-vsw0) を、制御ドメイン (primary) に追加します。

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

このコマンドによって、仮想スイッチに MAC アドレスが自動的に割り当てられます。ldm add-vsw コマンドに、オプションとして独自の MAC アドレスを指定できます。ただし、この場合、指定した MAC アドレスが既存の MAC アドレスと競合していないことの確認は、ユーザーが責任を持って行います。

追加された仮想スイッチが、基本となる物理アダプタに代わり主ネットワークインタフェースとなる場合は、動的ホスト構成プロトコル (DHCP) サーバーによってドメインに同じ IP アドレスが割り当てられるように、仮想スイッチに物理アダプタの MAC アドレスを割り当てる必要があります。53 ページの「制御ドメインまたはサービスドメインとその他のドメイン間のネットワークの有効化」 を参照してください。

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

4 list-services サブコマンドを使用して、サービスが作成されたことを確認します。

次のように出力されるはずです。

```
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
```

制御ドメインの初期構成

最初に、すべてのシステムリソースが制御ドメインに割り当てられます。その他の論理ドメインを作成できるように、一部のリソースを解放する必要があります。

▼ 制御ドメインを設定する

注 – この手順には、制御ドメイン用に設定するリソースの例も含まれています。ここで示す数値は単なる例であり、使用される値が制御ドメインに適していない場合があります。

---

1 制御ドメインに暗号化デバイスが割り当てられているかどうかを判断します。

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

2 暗号化リソースを制御ドメインに割り当てます。

次の例では、1 つの暗号化リソースが制御ドメイン primary に割り当てられます。これによって、残りの暗号化リソースをゲストドメインで使用できるようになります。

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

3 仮想 CPU を制御ドメインに割り当てます。

たとえば、次のコマンドでは、4 つの仮想 CPU が制御ドメイン primary に割り当てられます。これにより、残りの仮想 CPU をゲストドメインで使用できるようになります。

```
primary# ldm set-vcpu 4 primary
```

4 メモリーを制御ドメインに割り当てます。

たとえば、次のコマンドでは、4G バイトのメモリーが制御ドメイン primary に割り当てられます。これにより、残りのメモリーをゲストドメインで使用できるようになります。

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

5 論理ドメインのマシン構成をサービスプロセッサ (SP) に追加します。

たとえば、次のコマンドを使用して initial という名前の構成を追加します。

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

6 次回の再起動時に構成が使用できる状態であることを確認します。

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

この list サブコマンドでは、電源を再投入すると initial 構成設定が使用されることが示されています。

論理ドメインを使用するための再起動

構成の変更を有効にして、ほかの論理ドメインで使用できるようにリソースを解放するには、制御ドメインを再起動する必要があります。

▼ 再起動する

- 制御ドメインを停止して再起動します。

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

注 – 再起動または電源の再投入のいずれかによって、新しい構成がインスタンス化されます。サービスプロセッサ (SP) に保存されている構成が実際に起動されるのは、電源再投入後のみで、その際に `list-config` の出力に反映されます。

制御ドメインまたはサービスドメインとその他のドメイン間のネットワークの有効化

デフォルトでは、システムの制御ドメインとその他のドメイン間のネットワークは無効になっています。これを有効にするために、仮想スイッチデバイスをネットワークデバイスとして構成するようにしてください。仮想スイッチは、基本となる物理デバイス (この例では `nxge0`) に代わり主インタフェースとして構成するか、ドメインの追加のネットワークインタフェースとして構成することができます。

注 – この手順によってドメインへのネットワーク接続が一時的に中断される可能性があるため、次の手順は制御ドメインのコンソールから実行してください。

▼ 仮想スイッチを主インタフェースとして構成する

1. すべてのインタフェースのアドレス指定情報を表示します。

```
primary# ifconfig -a
```

2. 仮想スイッチを **plumb** します。この例では、構成する仮想スイッチは `vsw0` です。

```
primary# ifconfig vsw0 plumb
```

3. (省略可能) ドメイン内のすべての仮想スイッチインスタンスのリストを取得するために、仮想スイッチインスタンスを一覧で表示できます。

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

4. 仮想スイッチ (`net-dev`) に割り当てられた物理ネットワークデバイスを **unplumb** します。この例では、物理ネットワークデバイスは `nxge0` です。

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

5. 物理ネットワークデバイス (`nxge0`) のプロパティーを仮想スイッチ (`vsw0`) デバイスに移行するには、次のいずれかを実行します。

   - ネットワークが静的 IP アドレスを使用して構成されている場合は、`vsw0` に対して `nxge0` の IP アドレスとネットマスクを再利用します。

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

- ネットワークが DHCP を使用して構成されている場合は、vsw0 に対して DHCP を有効にします。

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

6 必要な構成ファイルに修正を加えて、この変更内容を確定します。

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

注 – 必要に応じて、物理ネットワークデバイスと同様に仮想スイッチも構成できます。この場合、手順 2 で記載されているように仮想スイッチを plumb して、物理デバイスは、unplumb しません (手順 4 をスキップする)。そのあと、仮想スイッチは、静的 IP アドレスまたは動的 IP アドレスを使用して構成する必要があります。動的 IP アドレスは DHCP サーバーから取得できます。この場合の詳細および例は、114 ページの「NAT およびルーティング用の仮想スイッチおよびサービスドメインの構成」 を参照してください。

仮想ネットワーク端末サーバーデーモンの有効化

各論理ドメインの仮想コンソールにアクセスするには、仮想ネットワーク端末サーバーデーモン (vntsd) を有効にする必要があります。このデーモンの使用法の詳細は、vntsd(1M) マニュアルページを参照してください。

▼ 仮想ネットワーク端末サーバーデーモンを有効にする

注 – vntsd を有効にする前に、制御ドメインにデフォルトのサービス vconscon (vcc) が作成されていることを確認してください。詳細は、50 ページの「デフォルトのサービスの作成」 を参照してください。

---

1 svcadm(1M) コマンドを使用して、仮想ネットワーク端末サーバーデーモン vntsd(1M) を有効にします。

```
primary# svcadm enable vntsd
```

2 svcs(1) コマンドを使用して、vntsd デーモンが有効であることを確認します。

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

ゲストドメインの作成と起動

ゲストドメインでは、sun4v プラットフォームとハイパーバイザによって提供される仮想デバイスの両方を認識するオペレーティングシステムを実行する必要があります。現時点では、Solaris 10 11/06 OS 以上を実行する必要があります。Solaris 10 10/09 OS を実行すると、Logical Domains 1.3 のすべての機能を使用できます。必要になる可能性があるパッチについては、『Logical Domains 1.3 リリースノート』を参照してください。デフォルトのサービスを作成し、制御ドメインからリソースを再度割り当てたら、ゲストドメインを作成して起動できます。

▼ ゲストドメインを作成および起動する

1 論理ドメインを作成します。

たとえば、次のコマンドを使用して ldg1 という名前のゲストドメインを作成します。

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

2 CPU をゲストドメインに追加します。

たとえば、次のコマンドを使用して 4 つの仮想 CPU をゲストドメイン ldg1 に追加します。

```
primary# ldm add-vcpu 4 ldg1
```

3 メモリーをゲストドメインに追加します。

たとえば、次のコマンドを使用して 2G バイトのメモリーをゲストドメイン ldg1 に追加します。

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

4 仮想ネットワークデバイスをゲストドメインに追加します。

たとえば、次のコマンドを使用して、次のように指定した仮想ネットワークデバイスをゲストドメイン ldg1 に追加します。

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

各表記の意味は次のとおりです。

- vnet1 は、後続の set-vnet または remove-vnet サブコマンドで参照するためにこの仮想ネットワークデバイスのインスタンスに割り当てられる、論理ドメインで一意のインタフェース名です。
- primary-vsw0 は、接続する既存のネットワークサービス (仮想スイッチ) の名前です。

注 – 手順 5 および 6 は、仮想ディスクサーバーデバイス (vdsdev) を primary ドメインに、および仮想ディスク (vdisk) をゲストドメインに追加するための簡略化された方法です。ZFS™ ボリュームおよびファイルシステムを仮想ディスクとして使用する方法については、81 ページの「ZFS ボリュームを 1 つのスライスディスクとしてエクスポートする」 および 91 ページの「仮想ディスクと ZFS の使用」 を参照してください。

5 仮想ディスクサーバーによってゲストドメインに仮想ディスクとしてエクスポートされるデバイスを指定します。

物理ディスク、ディスクスライス、ボリューム、またはファイルをブロック型デバイスとしてエクスポートできます。物理ディスクとファイルの例を次に示します。

- 物理ディスクの例。 最初の例では、次の指定で物理ディスクを追加します。

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

各表記の意味は次のとおりです。

- `/dev/dsk/c2t1d0s2` は、実際の物理デバイスのパス名です。デバイスを追加する場合、パス名にはデバイス名を組み合わせる必要があります。
- `vol1` は、仮想ディスクサーバーに追加するデバイスに指定する必要がある一意の名前です。ボリューム名は、この仮想ディスクサーバーによってクライアントにエクスポートされ追加されるため、ボリューム名はこの仮想ディスクサーバーのインスタンスに対して一意である必要があります。デバイスを追加する場合、ボリューム名には実際のデバイスのパス名を組み合わせる必要があります。
- `primary-vds0` は、このデバイスを追加する仮想ディスクサーバーの名前です。

- ファイルの例。 この 2 つめの例では、ファイルをブロック型デバイスとしてエクスポートします。

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

各表記の意味は次のとおりです。

- *backend* は、ブロック型デバイスとしてエクスポートされる実際のファイルのパス名です。デバイスを追加する場合、このバックエンドにデバイス名を組み合わせる必要があります。
- `vol1` は、仮想ディスクサーバーに追加するデバイスに指定する必要がある一意の名前です。ボリューム名は、この仮想ディスクサーバーによってクライアントにエクスポートされ追加されるため、ボリューム名はこの仮想ディスクサーバーのインスタンスに対して一意である必要があります。デバイスを追加する場合、ボリューム名には実際のデバイスのパス名を組み合わせる必要があります。
- `primary-vds0` は、このデバイスを追加する仮想ディスクサーバーの名前です。

6 仮想ディスクをゲストドメインに追加します。

次の例では、仮想ディスクをゲストドメイン ldg1 に追加します。

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

各表記の意味は次のとおりです。

- vdisk1 は、仮想ディスクの名前です。
- vol1 は、接続する既存のボリュームの名前です。
- primary-vds0 は、接続する既存の仮想ディスクサーバーの名前です。

注 – 仮想ディスクは、さまざまな種類の物理デバイス、ボリューム、またはファイルに関連付けられた総称的なブロック型デバイスです。仮想ディスクは SCSI ディスクと同義ではありません。そのため、ディスクラベル内のターゲット ID は除外されます。論理ドメインの仮想ディスクの形式は、c*N*d*N*s*N* です。c*N* は仮想コントローラ、d*N* は仮想ディスク番号、および s*N* はスライスを示します。

---

7 ゲストドメインの auto-boot および boot-device 変数を設定します。

最初の例のコマンドは、ゲストドメイン ldg1 の auto-boot\? を true に設定します。

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

2 つめの例のコマンドは、ゲストドメイン ldg1 の boot-device を vdisk に設定します。

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

8 ゲストドメイン ldg1 にリソースをバインドし、ドメインを一覧表示してリソースがバインドされていることを確認します。

```
primary# ldm bind-domain ldg1
primary# ldm list-domain ldg1
NAME          STATE    FLAGS  CONS   VCPU MEMORY   UTIL  UPTIME
ldg1          bound    -----  5000   4    2G
```

9 ゲストドメインのコンソールのポートを見つけるために、前述の list-domain サブコマンドの出力を調べます。

CONS という見出しの下で、論理ドメインゲスト 1 (ldg1) のコンソール出力がポート 5000 にバインドされていることがわかります。

10 制御ドメインにログインし、ローカルホストのコンソールポートに直接接続することによって、別の端末からゲストドメインのコンソールに接続します。

```
$ ssh admin@controldom.domain
$ telnet localhost 5000
```

11 ゲストドメイン ldg1 を起動します。

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

ゲストドメインへの Solaris OS のインストール

この節では、ゲストドメインに Solaris OS をインストールできる、いくつかの異なる方法について説明します。

▼ DVD からゲストドメインに Solaris OS をインストールする

1. **Solaris 10 OS DVD を DVD ドライブに挿入します。**

2. primary ドメインでボリューム管理デーモン vold(1M) を停止します。

```
primary# svcadm disable volfs
```

3. ゲストドメイン (ldg1) を停止し、バインドを解除します。次に、DVDROM メディアがマウントされた DVD を、たとえば二次ボリューム (dvd_vol@primary-vds0) および仮想ディスク (vdisk_cd_media) として追加します。

   c0t0d0s2 は、Solaris OS メディアのマウント先です。

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

4. DVD が二次ボリュームおよび仮想ディスクとして追加されていることを確認します。

```
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
....
```

5 ゲストドメイン (ldg1) をバインドし、起動します。

```
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 ..
```

6 クライアント **OpenBoot™ PROM** でデバイス別名を表示します。

この例で、vdisk_cd_media (Solaris DVD) および vdisk1 (Solaris OS をインストール可能な仮想ディスク) のデバイス別名を確認してください。

```
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
```

7 ゲストドメインのコンソールで、スライス f の vdisk_cd_media (disk@1) から起動します。

```
ok boot vdisk_cd_media:f -v
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 1983-2009 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
```

8 引き続き **Solaris OS** のインストールメニューに従います。

▼ Solaris ISO ファイルからゲストドメインに Solaris OS をインストールする

1 ゲストドメインのバインドを解除します。

次の例では、ldg1 をゲストドメインとして使用しています。

```
primary# ldm unbind ldg1
```

2 二次ボリュームおよび仮想ディスクとして **Solaris ISO** ファイルを追加します。

次の例では、`solarisdvd.iso` を Solaris ISO ファイル、`iso_vol@primary-vds0` を二次ボリューム、`vdisk_iso` を仮想ディスクとして使用します。

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

3 **Solaris ISO** ファイルが二次ボリュームおよび仮想ディスクとして追加されていることを確認します。

```
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 ゲストドメイン (`ldg1`) をバインドし、起動します。

```
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 クライアント **OpenBoot PROM** でデバイス別名を表示します。

この例で、`vdisk_iso` (Solaris ISO イメージ) および `vdisk_install` (ディスク領域) のデバイス別名を確認してください。

```
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 ゲストドメインのコンソールで、スライス f の vdisk_iso (disk@1) から起動します。

```
ok boot vdisk_iso:f -v
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 1983-2009 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
```

7 引き続き **Solaris OS** のインストールメニューに従います。

▼ ゲストドメインの JumpStart を実行する

- ゲストドメインの **JumpStart** を行うには、次の **2** つの例で示すように、正規の **Solaris OS** の **JumpStart** 手順にあるプロファイルの構文を **Logical Domains** 固有の **JumpStart** 手順に変更して、通常の **JumpStart** 手順を使用します。

  通常の **JumpStart** のプロファイル

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

  論理ドメインの仮想ディスクデバイス名は、デバイス名にターゲット ID (t*N*) が含まれないという点で、物理ディスクデバイス名とは異なります。通常の c*N*t*N*d*N*s*N* 形式の代わりに、仮想ディスクデバイス名は c*N*d*N*sN という形式になります。ここで、cN は仮想コントローラ、dN は仮想ディスク番号、および *s*N はスライスを示します。次のプロファイルの例のように、使用する JumpStart プロファイルを修正して、この変更を反映してください。

  論理ドメインで使用される実際のプロファイル

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

注 – 仮想ネットワーク (vnet) デバイスの MAC アドレスは、ゲストのバナーで報告されたものではなく、JumpStart 構成に対する ldm(1M) コマンドによって報告されたとおりに使用する必要があります。

◆ ◆ ◆ 第 5 章

I/O ドメインの設定

この章では、Logical Domains 環境で追加の I/O ドメインを設定する方法について説明します。

この章の内容は次のとおりです。

I/O ドメインと PCI EXPRESS バス

I/O ドメインとは、物理 I/O デバイスに対する直接の所有権と直接のアクセス権を持つドメインです。PCI EXPRESS (PCI-E) バスをドメインに割り当てることで、I/O ドメインを作成できます。サーバー上に存在する PCI-E バスは、`pci@400` (`pci_0`) などの名前で識別されます。それぞれのバスを別々のドメインに割り当てるには、`ldm` コマンドを使用します。

作成できる I/O ドメインの最大数は、サーバー上で使用可能な PCI-E バスの数に応じて異なります。Sun UltraSPARC T2 Plus ベースのサーバーは、最大 4 つの PCI-E バスを搭載しているため、最大 4 つの I/O ドメインを構成できます。

注 – Sun SPARC Enterprise T5120 サーバーや T5220 サーバーなどの Sun UltraSPARC T2 ベースのサーバーには、PCI-E バスは 1 つしかありません。そのため、このようなサーバーでは、物理デバイスに対する直接のアクセス権を持つドメインを複数構成することはできません。ただし、新しい I/O ドメインを構成する代わりに、ネットワークインタフェースユニット (NIU) をドメインに割り当てることができます。127 ページの「NIU ハイブリッド I/O の使用」 を参照してください。

サーバーで最初から Logical Domains が構成されている場合や、`factory-default` 構成を使用している場合、制御ドメインはすべての物理デバイスリソースにアクセスで

きます。つまり、制御 (primary) ドメインがシステム上に構成された唯一の I/O ドメインであり、すべての PCI-E バスを所有します。

▼ 新しい I/O ドメインを作成する

ここでは、例として、複数のバスが primary ドメインによって所有されている初期構成で新しい I/O ドメインを作成する手順について説明します。デフォルトでは、システム上に存在するすべてのバスを primary ドメインが所有しています。ここで示す例は、Sun SPARC Enterprise T5440 サーバーの場合です。この手順は、ほかのサーバーにも使用できます。別のサーバーではこれらの手順と若干異なる場合がありますが、この例では基本的な方針について理解できます。

最初に、primary ドメインの起動ディスクを持つバスを保持する必要があります。それから、その他のバスを primary ドメインから削除してほかのドメインに割り当てます。

注意 – サポートされたサーバーの内部ディスクはすべて、1 つの PCI バスに接続されています。ドメインが内部ディスクから起動する場合は、ドメインからそのバスを削除しないでください。また、ドメインで使用されているネットワークポートなどのデバイスが接続されたバスを削除していないことを確認してください。誤ったバスを削除すると、ドメインは必要なデバイスにアクセスできず、使用できなくなることがあります。ドメインで使用されているデバイスが接続されたバスを削除する場合は、ほかのバスのデバイスを使用するよう、そのドメインを再構成してください。たとえば、別のオンボードネットワークポートや、別の PCI スロットの PCI カードを使用するよう、ドメインを再構成する必要があります。

この例では、primary ドメインは 1 つの ZFS プール (rpool (c0t1d0s0)) と 1 つのネットワークインタフェース (nxge0) のみを使用します。primary ドメインで複数のデバイスを使用する場合は、デバイスごとに手順 2 - 4 を繰り返して、削除するバスにそれらのデバイスがないことを確認します。

1 primary ドメインが複数の **PCI** バスを所有していることを確認します。

```
primary# ldm list-bindings primary
...
IO
    DEVICE          PSEUDONYM       OPTIONS
    pci@400         pci_0
    pci@500         pci_1
    pci@600         pci_2
    pci@700         pci_3
...
```

2 起動ディスクのデバイスパスを確認します。これは保持する必要があります。

- **UFS** ファイルシステムの場合、`df /` コマンドを実行して、起動ディスクのデバイスパスを確認します。

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

- **ZFS** ファイルシステムの場合、まず `df /` コマンドを実行してプール名を確認してから、`zpool status` コマンドを実行して起動ディスクのデバイスパスを確認します。

```
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 ブロック型デバイスが接続されている物理デバイスを確認します。

ここでは、例としてブロック型デバイス `c1t0d0s0` を使用します。

```
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
```

この例では、`primary` ドメインの起動ディスクに対する物理デバイスは、前述の `pci_0` の一覧表示で対応しているバス `pci@400` に接続されています。つまり、`pci_0` (`pci@400`) を別のドメインに割り当てることはできません。

4 システムで使用されているネットワークインタフェースを確認します。

```
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
```

状態が `unknown` のインタフェースは構成されていないため、使用されていません。この例では、`nxge0` インタフェースが使用されます。

5 ネットワークインタフェースが接続されている物理デバイスを確認します。

次のコマンドでは、nxge0 ネットワークインタフェースを使用します。

```
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
```

この例では、primary ドメインで使用されているネットワークインタフェースに対する物理デバイスは、前述の pci@500 の一覧表示で対応しているバス pci@1 の配下にあります。そのため、ほかの 2 つのバス pci_2 (pci@600) と pci_3 (pci@700) は primary ドメインでは使用されていないため、ほかのドメインに安全に割り当てることができます。

primary ドメインで使用されているネットワークインタフェースが、別のドメインに割り当てようとしているバス上にある場合は、別のネットワークインタフェースを使用するように primary ドメインを再構成する必要があります。

6 起動ディスクを含まないバスをドメインから削除します。

この例では、バス pci@600 とバス pci@700 が primary ドメインから削除されます。

```
primary# ldm remove-io pci@600 primary
primary# ldm remove-io pci@700 primary
```

7 この構成をサービスプロセッサに保存します。

この例では、構成は io-domain です。

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

この構成 io-domain は、再起動後に使用される次の構成としても設定されます。

注 – 現在、SP に保存できる構成数の上限は 8 つです。この数には、factory-default 構成は含まれません。

---

8 primary ドメインを再起動して、変更を有効にします。

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

9 PCI バスを追加するドメインを停止します。

ここでは、例として ldg1 ドメインを停止します。

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

10 直接のアクセス権が必要なドメインに、使用可能なバスを追加します。

使用可能なバスは pci@600、ドメインは ldg1 です。

```
primary# ldm add-io pci@600 ldg1
```

InfiniBand カードが構成されていると、pci@600 バスでバイパスモードの有効化が必要になる場合があります。バイパスモードを有効にする必要があるかどうかについては、67 ページの「PCI バスでの I/O MMU バイパスモードの有効化」 を参照してください。

11 ドメインを再起動して、変更を有効にします。

次のコマンドでは、ldg1 ドメインを再起動します。

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

12 適切なバスが primary ドメインに割り当てられたままで、適切なバスがドメイン ldg1 に割り当てられていることを確認します。

```
primary# ldm list-bindings primary ldg1
NAME          STATE   FLAGS  CONS   VCPU  MEMORY  UTIL  UPTIME
primary       active  -n-cv  SP     4     4G      0.4%  18h 25m
...
IO
    DEVICE           PSEUDONYM        OPTIONS
    pci@400          pci_0
    pci@500          pci_1
...
------------------------------------------------------------------
NAME          STATE   FLAGS  CONS   VCPU  MEMORY  UTIL  UPTIME
ldg1          active  -n---  5000   4     2G      10%   35m
...
IO
    DEVICE           PSEUDONYM        OPTIONS
    pci@600          pci_2
...
```

この出力では、PCI-E バス pci_0 および pci_1 とそれらの配下のデバイスがドメイン primary に割り当てられており、pci_2 とそのデバイスが ldg1 に割り当てられていることを確認できます。

PCI バスでの I/O MMU バイパスモードの有効化

Infiniband ホストチャネルアダプタ (HCA) カードが構成されていると、I/O メモリー管理ユニット (MMU) のバイパスモードをオンにする必要がある場合があります。デフォルトでは、Logical Domains ソフトウェアが PCI-E トランザクションを制御して、特定の I/O デバイスまたは PCI-E オプションが I/O ドメイン内で割り当てられた物理メモリーにのみアクセス可能にします。別のゲストドメインのメモリーにアクセスしようとしても、I/O MMU によって阻止されます。これによって、I/O ドメインとその他すべてのドメインの間でより高いレベルのセキュリティーが得られます。ただし、I/O MMU バイパスモードがオフの状態で PCI-E または PCI-X オプ

ションカードが読み込まないまたは動作しないまれな状況では、このオプションを使用して I/O MMU バイパスモードをオンに設定できます。ただし、バイパスモードをオンに設定すると、I/O ドメインからのメモリーアクセスのハードウェアによる保護が実行されなくなります。

bypass=on オプションは、I/O MMU バイパスモードをオンに設定します。このバイパスモードは、それぞれの I/O ドメインおよびその I/O ドメイン内の I/O デバイスがすべてのゲストドメインに信頼されている場合にのみ有効にする必要があります。この例では、バイパスモードをオンにします。

```
primary# ldm add-io bypass=on pci@400 ldg1
```

出力では、OPTIONS ヘッダーの下に bypass=on が表示されます。

◆ ◆ ◆ 第 6 章

仮想ディスクの使用

この章では、Logical Domains ソフトウェアで仮想ディスクを使用する方法について説明します。

この章の内容は次のとおりです。

仮想ディスクの概要

仮想ディスクには、2 つの構成要素があります。ゲストドメインに表示される仮想ディスク自体と、データの格納先であり仮想 I/O の終端である仮想ディスクバックエンドです。仮想ディスクバックエンドは、仮想ディスクサーバー (`vds`) ドライバによって、サービスドメインからエクスポートされます。`vds` ドライバは、論理ドメインチャネル (LDC) を使用して、ハイパーバイザを介してゲストドメインの仮想ディスククライアント (`vdc`) ドライバと通信します。最終的には、仮想ディスクはゲストドメイン内の `/dev/[r]dsk/cXdYsZ` デバイスとして表示されます。

仮想ディスクバックエンドは、物理的でも論理的でもかまいません。物理デバイスには、次のものを含めることができます。

- 物理ディスクまたはディスク論理ユニット番号 (LUN)
- 物理ディスクスライス

論理デバイスは、次のいずれかにすることができます。

- ZFS、UFS などのファイルシステムのファイル
- ZFS、VxVM、Solaris™ Volume Manager (SVM) などのボリュームマネージャーからの論理ボリューム
- サービスドメインからアクセス可能な任意のディスク疑似デバイス

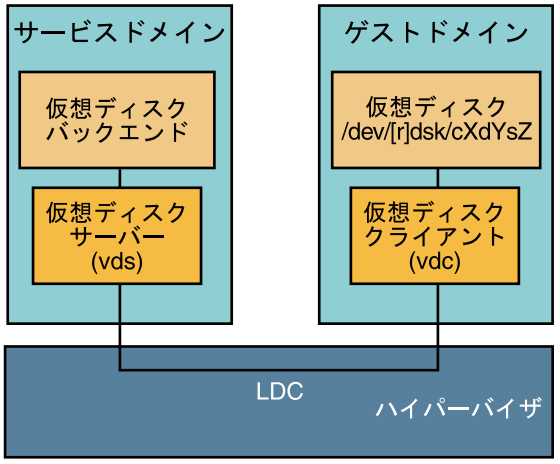

図 6–1　Logical Domains での仮想ディスク

仮想ディスクの管理

この節では、ゲストドメインへの仮想ディスクの追加、仮想ディスクオプションとタイムアウトオプションの変更、およびゲストドメインからの仮想ディスクの削除について説明します。仮想ディスクオプションの説明については、75 ページの「仮想ディスクバックエンドオプション」 を参照してください。仮想ディスクのタイムアウトの説明については、89 ページの「仮想ディスクのタイムアウト」 を参照してください。

▼ 仮想ディスクを追加する

1 仮想ディスクバックエンドをサービスドメインからエクスポートします。

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

2 このバックエンドをゲストドメインに割り当てます。

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

`id` プロパティーを設定して、新しい仮想ディスクデバイスの ID を指定できます。デフォルトでは ID 値は自動的に生成されるため、OS で既存のデバイス名に一致させる必要がある場合に、このプロパティーを設定します。73 ページの「仮想ディスクの識別子とデバイス名」 を参照してください。

注 – バックエンドは、ゲストドメイン (*ldom*) がバインドされたときに、実際にサービスドメインからエクスポートされ、ゲストドメインに割り当てられます。

▼ 仮想ディスクバックエンドを複数回エクスポートする

仮想ディスクバックエンドは、同じ仮想ディスクまたは別の仮想ディスクサーバーのいずれかを介して複数回エクスポートできます。仮想ディスクバックエンドのエクスポートされたインスタンスは、それぞれ同じゲストドメインまたは別のゲストドメインのいずれかに割り当てることができます。

仮想ディスクバックエンドを複数回エクスポートする場合は、排他 (`excl`) オプションを指定してエクスポートしないでください。`excl` オプションを指定すると、バックエンドのエクスポートは 1 回のみ許可されます。`ro` オプションを指定すると、バックエンドは読み取り専用デバイスとして問題なく複数回エクスポートできます。

注意 – 仮想ディスクバックエンドが複数回エクスポートされる際は、ゲストドメインで動作中のアプリケーションおよびその仮想ディスクを使用中のアプリケーションが、同時の書き込みアクセスを調整および同期化して、データの一貫性を確保する役割を果たします。

次の例では、同じ仮想ディスクサービスを介して 2 つの異なるゲストドメインに同じ仮想ディスクを追加する方法について説明します。

1. 次のコマンドを使用して、サービスドメインから仮想ディスクバックエンドを **2** 回エクスポートします。

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

   2 つめの `ldm add-vdsdev` コマンドでは、`-f` オプションを使用して、バックエンドの 2 回目のエクスポートを強制実行します。両方のコマンドに同じバックエンドパスを使用する場合や、仮想ディスクサーバーが同じサービスドメインに存在する場合に、このオプションを使用します。

2. 次のコマンドを使用して、エクスポートされたバックエンドを各ゲストドメインに割り当てます。

   `ldom1` と `ldom2` には、異なる *disk-name* を指定できます。

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

▼ 仮想ディスクオプションを変更する

- サービスドメインからバックエンドがエクスポートされたあとに、次のコマンドを使用して仮想ディスクオプションを変更できます。

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

▼ タイムアウトオプションを変更する

- 仮想ディスクがゲストドメインに割り当てられたあとに、次のコマンドを使用して仮想ディスクのタイムアウトを変更できます。

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

▼ 仮想ディスクを削除する

1. 次のコマンドを使用して、ゲストドメインから仮想ディスクを削除します。

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

2. 次のコマンドを使用して、サービスドメインからの対応するバックエンドのエクスポートを停止します。

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

仮想ディスクの識別子とデバイス名

`ldm add-vdisk` コマンドを使用してドメインに仮想ディスクを追加する際に、`id` プロパティーを設定して、その仮想ディスクのデバイス番号を指定できます。

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

ドメインの各仮想ディスクには、ドメインがバインドされるときに割り当てられる一意のデバイス番号があります。`id` プロパティーを設定して仮想ディスクを明示的なデバイス番号で追加した場合、指定したデバイス番号が使用されます。デバイス番号を指定しなかった場合、使用可能なもっとも小さいデバイス番号が自動的に割り当てられます。その場合、割り当てられるデバイス番号は、仮想ディスクがドメインに追加された方法によって異なります。仮想ディスクに最終的に割り当てられたデバイス番号は、ドメインがバインドされるときに `ldm list-bindings` コマンドの出力で確認できます。

仮想ディスクが構成されたドメインで Solaris OS を実行している場合、そのドメインでは、各仮想ディスクは `c0d`*n* ディスクデバイスとして表示されます。*n* は仮想ディスクのデバイス番号です。

次の例では、`ldg1` ドメインに、`rootdisk` と `pdisk` という 2 つの仮想ディスクがあります。`rootdisk` のデバイス番号は `0` (`disk@0`) で、ドメインではディスクデバイス `c0d0` として表示されます。`pdisk` のデバイス番号は `1` (`disk@1`) で、ドメインではディスクデバイス `c0d1` として表示されます。

```
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 – デバイス番号が仮想ディスクに明示的に割り当てられていない場合、ドメインのバインドがいったん解除されたあとで再びバインドされると、デバイス番号が変更されることがあります。その場合、ドメインで実行している OS によって割り当てられたデバイス名も変更され、システムの既存の構成が損なわれることがあります。これは、たとえば、仮想ディスクがドメインの構成から削除されたときに起こる場合があります。

仮想ディスクの表示

バックエンドが仮想ディスクとしてエクスポートされると、ゲストドメインにフルディスクまたは 1 つのスライスディスクとして表示可能になります。表示形式は、バックエンドの種類およびバックエンドのエクスポート時に使用したオプションによって異なります。

フルディスク

バックエンドをフルディスクとしてドメインにエクスポートすると、8 つのスライス (`s0` 〜 `s7`) を持つ通常のディスクとしてドメインに表示されます。このようなディスクは、`format(1M)` コマンドを使用して表示できます。ディスクのパーティションテーブルは、`fmthard(1M)` または `format`(1M) コマンドのいずれかを使用して変更できます。

また、フルディスクは OS インストールソフトウェアからも表示でき、OS のインストール先のディスクとして選択できます。

どのバックエンドも、フルディスクとしてエクスポートできます。ただし、1 つのスライスディスクとしてのみエクスポート可能な物理ディスクスライスは除きます。

1 つのスライスディスク

バックエンドを 1 つのスライスディスクとしてドメインにエクスポートすると、8 つのスライス (`s0` 〜 `s7`) を持つ通常のディスクとしてドメインに表示されます。ただし、使用できるのは 1 番めのスライス (`s0`) のみです。このようなディスクは、`format`(1M) コマンドで表示できますが、ディスクのパーティションテーブルは変更できません。

また、1 つのスライスディスクは OS インストールソフトウェアからも表示でき、OS のインストール先のディスクとして選択できます。この場合、UNIX ファイルシステム (UFS) を使用して OS をインストールするときは、ルートパーティション (`/`) のみを定義し、このパーティションがすべてのディスク領域を使用する必要があります。

どのバックエンドも、1 つのスライスディスクとしてエクスポートできます。ただし、フルディスクとしてのみエクスポートできる物理ディスクは除きます。

注 – Solaris 10 10/08 OS より前のリリースでは、1 つのスライスディスクは、1 つのパーティションを持つディスクとして表示されていました (s0)。このようなディスクは、format(1M) コマンドを使用して表示できませんでした。また、OS インストールソフトウェアからも表示できず、OS をインストール可能なディスクデバイスとして選択することができませんでした。

仮想ディスクバックエンドオプション

仮想ディスクのバックエンドをエクスポートする際には、さまざまなオプションを指定できます。これらのオプションは、ldm add-vdsdev コマンドの options= 引数にコンマ区切りのリストとして指定します。有効なオプションは、ro、slice、および excl です。

読み取り専用 (ro) オプション

読み取り専用 (ro) オプションは、バックエンドが読み取り専用デバイスとしてエクスポートされることを指定します。その場合、ゲストドメインに割り当てられるこの仮想ディスクに対しては読み取り操作のアクセスのみが可能で、仮想ディスクへの書き込み操作は失敗します。

排他 (excl) オプション

排他 (excl) オプションは、サービスドメインのバックエンドを仮想ディスクとして別のドメインにエクスポートするときに、仮想ディスクサーバーによって排他的に開かれる必要があることを指定します。バックエンドが排他的に開かれると、サービスドメインのほかのアプリケーションがこのバックエンドにアクセスすることはできません。これによって、サービスドメインで動作するアプリケーションが、ゲストドメインでも使用されているバックエンドを誤って使用することはなくなります。

注 – ドライバには excl オプションを受け入れないものもあるため、一部の仮想ディスクバックエンドを排他的に開くことが許可されません。excl オプションが物理ディスクおよびスライスで機能することはわかっていますが、このオプションはファイルでは機能しません。ディスクボリュームなどの擬似デバイスでは機能する場合と機能しない場合があります。バックエンドのドライバで排他的オープンが受け入れられない場合、バックエンドの excl オプションは無視され、バックエンドは排他的に開かれません。

excl オプションによって、サービスドメインで動作中のアプリケーションが、ゲストドメインにエクスポートされるバックエンドにアクセスできなくなるため、次の場合は excl オプションを設定しないでください。

- ゲストドメインの動作中に format(1M)、luxadm(1M) などのコマンドを使用して物理ディスクを管理できるようにする場合は、これらの物理ディスクをエクスポートする際に excl オプションを指定しないでください。
- RAID、ミラー化ボリュームなどの SVM ボリュームをエクスポートする場合は、excl オプションを設定しないでください。このようにしないと、RAID またはミラー化ボリュームのコンポーネントに障害が発生した場合に、SVM で一部の復旧処理の開始が妨げられる可能性があります。詳細は、97 ページの「SVM での仮想ディスクの使用」 を参照してください。
- Veritas Volume Manager (VxVM) がサービスドメインにインストールされていて、Veritas Dynamic Multipathing (VxDMP) が物理ディスクに対して有効な場合は、excl オプション (デフォルトではない) を指定せずに物理ディスクをエクスポートする必要があります。このようにしないと、仮想ディスクサーバー (vds) が物理ディスクデバイスを開くことができないため、エクスポートは失敗します。詳細は、98 ページの「VxVM のインストール時の仮想ディスクの使用」 を参照してください。
- 同じ仮想ディスクバックエンドを同じ仮想ディスクサービスから複数回エクスポートする場合の詳細は、71 ページの「仮想ディスクバックエンドを複数回エクスポートする」 を参照してください。

デフォルトでは、バックエンドは排他的ではない状態で開かれます。このため、バックエンドが別のドメインにエクスポートされている間でも、サービスドメインで動作中のアプリケーションはこのバックエンドを使用できます。これは、Solaris 10 5/08 OS リリースから導入された新しい動作です。Solaris 10 5/08 OS より前のリリースでは、ディスクバックエンドは常に排他的に開かれ、バックエンドを排他的でない状態で開くことはできませんでした。

スライス (slice) オプション

通常、バックエンドは、その種類に応じてフルディスクまたは 1 つのスライスディスクのいずれかとしてエクスポートされます。slice オプションを指定すると、バックエンドは強制的に 1 つのスライスディスクとしてエクスポートされます。

このオプションは、バックエンドの raw コンテンツをエクスポートする場合に便利です。たとえば、データを格納済みの ZFS または SVM ボリュームがある場合に、ゲストドメインでこのデータにアクセスするには、slice オプションを使用して ZFS または SVM ボリュームをエクスポートする必要があります。

このオプションの詳細は、77 ページの「仮想ディスクバックエンド」 を参照してください。

仮想ディスクバックエンド

仮想ディスクバックエンドは、仮想ディスクのデータの格納場所です。バックエンドには、ディスク、ディスクスライス、ファイル、またはボリューム (ZFS、SVM、VxVM など) を使用できます。バックエンドは、バックエンドをサービスドメインからエクスポートする際に `slice` オプションを設定するかどうかに応じて、フルディスクまたは 1 つのスライスディスクのいずれかとしてゲストドメインに表示されます。デフォルトでは、仮想ディスクバックエンドは読み取りおよび書き込み可能なフルディスクとして排他的でない状態でエクスポートされます。

物理ディスクまたはディスクの LUN

物理ディスクまたはディスク LUN は、常にフルディスクとしてエクスポートされます。この場合、仮想ディスクドライバ (`vds` および `vdc`) は仮想ディスクからの入出力を転送し、物理ディスクまたはディスク LUN へのパススルーとして動作します。

`slice` オプションを設定せずにそのディスクのスライス 2 (`s2`) に対応するデバイスをエクスポートすると、物理ディスクまたはディスク LUN はサービスドメインからエクスポートされます。`slice` オプションを指定してディスクのスライス 2 をエクスポートすると、ディスク全体ではなくこのスライスのみがエクスポートされます。

▼ 物理ディスクを仮想ディスクとしてエクスポートする

1. 物理ディスクを仮想ディスクとしてエクスポートします。

   たとえば、物理ディスク `c1t48d0` を仮想ディスクとしてエクスポートするには、そのディスクのスライス 2 (`c1t48d0s2`) をエクスポートする必要があります。

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

2. このディスクをゲストドメインに割り当てます。

   たとえば、ディスク `pdisk` をゲストドメイン `ldg1` に割り当てます。

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

3. ゲストドメインが起動されて **Solaris OS** が実行されたら、そのディスクがアクセス可能であり、フルディスクであることを確認します。

   フルディスクとは、8 つのスライスを持つ通常のディスクのことです。

   確認するディスクが `c0d1` の場合、次のようになります。

   ```
   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
```

物理ディスクスライス

物理ディスクスライスは、常に 1 つのスライスディスクとしてエクスポートされます。この場合、仮想ディスクドライバ (vds および vdc) は仮想ディスクから入出力を転送し、物理ディスクスライスへのパススルーとして動作します。

物理ディスクスライスは、対応するスライスデバイスをエクスポートすることで、サービスドメインからエクスポートされます。デバイスがスライス 2 と異なる場合は、slice オプションの指定の有無にかかわらず、自動的に 1 つのスライスディスクとしてエクスポートされます。デバイスがディスクのスライス 2 である場合は、slice オプションを設定して、スライス 2 のみを 1 つのスライスディスクとしてエクスポートする必要があります。このようにしないと、ディスク全体がフルディスクとしてエクスポートされます。

▼ 物理ディスクスライスを仮想ディスクとしてエクスポートする

1 物理ディスクのスライスを仮想ディスクとしてエクスポートします。

たとえば、物理ディスク c1t57d 0 のスライス 0 を仮想ディスクとしてエクスポートするには、そのスライス (c1t57d0s0) に対応するデバイスを次のようにエクスポートする必要があります。

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

スライスは常に 1 つのスライスディスクとしてエクスポートされるため、slice オプションを指定する必要はありません。

2 このディスクをゲストドメインに割り当てます。

たとえば、ディスク pslice をゲストドメイン ldg1 に割り当てます。

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

3 ゲストドメインが起動されて **Solaris OS** が実行されたら、ディスク (c0d13 など) を表示して、そのディスクがアクセス可能であることを確認できます。

```
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
```

デバイスは 8 つありますが、そのディスクは 1 つのスライスディスクであるため、使用できるのは 1 番めのスライス (`s0`) のみです。

▼ スライス 2 をエクスポートする

- スライス **2** (ディスク `c1t57d0s2` など) をエクスポートするには、`slice` オプションを指定する必要があります。このようにしないと、ディスク全体がエクスポートされます。

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

ファイルおよびボリューム

ファイルまたはボリューム (たとえば ZFS または SVM からの) は、`slice` オプションの指定の有無に応じて、フルディスクまたは 1 つのスライスディスクのいずれかとしてエクスポートされます。

フルディスクとしてエクスポートされるファイルまたはボリューム

`slice` オプションを設定しない場合、ファイルまたはボリュームはフルディスクとしてエクスポートされます。この場合、仮想ディスクドライバ (`vds` および `vdc`) は仮想ディスクから入出力を転送し、仮想ディスクのパーティション分割を管理します。最終的には、このファイルまたはボリュームは、仮想ディスクのすべてのスライスのデータ、およびパーティション分割とディスク構造の管理に使用されるメタデータを含むディスクイメージになります。

空のファイルまたはボリュームをフルディスクとしてエクスポートすると、未フォーマットのディスク、つまり、パーティションのないディスクとしてゲストドメインに表示されます。このため、ゲストドメインで `format(1M)` コマンドを実行して、使用可能なパーティションを定義し、有効なディスクラベルを書き込む必要があります。ディスクが未フォーマットの間、この仮想ディスクへの入出力はすべて失敗します。

注 – Solaris 10 5/08 OS より前のリリースでは、空のファイルが仮想ディスクとしてエクスポートされると、システムによってデフォルトのディスクラベルが書き込まれ、デフォルトのパーティションが作成されていました。Solaris 10 5/08 OS リリースではこの処理は行われなくなったため、ゲストドメインで `format`(1M) を実行してパーティションを作成する必要があります。

▼ ファイルをフルディスクとしてエクスポートする

1 サービスドメインから、ファイル (`fdisk0` など) を作成して仮想ディスクとして使用します。

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

ファイルのサイズによって、仮想ディスクのサイズが定義されます。この例では、100M バイトの空のファイルを作成して、100M バイトの仮想ディスクを取得しています。

2 制御ドメインから、ファイルを仮想ディスクとしてエクスポートします。

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

この例では、`slice` オプションを設定していないため、ファイルはフルディスクとしてエクスポートされます。

3 制御ドメインから、ディスクをゲストドメインに割り当てます。

たとえば、ディスク `fdisk` をゲストドメイン `ldg1` に割り当てます。

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

4 ゲストドメインが起動されて **Solaris OS** が実行されたら、そのディスクがアクセス可能であり、フルディスクであることを確認します。

フルディスクとは、8 つのスライスを持つ通常のディスクのことです。

次の例は、ディスク `c0d5` を表示して、そのディスクがアクセス可能であり、フルディスクであることを確認する方法を示しています。

```
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
```

1つのスライスディスクとしてエクスポートされるファイルまたはボリューム

slice オプションを設定すると、ファイルまたはボリュームは 1 つのスライスディスクとしてエクスポートされます。この場合、仮想ディスクには 1 つのパーティション (s0) のみが含まれ、このパーティションが直接ファイルまたはボリュームバックエンドにマップされます。ファイルまたはボリュームには仮想ディスクに書き込まれるデータのみが含まれ、パーティション情報やディスク構造などの追加データは含まれません。

ファイルまたはボリュームが 1 つのスライスディスクとしてエクスポートされると、システムは擬似的なディスクのパーティション分割のシミュレーションを行います。これにより、そのファイルまたはボリュームはディスクスライスとして表示されます。ディスクのパーティション分割のシミュレーションが行われるため、そのディスクに対してパーティションは作成しないでください。

▼ ZFS ボリュームを 1 つのスライスディスクとしてエクスポートする

1. **ZFS ボリュームを作成して、1 つのスライスディスクとして使用します。**

   次の例は、ZFS ボリューム zdisk0 を作成して、1 つのスライスディスクとして使用する方法を示しています。

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

   ボリュームのサイズによって、仮想ディスクのサイズが定義されます。この例では、100M バイトのボリュームを作成して、100M バイトの仮想ディスクを取得しています。

2. **制御ドメインから、その ZFS ボリュームに対応するデバイスをエクスポートします。このボリュームが 1 つのスライスディスクとしてエクスポートされるように slice オプションを設定します。**

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

3. **制御ドメインから、ボリュームをゲストドメインに割り当てます。**

   次の例は、ボリューム zdisk0 をゲストドメイン ldg1 に割り当てる方法を示しています。

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

4. **ゲストドメインが起動されて Solaris OS が実行されたら、ディスク (c0d9 など) を表示して、そのディスクがアクセス可能で、1 つのスライスディスク (s0) であることを確認できます。**

   ```
   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
```

ボリュームのエクスポートおよび下位互換性

Solaris 10 5/08 OS より前のリリースでは、`slice` オプションがなく、ボリュームは 1 つのスライスディスクとしてエクスポートされていました。ボリュームを仮想ディスクとしてエクスポートする構成である場合に、そのシステムを Solaris 10 5/08 OS にアップグレードすると、ボリュームは 1 つのスライスディスクではなくフルディスクとしてエクスポートされるようになります。アップグレード前の動作を保持して、ボリュームを 1 つのスライスディスクとしてエクスポートするには、次のいずれかを実行する必要があります。

- LogicalDomains1.3 ソフトウェアで `ldm set-vdsdev` コマンドを使用して、1 つのスライスディスクとしてエクスポートするすべてのボリュームに `slice` オプションを設定します。このコマンドの詳細は、`ldm(1M)` マニュアルページを参照してください。
- 次の行を、サービスドメインの `/etc/system` ファイルに追加します。

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

注 – この調整可能なオプションを設定すると、すべてのボリュームが強制的に 1 つのスライスディスクとしてエクスポートされ、ボリュームをフルディスクとしてエクスポートできなくなります。

各種のバックエンドのエクスポート方法の概要

バックエンド	スライスオプションなし	スライスオプションを設定
ディスク (ディスクスライス 2)	フルディスク[1]	1 つのスライスディスク[2]
ディスクスライス (スライス 2 以外)	1 つのスライスディスク[3]	1 つのスライスディスク
ファイル	フルディスク	1 つのスライスディスク

[1] ディスク全体をエクスポートします。

[2] スライス 2 のみをエクスポートします。

[3] スライスは常に 1 つのスライスディスクとしてエクスポートされます。

バックエンド	スライスオプションなし	スライスオプションを設定
ボリューム (ZFS、SVM、VxVM など)	フルディスク	1 つのスライスディスク

ファイルおよびディスクスライスを仮想ディスクとしてエクスポートする場合のガイドライン

この節では、ファイルおよびディスクスライスを仮想ディスクとしてエクスポートする場合のガイドラインを示します。

ループバックファイル (lofi) ドライバの使用

ループバックファイル (lofi) ドライバを使用すると、ファイルを仮想ディスクとしてエクスポートできます。ただし、これを行うと別のドライバ層が追加され、仮想ディスクのパフォーマンスに影響を及ぼします。代わりに、フルディスクまたは 1 つのスライスディスクとしてファイルを直接エクスポートすることができます。79 ページの「ファイルおよびボリューム」 を参照してください。

ディスクスライスの直接的または間接的なエクスポート

仮想ディスクとしてスライスを直接的に、または SVM ボリュームを介すなどして間接的にエクスポートするには、prtvtoc(1M) コマンドを使用して、スライスが物理ディスクの最初のブロック (ブロック 0) で開始されていないことを確認します。

物理ディスクの最初のブロックから始まるディスクスライスを直接的または間接的にエクスポートする場合は、物理ディスクのパーティションテーブルを上書きして、そのディスクのすべてのパーティションにアクセスできないようにすることもできます。

仮想ディスクマルチパスの構成

さまざまなサービスドメインを介して仮想ディスクバックエンドにアクセスできる場合は、仮想ディスクマルチパスを構成して、サービスドメインがダウンしても、ゲストドメイン内の仮想ディスクにアクセス可能にすることができます。さまざまなサービスドメインを介してアクセス可能な仮想ディスクバックエンドの例として、複数のサービスドメインに接続されたネットワークファイルシステム (NFS) サーバー上または共有物理ディスク上のファイルがあります。

仮想ディスクマルチパスを有効にするには、別のサービスドメインから仮想ディスクバックエンドをエクスポートし、同じマルチパスグループ (mpgroup) に追加する必要があります。仮想ディスクバックエンドがエクスポートされると、mpgroup は名前で識別され、構成されます。

次の図は、仮想ディスクマルチパスの構成方法を示しています。この例では、**foo** というマルチパスグループを使用して仮想ディスクを作成しています。そのバックエンドには、第一サービスドメインと代替サービスドメインの 2 つからアクセスできます。

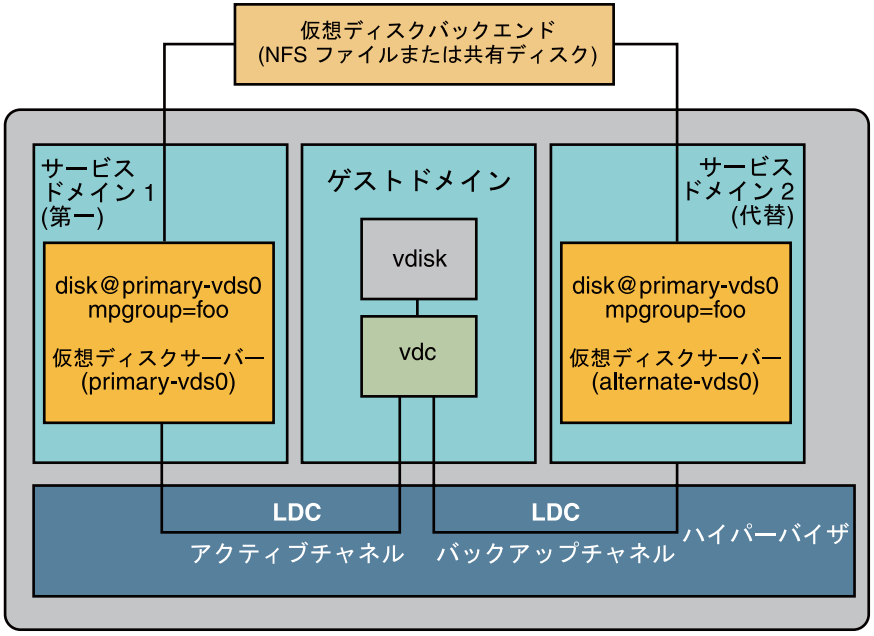

図 6–2　仮想ディスクマルチパスの構成

▼ 仮想ディスクマルチパスを構成する

1　仮想バックエンドを第一サービスドメインからエクスポートします。

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

backend-path1 は、第一サービスドメインから仮想ディスクバックエンドへのパスです。

2　同じ仮想バックエンドを代替サービスドメインからエクスポートします。

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

backend-path2 は、代替サービスドメインから仮想ディスクバックエンドへのパスです。

注 – *backend-path1* および *backend-path2* は、同じ仮想ディスクバックエンドへのパスですが、それらのエクスポート元は異なる 2 つのドメイン (第一と代替) です。これらのパスは、第一サービスドメインおよび代替サービスドメインの構成に応じて、同じ場合もあれば、異なる場合もあります。*volume* 名はユーザーが選択します。これは、両方のコマンドで同じ場合もあれば、異なる場合もあります。

3 仮想ディスクをゲストドメインにエクスポートします。

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

注 – 仮想ディスクバックエンドを複数のサービスドメインを介して複数回エクスポートしていますが、ゲストドメインに割り当てて、いずれかのサービスドメインを介して仮想ディスクバックエンドに関連付ける仮想ディスクは 1 つのみです。

参考 仮想ディスクマルチパスの結果

仮想ディスクをマルチパスで構成し、ゲストドメインを起動すると、仮想ディスクは関連付けられているサービスドメイン (この例では第一サービスドメイン) を介してバックエンドにアクセスします。このサービスドメインが利用できなくなると、仮想ディスクは、同じマルチパスグループに属する別のサービスドメインを介してバックエンドへのアクセスを試みます。

注意 – マルチパスグループ (`mpgroup`) を定義する場合、同じ `mpgroup` に属する仮想ディスクバックエンドは、事実上同じ仮想ディスクバックエンドにする必要があります。異なる仮想ディスクのバックエンドを同じ `mpgroup` に追加すると、予期しない動作が生じ、それらのバックエンドに格納されているデータが消失または破損する可能性があります。

CD、DVD および ISO イメージ

コンパクトディスク (CD) またはデジタル多用途ディスク (DVD) のエクスポートは、通常のディスクと同じ方法で実行できます。CD または DVD をゲストドメインにエクスポートするには、CD または DVD デバイスのスライス 2 をフルディスクとして、つまり `slice` オプションを指定しないでエクスポートします。

注 – CD または DVD ドライブ自体をエクスポートすることはできません。エクスポートできるのは、CD または DVD ドライブ内の CD または DVD のみです。このため、CD または DVD はエクスポート前にドライブ内に存在している必要があります。また、CD または DVD をエクスポートできるようにするには、その CD または DVD がサービスドメインで使用中になっていない必要があります。特に、ボリューム管理ファイルシステムの `volfs(7FS)` サービスが CD または DVD を使用していてはいけません。`volfs` によるデバイスの使用を解除する方法については、87 ページの「CD または DVD をサービスドメインからゲストドメインにエクスポートする」 を参照してください。

ファイルまたはボリュームに CD または DVD の ISO (国際標準化機構) イメージが格納されている場合に、そのファイルまたはボリュームをフルディスクとしてエクスポートすると、ゲストドメインで CD または DVD として表示されます。

CD、DVD、または ISO イメージをエクスポートすると、自動的にゲストドメインで読み取り専用デバイスとして表示されます。ただし、ゲストドメインから CD の制御操作を実行することはできません。つまり、ゲストドメインから CD の起動、停止、または取り出しは実行できません。エクスポートされた CD、DVD、または ISO イメージを起動可能な場合は、対応する仮想ディスクでゲストドメインを起動できます。

たとえば、Solaris OS インストール DVD をエクスポートした場合は、その DVD に対応する仮想ディスク上のゲストドメインを起動し、その DVD からゲストドメインをインストールすることができます。これを行うには、ゲストドメインで `ok` プロンプトが表示されたときに次のコマンドを使用します。

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

n は、エクスポートされた DVD を表す仮想ディスクのインデックスです。

注 – Solaris OS インストール DVD をエクスポートし、その DVD に対応する仮想ディスク上でゲストドメインを起動してゲストドメインをインストールする場合、インストール中に DVD を変更することはできません。このため、異なる CD または DVD を要求するインストール手順は省略する必要がある場合があります。または、要求されたメディアにアクセスするための代替パスを指定する必要があります。

▼ CD または DVD をサービスドメインからゲストドメインにエクスポートする

1 サービスドメインから、ボリューム管理デーモンの vold(1M) が動作中でオンラインかどうかを確認します。

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

2 次のいずれかを実行します。

- ボリューム管理デーモンが動作中またはオンラインでない場合は、手順 3 に進みます。
- 手順 1 の例に示すように、ボリューム管理デーモンが動作中でオンラインの場合は、次の手順を実行します。

a. /etc/vold.conf ファイルを編集して、次の文字列で始まる行をコメントアウトします。

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

vold.conf(4) マニュアルページを参照してください。

b. CD または DVD ドライブに CD または DVD を挿入します。

c. サービスドメインから、ボリューム管理ファイルシステムサービスを再起動します。

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

3 サービスドメインから、CD-ROM デバイスのディスクパスを検出します。

- Solaris 10 OS の場合、cdrw -l コマンドを実行します。

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

- OpenSolaris OS の場合、rmmount -l コマンドを実行します。

```
service# rmmount -l
/dev/dsk/c1t0d0s2    cdrom,cdrom0,cd,cd0,sr,sr0,OpenSolaris,/media/OpenSolaris
```

4 CD または DVD ディスクデバイスをフルディスクとしてエクスポートします。

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

5 エクスポートされた **CD** または **DVD** をゲストドメインに割り当てます。

次の例は、エクスポートされた CD または DVD をドメイン `ldg1` に割り当てる方法を示しています。

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

参考 CD または DVD の複数回のエクスポート

CD または DVD は複数回エクスポートし、異なるゲストドメインに割り当てることができます。詳細は、71 ページの「仮想ディスクバックエンドを複数回エクスポートする」 を参照してください。

▼ primary ドメインから ISO イメージをエクスポートしてゲストドメインをインストールする

ここでは、`primary` ドメインから ISO イメージをエクスポートし、それを使用してゲストドメインをインストールする手順について説明します。この手順では、`primary` ドメインとゲストドメインの両方が構成されていることを前提としています。

たとえば、次のように `ldm list` を実行すると、`primary` ドメインと `ldom1` ドメインの両方が構成されていることが表示されます。

```
# 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 仮想ディスクサーバーデバイスを追加して、**ISO** イメージをエクスポートします。

この例では、ISO イメージは `/export/images/sol-10-u8-ga-sparc-dvd.iso` です。

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

2 ゲストドメインを停止します。

この例では、論理ドメインは `ldom1` です。

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

3 **ISO** イメージの仮想ディスクを論理ドメインに追加します。

この例では、論理ドメインは `ldom1` です。

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

4 ゲストドメインを再起動します。

この例では、論理ドメインは ldom1 です。

```
# 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
```

この例では、ldm list コマンドにより、ldom1 ドメインが起動されたばかりであることが表示されています。

5 ゲストドメインに接続します。

```
# 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 ISO イメージが仮想ディスクとして追加されていることを確認します。

```
{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
```

この例では、新しく追加されたデバイスは /virtual-devices@100/channel-devices@200/disk@1 です。

7 ゲストドメインを起動して、ISO イメージからインストールします。

この例では、/virtual-devices@100/channel-devices@200/disk@1 ディスクの f スライスから起動します。

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

仮想ディスクのタイムアウト

デフォルトでは、仮想ディスクバックエンドへのアクセスを提供するサービスドメインが停止すると、ゲストドメインから対応する仮想ディスクへのすべての入出力がブロックされます。サービスドメインが動作していて、仮想ディスクバックエンドへの入出力要求が処理されている場合、入出力は自動的に再開されます。

ただし、サービスドメインの停止状態が長すぎる場合には、ファイルシステムまたはアプリケーションにとって、入出力処理がブロックされるよりも、入出力処理が

失敗してエラーが報告される方が望ましい場合があります。現在は、仮想ディスクごとに接続タイムアウト時間を設定することが可能になり、ゲストドメインの仮想ディスククライアントとサービスドメインの仮想ディスクサーバー間の接続確立に使用できます。タイムアウト時間に達した場合、サービスドメインが停止し、仮想ディスククライアントと仮想ディスクサーバー間の接続が再確立されていない間中、保留中の入出力および新規の入出力は失敗します。

このタイムアウトは、次のいずれかを実行すると設定できます。

- `ldm add-vdisk` コマンドを使用します。

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

- `ldm set-vdisk` コマンドを使用します。

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

タイムアウトは秒単位で指定します。タイムアウトを `0` に設定すると、タイムアウトは無効になり、サービスドメインの停止中は入出力がブロックされます (デフォルトの設定および動作)。

また、ゲストドメインの `/etc/system` ファイルに次の行を追加すると、タイムアウトを設定できます。

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

注 – この調整可能なオプションを設定すると、`ldm` CLI を使用して設定されたタイムアウトが上書きされます。また、この調整可能なオプションはゲストドメインのすべての仮想ディスクのタイムアウトを設定します。

仮想ディスクおよび**SCSI**

物理 SCSI ディスクまたは LUN をフルディスクとしてエクスポートする場合、対応する仮想ディスクでは、ユーザー SCSI コマンドインタフェース `uscsi(7I)` および多重ホストディスク制御操作 `mhd(7i)` がサポートされます。バックエンドとしてファイルまたはボリュームを含む仮想ディスクなど、その他の仮想ディスクでは、これらのインタフェースはサポートされません。

そのため、SCSI コマンド (SVM `metaset`、Solaris Cluster `shared devices` など) を使用するアプリケーションまたは製品機能は、バックエンドとして物理 SCSI ディスクを含む仮想ディスクのみを使用するゲストドメインで使用できます。

注 – SCSI 操作は、仮想ディスクバックエンドとして使用される物理 SCSI ディスクまたは LUN を管理するサービスドメインによって効果的に実行されます。特に、サービスドメインは SCSI の予約を行います。このため、サービスドメインおよびゲストドメインで動作するアプリケーションは、同じ物理 SCSI ディスクに対して SCSI コマンドを発行するべきではありません。そうでないと、ディスクが予期しない状態になる可能性があります。

仮想ディスクおよび format(1M) コマンド

format(1M) コマンドは、ドメイン上に存在するすべての仮想ディスクを認識します。ただし、1 つのスライスディスクとしてエクスポートされた仮想ディスクの場合、format コマンドでは、仮想ディスクのパーティションテーブルを変更できません。label などのコマンドは失敗しますが、書き込もうとするディスクラベルが仮想ディスクにすでに関連付けられているラベルに類似している場合は除きます。

バックエンドが SCSI ディスクである仮想ディスクでは、すべての format(1M) サブコマンドがサポートされています。バックエンドが SCSI ディスクでない仮想ディスクでは、repair、defect など、一部の format(1M) サブコマンドがサポートされていません。この場合、format(1M) の動作は、Integrated Drive Electronics (IDE) ディスクの動作に類似しています。

仮想ディスクと ZFS の使用

この節では、ゲストドメインにエクスポートされる仮想ディスクバックエンドを格納するために ZFS (Zettabyte File System) を使用する方法について説明します。ZFS は、仮想ディスクバックエンドを作成および管理するための便利で強力なソリューションです。ZFS では次のことを実行できます。

- ZFS ボリュームまたは ZFS ファイルにディスクイメージを格納する
- ディスクイメージのバックアップにスナップショットを使用する
- ディスクイメージの複製と、追加ドメインのプロビジョニングに複製を使用する

ZFS の使用法の詳細は、『Solaris ZFS 管理ガイド』 を参照してください。

次の説明および例で示す primary ドメインは、ディスクイメージが格納されるサービスドメインでもあります。

サービスドメインでの ZFS プールの構成

ディスクイメージを格納するには、まずサービスドメインに ZFS ストレージプールを作成します。たとえば、次のコマンドでは、primary ドメインにディスク c1t50d0 が格納された ZFS ストレージプール ldmpool が作成されます。

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

ZFS を使用したディスクイメージの格納

次のコマンドは、ゲストドメイン ldg1 にディスクイメージを作成します。このゲストドメイン用に ZFS ファイルシステムを作成し、このゲストドメインのすべてのディスクイメージをそのファイルシステムに格納します。

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

ディスクイメージは、ZFS ボリュームまたは ZFS ファイルに格納できます。ZFS ボリュームは、サイズにかかわらず、zfs create -V コマンドを使用すると迅速に作成できます。一方、ZFS ファイルは、mkfile コマンドを使用して作成する必要があります。このコマンドの完了まで少し時間がかかることがあります。特に、作成するファイルが非常に大きいときに時間がかかり、多くはディスクイメージの作成時に該当します。

ZFS ボリュームと ZFS ファイルはいずれも、スナップショットや複製など、ZFS 機能の利点を利用できますが、ZFS ボリュームは疑似デバイス、ZFS ファイルは通常のファイルです。

ディスクイメージを、Solaris OS のインストール先の仮想ディスクとして使用する場合は、次のものを収容できる容量を確保してください。

- インストールされるソフトウェア - 約 6G バイト
- スワップパーティション - 約 1G バイト
- システムデータを格納するための特別なスペース - 1G バイト以上

したがって、Solaris OS 全体をインストールするためのディスクイメージのサイズは、8G バイト以上になります。

ZFS によるディスクイメージの格納例

次の手順を実行します。

1. ZFS ボリュームまたは ZFS ファイルに 10G バイトのイメージを作成します。
2. ZFS ボリュームまたは ZFS ファイルを仮想ディスクとしてエクスポートします。ZFS ボリュームまたは ZFS ファイルをエクスポートする構文は同じですが、バックエンドへのパスは異なります。
3. エクスポートされた ZFS ボリュームまたは ZFS ファイルをゲストドメインに割り当てます。

ゲストドメインが起動すると、ZFS ボリュームまたは ZFS ファイルは、Solaris OS のインストールが可能な仮想ディスクとして表示されます。

▼ ZFS ボリュームを使用してディスクイメージを作成する

- たとえば、ZFS ボリュームに 10G バイトのディスクイメージを作成します。

```
primary# zfs create -V 10gb ldmpool/ldg1/disk0
```

▼ ZFS ファイルを使用してディスクイメージを作成する

- たとえば、ZFS ボリュームに 10G バイトのディスクイメージを作成します。

```
primary# zfs create ldmpool/ldg1/disk0
primary# mkfile 10g /ldmpool/ldg1/disk0/file
```

▼ ZFS ボリュームをエクスポートする

- ZFS ボリュームを仮想ディスクとしてエクスポートします。

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

▼ ZFS ファイルをエクスポートする

- ZFS ファイルを仮想ディスクとしてエクスポートします。

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

▼ ZFS ボリュームまたは ZFS ファイルをゲストドメインに割り当てる

- ZFS ボリュームまたは ZFS ファイルをゲストドメイン (次の例では `ldg1`) に割り当てます。

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

ディスクイメージのスナップショットの作成

ディスクイメージが ZFS ボリュームまたは ZFS ファイルに格納されている場合は、ZFS スナップショットコマンドを使用して、このディスクイメージのスナップショットを作成できます。

ディスクイメージに現在格納されているデータの一貫性を確保するため、ディスクイメージのスナップショットを作成する前に、ゲストドメインでそのディスクが現在使用されていないことを確認してください。ゲストドメインで確実にディスクが使用中ではない状態にするには、いくつかの方法があります。次のいずれかの手順を実行します。

- ゲストドメインを停止し、バインドを解除します。これはもっとも安全な対処方法であり、また、ゲストドメインの起動ディスクとして使用されているディスクイメージのスナップショットを作成する場合に実行可能な唯一の方法です。
- ゲストドメインで使用されていて、スナップショットの対象になるディスクのスライスのマウントを解除し、ゲストドメインで使用中のスライスがない状態にすることもできます。

この例では、ZFS レイアウトのため、ディスクイメージの格納場所が ZFS ボリュームまたは ZFS ファイルのどちらであっても、ディスクイメージのスナップショットを作成するコマンドは同じです。

▼ ディスクイメージのスナップショットを作成する

- たとえば、`ldg1` ドメインに作成されたディスクイメージのスナップショットを作成します。

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

複製を使用して新規ドメインをプロビジョニングする

ディスクイメージのスナップショットを作成したら、ZFS 複製コマンドを使用してこのディスクイメージを複製できます。そのあと、複製されたイメージを別のドメインに割り当てることができます。起動ディスクイメージを複製することによって、新規ゲストドメイン用の起動ディスクが迅速に作成され、Solaris OS インストールプロセス全体を実行する必要はなくなります。

たとえば、作成された `disk0` がドメイン `ldg1` の起動ディスクである場合、次の手順を実行してこのディスクを複製し、ドメイン `ldg2` の起動ディスクを作成します。

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

`ldompool/ldg2/disk0` は、仮想ディスクとしてエクスポートして、新規の `ldg2` ドメインに割り当てることができます。ドメイン `ldg2` は、OS のインストールプロセスを実行しなくても、この仮想ディスクから直接起動することができます。

起動ディスクイメージの複製

起動ディスクを複製した場合、新しいイメージは元の起動ディスクと全く同一であり、イメージの複製前に起動ディスクに格納されていたホスト名、IP アドレス、マウントされているファイルシステムテーブル、システム構成、チューニングなどの情報が含まれています。

マウントされているファイルシステムテーブルは、元の起動ディスクイメージ上と複製されたディスクイメージ上で同じであるため、複製されたディスクイメージは、元のドメインの場合と同じ順序で新規ドメインに割り当てる必要があります。たとえば、起動ディスクイメージが元のドメインの 1 番めのディスクとして割り当てられていた場合は、複製されたディスクイメージを新規ドメインの 1 番めのディスクとして割り当てる必要があります。このようにしない場合、新規ドメインは起動できなくなります。

元のドメインが静的 IP アドレスで構成されていた場合、複製されたイメージを使用する新規ドメインは、同じ IP アドレスで始まります。この場合は、`sys-unconfig(1M)` コマンドを使用すると、新規ドメインのネットワーク構成を変更できます。この問題を回避するために、未構成のシステムのディスクイメージのスナップショットを作成することもできます。

元のドメインが動的ホスト構成プロトコル (DHCP) で構成されていた場合は、複製されたイメージを使用する新規ドメインも、DHCP を使用します。この場合、新規ドメインの起動時に、IP アドレスとそのネットワーク構成を自動的に受け取るため、新規ドメインのネットワーク構成を変更する必要はありません。

注 – ドメインのホスト ID は起動ディスクには格納されませんが、ドメインの作成時に Logical Domains Manager によって割り当てられます。このため、ディスクイメージを複製した場合、その新規ドメインは元のドメインのホスト ID を保持しません。

▼ 未構成システムのディスクイメージのスナップショットを作成する

1. 元のドメインをバインドし、起動します。
2. `sys-unconfig` コマンドを実行します。
3. `sys-unconfig` コマンドが完了すると、このドメインは停止します。
4. ドメインを停止し、バインドを解除します。ドメインを再起動しないでください。

5 ドメインの起動ディスクイメージのスナップショットを作成します。
次に例を示します。

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

この時点でのスナップショットは、未構成システムの起動ディスクイメージです。

6 このイメージを複製して新規ドメインを作成します。このドメインの最初の起動時に、システムを構成するように求められます。

論理ドメイン環境でのボリュームマネージャーの使用

この節では、論理ドメイン環境でのボリュームマネージャーの使用法について説明します。

ボリュームマネージャーでの仮想ディスクの使用

ZFS (Zettabyte File System)、Solaris ボリュームマネージャー (SVM)、または Veritas Volume Manager (VxVM) は、サービスドメインからゲストドメインに仮想ディスクとしてエクスポートできます。ボリュームは、1 つのスライスディスク (`slice` オプションが `ldm add-vdsdev` コマンドで指定されている場合) またはフルディスクのいずれかとしてエクスポートできます。

注 – この節の残りの部分では、例として SVM ボリュームを使用します。ただし、説明は ZFS および VxVM ボリュームにも適用されます。

次の例に、ボリュームを 1 つのスライスディスクとしてエクスポートする方法を示します。

ゲストドメインの仮想ディスク (たとえば `/dev/dsk/c0d2s0`) は関連付けられたボリューム (たとえば `/dev/md/dsk/d0`) に直接割り当てられ、ゲストドメインからの仮想ディスクに格納されたデータは、メタデータを追加せずに関連付けられたボリュームに直接格納されます。そのためゲストドメインからの仮想ディスクに格納されたデータは、関連付けられたボリュームを介してサービスドメインから直接アクセスすることもできます。

例

- SVM ボリューム `d0` が `primary` ドメインから `domain1` にエクスポートされる場合、`domain1` の構成にはいくつかの手順が追加で必要になります。

```
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
```

- domain1 がバインドされて起動されると、エクスポートされたボリュームが /dev/dsk/c0d2s0 のように表示され、そのボリュームが使用可能になります。

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

- domain1 が停止してバインドが解除されると、domain1 からの仮想ディスクに格納されたデータは SVM ボリューム d0 を介して primary ドメインから直接アクセスできます。

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

SVM での仮想ディスクの使用

RAID またはミラー SVM ボリュームが別のドメインで仮想ディスクとして使用される場合は、排他 (excl) オプションを設定せずにエクスポートする必要があります。このようにしないと、SVM ボリュームのいずれかのコンポーネントで障害が発生したときに、metareplace コマンドまたはホットスペアを使用した SVM ボリュームの復旧が開始されません。metastat コマンドはそのボリュームを再同期化中と判断しますが、再同期化は進行していません。

たとえば、/dev/md/dsk/d0 は excl オプションを使用して別のドメインに仮想ディスクとしてエクスポートされた RAID SVM ボリュームで、d0 にはいくつかのホットスペアデバイスが構成されているとします。d0 のコンポーネントに障害が発生すると、SVM は障害の発生したコンポーネントをホットスペアに交換して、ふたたび SVM ボリュームとの同期をとります。ただし、再同期化は開始されません。ボリュームは再同期化中として報告されますが、再同期化は進行していません。

```
# 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/c10t600C0FF000000000000153153295A4B100d0s1  330  No  Resyncing  Yes
```

このような状況で再同期化を完了するには、SVM ボリュームを仮想ディスクとして使用しているドメインを停止してバインドを解除する必要があります。そのあと、metasync コマンドを使用して、SVM ボリュームを再同期化できます。

```
# metasync d0
```

VxVM のインストール時の仮想ディスクの使用

システムに Veritas Volume Manager (VxVM) がインストールされていて、仮想ディスクとしてエクスポートする物理ディスクまたはパーティションで Veritas Dynamic Multipathing (DMP) が有効な場合は、`excl` オプション (デフォルトではない) を設定せずにそのディスクまたはパーティションをエクスポートする必要があります。そうしない場合、このようなディスクを使用するドメインをバインドする間に `/var/adm/messages` にエラーが出力されます。

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

コマンド `vxdisk list` で出力されるマルチパス化情報を調べると、Veritas DMP が有効であるかどうかを確認できます。次に例を示します。

```
# 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%5F3032333948303144304E0000
site:      -
Multipathing information:
numpaths:  1
c4t12d0s2  state=enabled
```

また、`excl` オプションを設定して仮想ディスクとしてエクスポートするディスクまたはスライスで Veritas DMP が有効になっている場合は、`vxdmpadm` コマンドを使用して DMP を無効にすることもできます。次に例を示します。

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

仮想ディスクでのボリュームマネージャーの使用

この節では、仮想ディスクでのボリュームマネージャーの使用法について説明します。

仮想ディスクでの ZFS の使用

仮想ディスクは ZFS とともに使用できます。ZFS ストレージプール (`zpool`) は、この `zpool` の一部であるすべてのストレージデバイスを認識する任意のドメインにインポートできます。ドメインが、これらのすべてのデバイスを仮想デバイスまたは実デバイスのどちらで認識するかは関係ありません。

仮想ディスクでの SVM の使用

仮想ディスクは、SVM ローカルディスクセットで使用できます。たとえば、仮想ディスクは、ローカルディスクセットの SVM メタデバイス状態データベース metadb(1M) の格納またはローカルディスクセットでの SVM ボリュームの作成に使用できます。

バックエンドが SCSI ディスクであるすべての仮想ディスクは、SVM 共有ディスクセット metaset(1M) で使用できます。バックエンドが SCSI ディスクでない仮想ディスクは、SVM 共有ディスクセットに追加できません。バックエンドが SCSI ディスクでない仮想ディスクを SVM 共有ディスクセットに追加しようとすると、次のようなエラーが表示されて失敗します。

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

仮想ディスクでの VxVM の使用

ゲストドメインでの VxVM サポートについては、Symantec 社の VxVM ドキュメントを参照してください。

◆ ◆ ◆ 第 7 章

仮想ネットワークの使用

この章では、Logical Domains ソフトウェアで仮想ネットワークを使用する方法について説明します。この章の内容は次のとおりです。

仮想ネットワークの概要

仮想ネットワークでは、ドメインが外部の物理ネットワークを使用しないで相互に通信できます。仮想ネットワークでは、複数のドメインが同じ物理ネットワークインタフェースを使用して物理ネットワークにアクセスし、遠隔システムと通信することもできます。仮想ネットワークは、仮想ネットワークデバイスを接続できる仮想スイッチを備えることで構築します。

仮想スイッチ

仮想スイッチ (vsw) とは、サービスドメインで動作し、仮想スイッチドライバによって管理されるコンポーネントのことです。仮想スイッチを複数のゲストドメインに接続すると、これらのドメイン間のネットワーク通信を可能にできます。また、仮想スイッチが物理ネットワークインタフェースにも関連付けられている場合は、物理ネットワークインタフェースを介して、ゲストドメインと物理ネットワークの間のネットワーク通信が有効になります。仮想スイッチはネットワークインタフェース vsw*n* も備えています。このインタフェースによって、サービスドメインは、仮想スイッチに接続されたほかのドメインと通信できます。このインタフェースは通常のネットワークインタフェースと同様に使用でき、`ifconfig(1M)` コマンドで構成できます。

注 – サービスドメインに仮想スイッチを追加する際、そのネットワークインタフェースは plumb されません。このため、デフォルトでは、サービスドメインは仮想スイッチに接続されたゲストドメインと通信できません。ゲストドメインとサービスドメインの間のネットワーク通信を有効にするには、関連付けられた仮想スイッチのネットワークインタフェースを plumb し、サービスドメイン内で構成する必要があります。手順については、53 ページの「制御ドメインまたはサービスドメインとその他のドメイン間のネットワークの有効化」 を参照してください。

仮想ネットワークデバイス

仮想ネットワーク (vnet) デバイスとは、仮想スイッチに接続されたドメイン内で定義されている仮想デバイスのことです。仮想ネットワークデバイスは、仮想ネットワークドライバによって管理され、論理ドメインチャネル (LDC) を使用するハイパーバイザを介して仮想ネットワークに接続されます。

仮想ネットワークデバイスは、vnet*n* という名前のネットワークインタフェースとして使用できます。このネットワークデバイスは通常のネットワークインタフェースと同様に使用でき、`ifconfig(1M)` コマンドで構成できます。

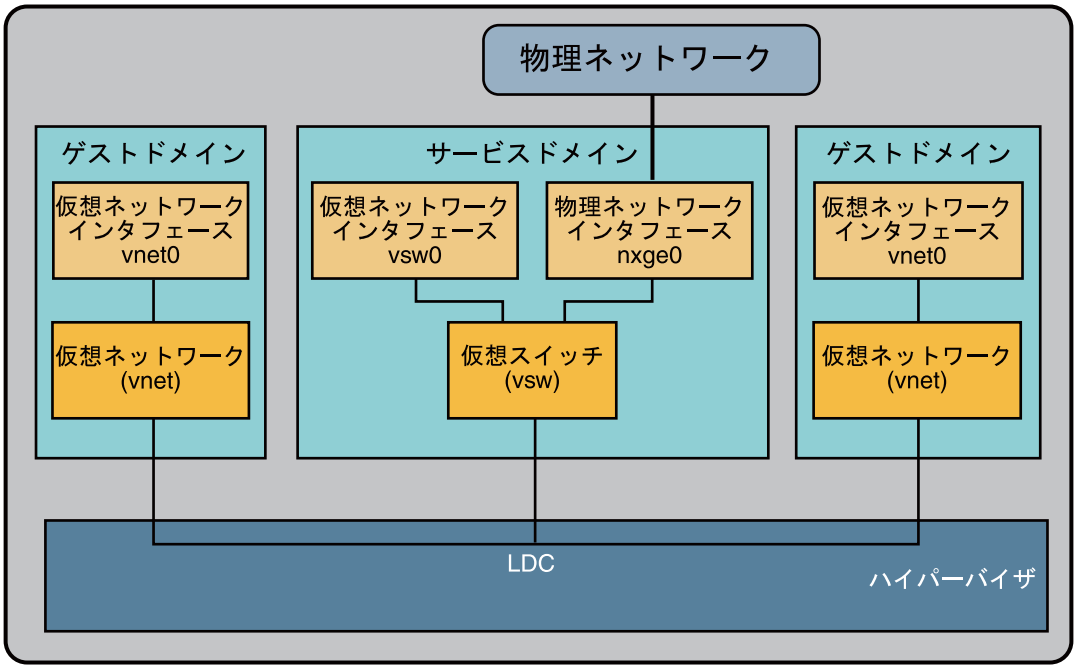

図 7–1　仮想ネットワークの設定

図 7–1 の例の説明は、次のとおりです。

- サービスドメイン内の仮想スイッチは、ゲストドメインに接続されます。この接続によって、ゲストドメイン間で相互に通信することができます。
- 仮想スイッチは物理ネットワークインタフェース `nxge0` にも接続されています。この接続によって、ゲストドメインは物理ネットワークと通信できます。
- 仮想スイッチネットワークインタフェース `vsw0` はサービスドメイン内で plumb されているため、2 つのゲストドメインはサービスドメインと通信できます。
- サービスドメイン内の仮想スイッチネットワークインタフェース `vsw0` は、`ifconfig`(1M) コマンドを使用して構成できます。
- ゲストドメイン内の仮想ネットワークインタフェース `vnet0` は、`ifconfig`(1M) コマンドを使用して構成できます。

基本的に仮想スイッチは、通常の物理ネットワークスイッチと同様に機能し、接続されているゲストドメイン、サービスドメイン、物理ネットワークなど異なるシステム間のネットワークパケットをスイッチングします。

仮想スイッチの管理

この節では、ドメインへの仮想スイッチの追加、仮想スイッチのオプションの設定、および仮想スイッチの削除について説明します。

▼ 仮想スイッチを追加する

- 仮想スイッチを追加するには、次のコマンド構文を使用します。

```
# 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
```

各表記の意味は次のとおりです。

- `default-vlan-id=`*vlan-id* は、仮想スイッチとそれに関連する仮想ネットワークデバイスが暗黙的にタグなしモードで属するデフォルトの仮想ローカルエリアネットワーク (VLAN) を指定します。これは、仮想スイッチおよび仮想ネットワークデバイスのデフォルトのポート VLAN id (pvid) として機能します。このオプションを指定しない場合、このプロパティーのデフォルト値は 1 です。通常、このオプションを指定する必要はありません。このオプションは、単にデフォルト値の 1 を変更する手段として用意されています。詳細は、123 ページの「Logical Domains ソフトウェアでの VLAN のタグ付けの使用」 を参照してください。
- `pvid=`*port-vlan-id* には、仮想スイッチをメンバーにする必要のある VLAN をタグなしモードで指定します。詳細は、123 ページの「Logical Domains ソフトウェアでの VLAN のタグ付けの使用」 を参照してください。
- `vid=`*vlan-id* は、仮想スイッチがタグ付きモードでメンバーとして属する必要のある 1 つ以上の VLAN を指定します。詳細は、123 ページの「Logical Domains ソフトウェアでの VLAN のタグ付けの使用」 を参照してください。
- `linkprop=phys-state` では、配下の物理ネットワークデバイスに基づいて、仮想デバイスがリンクステータスをレポートするかどうかを指定できます。コマンドラインで `linkprop=phys-state` を指定すると、仮想デバイスのリンクステータスは物理リンクステータスを反映します。デフォルトでは、仮想デバイスのリンクステータスは物理リンクステータスを反映しません。

  リンクベースの IPMP を使用するには、このオプションを指定します。118 ページの「Logical Domains 仮想ネットワークでのリンクベースの IPMP の使用」 を参照してください。
- `mac-addr=`*num* は、このスイッチで使用される MAC アドレスです。番号は、標準のオクテット記述法で指定する必要があります。たとえば、80:00:33:55:22:66 とします。MAC アドレスを指定しない場合、スイッチには、Logical Domains Manager

に割り当てられているパブリック MAC アドレス範囲のアドレスが自動的に割り当てられます。詳細は、110 ページの「自動または手動による MAC アドレスの割り当て」 を参照してください。

- `net-dev=`*device* は、このスイッチが動作するネットワークデバイスへのパスです。
- `mode=sc` を指定すると、論理ドメイン環境での Solaris Cluster のハートビートパケットの優先処理用の仮想ネットワークサポートが有効になります。Solaris Cluster などのアプリケーションでは、輻輳した仮想ネットワークおよびスイッチデバイスによって高優先度のハートビートパケットがドロップされないようにする必要があります。このオプションを使用して、Solaris Cluster のハートビートフレームが優先され、これらのフレームが信頼性の高い方法で転送されるようにします。

  論理ドメイン環境で Solaris Cluster を動作させ、ゲストドメインを Solaris Cluster ノードとして使用する場合は、このオプションを設定する必要があります。ゲストドメインで Solaris Cluster ソフトウェアを実行していない場合には、仮想ネットワークのパフォーマンスに影響を与える可能性があるため、このオプションを設定しないでください。
- `mtu=`*size* は、仮想スイッチデバイスの最大転送単位 (MTU) を指定します。有効な値の範囲は 1500 ～ 16000 です。
- `id=`*switch-id* は、新しい仮想スイッチデバイスの ID です。デフォルトでは ID 値は自動的に生成されるため、OS で既存のデバイス名に一致させる必要がある場合に、このプロパティーを設定します。108 ページの「仮想デバイス識別子およびネットワークインタフェース名」 を参照してください。
- *vswitch-name* は、サービスとしてエクスポートされるスイッチの一意の名前です。クライアント (ネットワーク) は、このサービスに接続できます。
- *ldom* には、仮想スイッチを追加する論理ドメインを指定します。

▼ 既存の仮想スイッチのオプションを設定する

- すでに存在している仮想スイッチのオプションを設定するには、次のコマンド構文を使用します。

  ```
  # 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
  ```

  各表記の意味は次のとおりです。

  - `mode=` (空白のまま) では、Solaris Cluster のハートビートパケットの特殊処理が停止されます。
  - それ以外のコマンド引数は、104 ページの「仮想スイッチを追加する」 で説明しているものと同じです。

▼ 仮想スイッチを削除する

- 仮想スイッチを削除するには、次のコマンド構文を使用します。

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

  各表記の意味は次のとおりです。

  - `-f` は、仮想スイッチの強制削除を試行します。削除は失敗することがあります。
  - *vswitch-name* は、サービスとして削除されるスイッチの名前です。

仮想ネットワークデバイスの管理

この節では、ドメインへの仮想ネットワークデバイスの追加、既存の仮想ネットワークデバイスのオプションの設定、および仮想ネットワークデバイスの削除について説明します。

▼ 仮想ネットワークデバイスを追加する

- 仮想ネットワークデバイスを追加するには、次のコマンド構文を使用します。

  # 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

  各表記の意味は次のとおりです。

  - `mac-addr=`*num* は、このネットワークデバイスの MAC アドレスです。数字は、`80:00:33:55:22:66` など標準の 8 ビット表記にする必要があります。詳細は、110 ページの「自動または手動による MAC アドレスの割り当て」 を参照してください。
  - `mode=hybrid` は、可能な場合に、この `vnet` で NIU ハイブリッド I/O を使用するようにシステムに要求します。可能でない場合、システムは仮想 I/O に戻ります。このハイブリッドモードは、アクティブな `vnet` で設定すると、遅延再構成とみなされます。詳細は、127 ページの「NIU ハイブリッド I/O の使用」 を参照してください。
  - `pvid=`*port-vlan-id* には、仮想ネットワークデバイスをメンバーにする必要のある VLAN をタグなしモードで指定します。詳細は、123 ページの「Logical Domains ソフトウェアでの VLAN のタグ付けの使用」 を参照してください。
  - `linkprop=phys-state` では、配下の物理ネットワークデバイスに基づいて、仮想ネットワークデバイスがリンクステータスをレポートするかどうかを指定できます。コマンドラインで `linkprop=phys-state` を指定すると、仮想ネットワークデ

バイスのリンクステータスは物理リンクステータスを反映します。デフォルトでは、仮想ネットワークデバイスのリンクステータスは物理リンクステータスを反映しません。

リンクベースの IPMP を使用するには、このオプションを指定します。118 ページの「Logical Domains 仮想ネットワークでのリンクベースの IPMP の使用」 を参照してください。

- `vid=`*vlan-id* は、仮想ネットワークデバイスがタグ付きモードでメンバーとして属する必要のある 1 つ以上の VLAN を指定します。詳細は、123 ページの「Logical Domains ソフトウェアでの VLAN のタグ付けの使用」 を参照してください。
- `mtu=`*size* は、仮想ネットワークデバイスの最大転送単位 (MTU) を指定します。有効な値の範囲は 1500 ～ 16000 です。
- `id=`*network-id* は、新しい仮想ネットワークデバイスの ID です。デフォルトでは ID 値は自動的に生成されるため、OS で既存のデバイス名に一致させる必要がある場合に、このプロパティーを設定します。108 ページの「仮想デバイス識別子およびネットワークインタフェース名」 を参照してください。
- *if-name* (インタフェースの名前) は、後続の `ldm set-vnet` または `ldm rm-vnet` コマンドで参照するために仮想ネットワークデバイスのインスタンスに割り当てられる、論理ドメインで一意の名前です。
- *vswitch-name* は、接続する既存のネットワークサービス (仮想スイッチ) の名前です。
- *ldom* には、仮想ネットワークデバイスを追加する論理ドメインを指定します。

▼ 既存の仮想ネットワークデバイスのオプションを設定する

- すでに存在している仮想ネットワークデバイスのオプションを設定するには、次のコマンド構文を使用します。

```
# 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
```

各表記の意味は次のとおりです。

- `mode=` (空白のまま) では、NIU ハイブリッド I/O が無効になります。
- *if-name* (インタフェースの名前) は、設定する仮想ネットワークデバイスに割り当てられている一意の名前です。
- *ldom* には、仮想ネットワークデバイスを削除する論理ドメインを指定します。
- それ以外のコマンド引数は、106 ページの「仮想ネットワークデバイスを追加する」 で説明しているものと同じです。

▼ 仮想ネットワークデバイスを削除する

- 仮想ネットワークデバイスを削除するには、次のコマンド構文を使用します。

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

  各表記の意味は次のとおりです。

  - `-f` は、論理ドメインからの仮想ネットワークデバイスの強制削除を試行します。削除は失敗することがあります。
  - *if-name* (インタフェースの名前) は、削除する仮想ネットワークデバイスに割り当てられている一意の名前です。
  - *ldom* には、仮想ネットワークデバイスを削除する論理ドメインを指定します。

仮想デバイス識別子およびネットワークインタフェース名

ドメインに仮想スイッチまたは仮想ネットワークデバイスを追加する場合、`id` プロパティーを設定することでデバイス番号を指定できます。

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

ドメインの各仮想スイッチおよび仮想ネットワークデバイスには、ドメインがバインドされるときに割り当てられる一意のデバイス番号があります。`id` プロパティーを設定して仮想スイッチまたは仮想ネットワークデバイスを明示的なデバイス番号で追加した場合、指定したデバイス番号が使用されます。デバイス番号を指定しなかった場合、使用可能なもっとも小さいデバイス番号が自動的に割り当てられます。その場合、割り当てられるデバイス番号は、仮想スイッチまたは仮想ネットワークデバイスがシステムに追加された方法によって異なります。仮想スイッチまたは仮想ネットワークデバイスに最終的に割り当てられたデバイス番号は、ドメインがバインドされるときに `ldm list-bindings` コマンドの出力で確認できます。

次の例は、`primary` ドメインに 1 つの仮想スイッチ `primary-vsw0` が構成されていることを示しています。この仮想スイッチのデバイス番号は `0` (`switch@0`) です。

```
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
...
```

次の例は、ldg1 ドメインには 2 つの仮想ネットワークデバイス vnet および vnet1 が構成されていることを示しています。デバイス vnet のデバイス番号は 0 (network@0) で、デバイス vnet1 のデバイス番号は 1 (network@1) です。

```
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
...
```

仮想スイッチが構成されたドメインで Solaris OS を実行している場合、仮想スイッチはネットワークインタフェース vsw*N* を備えています。ただし、仮想スイッチのネットワークインタフェース番号 *N* は、仮想スイッチのデバイス番号 *n* と同じとはかぎりません。

同様に、仮想ネットワークデバイスが構成されたドメインで Solaris OS を実行している場合、仮想ネットワークデバイスはネットワークインタフェース vnet*N* を備えています。ただし、仮想ネットワークデバイスのネットワークインタフェース番号 *N* は、仮想ネットワークデバイスのデバイス番号 *n* と同じとはかぎりません。

注意 – Solaris OS では、ネットワークインタフェースの名前と、仮想スイッチまたは仮想ネットワークとの間のマッピングが、デバイス番号に基づいて保存されます。デバイス番号が仮想スイッチまたは仮想ネットワークデバイスに明示的に割り当てられていない場合、ドメインのバインドがいったん解除されたあとで再びバインドされると、デバイス番号が変更されることがあります。その場合、ドメインで動作している OS によって割り当てられたネットワークインタフェース名も変更され、システムの既存の構成が損なわれることがあります。これは、たとえば、仮想スイッチまたは仮想ネットワークインタフェースがドメインの構成から削除されたときに起こる場合があります。

ldm list-* コマンドを使用して、仮想スイッチまたは仮想ネットワークデバイスに対応する Solaris OS のネットワークインタフェース名を直接判定することはできません。ただし、ldm list -l コマンドの出力と、Solaris OS の /devices 配下のエントリの出力を組み合わせて使用すると、この情報を取得できます。

▼ Solaris OS ネットワークインタフェース名を確認する

次の例では、ゲストドメイン ldg1 には net-a および net-c の 2 つの仮想ネットワークデバイスが含まれています。net-c に対応する、ldg1 での Solaris OS ネットワークイ

ンタフェース名を確認するには、次の手順を実行します。この例では、仮想ネットワークデバイスではなく仮想スイッチのネットワークインタフェース名を検索する場合の相違点も示します。

1 ldm コマンドを使用して、net-c の仮想ネットワークデバイス番号を探します。

```
# 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 の仮想ネットワークデバイス番号は 2 (network@2) です。

仮想スイッチのネットワークインタフェース名を判定するには、switch@*n* の *n* に示された仮想スイッチデバイス番号を探します。

2 ldg1 で対応するネットワークインタフェースを検出するには、ldg1 にログインして、/devices 配下でこのデバイス番号に対するエントリを探します。

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

ネットワークインタフェース名は、コロンのあとのエントリの部分で、この場合は vnet1 です。

仮想スイッチのネットワークインタフェース名を判定するには、-name オプションの引数を virtual-network-switch@*n** に置換します。次に、vsw*N* という名前のネットワークインタフェースを探します。

3 vnet1 を **plumb** して、手順 **1** の net-c に対する ldm list -l の出力で示されたように、**MAC** アドレスが 00:14:4f:f8:dd:68 であることを確認します。

```
# 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
```

自動または手動による MAC アドレスの割り当て

使用する予定の論理ドメイン、仮想スイッチ、および仮想ネットワークに割り当てられるだけの十分な数のメディアアクセス制御 (MAC) アドレスが必要です。Logical Domains Manager から論理ドメイン、仮想ネットワーク (vnet)、および仮想スイッチ (vsw) に自動的に MAC アドレスを割り当てるか、割り当てられた MAC アドレスの自身のプールから手動で MAC アドレスを割り当てることができます。MAC アドレス

を設定する ldm のサブコマンドは、add-domain、add-vsw、set-vsw、add-vnet、および set-vnet です。これらのサブコマンドで MAC アドレスを指定しない場合は、Logical Domains Manager が自動的に MAC アドレスを割り当てます。

Logical Domains Manager に MAC アドレスの割り当てを実行させる利点は、論理ドメインで使用するための専用の MAC アドレスのブロックを利用できることです。また、Logical Domains Manager は、同じサブネットにあるほかの Logical Domains Manager インスタンスと競合する MAC アドレスを検出し、これを回避します。これにより、手動で MAC アドレスのプールを管理する必要がなくなります。

論理ドメインが作成されたり、ドメインにネットワークデバイスが構成されたりするとすぐに、MAC アドレスの割り当てが発生します。また、割り当ては、デバイスまたは論理ドメイン自体が削除されるまで保持されます。

Logical Domains ソフトウェアに割り当てられる MAC アドレスの範囲

論理ドメインには、次の 512K の MAC アドレスのブロックが割り当てられています。

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

下位の 256K のアドレスは、Logical Domains Manager による *MAC* アドレスの自動割り当てに使用されるため、この範囲のアドレスを手動で要求することはできません。

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

MAC アドレスを手動で割り当てる場合は、この範囲の上位半分を使用できます。

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

自動割り当てのアルゴリズム

論理ドメインまたはネットワークデバイスの作成時に MAC アドレスを指定しない場合、Logical Domains Manager は MAC アドレスを自動的に確保して、その論理ドメインまたはネットワークデバイスに割り当てます。この MAC アドレスを取得するために、Logical Domains Manager はアドレスの選択を繰り返し試みて、潜在的な競合がないか確認します。

可能性のあるアドレスを選択する前に、Logical Domains Manager は、自動的に割り当てられ、最近解放されたアドレスが、ここで使用するためにデータベースに保存されているかどうかをまず確認します (113 ページの「解放された MAC アドレス」 を参照)。保存されていた場合、Logical Domains Manager はデータベースから候補となるアドレスを選択します。

最近解放されたアドレスが使用できない場合、MAC アドレスはこの用途のために確保された 256K の範囲のアドレスからランダムに選択されます。候補として選択される MAC アドレスが重複する可能性を少なくするために、MAC アドレスはランダムに選択されます。

選択されたアドレスは、ほかのシステムのその他の Logical Domains Manager に対して確認され、重複した MAC アドレスが実際に割り当てられることを防止します。使用されているアルゴリズムは、112 ページの「重複した MAC アドレスの検出」 に記載されています。アドレスがすでに割り当てられている場合、Logical Domains Manager は、ほかのアドレスの選択および競合の再確認を繰り返し行います。この動作は、まだ割り当てられていない MAC アドレスが見つかるか、30 秒の制限時間が経過するまで続きます。制限時間に達すると、デバイスの作成が失敗し、次のようなエラーメッセージが表示されます。

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

重複した MAC アドレスの検出

同じ MAC アドレスが別のデバイスに割り当てられないようにするために、Logical Domains Manager がデバイスに割り当てようとしているアドレスを含むマルチキャストメッセージを、制御ドメインのデフォルトのネットワークインタフェースを介して送信することで、Logical Domains Manager はほかのシステム上の Logical Domains Manager に確認します。MAC アドレスの割り当てを試行している Logical Domains Manager は、応答が返されるまで 1 秒待機します。LDoms が有効な別のシステムの異なるデバイスにその MAC アドレスがすでに割り当てられている場合は、そのシステムの Logical Domains Manager が対象となっている MAC アドレスを含む応答を送信します。要求を送信した Logical Domains Manager は応答を受け取ると、選択した MAC アドレスがすでに割り当てられていることを認識し、別のアドレスを選択して処理を繰り返します。

デフォルトでは、これらのマルチキャストメッセージは、デフォルトの生存期間 (TTL) が 1 である同じサブネット上のほかのマネージャーにのみ送信されます。TTL は、サービス管理機能 (SMF) プロパティー `ldmd/hops` を使用して設定できます。

各 Logical Domains Manager は、次の処理を行います。

- マルチキャストメッセージの待機
- ドメインに割り当てられた MAC アドレスの追跡
- 重複の検索
- 重複が発生しないようにするための応答

何らかの理由でシステム上の Logical Domains Manager が停止すると、Logical Domains Manager が停止している間に MAC アドレスの重複が発生する可能性があります。

論理ドメインまたはネットワークデバイスが作成されるときに MAC の自動割り当てが行われ、そのデバイスまたは論理ドメインが削除されるまで保持されます。

解放されたMACアドレス

自動のMACアドレスに関連付けられた論理ドメインまたはデバイスが削除されると、そのMACアドレスはそのシステムであとで使用する場合に備えて、最近解放されたMACアドレスのデータベースに保存されます。これらのMACアドレスを保存して、動的ホスト構成プロトコル(DHCP)サーバーのインターネットプロトコル(IP)アドレスが使い果たされないようにします。DHCPサーバーがIPアドレスを割り当てるとき、しばらくの間(リース期間中)その動作が行われます。多くの場合、リース期間は非常に長く構成されており、通常は数時間または数日間です。ネットワークデバイスが作成および削除される割合が高く、Logical Domains Managerが自動的に割り当てられたMACアドレスを再利用しない場合、割り当てられるMACアドレスの数によって典型的な構成のDHCPサーバーがすぐに圧迫される可能性があります。

Logical Domains Managerは、論理ドメインまたはネットワークデバイスのMACアドレスを自動的に取得するように要求されると、以前に割り当てられた再利用可能なMACアドレスが存在するかどうかを確認するために、解放されたMACアドレスデータベースを最初に参照します。このデータベースに使用可能なMACアドレスが存在する場合、重複したMACアドレスの検出アルゴリズムが実行されます。以前に解放されたMACアドレスが、そのあと割り当てられていない場合は、そのMACアドレスが再利用され、データベースから削除されます。競合が検出された場合、そのアドレスは単にデータベースから削除されます。Logical Domains Managerは、データベース内の次のアドレスを試行するか、使用可能なアドレスがない場合は、新しいMACアドレスをランダムに選択します。

LDomsでのネットワークアダプタの使用

論理ドメイン環境のサービスドメイン内で動作する仮想スイッチサービスは、GLDv3準拠のネットワークアダプタと直接対話できます。GLDv3に準拠していないネットワークアダプタは、これらのシステムで使用できますが、仮想スイッチと直接対話することはできません。GLDv3に準拠していないネットワークアダプタを使用する方法については、114ページの「NATおよびルーティング用の仮想スイッチおよびサービスドメインの構成」を参照してください。

リンク集積体の使用法については、130ページの「仮想スイッチでのリンク集積体の使用」を参照してください。

▼ ネットワークアダプタが GLDv3 準拠かどうかを判別する

1. **Solaris OS** `dladm`**(1M)** コマンドを使用します。ここでは、たとえば、ネットワークデバイス名として `bge0` を指定します。

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

2. 出力結果の `type:` を確認します。
   - GLDv3 に準拠しているドライバの種類は、`non-vlan` または `vlan` です。
   - GLDv3 に準拠していないドライバの種類は、`legacy` です。

NAT およびルーティング用の仮想スイッチおよびサービスドメインの構成

仮想スイッチ (`vsw`) はレイヤー 2 スイッチで、サービスドメインでネットワークデバイスとしても使用できます。仮想スイッチは、さまざまな論理ドメインで仮想ネットワーク (`vnet`) デバイス間のスイッチとしてのみ動作するように構成できますが、物理デバイスを介してネットワークの外部に接続することはできません。このモードで、`vsw` をネットワークデバイスとして plumb し、サービスドメインで IP ルーティングを有効にすると、仮想ネットワークでサービスドメインをルーターとして使用して外部と通信することができます。このモードでの操作は、物理ネットワークアダプタが GLDv3 に準拠していない場合、ドメインが外部に接続できるようにするために非常に重要です。

この構成の利点は次のとおりです。

- 仮想スイッチは物理デバイスを直接使用する必要がなく、基本となるデバイスが GLDv3 に準拠していない場合でも外部と接続できます。
- この構成では、Solaris OS の IP ルーティングとフィルタリング機能を利用できます。

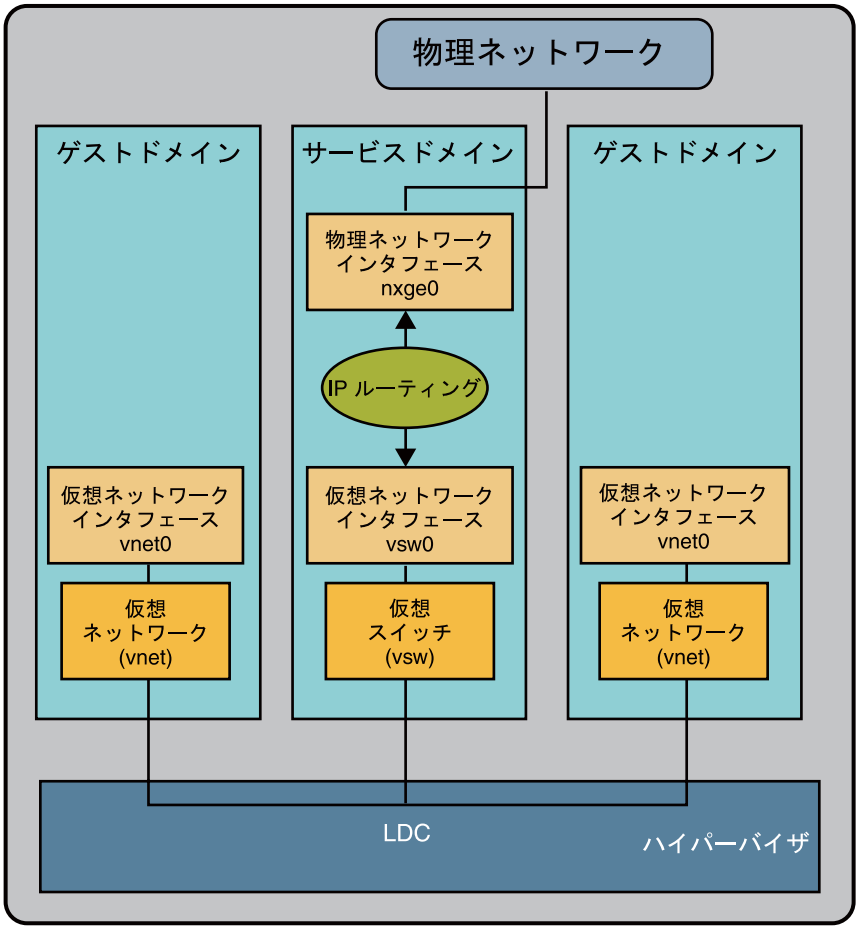

図 7–2 仮想ネットワークルーティング

▼ ドメインが外部に接続できるように仮想スイッチを設定する

1 物理デバイスを関連付けずに仮想スイッチを作成します。

アドレスを割り当てる場合は、仮想スイッチに一意の MAC アドレスが割り当てられるようにしてください。

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

2 ドメインによって使用される物理ネットワークデバイスに加えて、仮想スイッチをネットワークデバイスとして **plumb** します。

仮想スイッチの plumb の詳細は、53 ページの「仮想スイッチを主インタフェースとして構成する」 を参照してください。

3 必要に応じて、**DHCP** で仮想スイッチデバイスを構成します。

DHCP での仮想スイッチデバイスの構成については、53 ページの「仮想スイッチを主インタフェースとして構成する」 を参照してください。

4 必要に応じて、`/etc/dhcp.vsw` ファイルを作成します。

5 サービスドメインで **IP** ルーティングを構成し、すべてのドメインで必要なルーティングテーブルを設定します。

この実行方法については、『Solaris のシステム管理 (IP サービス)』の「IPv4 ネットワーク上でのパケット転送と経路制御」を参照してください。

論理ドメイン環境での IPMP の構成

Logical Domains 1.3 リリースでは、仮想ネットワークデバイスでのリンクベースの IPMP のサポートが導入されています。仮想ネットワークデバイスで IPMP グループを構成する場合は、リンクベースの検出を使用するようにグループを構成します。Logical Domains ソフトウェアの以前のバージョンを使用している場合、仮想ネットワークデバイスでプローブベースの検出のみを構成できます。

論理ドメインの IPMP グループへの仮想ネットワークデバイスの構成

次の図に、サービスドメインで個別の仮想スイッチインスタンス (`vsw0` および `vsw1`) に接続された 2 つの仮想ネットワーク (`vnet0` および `vnet1`) を示します。これらは、同様に、2 つの異なる物理インタフェース (`nxge0` および `nxge1`) を使用します。サービスドメインの物理リンクに障害が発生した場合、その物理デバイスにバインドされた仮想スイッチデバイスがリンクの障害を検出します。次に、仮想スイッチデバイスは、その仮想スイッチにバインドされた対応する仮想ネットワークデバイスに障害を伝播します。仮想ネットワークデバイスは、このリンクイベントの通知をゲスト `LDom_A` の IP 層に送信し、その結果、IPMP グループのもう一方の仮想ネットワークデバイスにフェイルオーバーします。

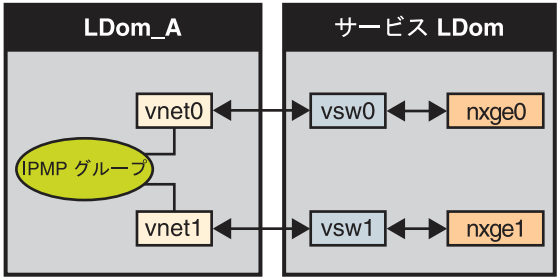

図 7–3 個別の仮想スイッチインスタンスに接続された 2 つの仮想ネットワーク

次の図に示すように、各仮想ネットワークデバイス (`vnet0` および `vnet1`) を異なるサービスドメインの仮想スイッチインスタンスに接続すると、論理ドメインでの信頼性をさらに高めることができます。この場合、物理ネットワークの障害に加えて、`LDom_A` が仮想ネットワークの障害を検出し、サービスドメインがクラッシュまたは停止したあとでフェイルオーバーを引き起こすことができます。

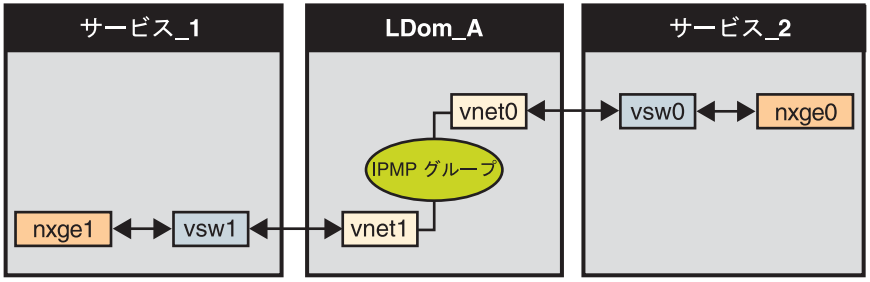

図 7–4 異なるサービスドメインに接続された各仮想ネットワークデバイス

IPMP グループの構成と使用法の詳細は、Solaris 10 の『Solaris のシステム管理 (IP サービス)』を参照してください。

サービスドメインでの IPMP の構成と使用

仮想スイッチインタフェースをグループに構成することで、サービスドメインで IPMP を構成できます。次の図に、2 つの異なる物理デバイスにバインドされた 2 つの仮想スイッチインスタンス (`vsw0` および `vsw1`) を示します。この場合、この 2 つの仮想スイッチインタフェースを plumb して IPMP グループに構成できます。物理リンクに障害が発生した場合、その物理デバイスにバインドされた仮想スイッチデバイスがリンクの障害を検出します。次に、仮想スイッチデバイスは、このリンクイベントの通知をサービスドメインの IP 層に送信し、その結果、IPMP グループのもう一方の仮想スイッチデバイスにフェイルオーバーします。

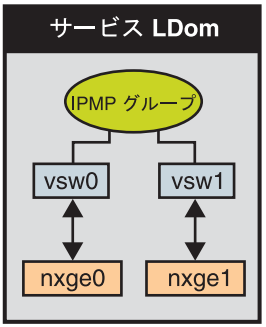

図 7–5　IPMP グループの一部として構成された 2 つの仮想スイッチインタフェース

Logical Domains 仮想ネットワークでのリンクベースの IPMP の使用

Logical Domains 1.3 では、仮想ネットワークおよび仮想スイッチデバイスがネットワークスタックへのリンクステータスの更新をサポートします。デフォルトでは、仮想ネットワークデバイスはその仮想リンク (仮想スイッチへの LDC) のステータスをレポートします。この設定はデフォルトで有効になり、追加構成手順を実行する必要はありません。

場合によっては、物理ネットワークのリンクステータスの変更を検出する必要があります。たとえば、物理デバイスが仮想スイッチに割り当てられている場合、仮想ネットワークデバイスからその仮想スイッチデバイスへのリンクが動作していても、サービスドメインから外部ネットワークへの物理ネットワークリンクは停止している可能性があります。このような場合、物理リンクステータスを取得して仮想ネットワークデバイスとそのスタックにレポートする必要がある可能性があります。

`linkprop=phys-state` オプションを使用すると、仮想ネットワークデバイスおよび仮想スイッチデバイスに対して物理リンクステータスの追跡を構成できます。このオプションを有効にすると、仮想デバイス (仮想ネットワークまたは仮想スイッチ) が、ドメインでインタフェースとして plumb されている間、物理リンクステータスに基づいてリンクステータスをレポートします。`dladm`、`ifconfig` などの、Solaris の標準ネットワーク管理コマンドを使用して、リンクステータスを確認できます。`dladm(1M)` および `ifconfig(1M)` マニュアルページを参照してください。また、リンクステータスは `/var/adm/messages` ファイルにも記録されます。

注 – 1 つの Logical Domains システムで、リンクステータスを認識しないものとリンクステータスを認識するものの両方の vnet および vsw ドライバを同時に実行できます。ただし、リンクベースの IPMP を構成する場合、リンクステータスを認識するドライバをインストールする必要があります。物理リンクステータスの更新を有効にする場合、vnet および vsw の両方のドライバを Solaris 10 10/09 OS にアップグレードして、Logical Domains Manager の Version 1.3 以上を実行します。

▼ 物理リンクステータスの更新を構成する

この手順では、仮想ネットワークデバイスで物理リンクステータスの更新を有効にする方法を示します。

同様の手順に従い、ldm add-vsw および ldm set-vsw コマンドに linkprop=phys-state オプションを指定することで、仮想スイッチデバイスで物理リンクステータスの更新を有効にすることもできます。

注 – linkprop=phys-state オプションは、仮想スイッチデバイス自体がインタフェースとして plumb されている場合にのみ使用する必要があります。linkprop=phys-state が指定され、物理リンクが停止している場合、仮想スイッチへの接続が有効であっても、仮想ネットワークデバイスはリンクステータスを停止状態とレポートします。この状況が発生するのは、Solaris OS は現在、仮想リンクステータスと物理リンクステータスなど、2 つの異なるリンクステータスをレポートするインタフェースを備えていないためです。

1 スーパーユーザーになるか、同等の役割を取得します。

役割には、承認および特権付きコマンドが含まれます。役割の詳細は、『Solaris のシステム管理 (セキュリティサービス)』の「RBAC の構成 (作業マップ)」 を参照してください。

2 仮想デバイスで物理リンクステータスの更新を有効にします。

仮想ネットワークデバイスで物理リンクステータスの更新を有効にするには、次の手順に従います。

- ldm add-vnet コマンド実行時に linkprop=phys-state を指定し、仮想ネットワークデバイスを作成します。

  linkprop=phys-state オプションを指定すると、仮想ネットワークデバイスが物理リンクステータスの更新を取得してスタックにレポートするように構成されます。

注 – linkprop=phys-state が指定され、物理リンクが停止している場合、仮想スイッチへの接続が有効であっても、仮想ネットワークデバイスはリンクステータスを down とレポートします。この状況が発生するのは、Solaris OS は現在、仮想リンクステータスと物理リンクステータスなど、2 つの異なるリンクステータスをレポートするインタフェースを備えていないためです。

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

次の例では、論理ドメイン ldom1 の primary-vsw0 に接続された vnet0 で物理リンクステータスの更新を有効にします。

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

- ldm set-vnet コマンド実行時に linkprop=phys-state を指定し、既存の仮想ネットワークデバイスを変更します。

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

次の例では、論理ドメイン ldom1 の vnet0 で物理リンクステータスの更新を有効にします。

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

物理リンクステータスの更新を無効にするには、ldm set-vnet コマンドを実行して linkprop= を指定します。

次の例では、論理ドメイン ldom1 の vnet0 で物理リンクステータスの更新を無効にします。

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

例 7–1 リンクベースの IPMP の構成

次の例は、物理リンクステータスの更新を有効にする方法と有効にしない方法の両方を使用してリンクベースの IPMP を構成する方法を示します。

- 次の例では、1 つのドメインで 2 つの仮想ネットワークデバイスを構成します。各仮想ネットワークデバイスは、リンクベースの IPMP を使用するためにサービスドメインの個別の仮想スイッチデバイスに接続されます。

注 – これらの仮想ネットワークデバイスでテストアドレスは構成されません。また、ldm add-vnet コマンドを使用してこれらの仮想ネットワークデバイスを作成する場合に、追加構成を実行する必要はありません。

次のコマンドは、仮想ネットワークデバイスをドメインに追加します。linkprop=phys-state が指定されていないため、仮想スイッチへのリンクのみでステータスの変更が監視されることに注意してください。

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

次のコマンドは、仮想ネットワークデバイスをゲストドメインで構成して IPMP グループに割り当てます。リンクベースの障害検出が使用されているためにこれらの仮想ネットワークデバイスでテストアドレスが構成されていないことに注意してください。

```
# 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
```

- 次の例では、1 つのドメインで 2 つの仮想ネットワークデバイスを構成します。各ドメインは、リンクベースの IPMP を使用するためにサービスドメインの個別の仮想スイッチデバイスに接続されます。また、仮想ネットワークデバイスは、物理リンクステータスの更新を取得するように構成されます。

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

注 – ドメインを正常にバインドするために、仮想スイッチに物理ネットワークデバイスを割り当てる必要があります。ドメインがすでにバインドされており、仮想スイッチに物理ネットワークデバイスが割り当てられていない場合、`ldm add-vnet` コマンドは失敗します。

次のコマンドは、仮想ネットワークデバイスを plumb して IPMP グループに割り当てます。

```
# 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
```

Logical Domains 1.3 より前のリリースでの IPMP の構成と使用

Logical Domains 1.3 より前のリリースでは、仮想スイッチおよび仮想ネットワークデバイスはリンク障害の検出を実行できません。それらのリリースでは、プローブベースの IPMP を使用してネットワーク障害の検出と復旧を設定できます。

ゲストドメインでの IPMP の構成

ゲストドメインの仮想ネットワークデバイスは、図 7–3 および図 7–4 に示す方法で IPMP グループに構成できます。唯一の相違点は、仮想ネットワークデバイスでテストアドレスを構成することでプローブベースの障害検出が使用されることです。プローブベースの IPMP の設定の詳細は、『Solaris のシステム管理 (IP サービス)』 を参照してください。

サービスドメインでの IPMP の構成

Logical Domains 1.3 より前のリリースでは、仮想スイッチデバイスは物理リンク障害の検出を実行できません。このような場合、サービスドメインの物理インタフェースを IPMP グループに構成することで、ネットワーク障害の検出と復旧を設定できます。これを行うには、物理ネットワークデバイスを割り当てずにサービスドメインの仮想スイッチを構成します。特に、`ldm add-vswitch` コマンドを使用して仮想スイッチを作成するときに、`net-dev` (`net-dev=`) プロパティーに値を指定しないでください。サービスドメインの仮想スイッチインタフェースを plumb して、サービスドメイン自体が IP ルーターとして機能するように構成します。IP ルーティングの設定については、Solaris 10 の『Solaris のシステム管理 (IP サービス)』 を参照してください。

いったん仮想スイッチが構成されると、仮想ネットワークから発生し外部のマシンに送信される予定のすべてのパケットは、物理デバイスを使用して直接送信されるのではなく、IP 層に送信されます。物理インタフェースに障害が発生した場合、IP 層は障害を検出し、自動的に二次インタフェースを使用してパケットをふたたび経路指定します。

物理インタフェースは直接 IPMP グループに構成されているため、グループは、リンクベースまたはプローブベースのいずれかの検出用に設定できます。次の図に、IPMP グループの一部として構成された 2 つのネットワークインタフェース (`nxge0` および `nxge1`) を示します。仮想スイッチインスタンス (`vsw0`) は、IP 層にパケットを送信するネットワークデバイスとして plumb されています。

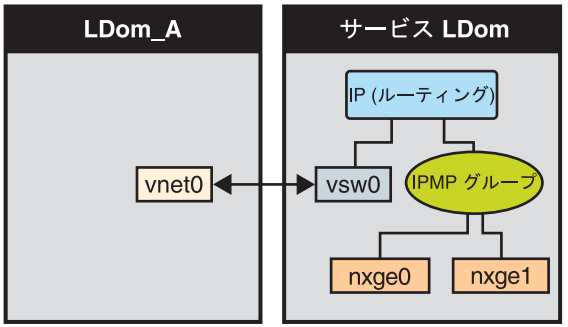

図 7–6 IPMP グループの一部として構成された 2 つのネットワークインタフェース

▼ プローブベースの IPMP 用のホストルートを構成する

注 – この手順は、ゲストドメインおよび 1.3 より前のリリースのみに適用されます。1.3 より前のリリースでは、プローブベースの IPMP のみがサポートされています。

ネットワーク内の IPMP インタフェースに対応するルーターに明示的なルートが構成されていない場合、IPMP プローブベースの検出を目的どおりに動作させるには、ターゲットシステムへの明示的なホストルートを 1 つ以上構成する必要があります。このようにしない場合、プローブ検出がネットワーク障害を検出できないことがあります。

- ホストルートを構成します。

```
# route add -host destination-IP gateway-IP -static
```

次に例を示します。

```
# route add -host 192.168.102.1 192.168.102.1 -static
```

詳細は、『Solaris のシステム管理 (IP サービス)』の「ターゲットシステムの構成」を参照してください。

Logical Domains ソフトウェアでの VLAN のタグ付けの使用

Solaris 10 10/08 OS および LDoms 1.1 ソフトウェアのリリース以降は、Logical Domains ネットワークインフラストラクチャーで 802.1Q VLAN のタグ付けがサポートされます。

注 – タグ付き VLAN は、以前のリリースの LDoms ネットワークコンポーネント用ではサポートされていません。

仮想スイッチ (vsw) および仮想ネットワーク (vnet) デバイスは、仮想ローカルエリアネットワーク (VLAN) 識別子 (ID) に基づいて Ethernet パケットのスイッチングをサポートし、Ethernet フレームの必要なタグ付けまたはタグなし処理を行います。

ゲストドメインの vnet デバイスには複数の VLAN インタフェースを作成できます。Solaris OS ifconfig(1M) コマンドを使用すると、ほかの物理ネットワークデバイスに VLAN インタフェースを構成する場合と同じ方法で、仮想ネットワークデバイスに VLAN インタフェースを作成できます。LDoms 環境では、この手順のほかに Logical Domains Manager CLI コマンドを使用して、対応する VLAN に vnet を割り当てる必要があります。Logical Domains Manager CLI コマンドの詳細は、ldm(1M) を参照してください。

同様に、サービスドメインの仮想スイッチデバイスに VLAN インタフェースを構成することができます。VLAN ID 2 ～ 4094 が有効です。VLAN ID 1 は default-vlan-id として予約されています。

ゲストドメインに vnet デバイスを作成する場合は、そのデバイスを必要な VLAN に割り当てる必要があります。それには、ldm add-vnet コマンドで pvid= 引数および vid= 引数を使用して、この vnet にポート VLAN ID および 0 個以上の VLAN ID を指定します。これによって、仮想スイッチは、LDoms ネットワークで複数の VLAN をサポートし、ネットワークで MAC アドレスと VLAN ID の両方を使用してパケットをスイッチングするように構成されます。

同様に、vsw デバイス自体が属することになる VLAN を、ネットワークインタフェースとして plumb するときに、ldm add-vsw コマンドで pvid= 引数および vid= 引数を使用して、vsw デバイス内に構成する必要があります。

デバイスが属する VLAN は、ldm set-vnet または ldm set-vsw コマンドを使用して変更できます。

ポート VLAN ID (PVID)

PVID は、仮想ネットワークデバイスをメンバーにする必要のある VLAN を、タグなしモードで示します。この場合、PVID で指定した VLAN の vnet デバイスのために必要なフレームのタグ付けまたはタグなし処理は、vsw デバイスによって行われます。仮想ネットワークからのタグなしのアウトバウンドフレームは、仮想スイッチによって PVID でタグ付けされます。この PVID でタグ付けされたインバウンドフレームは、仮想スイッチによってタグが削除されてから、vnet デバイスに送信されます。このため、PVID を vnet に暗黙に割り当てることは、仮想スイッチの対応する仮想ネットワークポートが、PVID で指定された VLAN に対してタグなしとしてマークされることを意味します。vnet デバイスに設定できる PVID は 1 つだけです。

対応する仮想ネットワークインタフェースは、VLAN ID なしで ifconfig(1M) コマンドを使用して、そのデバイスインスタンスだけを使用して構成した場合、仮想ネットワークの PVID によって指定された VLAN に暗黙に割り当てられます。

たとえば、次のコマンドを使用して vnet インスタンス 0 を plumb する場合に、この vnet の pvid= 引数が 10 として指定されているときは、vnet0 インタフェースが VLAN 10 に属するように暗黙に割り当てられます。

```
# ifconfig vnet0 plumb
```

VLAN ID (VID)

VID は、仮想ネットワークデバイスまたは仮想スイッチをメンバーにする必要のある VLAN を、タグ付きモードで示します。仮想ネットワークデバイスは、その VID で指定されている VLAN でタグ付きフレームを送受信します。仮想スイッチは、仮想ネットワークデバイスと外部ネットワークの間で、指定の VID でタグ付けされたフレームを通過させます。

▼ VLAN を仮想スイッチおよび仮想ネットワークデバイスに割り当てる

1 **仮想スイッチ (vsw) を 2 つの VLAN に割り当てます。**

たとえば、VLAN 21 をタグなし、VLAN 20 をタグ付きとして構成します。仮想ネットワーク (vnet) を 3 つの VLAN に割り当てます。VLAN 20 をタグなし、VLAN 21 および VLAN 22 をタグ付きとして構成します。

```
# 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 **VLAN インタフェースを plumb します。**

この例では、ドメイン内のこれらのデバイスのインスタンス番号は 0 で、VLAN はこれらのサブネットに対応づけられていることを前提としています。

VLAN	サブネット
20	192.168.1.0 (ネットマスク: 255.255.255.0)
21	192.168.2.0 (ネットマスク: 255.255.255.0)

VLAN	サブネット
22	192.168.3.0 (ネットマスク: 255.255.255.0)

a. サービス (`primary`) ドメインで **VLAN** インタフェースを **plumb** します。

```
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. ゲスト (`ldom1`) ドメインで **VLAN** インタフェースを **plumb** します。

```
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
```

Solaris OS で VLAN インタフェースを構成する方法の詳細は、『Solaris のシステム管理 (IP サービス)』の「仮想ローカルエリアネットワークの管理」を参照してください。

▼ インストールサーバーが **VLAN** に存在する場合にゲストドメインをインストールする

ネットワークを介してゲストドメインをインストールしており (JumpStart)、インストールサーバーが VLAN に存在する場合は注意してください。インストールサーバーに関連付けられた VLAN ID を、仮想ネットワークデバイスの PVID として指定します。その仮想ネットワークデバイスにタグ付き VLAN (`vid`) を構成しないでください。OBP は VLAN を認識せず、VLAN のタグ付きのネットワークパケットを処理できないため、このようにする必要があります。仮想スイッチは、ネットワークインストールの実行中、ゲストドメインから送受信されるパケットのタグ付きおよびタグなし処理を行います。ネットワークインストールが完了して Solaris OS が起動したら、仮想ネットワークデバイスがその VLAN でタグ付けされるように構成できます。その後、その仮想ネットワークデバイスをタグ付きモードでほかの VLAN に追加できます。

JumpStart を使用したゲストドメインのインストールについては、61 ページの「ゲストドメインの JumpStart を実行する」 を参照してください。

1 最初にネットワークデバイスをタグなしモードで構成します。

たとえば、インストールサーバーが VLAN 21 にある場合、最初に仮想ネットワークを次のように構成します。

```
primary# ldm add-vnet pvid=21 vnet01 primary-vsw0 ldom1
```

2 インストールが完了して **Solaris OS** が起動したら、仮想ネットワークをタグ付きモードで構成します。

```
primary# ldm set-vnet pvid= vid=21, 22, 23 vnet01 primary-vsw0 ldom1
```

NIU ハイブリッド I/O の使用

仮想 I/O フレームワークは、機能およびパフォーマンスを向上させるために、「ハイブリッド」I/O モデルを実装しています。ハイブリッド I/O モデルでは、ダイレクト I/O および仮想化 I/O を組み合わせることで、仮想マシンへの柔軟な I/O リソース配備が可能になっています。これは、仮想マシンに対してダイレクト I/O の機能が十分に提供されない場合、または仮想マシンが持続的にあるいは一貫してダイレクト I/O を利用できない場合に特に便利です。この状況は、リソースの可用性または仮想マシンの移行が原因で発生する可能性があります。ハイブリッド I/O アーキテクチャーは、Sun UltraSPARC T2 ベースのプラットフォームでのチップに統合されたネットワーク I/O インタフェースであるネットワークインタフェースユニット (NIU) に適しています。これにより、ダイレクトメモリーアクセス (DMA) リソースを仮想ネットワークデバイスに動的に割り当てることができ、ドメイン内のアプリケーションのパフォーマンスが安定します。

NIU ハイブリッド I/O は、Sun UltraSPARC T2 ベースのプラットフォームで使用できます。この機能は、仮想ネットワーク (vnet) デバイスに提供されるオプションのハイブリッドモードによって有効になります。このモードでは、DMA ハードウェアリソースが、パフォーマンスを向上させるために、ゲストドメインの vnet デバイスに貸し出されます。ハイブリッドモードでは、ゲストドメインの vnet デバイスは、この DMA ハードウェアリソースを使用して、外部ネットワークとゲストドメインの間で、ユニキャストトラフィックを直接送受信することができます。同じシステム内のほかのゲストドメインへのブロードキャストトラフィック、マルチキャストトラフィック、およびユニキャストトラフィックは、仮想 I/O 通信機構を使用して引き続き送信されます。

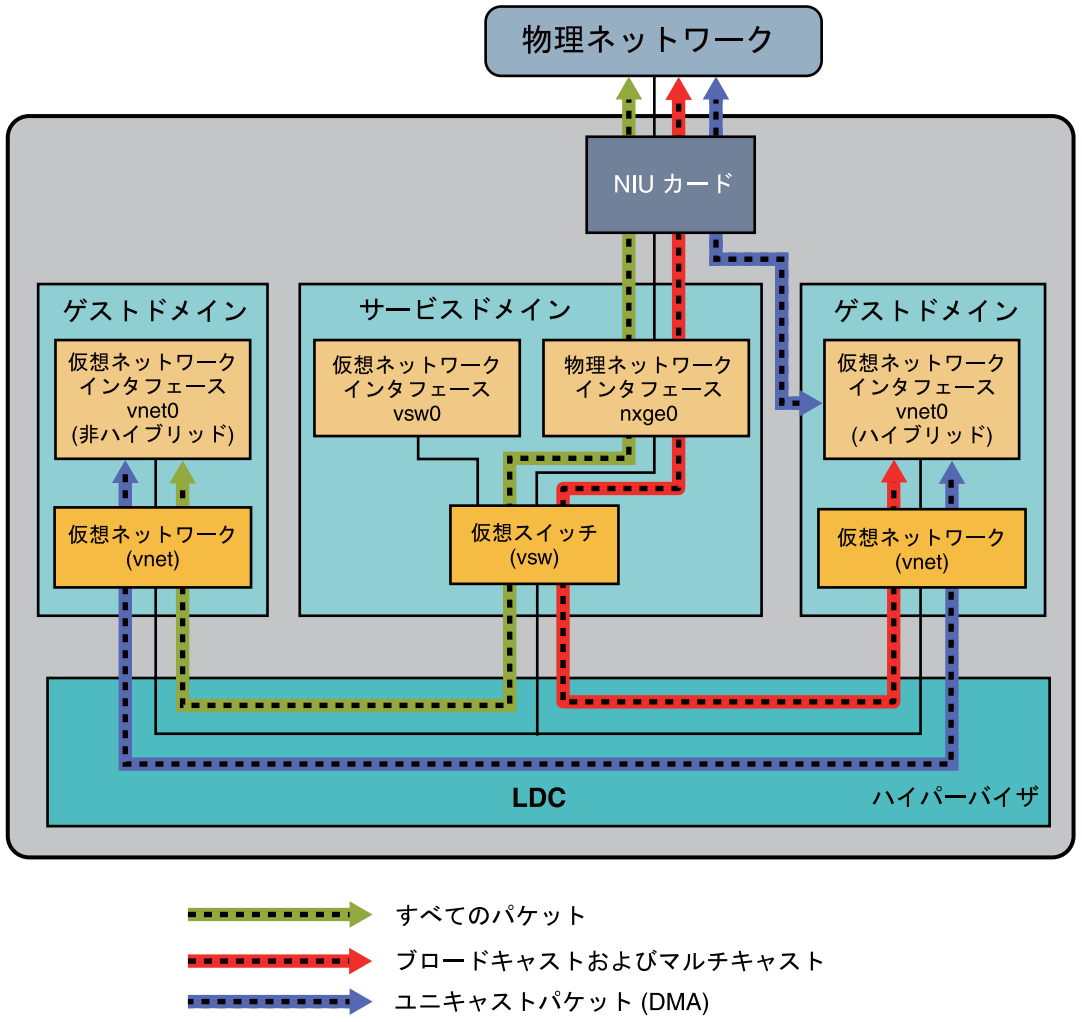

図 7–7 ハイブリッド仮想ネットワーク接続

ハイブリッドモードは、NIU ネットワークデバイスを使用するように構成された仮想スイッチ (vsw) に関連付けられた vnet デバイスだけに適用されます。共有可能な DMA ハードウェアリソースには制限があるため、DMA ハードウェアリソースの割り当てを受けられるのは、一度に、1 つの vsw あたり最大 3 つの vnet デバイスのみです。4 つ以上の vnet デバイスでハイブリッドモードを有効にすると、割り当ては先着順に行われます。1 つのシステムに 2 つの NIU ネットワークデバイスがあるため、DMA ハードウェアリソースが割り当てられている 2 つの異なる仮想スイッチで、合計 6 つの vnet デバイスが存在できます。

この機能を使用する場合の注意事項は、次のとおりです。

- vnet デバイスのハイブリッドモードオプションは、提案のみとして扱われます。つまり、DMA リソースが割り当てられるのは、DMA リソースが利用可能で、デバイスがこれらを使用できる場合だけです。
- Logical Domains Manager CLI コマンドは、ハイブリッドモードオプションを検証しません。つまり、どの vnet にも、いくつの vnet デバイスにもハイブリッドモードを設定することができます。
- ゲストドメインおよびサービスドメインでは、Solaris 10 10/08 以上の OS を実行する必要があります。
- DMA ハードウェアリソースの貸し出しを受けられるのは、一度に、1 つの vsw あたり最大 3 つの vnet デバイスのみです。2 つの NIU ネットワークデバイスがあるため、DMA ハードウェアリソースの貸し出しを受けられるのは合計 6 つの vnet デバイスです。

注 – 1 つの vsw あたり 3 つの vnet デバイスのみにハイブリッドモードを設定して、DMA ハードウェアリソースが確実に割り当てられるようにしてください。

- デフォルトでは、vnet デバイスのハイブリッドモードは無効になっています。Logical Domains Manager CLI コマンドを使用して明示的に有効にする必要があります。130 ページの「ハイブリッドモードを有効にする」 を参照してください。

  詳細は、ldm(1M) マニュアルページを参照してください。
- ゲストドメインがアクティブの間、ハイブリッドモードオプションを動的に変更することはできません。
- DMA ハードウェアリソースが割り当てられるのは、ゲストドメインで plumb されている vnet デバイスがアクティブの場合のみです。
- Sun x8 Express 1/10G Ethernet アダプタ (nxge) のドライバは NIU カードで使用されていますが、同じドライバは、ほかの 10 ギガビットネットワークカードでも使用されています。ただし、NUI ハイブリッド I/O 機能は、NIU ネットワークデバイスのみで利用可能です。

▼ NIU ネットワークデバイスで仮想スイッチを構成する

- たとえば、NIU ネットワークデバイスを使用して仮想スイッチを構成するには、次の手順を実行します。

  a. NIU ネットワークデバイスを調べます。

  ```
  # grep nxge /etc/path_to_inst
  "/niu@80/network@0" 0 "nxge"
  "/niu@80/network@1" 1 "nxge"
  ```

  b. 仮想スイッチを構成します。

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

▼ ハイブリッドモードを有効にする

- たとえば、作成中に vnet デバイスのハイブリッドモードを有効にします。

  ```
  # ldm add-vnet mode=hybrid vnet01 primary-vsw0 ldom01
  ```

▼ ハイブリッドモードを無効にする

- たとえば、vnet デバイスのハイブリッドモードを無効にします。

  ```
  # ldm set-vnet mode= vnet01 ldom01
  ```

仮想スイッチでのリンク集積体の使用

Solaris 10 10/08 OS および Logical Domains 1.1 ソフトウェアのリリース以降は、仮想スイッチでリンク集積体を使用するように構成できます。リンク集積体は、物理ネットワークに接続するための仮想スイッチのネットワークデバイスとして使用します。この構成を使用すると、仮想スイッチで IEEE 802.3ad Link Aggregation Standard によって提供される機能を利用できます。この機能には、帯域幅の増加、負荷分散、フェイルオーバーなどが含まれます。リンク集積体を構成する方法の詳細は、『Solaris のシステム管理 (IP サービス)』を参照してください。

リンク集積体を作成したら、そのリンク集積体を仮想スイッチに割り当てることができます。この割り当て方法は、仮想スイッチへの物理ネットワークデバイスの割り当てに似ています。`ldm add-vswitch` または `ldm set-vswitch` コマンドを使用して `net-dev` プロパティーを設定します。

リンク集積体を仮想スイッチに割り当てると、物理ネットワークに対して送受信されるトラフィックは集積体を通過してます。必要な負荷分散またはフェイルオーバーは、ベースとなる集積体のフレームワークによって透過的に処理されます。リンク集積体は、ゲストドメイン上の仮想ネットワーク (vnet) デバイスに対して、および集積体を使用する仮想スイッチにバインドされた仮想ネットワークデバイスに対して、完全に透過的です。

注 – 仮想ネットワークデバイス (vnet および vsw) をリンク集積体にグループ化することはできません。

サービスドメインでリンク集積体を使うように構成された仮想スイッチを、plumb して使用できます。53 ページの「仮想スイッチを主インタフェースとして構成する」 を参照してください。

次の図に、物理インタフェース nxge0 および nxge1 上で集積体 aggr1 を使用するように構成された仮想スイッチを示します。

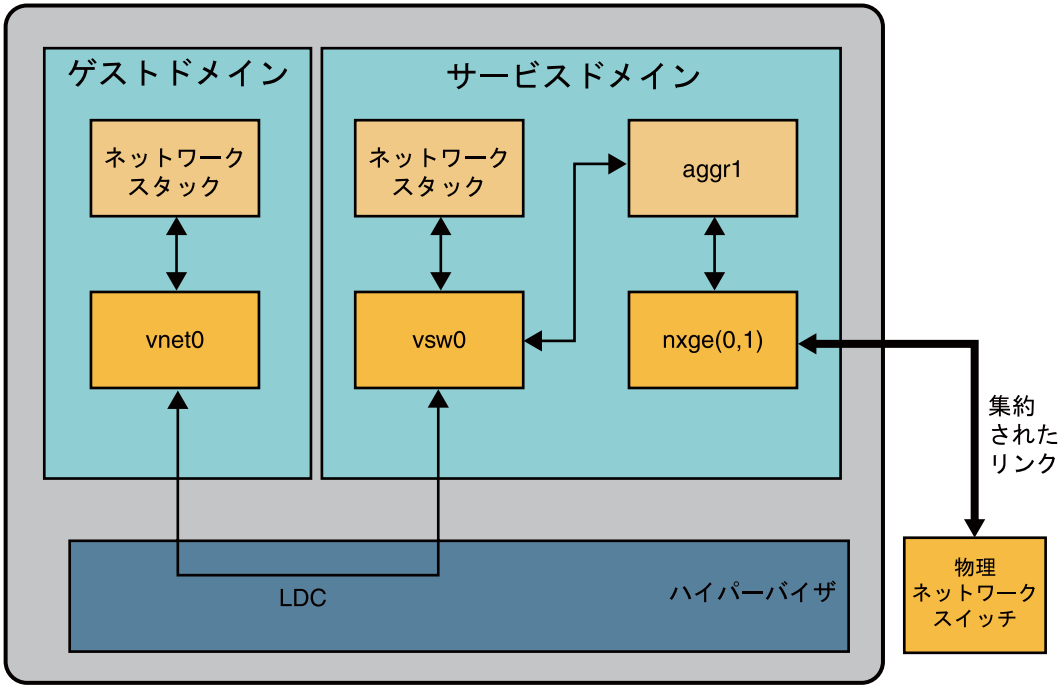

図 7–8 リンク集積体を使用する仮想スイッチの構成

ジャンボフレームの構成

Logical Domains の仮想スイッチ (vsw) および仮想ネットワーク (vnet) デバイスで、1500 バイトを超えるペイロードサイズの Ethernet フレームをサポートできるようになりました。この変更によって、これらのドライバのネットワークスループットが向上します。

▼ ジャンボフレームを使用するように仮想ネットワークおよび仮想スイッチデバイスを構成する

ジャンボフレームを有効にするには、仮想スイッチデバイスの最大転送単位 (MTU) を指定します。このような場合、仮想スイッチデバイスとその仮想スイッチデバイスにバインドされているすべての仮想ネットワークデバイスで、指定した MTU 値が使用されます。

特定の状況では、仮想ネットワークデバイス上で MTU 値を直接指定できます。これは、仮想ネットワークデバイスで必要な MTU 値が仮想スイッチによってサポートされる MTU 値よりも小さい場合に行うことがあります。

注 – Solaris 10 5/09 OS では、物理デバイスの MTU は仮想スイッチの MTU と一致するように構成する必要があります。特定のドライバの構成については、Solaris リファレンスマニュアルの 7D 節にある、そのドライバに対応するマニュアルページを参照してください。たとえば、nxge ドライバの情報については、nxge(7D) マニュアルページを参照してください。

OpenSolaris™ OS では、vsw ドライバは物理デバイスの MTU を自動的に構成します。したがって、追加構成は必要はありません。

---

1. 制御ドメインにログインします。

2. スーパーユーザーになるか、同等の役割を取得します。

   役割には、承認および特権付きコマンドが含まれます。役割の詳細は、『Solaris のシステム管理 (セキュリティサービス)』の「RBAC の構成 (作業マップ)」 を参照してください。

3. 仮想ネットワークで使用する **MTU** の値を決定します。

   1500 ～ 16000 バイトの MTU 値を指定できます。指定する MTU は、仮想スイッチに割り当てられた物理ネットワークデバイスの MTU と一致する必要があります。

4. 仮想スイッチデバイスまたは仮想ネットワークデバイスの **MTU** 値を指定します。

   次のいずれかを実行します。

- MTU を mtu プロパティーの値として指定することで、サービスドメインの新しい仮想スイッチデバイスでジャンボフレームを有効にします。

  ```
  # ldm add-vsw mtu=value vswitch-name ldom
  ```

  このコマンドは、仮想スイッチの構成に加えて、この仮想スイッチにバインドされる各仮想ネットワークデバイスの MTU 値を更新します。
- MTU を mtu プロパティーの値として指定することで、サービスドメインの既存の仮想スイッチデバイスでジャンボフレームを有効にします。

  ```
  # ldm set-vsw mtu=value vswitch-name
  ```

  このコマンドは、仮想スイッチの構成に加えて、この仮想スイッチにバインドされる各仮想ネットワークデバイスの MTU 値を更新します。

まれに、ldm add-vnet または ldm set-vnet コマンドを使用して、仮想スイッチの MTU 値と異なる MTU 値を仮想ネットワークデバイスに指定する必要がある場合があります。たとえば、VLAN を仮想ネットワークデバイス上で構成し、VLAN の MTU の最大値が仮想スイッチの MTU 値よりも小さい場合、仮想ネットワークデバイスの MTU 値を変更する場合があります。デフォルトの MTU 値のみが使用されているドメインでは、ジャンボフレームをサポートしている vnet ドライバは必要ない場合があります。ただし、ジャンボフレームを使用する仮想スイッチにバインドされた仮想ネットワークデバイスがドメインに存在する場合、vnet ドライバがジャンボフレームをサポートしていることを確認してください。

ldm set-vnet コマンドを使用して仮想ネットワークデバイスで mtu 値を指定する場合、あとで仮想スイッチデバイスの MTU 値が更新されても、仮想ネットワークデバイスには更新値は伝播されません。仮想ネットワークデバイスを再度有効にして仮想スイッチデバイスから MTU 値を取得するには、次のコマンドを実行します。

```
# ldm set-vnet mtu= vnet-name ldom
```

仮想ネットワークデバイスでジャンボフレームを有効にすると、その仮想ネットワークデバイスに割り当てられているハイブリッド I/O リソースでもジャンボフレームが自動的に有効になります。

制御ドメインでは、Logical Domains Manager が、ldm set-vsw および ldm set-vnet コマンドによって設定された MTU 値を遅延再構成処理として更新します。制御ドメイン以外のドメインの MTU を更新するには、ドメインを停止してから ldm set-vsw または ldm set-vnet コマンドを実行して MTU 値を変更する必要があります。

注 – OpenSolaris 2009.06 の dladm set-linkprop コマンドは、Logical Domains 仮想ネットワークデバイスの MTU 値の変更には使用できません。

例 7–2 仮想スイッチおよび仮想ネットワークデバイスでのジャンボフレームの構成

- 次の例に、MTU 値が 9000 の新しい仮想スイッチデバイスを追加する方法を示します。この MTU 値は、仮想スイッチデバイスからすべてのクライアントの仮想ネットワークデバイスに伝播されます。

  まず、`ldm add-vsw` コマンドによって、仮想スイッチデバイス `primary-vsw0` を MTU 値 9000 で作成します。ネットワークデバイス `nxge 0` のインスタンス 0 は、`net-dev` プロパティーの値として指定されています。

  ```
  # ldm add-vsw net-dev=nxge0 mtu=9000 primary-vsw0 primary
  ```

  次に、`ldm add-vnet` コマンドによって、クライアントの仮想ネットワークデバイスをこの仮想スイッチ `primary-vsw0` に追加します。仮想ネットワークデバイスの MTU は、バインドされている仮想スイッチから暗黙に割り当てられます。そのため、`ldm add-vnet` コマンドで `mtu` プロパティーの値を指定する必要はありません。

  ```
  # ldm add-vnet vnet01 primary-vsw0 ldom1
  ```

  `ifconfig` コマンドによって、サービスドメイン `primary` の仮想スイッチインタフェースを plumb します。`ifconfig vsw0` コマンドの出力には、`mtu` プロパティーの値が 9000 であることが示されます。

  ```
  # 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` コマンドによって、ゲストドメイン `ldom1` の仮想ネットワークインタフェースを plumbします。`ifconfig vnet0` コマンドの出力には、`mtu` プロパティーの値が 9000 であることが示されます。

  ```
  # 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
  ```

- 次の例に、`ifconfig` コマンドを使用してインタフェースの MTU を 4000 に変更する方法を示します。

  インタフェースの MTU は、Logical Domains Manager によってデバイスに割り当てられた MTU よりも小さい値にのみ変更できます。この方法は、VLAN が構成されていて各 VLAN インタフェースに異なる MTU が必要なときに便利です。

```
# 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
```

ジャンボフレームに対応していない旧バージョンの vnet および vsw ドライバとの互換性

ジャンボフレームをサポートしているドライバとジャンボフレームをサポートしていないドライバを、同じシステム上で相互運用できます。この相互運用は、仮想スイッチを作成するときにジャンボフレームのサポートが有効になっていない場合にかぎり可能です。

注 – 仮想スイッチに関連付けられたゲストドメインまたはサービスドメインがジャンボフレームをサポートしている Logical Domains ドライバを使用していない場合、`mtu` プロパティーを設定しないでください。

ジャンボフレームを有効にするには、仮想スイッチの `mtu` プロパティーをデフォルト値の 1500 から変更します。この場合、旧バージョンのドライバは `mtu` 設定を無視し、デフォルト値を引き続き使用します。`ldm list` の出力には、デフォルト値ではなく、指定した MTU 値が示されます。デフォルトの MTU よりも大きいフレームはそれらのデバイスには送られず、新しいドライバによって破棄されます。この場合、旧ドライバを使用し続けているゲストがあると、一貫性のないネットワーク動作につながる場合があります。これは、クライアントゲストドメインおよびサービスドメインの両方に当てはまります。

そのため、ジャンボフレームが有効な場合は、Logical Domains ネットワークのすべての仮想デバイスをアップグレードし、ジャンボフレームをサポートしている新しいドライバが使用されるようにしてください。また、ジャンボフレームを構成できるように、Logical Domains Manager 1.2 以上にアップグレードしてください。

◆ ◆ ◆ 第 8 章

8 論理ドメインの移行

この章では、今回のリリースの LogicalDomains1.3 ソフトウェア以降で、ホストマシン間で論理ドメインを移行する方法について説明します。

この章の内容は次のとおりです。

論理ドメインの移行の概要

論理ドメインの移行を行うと、ホストマシン間で論理ドメインを移行できます。移行が開始されるホストはソースマシン、ドメインの移行先のホストはターゲットマシンと呼ばれます。同様に、移行が開始されてから移行が進行中の間、移行されるドメインはソースドメイン、ターゲットマシン上に作成されるドメインのシェルはターゲットドメインと呼ばれます。

移行処理の概要

ソースマシン上の Logical Domains Manager はドメインの移行要求を受け入れ、ターゲットマシン上で動作している Logical Domains Manager とのセキュリティー保護されたネットワーク接続を確立します。この接続が確立されると、移行が行われます。移行自体は、複数のフェーズに分解できます。

フェーズ **1:** ターゲットホストで動作している Logical Domains Manager との接続後、ソースマシンおよびソースドメインに関する情報がターゲットホストに転送されます。この情報を使用して、移行が可能かどうかを判断する一連のチェックが実行されます。チェックは、ソースドメインの状態によって異なります。たとえば、ソースドメインがアクティブになっている場合と、ドメインがバインドされているかアクティブでない場合では、実行される一連のチェックが異なります。

フェーズ **2:** フェーズ 1 のすべてのチェックに合格すると、ソースマシンおよびターゲットマシンで移行の準備が行われます。ソースドメインがアクティブな場合は、この準備に CPU の数を 1 つに縮小する処理と、ドメインの一時停止が含まれます。ターゲットマシンでは、ソースドメインを受け入れるためにドメインが作成されます。

フェーズ **3:** アクティブなドメインの場合、次のこのフェーズでは、ドメインのすべての実行時の状態情報がターゲットに転送されます。この情報は、ハイパーバイザから取得されます。ターゲットで、状態情報がハイパーバイザにインストールされます。

フェーズ **4:** ハンドオフが行われます。すべての状態情報が転送されたあと、ソースがアクティブな場合はターゲットドメインが実行を再開するときにハンドオフが行われ、ソースドメインが削除されます。この時点で、ターゲットドメインは唯一の動作中のドメインになります。

ソフトウェアの互換性

移行が行われるためには、ソースマシンとターゲットマシンの両方で互換性のあるソフトウェアが動作している必要があります。

- ソースマシンとターゲットマシンの両方のハイパーバイザに、ドメインの移行をサポートするファームウェアが必要です。

  次のエラーが発生した場合、ソースマシンまたはターゲットマシンのいずれかのシステムファームウェアのバージョンが適切ではありません。

  ```
  System Firmware version on <downrev machine> does not support Domain Migration
  Domain Migration of LDom <source domain> failed
  ```

- 互換性のあるバージョンの Logical Domains Manager が両方のマシンで動作している必要があります。

注 – 移行機能は、Logical Domains 1.1 ソフトウェアおよび対応するファームウェアではじめてリリースされました。プラットフォームの最新のファームウェアについては、『Logical Domains 1.3 リリースノート』を参照してください。

移行処理の認証

移行処理は 2 つのマシンで実行されるため、ユーザーはソースホストとターゲットホストの両方で認証される必要があります。特に、スーパーユーザー以外のユーザーは、両方のマシンで `file_dac_read` 特権と `file_dac_search` 特権を持ち、`solaris.ldoms.read` 認証と `solaris.ldoms.write` 認証を受ける必要があります。46 ページの「ドメインの移行に必要な特権の追加」を参照してください。

移行に `ldm` コマンド行インタフェースを使用すると、ターゲットホストでの認証に任意の代替ユーザー名を指定できます。この代替ユーザー名を指定しない場合、移行コマンドを実行するユーザーの名前が使用されます。どちらの場合にも、`-p` オプションを使用して自動的な移行を開始しないかぎり、ターゲットマシンのパスワードの入力を求めるプロンプトが表示されます。

アクティブなドメインの移行

LogicalDomains1.3 ソフトウェアを使用してアクティブなドメインの移行を行うには、ソース論理ドメイン、ソースマシン、およびターゲットマシンに特定の一連の要件および制限が課せられます。以降の節では、各リソースタイプに対するこれらの要件および制限について説明します。

注 – ソースシステムおよびターゲットシステムの `primary` ドメインに暗号化装置が割り当てられていると、移行処理が高速になります。Logical Domains 1.3 では、ソースシステムとターゲットシステムの両方の `primary` ドメインに仮想 CPU を追加することで、移行を高速に実行できます。

アクティブなドメインの CPU の移行

次に、移行を実行する場合の CPU に対する要件および制限を示します。

- ソースマシンおよびターゲットマシンには、同じ周波数で動作する同じタイプのプロセッサが搭載されている必要があります。
- ターゲットマシンには、ドメインによって使用されるストランドの数に対応できる十分な空きストランドが存在する必要があります。また、移行されるドメインにはフルコアが割り当てられている必要があります。ソースのストランドの数がフルコアより少ない場合、移行されたドメインの再起動後までドメインに追加のストランドを使用することはできません。
- 移行後、ターゲットドメインが再起動されるまで、ターゲットドメインでの CPU の動的再構成 (DR) は無効になります。再起動が完了すると、そのドメインでの CPU の DR が可能になります。
- 移行前にドメインが 1 つのストランドに縮小できるように、ソースドメインのストランドを 1 つのみにするか、またはゲスト OS で CPU の DR をサポートしている必要があります。ゲストドメインが CPU の DR による削除が失敗する状態の場合、移行の試みも失敗することがあります。たとえば、ゲストドメイン内の CPU にバインドされた処理、またはソース論理ドメインに構成されたプロセッサセットによって、移行処理が失敗する可能性があります。

アクティブなドメインのメモリーの移行

ターゲットマシン上に、ソースドメインの移行に対応できる十分な空きメモリーが存在する必要があります。さらに、移行が終了するまで次に示すいくつかのプロパティーが維持される必要があります。

- 同じ数、同じサイズのメモリーブロックを作成できる必要があります。
- メモリーブロックの物理アドレスが一致する必要はありませんが、移行が終了するまで同じ実アドレスが維持される必要があります。

ターゲットマシンには、ソースドメインの移行に対応できる十分な空きメモリーが存在する必要があります。また、ターゲットマシンの使用可能メモリーのレイアウトとソースドメインのメモリーのレイアウトに互換性がある必要があります。互換性がないと、移行は失敗します。

特に、ターゲットマシンのメモリーが複数の小さいアドレス範囲に分割されているのに、ソースドメインには単一の大きいアドレス範囲が必要な場合、移行は失敗します。次の例は、この場合について示したものです。ターゲットドメインの 2 つのメモリーブロックに、2G バイトの空きメモリーがあるとします。

```
# ldm list-devices memory
MEMORY
    PA                   SIZE
```

```
    0x108000000          1G
    0x188000000          1G
```

ソースドメイン ldg-src にも 2G バイトの空きメモリーがありますが、これは単一のメモリーブロックに配置されています。

```
# ldm list -o memory ldg-src
NAME
ldg-src

MEMORY
    RA               PA               SIZE
    0x8000000        0x208000000      2G
```

このようなメモリーレイアウトの場合、移行は失敗します。

```
# 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
```

アクティブなドメインの物理 I/O デバイスの移行

物理デバイスが関連付けられている仮想デバイスは移行できます。ただし、物理デバイスに直接アクセスするドメインは移行できません。たとえば、I/O ドメインは移行できません。

アクティブなドメインの仮想 I/O デバイスの移行

ソースドメインが使用するすべての仮想 I/O (VIO) サービスが、ターゲットマシン上で使用可能である必要があります。つまり、次に示す状態になっている必要があります。

- ソース論理ドメインで使用されている各論理ボリュームは、ターゲットホスト上でも使用可能で、同じストレージを参照している必要があります。

注意 – ソースによって起動デバイスとして使用されている論理ボリュームがターゲット上に存在するにもかかわらず、同じストレージを参照していない場合、移行は正常に実行されたように見えますが、マシンから起動デバイスにアクセスできないため、このマシンは使用できません。ドメインを停止し、構成の問題を修正したあとで、ドメインを再起動する必要があります。この操作を行わない場合、ドメインが矛盾した状態のままになる可能性があります。

- ソースドメインの各仮想ネットワークデバイスに対して、ターゲットホスト上に仮想ネットワークスイッチが存在し、ソースホスト上でそのデバイスが接続されている仮想ネットワークスイッチと同じ名前が指定されている必要があります。

  たとえば、ソースドメインの `vnet0` が `switch-y` という名前の仮想スイッチサービスに接続されていた場合、ターゲットホスト上に `switch-y` という名前の仮想スイッチサービスを提供する論理ドメインが存在する必要があります。

  注 – これらのスイッチが同じネットワークに接続されていなくても移行は実行されますが、スイッチが同じネットワークに接続されていない場合、移行されたドメインでネットワークの問題が発生する可能性があります。

  ソースドメインによって使用されていた、自動的に割り当てられる範囲内の MAC アドレスは、ターゲットホストで使用可能である必要があります。
- 仮想コンソール端末集配信装置 (`vcc`) サービスがターゲットホスト上に存在し、1 つ以上のポートが空いている必要があります。移行時には明示的なコンソール制約は無視されます。ターゲットドメイン名をコンソールグループとして使用し、制御ドメインの最初の `vcc` デバイスで使用可能なポートを使用して、ターゲットドメインのコンソールが作成されます。デフォルトのグループ名と競合する場合、移行は失敗します。

アクティブなドメインの NIU ハイブリッド I/O の移行

NIU ハイブリッド I/O リソースを使用するドメインを移行できます。NIU ハイブリッド I/O リソースを指定する制約は、論理ドメインの必須要件ではありません。使用可能な NIU リソースが存在しないマシンにこのようなドメインを移行した場合、制約は維持されますが、この制約が満たされることはありません。

アクティブなドメインの暗号化装置の移行

Logical Domains 1.3 では、暗号化装置をバインドしたゲストドメインが暗号化装置の動的再構成 (DR) をサポートしているオペレーティングシステムを実行している場合、そのゲストドメインを移行できます。

次の Solaris OS バージョンでは、暗号化装置の DR がサポートされています。

- Solaris 10 10/09 OS 以上
- OpenSolaris 2009.06 OS 以上
- Solaris 10 5/08 OS とパッチ ID 142245-01 以上

移行の開始時点で、Logical Domains Manager は、ソースドメインが暗号化装置の DR をサポートしているかどうかを判断します。サポートしている場合、Logical Domains Manager はドメインからの暗号化装置の削除を試行します。移行の完了後、移行したドメインに暗号化装置が再度追加されます。

注 – ターゲットマシンで暗号化装置の制約を満たすことができない場合でも、移行処理は正常に完了する場合があります。このような場合、ドメインの暗号化装置の数が移行処理前よりも減少する可能性があります。

アクティブなドメインの遅延再構成

ソースホストまたはターゲットホスト上でアクティブな遅延再構成処理が実行されている場合、移行を開始できません。移行の進行中、遅延再構成処理はブロックされます。

ほかのドメインの操作

マシンでの移行が終了するまで、移行中のドメインのマシン記述 (MD) が変更されるような操作はブロックされます。このような操作には、このドメイン自体でのすべての操作のほか、マシン上のほかのドメインでのバインド、停止などの操作も含まれます。

バインドされたドメインまたはアクティブでないドメインの移行

バインドされたドメインまたはアクティブでないドメインは移行時に実行されていないため、アクティブなドメインを移行する場合より制約が少なくなります。

バインドされたドメインを移行するには、ターゲットがソースドメインの CPU、メモリー、および入出力の制約を満たす必要があります。満たしていないと、移行は失敗します。アクティブでないドメインの移行には、このような要件はありません。ただし、バインドが行われた場合、ターゲットはそのドメインの制約を満たす必要があります。満たしていないと、ドメインのバインドは失敗します。

バインドされたドメインまたはアクティブでないドメインの **CPU** の移行

バインドされたドメインまたはアクティブでないドメインは、異なるタイプのプロセッサが動作しているマシンおよび異なる周波数で動作しているマシン間で移行できます。

ゲストの Solaris OS イメージで、ターゲットマシン上のプロセッサタイプがサポートされている必要があります。

バインドされたドメインまたはアクティブでないドメインの仮想入出力の移行

アクティブでないドメインの場合、仮想入出力 (VIO) 制約に対して実行されるチェックはありません。そのため、VIO サーバーが存在しなくても移行は正常に実行されます。アクティブでないドメインと同様に、そのドメインがバインドされる時点では、VIO サーバーが存在し、使用可能になっている必要があります。

予行演習の実行

`migrate-domain` サブコマンドに `-n` オプションを指定すると、移行のチェックが実行されますが、ソースドメインの移行は行われません。満たしていない要件がある場合、エラーとして報告されます。これによって、実際に移行を試行する前に構成エラーを修正できます。

注 – 論理ドメインには動的な性質があるため、予行演習が正常に実行されても移行が失敗したり、逆に予行演習が失敗しても移行が成功する可能性があります。

進行中の移行の監視

移行が進行中の場合、ソースドメインとターゲットドメインでは状態出力での表示が異なります。`ldm list` コマンドの出力には、移行中のドメインの状態が表示されます。

`FLAGS` フィールドの 6 列目は、次のいずれかの値になります。

- ソースドメインの場合は、移行のソースであることを示す `s` が表示されます。
- ターゲットドメインの場合は、移行のターゲットであることを示す `t` が表示されます。
- ユーザーによる介入を必要とするエラーが発生した場合、`e` が表示されます。

次の出力は、`ldg-src` が移行のソースドメインであることを示しています。

```
# ldm list ldg-src
NAME       STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg-src    suspended  -n---s          1     1G       0.0%  2h 7m
```

次の出力は、`ldg-tgt` が移行のターゲットドメインであることを示しています。

```
# ldm list ldg-tgt
NAME             STATE      FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg-tgt          bound      -----t  5000    1     1G
```

長形式の状態出力では、移行に関する詳細情報が表示されます。ソースの場合は、完了した処理の割合とともに、ターゲットホストとターゲットドメイン名が表示されます。同様に、ターゲットの場合は、完了した処理の割合とともに、ソースホストとソースドメイン名が表示されます。

例 8–1 進行中の移行の監視

```
# ldm list -o status ldg-src
NAME
ldg-src

STATUS
    OPERATION    PROGRESS    TARGET
    migration    17%         t5440-sys-2
```

進行中の移行の取り消し

移行が開始されたあとに KILL 信号によって `ldm` コマンドが中断されると、移行は終了します。ターゲットドメインは削除され、ソースドメインがアクティブだった場合は再開されます。`ldm` コマンドの制御シェルが失われた場合、移行はバックグラウンドで続行されます。

移行処理は、`ldm cancel-operation` コマンドを使用して、外部から取り消すこともできます。これによって、進行中の移行が終了され、ソースドメインはアクティブなドメインとして再開されます。`ldm cancel-operation` コマンドはソースシステムから開始するようにしてください。あるシステム上で実行される移行関連のコマンドは、そのシステムから開始された移行処理に影響を及ぼします。システムがターゲットシステムの場合、移行処理は制御できません。

注 – 移行が開始されたあとに `ldm(1M)` プロセスを中断しても、移行に影響を与えるのはソースマシンおよびターゲットマシン上の Logical Domains Manager デーモン (`ldmd`) であるため、処理は中断されません。`ldm` プロセスは、戻る前に、移行が完了したことを示す `ldmd` からの信号を待機します。

移行の失敗からの回復

ソースからターゲットへのすべての実行時の状態情報の送信が完了してから、ドメインが再開されたことをターゲットが認識する前にネットワーク接続が切断された場合、移行処理が終了され、ソースがエラー状態になります。これは、移行が正常に完了したかどうかを判断するためにユーザーによる介入が必要であることを示しています。このような状況では、次の手順を実行します。

- ターゲットドメインが正常に再開されているかどうかを判断します。ターゲットドメインは次の 2 つのいずれかの状態になります。
  - 移行が正常に完了した場合、ターゲットドメインは通常の状態になっています。
  - 移行が失敗した場合、ターゲットではターゲットドメインがクリーンアップされ、削除されます。
- ターゲットが再開されている場合、エラー状態のソースドメインを安全に削除できます。ターゲットが存在しない場合、ソースドメインはまだマスターバージョンのドメインであり、回復する必要があります。これを行うには、ソースマシンで取り消しコマンドを実行します。これによって、エラー状態がクリアーされ、ソースドメインが元の状態に復元されます。

自動化された移行の実行

Logical Domains 1.3 ソフトウェアのリリースまでは、移行は対話型の処理でした。移行を開始すると、ターゲットマシンに使用するパスワードの入力を求めるプロンプトが表示されていました。Logical Domains 1.3 では、`ldm migrate-domain -p` *filename* コマンドを使用して、自動化された移行処理を開始できます。

`-p` オプションの引数として指定するファイル名には、次のプロパティーが必要です。

- ファイルの最初の行にパスワードが指定されている必要があります
- パスワードは平文である必要があります
- パスワードの長さは 256 文字以下である必要があります

パスワード末尾の改行文字と最初の行のあとのすべての行は無視されます。

ターゲットマシンのパスワードを格納するファイルは、適切にセキュリティー保護する必要があります。この方法でパスワードを格納する場合は、ファイルのアクセス権の設定が 400 または 600 であること、つまり root 所有者 (特権ユーザー) のみがファイルの読み取りまたは書き込みを許可されていることを確認します。

移行の例

例 8–2 に、ldg1 というドメインを t5440-sys-2 というマシンに移行する方法を示します。

例 8–2 ゲストドメインの移行

```
# ldm migrate-domain ldg1 t5440-sys-2
Target Password:
```

ターゲットのパスワードの入力を要求されることなく、この移行を自動的に実行するには、次のコマンドを使用します。

```
# ldm migrate-domain -p pfile ldg1 t5440-sys-2
```

-p オプションには、引数としてファイル名を指定します。指定するファイルには、ターゲットのスーパーユーザーパスワードを指定します。この例では、pfile はターゲット t5440-sys-2 のパスワードを格納しています。

例 8–3 に示すように、移行処理の一環としてドメインの名前を変更できます。この例では、ldg-src がソースドメインで、移行処理の一環として、ターゲットマシン (t5440-sys-2) 上でこのドメインの名前を ldg-tgt に変更しています。また、ターゲットマシンでのユーザー名 (root) を明示的に指定しています。

例 8–3 ゲストドメインの移行と名前の変更

```
# ldm migrate ldg-src root@t5440-sys-2:ldg-tgt
Target Password:
```

例 8–4 に、ターゲットドメインで移行がサポートされていない場合、すなわち Version 1.1 より前のバージョンの LDoms を実行している場合に表示される失敗メッセージの例を示します。

例 8–4 移行の失敗メッセージ

```
# 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).
```

例 8–5 に、移行が進行中のターゲットドメインの状態を取得する方法を示します。この例では、ソースマシンは t5440-sys-1 です。

例 8–5 ターゲットドメインの状態の取得

```
# ldm list -o status ldg-tgt
NAME
ldg-tgt

STATUS
    OPERATION    PROGRESS    SOURCE
    migration    55%         t5440-sys-1
```

例 8–6 に、移行が進行中のソースドメインの解析可能な状態を取得する方法を示します。この例では、ターゲットマシンは t5440-sys-2 です。

例 8–6 ソースドメインの解析可能な状態の取得

```
# ldm list -o status -p ldg-src
VERSION 1.3
DOMAIN|name=ldg-src|
STATUS
|op=migration|progress=42|error=no|target=t5440-sys-2
```

◆ ◆ ◆ 第 9 章

リソースの管理

この章では、Logical Domains システムでのリソース管理の実行について説明します。

この章の内容は次のとおりです。

CPU Power Management ソフトウェアの使用

CPU Power Management (PM) ソフトウェアを使用するには、まず ILOM 3.0 ファームウェアで電源管理ポリシーを設定する必要があります。この節では、LDoms ソフトウェアで Power Management を使用できるようにするために必要な情報の概要を示します。詳細は、『Sun Integrated Lights Out Manager (ILOM) 3.0 CLI 手順ガイド (http://dlc.sun.com/pdf/820-7376-10/820-7376-10.pdf)』の「消費電力の監視」を参照してください。

電源ポリシーは、任意の時点でのシステムの電力使用量を管理する設定です。Logical Domains Manager (Version 1.3) では、ベースとなるプラットフォームに Power Management 機能が実装されていることを前提として、2 つの電源ポリシーがサポートされます。

- **Performance** - システムは、利用可能なすべての電力を使用できます。
- **Elastic** - システムの電力使用量は、現在の使用率のレベルに合わせて変化します。たとえば、作業負荷が変動しても使用率が常にしきい値の範囲内に維持されるように、必要な分だけシステムコンポーネントの電源を入れたり切ったりします。

ILOM 3.0 ファームウェアの CLI を使用して電源ポリシーを設定する手順については、『Sun Integrated Lights Out Manager (ILOM) 3.0 CLI 手順ガイド (http://dlc.sun.com/pdf/820-7376-10/820-7376-10.pdf)』の「消費電力の監視」を参照してください。

注 – 電力を最大限節約するには、`ldm bind-domain` コマンドを実行してドメインを長時間バインドされたままの状態にしないでください。ドメインがバインドされた状態になっていると、ドメインのすべての CPU の電源がオンになります。

CPU で電源管理されているストランドの表示

この節では、電源管理されているストランドおよび仮想 CPU を一覧表示する方法について説明します。

▼ CPU で電源管理されているストランドを一覧表示する

- 電源管理されているストランドを一覧表示するには、次のいずれかの手順を実行します。

  a. `list -l` サブコマンドを使用します。

  CPU の `UTIL` 列にダッシュ (`---`) が表示されている場合、ストランドが電源管理されていることを意味します。

```
# 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. **list** `-l` サブコマンドに解析可能なオプション (`-p`) を使用します。

`util=` のあとが空白になっている場合、ストランドが電源管理されていることを意味します。

```
# 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
```

▼ 電源管理されている CPU を一覧表示する

- 電源管理されている **CPU** を一覧表示するには、次のいずれかの手順を実行します。

a. `list-devices -a cpu` サブコマンドを使用します。

Power Management (`PM`) 列に `yes` が表示されている場合は CPU が電源管理されていることを意味し、`no` が表示されている場合は CPU の電源が投入されていることを意味します。100 パーセント未使用の CPU はデフォルトで電源管理されることが前提となっているので、PM の下にダッシュ (`---`) が表示されます。

```
# 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. `list-devices -a cpu` サブコマンドに解析可能なオプション (`-p`) を使用します。

Power Management (`pm=`) フィールドに `yes` が表示されている場合は CPU が電源管理されていることを意味し、`no` が表示されている場合は CPU の電源が投入されていることを意味します。100 パーセント未使用の CPU はデフォルトで電源管理されることが前提となっているので、このフィールドは空白になります。

```
# 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=
```

動的資源管理ポリシーの使用

Logical Domains 1.3 ソフトウェア以降では、ポリシーを使用して、動的再構成活動を自動的に実行する方法を決定できます。現時点では、仮想 CPU の動的資源管理を制御するポリシーのみを作成できます。

注意 – 次の問題が CPU の動的資源管理 (DRM) に影響します。

- Power Management (PM) が Elastic モードの場合、DRM は有効にできません。
- DRM が有効になっている場合、PM を Performance モードから Elastic モードに変更できません。
- ドメインの移行処理を実行する前に、CPU の DRM を無効にしていることを確認してください。

資源管理ポリシーでは、論理ドメインで仮想 CPU を自動的に追加および削除できる条件について指定します。ポリシーを管理するには、`ldm add-policy`、`ldm set-policy`、および `ldm remove-policy` コマンドを使用します。

```
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
```

これらのコマンドの詳細および資源管理ポリシーの作成については、`ldm(1M)` マニュアルページを参照してください。

ポリシーは、`tod-begin` プロパティーと `tod-end` プロパティーで指定された期間の間有効です。複数のポリシーが同時に有効になっている場合、ポリシーは、`priority` プロパティーの値を使用して、使用するポリシーを決定します。

ポリシーは、`util-high` および `util-low` プロパティーの値を使用して、CPU 利用率の高位境界値と低位境界値を指定します。利用率が `util-high` の値を超えた場合、仮想 CPU の数が `vcpu-min` から `vcpu-max` までの値の範囲に収まるまで、仮想 CPU がドメインに追加されます。利用率が `util-low` の値を下回った場合、仮想 CPU の数が `vcpu-min` から `vcpu-max` までの値の範囲に収まるまで、仮想 CPU がドメインから削除されます。`vcpu-min` に達すると、仮想 CPU をそれ以上動的に削除できません。`vcpu-max` に達すると、仮想 CPU をそれ以上動的に追加できません。

例 9–1 資源管理ポリシーの追加

たとえば、数週間に渡ってシステムの標準利用率を観測したあと、資源使用状況を最適化するためにポリシーを設定する場合があります。使用率が最も高いのは、毎日太平洋標準時の午前 9:00 〜午後 6:00、使用率が低いのは、毎日太平洋標準時の午後 6:00 〜午前 9:00 です。

このシステム利用率の観測に基づき、システム全体の利用率に従って次の高利用率ポリシーと低利用率ポリシーを作成することにします。

- **高:** 毎日太平洋標準時の午前 9:00 〜午後 6:00
- **低:** 毎日太平洋標準時の午後 6:00 〜午前 9:00

次の `ldm add-policy` コマンドで、高利用率時に `ldom1` ドメインで使用される `high-usage` ポリシーを作成します。

次の `high-usage` ポリシーは次のことを行います。

- `tod-begin` プロパティーと `tod-end` プロパティーを設定することで、開始時間と終了時間がそれぞれ午前 9:00 と午後 6:00 であることを指定します。
- `util-lower` プロパティーと `util-upper` プロパティーを設定することで、ポリシー分析を実行する上限と下限がそれぞれ 25 パーセントと 75 パーセントであることを指定します。
- `vcpu-min` プロパティーと `vcpu-max` プロパティーを設定することで、仮想 CPU の最小数と最大数がそれぞれ 2 と 16 であることを指定します。
- `attack` プロパティーを設定することで、任意の 1 回のリソース制御サイクルで追加される仮想 CPU の最大数は 1 であることを指定します。

例 9–1 資源管理ポリシーの追加 (続き)

- decay プロパティーを設定することで、任意の 1 回のリソース制御サイクルで削除される仮想 CPU の最大数は 1 であることを指定します。
- priority プロパティーを設定することで、このポリシーの優先順位が 1 であることを指定します。優先順位が 1 であるため、別のポリシーが有効になることが可能であっても、このポリシーが実施されます。
- name プロパティーを設定することで、ポリシーファイルの名前が high-usage であることを指定します。
- enable や sample-rate など、指定されていないプロパティーではデフォルト値を使用します。ldm(1M) マニュアルページを参照してください。

```
# 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
```

次の ldm add-policy コマンドで、低利用率時に ldom1 ドメインで使用される med-usage ポリシーを作成します。

次の med-usage ポリシーは次のことを行います。

- tod-begin プロパティーと tod-end プロパティーを設定することで、開始時間と終了時間がそれぞれ午後 6:00 と午前 9:00 であることを指定します。
- util-lower プロパティーと util-upper プロパティーを設定することで、ポリシー分析を実行する上限と下限がそれぞれ 10 パーセントと 50 パーセントであることを指定します。
- vcpu-min プロパティーと vcpu-max プロパティーを設定することで、仮想 CPU の最小数と最大数がそれぞれ 2 と 16 であることを指定します。
- attack プロパティーを設定することで、任意の 1 回のリソース制御サイクルで追加される仮想 CPU の最大数は 1 であることを指定します。
- decay プロパティーを設定することで、任意の 1 回のリソース制御サイクルで削除される仮想 CPU の最大数は 1 であることを指定します。
- priority プロパティーを設定することで、このポリシーの優先順位が 1 であることを指定します。優先順位が 1 であるため、別のポリシーが有効になることが可能であっても、このポリシーが実施されます。
- name プロパティーを設定することで、ポリシーファイルの名前が high-usage であることを指定します。
- enable や sample-rate など、指定されていないプロパティーではデフォルト値を使用します。ldm(1M) マニュアルページを参照してください。

```
# 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
```

論理ドメインのリソースの一覧表示

この節では、ldm サブコマンドの構文の使用法、フラグや利用統計情報などの出力項目の定義、および実際と同様の出力例について説明します。

マシンが読み取り可能な出力

ldm list コマンドの出力を使用するスクリプトを作成する場合は、常に -p オプションを使用してマシンが読み取り可能な形式で出力を生成します。詳細は、157 ページの「解析可能でマシンが読み取り可能なリストを生成する (-p)」 を参照してください。

▼ ldm サブコマンドの構文の使用法を表示する

- ldm のすべてのサブコマンドの構文の使用法を確認します。

```
primary# ldm --help
```

ldm サブコマンドの詳細は、ldm(1M) マニュアルページを参照してください。

フラグの定義

ドメインの出力 (ldm list) では、次のフラグを表示できます。コマンドに長形式および解析可能オプション (-l -p) を使用すると、flags=normal,control,vio-service のように、フラグが省略されずに表示されます。このオプションを使用しない場合は、-n-cv- のように略語が表示されます。リストフラグ値は位置に依存します。次に、左から順に 6 つの列のそれぞれに表示される可能性のある値を示します。

列 1

- s 起動または停止
- - 可変部分

列 2

- n 通常
- t 切り替え

列 3

- d 遅延再構成
- - 可変部分

列 **4**

- c 制御ドメイン
- - 可変部分

列 **5**

- v 仮想 I/O サービスドメイン
- - 可変部分

列 **6**

- s 移行のソースドメイン
- t 移行のターゲットドメイン
- e 移行時に発生したエラー
- - 可変部分

利用統計情報の定義

`ldm list` コマンドの長形式 (`-l`) オプションでは、仮想 CPU ごとの利用統計情報 (`UTIL`) が表示されます。この統計情報は、ゲストオペレーティングシステムの代わりに仮想 CPU が実行に費やした時間の割合です。仮想 CPU は、ハイパーバイザに制御が渡される場合を除き、ゲストオペレーティングシステムに代わって実行するものと考えられます。ゲストオペレーティングシステムが仮想 CPU の制御をハイパーバイザに渡さない場合、ゲストオペレーティングシステムの CPU の利用率は常に `100%` として表示されます。

論理ドメインについて報告された利用統計情報は、ドメインの仮想 CPU に対する仮想 CPU 利用率の平均です。`UTIL` 列にダッシュ (`---`) が表示されている場合、ストランドが電源管理されていることを意味します。

さまざまなリストの表示

▼ ソフトウェアのバージョンを表示する (-V)

- インストールされている現在のソフトウェアのバージョンを表示します。

```
primary# ldm -V
```

▼ 省略形式のリストを生成する

- すべてのドメインの省略形式のリストを生成します。

```
primary# ldm list
```

▼ 長形式のリストを生成する (-l)

- すべてのドメインの長形式のリストを生成します。

```
primary# ldm list -l
```

▼ 拡張リストを生成する (-e)

- すべてのドメインの拡張リストを生成します。

```
primary# ldm list -e
```

▼ 解析可能でマシンが読み取り可能なリストを生成する (-p)

- すべてのドメインの解析可能でマシンが読み取り可能なリストを生成します。

```
primary# ldm list -p
```

▼ 長形式のリストのサブセットを生成する (-o *format*)

- 次に示す 1 つ以上の *format* オプションを入力して、出力をリソースのサブセットとして生成します。1 つ以上の形式を指定する場合、スペースなしでコンマを使用して項目を区切ります。

```
primary# ldm list -o resource[,resource...] ldom
```

  - `console` - 出力には、仮想コンソール (`vcons`) および仮想コンソール端末集配信装置 (`vcc`) サービスが含まれます。
  - `cpu` - 出力には、仮想 CPU (`vcpu`) および物理 CPU (`pcpu`) が含まれます。
  - `crypto` - 暗号化装置の出力には、モジュラー演算ユニット (`mau`) と、Control Word Queue (CWQ) など、LDoms がサポートするその他の暗号化装置が含まれます。
  - `disk` - 出力には、仮想ディスク (`vdisk`) および仮想ディスクサーバー (`vds`) が含まれます。
  - `domain` - 出力には、変数 (`var`)、ホスト ID (`hostid`)、ドメインの状態、フラグ、およびソフトウェアの状態が含まれます。
  - `memory` - 出力には、`memory` が含まれます。
  - `network` - 出力には、メディアアクセス制御 (`mac`) アドレス、仮想ネットワークスイッチ (`vsw`)、および仮想ネットワーク (`vnet`) デバイスが含まれます。
  - `physio` - 物理入出力には、Peripheral Component Interconnect (`pci`) およびネットワークインタフェースユニット (`niu`) が含まれます。
  - `resmgmt` - 出力には、動的資源管理 (DRM) ポリシー情報が含まれます。
  - `serial` - 出力には、仮想論理ドメインチャネル (`vldc`) サービス、仮想論理ドメインチャネルクライアント (`vldcc`)、仮想データプレーンチャネルクライアント (`vdpcc`)、仮想データプレーンチャネルサービス (`vdpcs`) が含まれます。

- `stats` - 出力には、資源管理ポリシーに関連する統計が含まれます。
- `status` - 出力には、進行中のドメインの移行に関する状態情報が含まれます。

次の例に、指定可能なさまざまな出力のサブセットを示します。

- 制御ドメインの CPU 情報のリスト

```
# ldm list -o cpu primary
```

- ゲストドメインのドメイン情報のリスト

```
# ldm list -o domain ldm2
```

- ゲストドメインのメモリーおよびネットワーク情報のリスト

```
# ldm list -o network,memory ldm1
```

- ゲストドメインの DRM ポリシー情報のリスト

```
# ldm list -o resmgmt,stats ldm1
```

▼ 変数を一覧表示する

- ドメインの変数とその値を表示します。

  `primary# ldm list-variable` *variable-name ldom*

  たとえば、次のコマンドは、`ldg1` ドメインの `boot-device` 変数の値を表示します。

```
primary# ldm list-variable boot-device ldg1
boot-device=/virtual-devices@100/channel-devices@200/disk@0:a
```

▼ バインドを一覧表示する

- ドメインにバインドされたリソースを一覧表示します。

  `primary# ldm list-bindings` *ldom*

▼ 構成を一覧表示する

- SP に格納されている論理ドメイン構成を一覧表示します。

例 9–2 構成のリスト

`ldm list-config` コマンドは、サービスプロセッサに格納されている論理ドメイン構成を一覧表示します。`-r` オプションとともに使用する場合、このコマンドは、制御ドメインに存在する自動保存ファイルの構成を一覧表示します。

構成の詳細は、170 ページの「Logical Domains 構成の管理」 を参照してください。ほかの例については、`ldm(1M)` マニュアルページを参照してください。

```
primary# ldm list-config
factory-default
3guests
foo [next poweron]
primary
reconfig-primary
```

参考 ラベルの意味

構成名の右にあるラベルの意味は、次のとおりです。

- [current] - 最後に起動された構成。これは、現在動作している構成に一致する間、つまり再構成を開始するまでの間のみ表示されます。再構成を行なったあとは、注釈が [next poweron] に変更されます。
- [next poweron] - 次回電源を再投入するときに使用される構成。

▼ デバイスを一覧表示する

- すべてのサーバーリソース (バインドされたリソースおよびバインドされていないリソース) を一覧表示します。

```
primary# ldm list-devices -a
```

▼ 使用可能なメモリーを一覧表示する

- 割り当て可能なメモリーの量を一覧表示します。

```
primary# ldm list-devices mem
MEMORY
    PA                   SIZE
    0x14e000000          2848M
```

▼ サービスを一覧表示する

- 使用可能なサービスを一覧表示します。

```
primary# ldm list-services
```

制約の一覧表示

Logical Domains Manager に対する制約とは、特定のドメインに割り当てられる 1 つ以上のリソースです。使用可能なリソースに応じて、ドメインに追加するように要求したすべてのリソースを受け取るか、まったく受け取らないかのいずれかです。list-constraints サブコマンドは、ドメインに割り当てるように要求したリソースを一覧表示します。

▼ 1つのドメインの制約を一覧表示する

- 1つのドメインの制約を一覧表示します。

```
primary# ldm list-constraints ldom
```

▼ 制約を XML 形式で一覧表示する

- 特定のドメインの制約を XML 形式で一覧表示します。

```
primary# ldm list-constraints -x ldom
```

▼ 制約をマシンが読み取り可能な形式で一覧表示する

- すべてのドメインの制約を解析可能な形式で一覧表示します。

```
primary# ldm list-constraints -p
```

◆ ◆ ◆ 第 10 章

10 構成の管理

この章では、構成の管理について説明します。

この章の内容は次のとおりです。

将来の再構築用の論理ドメイン構成の保存

基本的な処理は、各ドメインの制約情報を XML ファイルに保存することです。たとえば、ハードウェアの障害のあとに、この XML ファイルを Logical Domains Manager に対して再実行して、必要な設定を再構築できます。

162 ページの「ゲストドメイン構成を再構築する」 は、制御ドメインではなく、ゲストドメインに対して有効です。制御 (`primary`) ドメインの制約を XML ファイルに保存することはできますが、それを `ldm add-domain i` コマンドに指定することはできません。ただし、XML ファイルのリソース制約を使用して、primary ドメインを再構成する CLI コマンドを作成することはできます。`ldm list-constraints -x` primary コマンドの標準的な XML 出力を、primary ドメインの再構成に必要な CLI コマンドに変換する方法については、162 ページの「制御ドメインの再構築」 を参照してください。

次に示す方法では、実際のバインドは保持されず、それらのバインドを作成するために使用した制約だけが保持されます。つまり、この手順を行うと、ドメインは同じ仮想リソースを持ちますが、同じ物理リソースにバインドされるとはかぎりません。

▼ すべての論理ドメイン構成を保存する

- 各論理ドメインで、ドメインの制約を含む **XML** ファイルを作成します。

  # ldm list-constraints -x *ldom* **>** *ldom*.**xml**

  次の例は、primary ドメインの制約を含む XML ファイル primary.xml を作成する方法を示しています。

  # ldm list-constraints -x primary > primary.xml

▼ ゲストドメイン構成を再構築する

- 作成した各ゲストドメインの **XML** ファイルに対して、次のコマンドを実行します。

  # ldm add-domain -i *ldom*.**xml**
  # ldm bind-domain *ldom*
  # ldm start-domain *ldom*

制御ドメインの再構築

この節では、ldm list-constraints -x primary コマンドの標準的なの XML 出力を、primary ドメインの再構成に必要な CLI コマンドに変換する方法について説明します。XML 出力のサンプルでは、XML から CLI コマンドを作成するために使用するリソースおよびプロパティーが太字で示されています。CLI コマンドの詳細は、ldm(1M) マニュアルページを参照してください。

ldm list-constraints -x primary コマンドの出力のサンプルを次に示します。

例 **10–1** list-constraints サブコマンドの XML 出力のサンプル

```
<?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">
  <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>
```

例 10–1 list-constraints サブコマンドの XML 出力のサンプル (続き)

```
    <rasd:Address>00:03:ba:d8:ba:f6</rasd:Address>
    <gprop:GenericProperty key="hostid">0x83d8baf6</gprop:GenericProperty>
  </Item>
</Section>
<Section xsi:type="ovf:VirtualHardwareSection_Type">
  <Item>
    <rasd:OtherResourceType>cpu</rasd:OtherResourceType>
    <rasd:AllocationUnits>4</rasd:AllocationUnits>
  </Item>
</Section>
<Section xsi:type="ovf:VirtualHardwareSection_Type">
  <Item>
    <rasd:OtherResourceType>mau</rasd:OtherResourceType>
    <rasd:AllocationUnits>1</rasd:AllocationUnits>
  </Item>
</Section>
<Section xsi:type="ovf:VirtualHardwareSection_Type">
  <Item>
    <rasd:OtherResourceType>memory</rasd:OtherResourceType>
    <rasd:AllocationUnits>4G</rasd:AllocationUnits>
  </Item>
</Section>
<Section xsi:type="ovf:VirtualHardwareSection_Type">
  <Item>
    <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
    <gprop:GenericProperty key="name">pci@7c0</gprop:GenericProperty>
  </Item>
</Section>
<Section xsi:type="ovf:VirtualHardwareSection_Type">
  <Item>
    <rasd:OtherResourceType>vsw</rasd:OtherResourceType>
    <rasd:Address>auto-allocated</rasd:Address>
    <gprop:GenericProperty key="service_name">primary-vsw0</gprop:GenericProperty>
    <gprop:GenericProperty key="dev_path">nxge0</gprop:GenericProperty>
    <gprop:GenericProperty key="default-vlan-id">1</gprop:GenericProperty>
    <gprop:GenericProperty key="pvid">1</gprop:GenericProperty>
  </Item>
</Section>
<Section xsi:type="ovf:VirtualHardwareSection_Type">
  <Item>
    <rasd:OtherResourceType>vcc</rasd:OtherResourceType>
    <gprop:GenericProperty key="service_name">primary-vcc0</gprop:GenericProperty>
    <gprop:GenericProperty key="min_port">5000</gprop:GenericProperty>
    <gprop:GenericProperty key="max_port">6000</gprop:GenericProperty>
  </Item>
</Section>
```

例 10–1 list-constraints サブコマンドの XML 出力のサンプル (続き)

```
        <Section xsi:type="ovf:VirtualHardwareSection_Type">
          <Item>
            <rasd:OtherResourceType>vds</rasd:OtherResourceType>
            <gprop:GenericProperty key="service_name">primary-vds0</gprop:GenericProperty>
          </Item>
        </Section>
        <Section xsi:type="ovf:VirtualHardwareSection_Type">
          <Item>
            <rasd:OtherResourceType>vds_volume</rasd:OtherResourceType>
            <gprop:GenericProperty key="vol_name">primary-vds0-vol0</gprop:GenericProperty>
            <gprop:GenericProperty
              key"block_dev">/opt/SUNWldm/domain_disks/testdisk.nv.53.1</gprop:GenericProperty>
            <gprop:GenericProperty key="service_name">primary-vds0</gprop:GenericProperty>
          </Item>
        </Section>
      </Content>
    </Envelope>
  </data>
</LDM_interface>
```

<Content> タグおよび <Content> タグ内の <Section> には、primary ドメイン、および primary ドメインに含まれるすべてのリソースが記述されています。<Item> 内の <rasd:...> タグおよび <gprop:GenericProperty...> タグには、各リソースに必要なプロパティーが記述されています。各 <Section> の各リソースを確認して、リソースの制約に基づいて CLI コマンドを作成できます。以降の節では、ドメインの XML 記述でより一般的ないくつかのリソースと、そのリソースに対する同等の CLI コマンドを示します。

論理ドメインの情報 (ldom_info) セクション

このセクションには、primary ドメインの MAC アドレスおよびホスト ID の情報が記述されます。これは primary ドメインであるため、この情報を設定することはできません。この情報は自動的に設定されます。

例 10–2 論理ドメインの情報 (ldom_info) セクション

```
<Section> xsi:type="ovf:ResourceAllocationSection_Type">
  <Item>
    <rasd:OtherResourceType>ldom_info</rasd:OtherResourceType>
    <rasd:Address>00:03:ba:d8:ba:f6</rasd:Address>
    <gprop:GenericProperty key="hostid">0x83d8baf6</gprop:GenericProperty>
  </Item>
</Section>
```

この例での論理ドメインの情報 (ldom_info) は、次のとおりです。

- (MAC) Address - 00:03:ba:d8:ba:f6
- hostid - 0x83d8baf6

暗号化 (mau) セクション

このセクションには、primary ドメインに割り当てられた暗号化装置 (mau) の数が記述されます。

注 – XML の一覧では mau セクションは cpu セクションのあとに記述されていますが、set-mau サブコマンドは set-cpu サブコマンドの前に実行する必要があります。これは、対応する暗号化装置を削除しないかぎりドメインから CPU を削除できないためです。

例 10–3　暗号化 (mau) セクション

```
<Section> xsi:type="ovf:VirtualHardwareSection_Type"
  <Item>
    <rasd:OtherResourceType>mau</rasd:OtherResourceType>
    <rasd:AllocationUnits>1</rasd:AllocationUnits>
  </Item>
</Section>
```

このセクションは、次の CLI コマンドに相当します。

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

CPU (cpu) セクション

このセクションには、primary ドメインに割り当てられた仮想 cpu の数が記述されます。

例 10–4　CPU (cpu) セクション

```
<Section> xsi:type="ovf:VirtualHardwareSection_Type"
  <Item>
    <rasd:OtherResourceType>cpu</rasd:OtherResourceType>
    <rasd:AllocationUnits>4</rasd:AllocationUnits>
  </Item>
</Section>
```

このセクションは、次の CLI コマンドに相当します。

```
# ldm set-vcpu 4 primary
```

メモリー (memory) セクション

このセクションには、primary ドメインに割り当てられたメモリーの量が記述されます。

例 10–5　メモリー (memory) セクション

```
<Section> xsi:type="ovf:VirtualHardwareSection_Type"
  <Item>
    <rasd:OtherResourceType>memory</rasd:OtherResourceType>
    <rasd:AllocationUnits>4G</rasd:AllocationUnits>
  </Item>
</Section>
```

このセクションは、次の CLI コマンドに相当します。

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

物理入出力 (physio_device) セクション

このセクションには、primary ドメインに残す物理 I/O バスが記述されます。

例 10–6　物理 I/O (physio_device) セクション

```
<Section> xsi:type="ovf:VirtualHardwareSection_Type"
  <Item>
    <rasd:OtherResourceType>physio_device</rasd:OtherResourceType>
    <gprop:GenericProperty key="name">pci@7c0</gprop:GenericProperty>
  </Item>
</Section>
```

以前の構成どおりに、同じ I/O デバイスを primary ドメインに設定するには、まず、起動時に構成される I/O デバイスを一覧表示する必要があります。

```
# ldm list -l primary
....
IO
    DEVICE          PSEUDONYM       OPTIONS
    pci@7c0         bus_b
    pci@780         bus_a
....
```

例 10–6 で、primary ドメインに残るように以前に構成されていたバスは、pci@7c0 です。XML に他の physio-device セクションが含まれていない場合、pci@780 バスを削除する必要があります。

このセクションは、次の CLI コマンドに相当します。

```
# ldm remove-io pci@780 primary
```

仮想スイッチ (vsw) セクション

このセクションには、primary ドメインに割り当てられた仮想スイッチ (vsw) が記述されます。

```
<Section xsi:type="ovf:VirtualHardwareSection_Type">
  <Item>
    <rasd:OtherResourceType>vsw</rasd:OtherResourceType>
    <rasd:Address>auto-allocated</rasd:Address>
    <gprop:GenericProperty key="service_name">primary-vsw0</gprop:GenericProperty>
    <gprop:GenericProperty key="dev_path">nxge0</gprop:GenericProperty>
    <gprop:GenericProperty key="mode">sc</gprop:GenericProperty>
    <gprop:GenericProperty key="default-vlan-id">1</gprop:GenericProperty>
    <gprop:GenericProperty key="pvid">1</gprop:GenericProperty>
  </Item>
</Section>
```

各表記の意味は次のとおりです。

- <rasd:Address> タグには、仮想スイッチに使用される MAC アドレスが記述されます。このタグの値が auto-allocated である場合、MAC アドレスを指定する必要はありません。
- XML のキープロパティー service_name は、仮想スイッチの名前 (この場合は、primary-vsw0) を示します。
- XML のキープロパティー dev_path は、実際のネットワークデバイスのパス名 (この場合は、net-dev=nxge) を示します。
- XML のキープロパティー mode は、SunCluster のハートビートサポートのための sc を示します。

default-vlan-id (1)、pvid (1) など、このセクションの一部の値にはデフォルト値が使用されるため、このセクションは次の CLI コマンドに相当します。

```
# ldm add-vswitch net-dev=nxge primary-vsw0 primary
```

仮想コンソール端末集配信装置 (vcc) セクション

このセクションには、primary ドメインに割り当てられた仮想コンソール端末集配信装置 (vcc) が記述されます。

```
<Section xsi:type="ovf:VirtualHardwareSection_Type">
  <Item>
    <rasd:OtherResourceType>vcc</rasd:OtherResourceType>
    <gprop:GenericProperty key="service_name">primary-vcc0</gprop:GenericProperty>
    <gprop:GenericProperty key="min_port">5000</gprop:GenericProperty>
    <gprop:GenericProperty key="max_port">6000</gprop:GenericProperty>
  </Item>
</Section>
```

XML のキープロパティー service_name は、vcc サービスの名前 (この場合は、primary-vcc0) を示します。

このセクションは、次の CLI コマンドに相当します。

```
# ldm add-vcc port-range=5000-6000 primary-vcc0 primary
```

仮想ディスクサーバー (vds) セクション

このセクションには、primary ドメインに割り当てられた仮想ディスクサーバー (vds) が記述されます。

```
<Section xsi:type="ovf:VirtualHardwareSection_Type">
  <Item>
    <rasd:OtherResourceType>vds</rasd:OtherResourceType>
    <gprop:GenericProperty key="service_name">primary-vds0</gprop:GenericProperty>
  </Item>
</Section>
```

XML のキープロパティー service_name は、仮想ディスクサーバーのこのインスタンスのサービス名 (この場合は、primary-vds0) を示します。この service_name は、サーバー上のすべての仮想ディスクサーバーインスタンスの中で一意である必要があります。

このセクションは、次の CLI コマンドに相当します。

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

仮想ディスクサーバーデバイス (vdsdev) セクション

このセクションには、primary ドメインに割り当てられた仮想ディスクサーバーによってエクスポートされたデバイス (vdsdev) が記述されます。

```
<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>
```

各表記の意味は次のとおりです。

- XML のキープロパティーであるボリューム名 (vol_name) とサービス名 (service_name) は、CLI コマンドでは組み合わせて使用します (この場合は、vdsdev0@primary-vds0)。
- XML のキープロパティー block_dev は、相当する CLI コマンドでの *backend* 引数となります。これは、仮想ディスクのデータの格納場所を示し、この場合は、/opt/SUNWldm/domain_disks/testdisk1 となります。
- XML の省略可能なキープロパティー vol_opts は、{ro,slice,excl} のように、これらの項目の 1 つ以上がコンマで区切られて、1 つの文字列となっているものです。
- XML の省略可能なキープロパティー mpgroup は、マルチパス (フェイルオーバー) グループの名前を示します。

このセクションは、次の CLI コマンドに相当します。

```
# ldm add-vdsdev options=ro mpgroup=mpgroup-name
/opt/SUNWldm/domain_disks/testdisk1 vdsdev0@primary-vds0
```

Logical Domains 構成の管理

Logical Domains 構成とは、単一システム内でのすべてのドメインとそのリソース割り当てをすべて記述したものです。構成は、サービスプロセッサ (SP) に保存および格納し、あとで使用することができます。

システムに電源を投入すると、SP は選択された構成を起動します。特定の構成を起動することで、システムは、同じドメインセットを実行し、その構成に指定されている同じ仮想化およびリソース割り当てのパーティション分割を使用します。デフォルトの構成は、最後に保存された構成です。

Logical Domains 1.2 リリース以降は、Logical Domains 構成が変更された場合は常に、現在の構成のコピーが制御ドメインに自動的に保存されます。

次の状況でも、自動保存処理はただちに行われます。

- 新しい構成が、SP に明示的に保存されていない場合
- 実際の構成の変更が、影響を受けるドメインの再起動時まで行われない場合

SP に保存されている構成が失われた場合、この自動保存処理によって構成を回復できます。また、システムの電源再投入時に現在の構成が SP に明示的に保存されなかった場合も、この処理によって構成を回復できます。このような場合、現在の構成が次回の起動時用としてマークされている構成よりも新しければ、Logical Domains Manager は再起動時にこの構成を回復できます。

注 – 電源管理、FMA、ASR、および PRI 更新イベントでは、自動保存ファイルは更新されません。

自動保存ファイルは、自動または手動で新規または既存の構成に復元できます。デフォルトでは、自動保存構成が、対応する実行中の構成よりも新しい場合、メッセージが LDoms ログに書き込まれます。したがって、`ldm add-spconfig -r` コマンドを使用して既存の構成を手動で更新するか、または自動保存データに基づいて新しい構成を作成する必要があります。

注 – 遅延再構成が保留中の場合、構成の変更はただちに自動保存されます。そのため、`ldm list-config -r` コマンドを実行すると、自動保存構成は、現在の構成より新しいものとして表示されます。

`ldm *-spconfig` コマンドを使用して構成を管理する方法と、自動保存ファイルを手動で回復する方法については、`ldm(1M)` マニュアルページを参照してください。

起動する構成を選択する方法については、178 ページの「LDoms とサービスプロセッサの使用」を参照してください。

▼ 自動回復ポリシーを変更する

自動回復ポリシーには、制御ドメインに自動的に保存された 1 つの構成が対応する実行中の構成よりも新しい場合に、構成の回復を処理する方法を指定します。自動回復ポリシーを指定するには、ldmd SMF サービスの autorecovery_policy プロパティーを設定します。autorecovery_policy プロパティーには次の値を使用できます。

- autorecovery_policy=1 – 自動保存構成が、対応する実行中の構成よりも新しい場合に、警告メッセージをログに記録します。これらのメッセージは、ldmd SMF ログファイルに記録されます。ユーザーは、構成の回復を手動で実行する必要があります。これはデフォルトのポリシーです。
- autorecovery_policy=2 – 自動保存構成が、対応する実行中の構成よりも新しい場合に、通知メッセージを表示します。この通知メッセージは、Logical Domains Manager の毎回の再起動後にはじめて ldm コマンドが実行されるときに、ldm コマンドの出力結果中に出力されます。ユーザーは、構成の回復を手動で実行する必要があります。
- autorecovery_policy=3 – 自動保存構成が、対応する実行中の構成よりも新しい場合に、構成を自動的に更新します。この処理は、次回の電源の再投入時に使用される SP 構成を上書きします。この構成は、制御ドメインに保存されている、より新しい構成で更新されます。この処理は、現在実行中の構成には影響を与えません。次回の電源再投入時に使用される構成にのみ影響を与えます。メッセージもログに記録されます。このメッセージには、より新しい構成が SP に保存され、次回のシステム電源の再投入時にはその構成が起動されるということが示されます。これらのメッセージは、ldmd SMF ログファイルに記録されます。

1 制御ドメインにログインします。

2 スーパーユーザーになるか、同等の役割を取得します。

役割には、承認および特権付きコマンドが含まれます。役割の詳細は、『Solaris のシステム管理 (セキュリティサービス)』の「RBAC の構成 (作業マップ)」 を参照してください。

3 autorecovery_policy プロパティー値を表示します。

```
# svccfg -s ldmd listprop ldmd/autorecovery_policy
```

4 ldmd サービスを停止します。

```
# svcadm disable ldmd
```

5 autorecovery_policy プロパティー値を変更します。

```
# svccfg -s ldmd setprop ldmd/autorecovery_policy=value
```

たとえば、自動回復を実行するようにポリシーを設定するには、プロパティー値を 3 に設定します。

```
# svccfg -s ldmd setprop ldmd/autorecovery_policy=3
```

6 ldmd サービスを更新して再起動します。

```
# svcadm refresh ldmd
# svcadm enable ldmd
```

例 10–7 ログへの記録から自動回復への自動回復ポリシーの変更

次の例は、autorecovery_policy プロパティーの現在の値を表示し、その値を新しい値に変更する方法を示しています。このプロパティーの元の値は 1 です。この場合、自動保存の変更はログに記録されます。ldmd サービスの停止および再起動には svcadm コマンド、プロパティー値の表示および設定には svccfg コマンドが使用されます。

```
# 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 11 章

11 その他の管理タスクの実行

この章では、ここまでの章では説明していない Logical Domains ソフトウェアの使用に関する情報とタスクについて説明します。

この章の内容は次のとおりです。

CLI での名前の入力

次の節では、Logical Domains Manager CLI で名前を入力する場合の制限について説明します。

ファイル名 (*file*) と変数名 (*var-name*)

- 最初の文字は、英字、数字、またはスラッシュ (/) である必要があります。
- 以降の文字は、英字、数字、または句読点である必要があります。

仮想ディスクサーバー *backend* および仮想スイッチデバイス名

名前は、英字、数字、または句読点を含む必要があります。

構成名 (*config-name*)

サービスプロセッサ (SP) に格納されている構成に割り当てる論理ドメイン構成名 (*config-name*) は、64 文字以下である必要があります。

その他のすべての名前

論理ドメイン名 (*ldom*)、サービス名 (*vswitch-name*、*service-name*、*vdpcs-service-name*、および *vcc-name*)、仮想ネットワーク名 (*if-name*)、仮想ディスク名 (*disk-name*) など、その他の名前は、次の形式である必要があります。

- 最初の文字は、英字または数字である必要があります。
- 以降の文字は、英字、数字、または「`-_+#.:^;()`」のいずれかの文字である必要があります。

ネットワークを介したゲストコンソールへの接続

vntsd(1M) の SMF マニフェストで `listen_addr` プロパティーが制御ドメインの IP アドレスに設定されている場合は、ネットワークを介してゲストコンソールに接続できます。次に例を示します。

```
$ telnet host-name 5001
```

注 – コンソールへのネットワークアクセスを有効にすることには、セキュリティー上の問題があります。すべてのユーザーがコンソールに接続できるようになるため、デフォルトではこの設定は無効になっています。

サービス管理機能マニフェストは、サービスが記述された XML ファイルです。SMF マニフェストの作成については、「Solaris 10 System Administrator Collection (`http://docs.sun.com/app/docs/coll/47.16`)」を参照してください。

注 – コンソールを使用してゲストドメインの英語版以外の OS にアクセスするには、コンソールの端末が、その OS が必要とするロケールになっている必要があります。

コンソールグループの使用

仮想ネットワーク端末サーバーデーモン vntsd(1M) を使用すると、1 つの TCP ポートを使用して複数のドメインのコンソールにアクセスできるようになります。Logical Domains Manager は、ドメインの作成時に、そのドメインのコンソール用の新しいデフォルトグループを作成することにより、各コンソールに一意の TCP ポートを割り当てます。TCP ポートは、コンソール自体ではなくコンソールグループに割り当てられます。コンソールは、set-vcons サブコマンドを使用して既存のグループにバインドできます。

▼ 複数のコンソールを 1 つのグループにまとめる

1 ドメインのコンソールを 1 つのグループにバインドします。

次の例では、3 つの異なるドメイン (ldg1、ldg2、ldg3) のコンソールを同じコンソールグループ (group1) にバインドします。

```
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 関連付けられた TCP ポート (この例ではポート 5000 の localhost) に接続します。

```
# telnet localhost 5000
primary-vnts-group1: h, l, c{id}, n{name}, q:
```

いずれかのドメインコンソールの選択を求めるプロンプトが表示されます。

3 l (list) を選択して、グループ内のドメインを一覧表示します。

```
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 – コンソールを別のグループまたは vcc インスタンスに再度割り当てるには、ドメインがバインドされていない状態、つまり、アクティブでない状態である必要があります。vntsd を管理するための SMF の構成と使用法、およびコンソールグループの使用法については、Solaris 10 OS の vntsd(1M) マニュアルページを参照してください。

負荷が大きいドメインの停止処理がタイムアウトする可能性

`ldm stop-domain` コマンドは、ドメインが完全に停止する前にタイムアウトする可能性があります。このような状況が発生すると、Logical Domains Manager によって次のようなエラーが返されます。

```
LDom ldg8 stop notification failed
```

しかし、ドメインが停止要求をまだ処理している可能性があります。`ldm list-domain` コマンドを使用して、ドメインの状態を確認します。次に例を示します。

```
# ldm list-domain ldg8
NAME         STATE   FLAGS  CONS   VCPU MEMORY  UTIL UPTIME
ldg8         active  s----  5000   22   3328M   0.3% 1d 14h 31m
```

前述のリストには、ドメインがアクティブと表示されていますが、`s` フラグはドメインが停止処理中であることを示しています。これは、一時的な状態であるはずです。

次の例は、ドメインがすでに停止していることを示しています。

```
# ldm list-domain ldg8
NAME         STATE   FLAGS  CONS   VCPU MEMORY  UTIL UPTIME
ldg8         bound   -----  5000   22   3328M
```

論理ドメインを使用した Solaris OS の操作

この節では、Logical Domains Manager によって作成された構成がインスタンス化されるときに発生する、Solaris OS を使用した場合の動作の変更について説明します。

Solaris OS の起動後に OpenBoot ファームウェアを使用できない

Solaris OS を起動したあとに OpenBoot ファームウェアを使用できません。これは、OpenBoot ファームウェアがメモリーから削除されるためです。

Solaris OS から `ok` プロンプトを表示するには、ドメインを停止する必要があります。Solaris OS の `halt` コマンドを使用すると、ドメインを停止することができます。

サーバーの電源の再投入

LDoms ソフトウェアを実行しているシステムでサーバーの電源の再投入を必要とする保守作業を行うときは常に、最初に現在の論理ドメイン構成を SP に保存する必要があります。

▼ 現在の論理ドメイン構成を SP に保存する

- 次のコマンドを使用します。

```
# ldm add-config config-name
```

電源管理されているドメインのアクティブな CPU での psradm(1M) コマンドの使用禁止

電源管理されているドメインのアクティブな CPU の動作状態を、psradm(1M) コマンドを使用して変更しようとしないでください。

Solaris OS のブレークの結果

この節で説明する動作は、次の処理を行なった場合に発生します。

1. 入力デバイスが keyboard に設定されているときに、L1-A キーシーケンスを押した場合。
2. 仮想コンソールが telnet プロンプトにあるときに、send break コマンドを入力した場合。

これらのタイプのブレーク後に次のプロンプトが表示されます。

```
c)ontinue, s)ync, r)eset, h)alt?
```

これらのタイプのブレーク後のシステムの動作を表す文字を入力します。

制御ドメインの停止または再起動の結果

次の表に、制御 (primary) ドメインの停止時または再起動時に予想される動作を示します。

表 11–1　制御 (primary) ドメインの停止または再起動時に予想される動作

コマンド	他のドメインが構成されているか	動作
halt	未構成	ホストの電源が切断され、SP で電源が投入されるまで切断されたままです。
	構成	変数 auto-boot? が true である場合は、ソフトリセットが行われて起動します。変数 auto-boot? が false である場合は、ソフトリセットが行われて ok プロンプトで停止します。
reboot	未構成	ホストを再起動し、電源は切断されません。
	構成	ホストを再起動し、電源は切断されません。
shutdown -i 5	未構成	ホストの電源が切断され、SP で電源が投入されるまで切断されたままです。
	構成	ソフトリセットが行われて再起動します。

LDoms とサービスプロセッサの使用

この節では、Integrated Lights Out Manager (ILOM) サービスプロセッサ (SP) を Logical Domains Manager とともに使用する場合の注意事項について説明します。ILOM ソフトウェアの使用については、使用しているプラットフォーム固有のドキュメントを参照してください。たとえば、Sun SPARC Enterprise T5120 および T5220 サーバーの場合は、『Sun SPARC Enterprise T5120 and T5220 Servers Topic Set』 を参照してください。

既存の ILOM コマンドに、オプションを 1 つ追加できます。

```
-> set /HOST/bootmode config=config-name
```

config=*config-name* オプションを使用すると、次回の電源投入時の構成を出荷時構成 (factory-default) などの別の構成に設定できます。

ホストの電源が投入されているか切断されているかにかかわらず、このコマンドを実行できます。次回のホストリセットまたは電源投入時に有効になります。

▼ 論理ドメインの構成をデフォルトまたは別の構成にリセットする

- 次のコマンドを実行して、次回の電源投入時に論理ドメインの構成をデフォルトの出荷時構成にリセットします。

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

また、ldm add-config コマンドを使用して Logical Domains Manager で作成され、サービスプロセッサ (SP) に保存されているほかの構成を選択することもできます。Logical Domains Manager の ldm add-config コマンドで指定する名前は、ILOM の bootmode コマンドで構成を選択する際に使用できます。たとえば、ldm-config1 という名前の構成が保存されているとすると、次のように指定します。

```
-> set /HOST/bootmode config=ldm-config1
```

この場合、システムの電源を切ってすぐに入れ直し、新しい設定をロードする必要があります。

ldm add-config コマンドの詳細は、ldm(1M) マニュアルページを参照してください。

ドメインの依存関係の構成

Logical Domains Manager を使用して、ドメイン間の依存関係を確立できます。依存する 1 つ以上のドメインを持つドメインは、マスタードメインと呼ばれます。別のドメインに依存するドメインは、スレーブドメインと呼ばれます。

master プロパティーを設定することによって、各スレーブドメインに最大 4 つのマスタードメインを指定できます。たとえば、次に示すコンマで区切られたリストでは、pine スレーブドメインに 4 つのマスタードメインを指定しています。

```
# ldm add-domain master=apple,lemon,orange,peach pine
```

各マスタードメインには、マスタードメインに障害が発生した場合のスレーブドメインの動作を指定できます。たとえば、マスタードメインに障害が発生した場合、そのスレーブドメインでパニックを発生させる必要があることがあります。1 つのスレーブドメインに複数のマスタードメインが指定されている場合、最初のマスタードメインに障害が発生すると、そのすべてのスレーブドメインに対して定義済みの障害ポリシーがトリガーされます。

注 – 複数のマスタードメインに同時に障害が発生した場合、指定された障害ポリシーのうち 1 つのみが、影響を受けるすべてのスレーブドメインに対して実施されます。たとえば、障害が発生したマスタードメインに stop および panic という障害ポリシーが定義されている場合、すべてのスレーブドメインが停止するか、パニックが発生します。

マスタードメインの障害ポリシーは、failure-policy プロパティーに次のいずれかの値を設定することによって制御できます。

- ignore は、マスタードメインに障害が発生した場合、すべてのスレーブドメインを無視します。
- panic は、マスタードメインに障害が発生した場合、すべてのスレーブドメインにパニックを発生させます。
- reset は、マスタードメインに障害が発生した場合、すべてのスレーブドメインをリセットします。
- stop は、マスタードメインに障害が発生した場合、すべてのスレーブドメインを停止します。

この例では、マスタードメインの障害ポリシーが次のように指定されています。

```
# 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
```

このメカニズムを使用して、ドメイン間の明示的な依存関係を作成できます。たとえば、ゲストドメインが、サービスドメインに暗黙に依存し、その仮想デバイスを提供しているとします。ゲストドメインが依存しているサービスドメインが実行されていない場合、ゲストドメインの入出力はブロックされます。ゲストドメインをサービスドメインのスレーブドメインとして定義することによって、サービスドメインが停止した場合のゲストドメインの動作を指定できます。このような依存関係が確立されていない場合、ゲストドメインはサービスドメインが使用可能になるのを待機します。

注 – Logical Domains Manager では、依存サイクルが生じるようなドメイン関係を作成することはできません。詳細は、182 ページの「依存サイクル」 を参照してください。

ドメインの依存関係の XML の例は、例 12–6 を参照してください。

ドメインの依存関係の例

次の例は、ドメインの依存関係を構成する方法を示します。

- 最初のコマンドは、twizzle というマスタードメインを作成します。このコマンドは、failure-policy=reset を使用して、twizzle ドメインに障害が発生した場合にスレーブドメインをリセットするように指定します。2 つめのコマンドは、primary というマスタードメインに変更を加えます。このコマンドは、failure-policy=panic を使用して、primary ドメインに障害が発生した場合にスレーブドメインにパニックを発生させるように指定します。3 つめのコマンドは、2 つのマスタードメイン twizzle と primary に依存する、chocktaw というスレーブドメインを作成します。このスレーブドメインは、master=twizzle,primary を使用して、マスタードメインを指定します。twizzle または primary のいずれかのドメインに障害が発生した場合、chocktaw ドメインはリセットされるか、パニックが発生します。最初に障害が発生したマスタードメインによって、スレーブドメインの動作が決定されます。

```
# ldm add-domain failure-policy=reset twizzle
# ldm set-domain failure-policy=panic primary
# ldm add-domain master=twizzle,primary chocktaw
```

- この例は、ldm set-domain コマンドを使用して orange ドメインに変更を加え、primary をマスタードメインとして割り当てます。2 つめのコマンドは、ldm set-domain コマンドを使用して、orange および primary を tangerine ドメインのマスタードメインとして割り当てます。3 つめのコマンドは、これらすべてのドメインに関する情報を一覧表示します。

```
# 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
```

- 次に、解析可能な出力を使用した一覧表示の例を示します。

```
# ldm list -o domain -p
```

依存サイクル

Logical Domains Manager では、依存サイクルが生じるようなドメイン関係を作成することはできません。依存サイクルとは、スレーブドメインが自身に依存したり、マスタードメインがそのスレーブドメインのいずれかに依存したりすることになる、2 つ以上のドメイン間の関係です。

Logical Domains Manager は、依存関係を追加する前に、依存サイクルが存在しないかを判断します。Logical Domains Manager は、まずスレーブドメインについて、マスター配列によって指定されたすべてのパスを最初から最後まで検索します。途中で依存サイクルが見つかると、エラーとして報告されます。

次の例は、依存サイクルがどのように作成されるかを示します。最初のコマンドは、mohawk というスレーブドメインを作成します。このドメインは、マスタードメインに primary を指定します。その結果、mohawk は、次のような依存関係の連鎖で primary に依存します。

図 11–1　単一のドメインの依存関係

2 つめのコマンドは、primary というスレーブドメインを作成します。このドメインは、マスタードメインに counter を指定します。その結果、次のような依存関係の連鎖で、mohawk が primary に依存し、primary が counter に依存します。

図 11–2　複数のドメインの依存関係

3 つめのコマンドは、counter ドメインと mohawk ドメインとの間に依存関係の作成を試みます。これによって、次のような依存サイクルが生成されます。

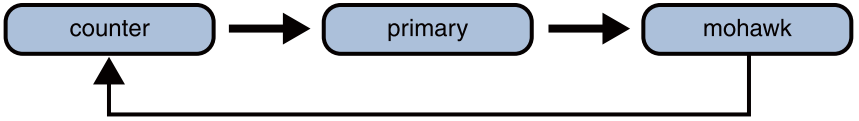

図 11–3　ドメインの依存サイクル

次のエラーメッセージが表示されて ldm set-domain コマンドが失敗します。

```
# 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
```

CPU およびメモリーアドレスのマッピングによるエラー発生箇所の確認

この節では、Solaris の障害管理アーキテクチャー (FMA) によって報告される情報と、障害の発生が示されている論理ドメインリソースを関係付ける方法について説明します。

FMA では、物理 CPU 番号に関する CPU エラーと、物理メモリーアドレスに関するメモリーエラーを報告します。

エラーが発生した論理ドメインと、そのドメイン内の対応する仮想 CPU 番号または実メモリーアドレスを確認する場合は、マッピングを実行する必要があります。

CPU マッピング

ドメインとそのドメイン内の仮想 CPU 番号は、特定の物理 CPU 番号に対応しており、次の手順を使用して確認できます。

▼ CPU 番号を確認する

1 すべてのドメインの解析可能な長形式のリストを生成します。

```
primary# ldm list -l -p
```

2 リストの VCPU セクションで、物理 CPU 番号に等しい pid フィールドを持つエントリを探します。

- このようなエントリが見つかった場合、CPU はそのエントリが表示されたドメインに存在し、そのドメイン内の仮想 CPU 番号がエントリの vid フィールドに指定されています。
- このようなエントリが見つからない場合、CPU はどのドメインにも存在しません。

メモリーのマッピング

ドメインとそのドメイン内の実メモリーアドレスは、特定の物理メモリーアドレス (PA) に対応しており、次のように確認できます。

▼ 実メモリーアドレスを確認する

1 すべてのドメインの解析可能な長形式のリストを生成します。

```
primary# ldm list -l -p
```

2 リストの MEMORY セクションの行を探します。この場合、**PA** は *pa* から (*pa* + *size* - **1**) の包括範囲内にあります。つまり、*pa* <= **PA** < (*pa* + *size* - **1**) です。
ここでの *pa* と *size* は、その行の対応するフィールドの値を指します。

- このようなエントリが見つかった場合、**PA** はそのエントリが表示されたドメインに存在し、そのドメイン内の対応する実アドレスが *ra* + (**PA** - *pa*) によって求められます。
- このようなエントリが見つからない場合、**PA** はどのドメインにも存在しません。

CPU およびメモリーのマッピングの例

例 11–1 に示すような論理ドメインの構成があり、物理 CPU 番号 5 に対応するドメインと仮想 CPU、および物理アドレス 0x7e816000 に対応するドメインと実アドレスを確認すると仮定します。

リストで pid フィールドが 5 である VCPU エントリを探すと、論理ドメイン ldg1 の下に次のエントリが見つかります。

```
|vid=1|pid=5|util=29|strand=100
```

したがって、物理 CPU 番号 5 はドメイン ldg1 に存在し、そのドメイン内には仮想 CPU 番号 1 があります。

リストの MEMORY エントリを探すと、ドメイン ldg2 の下に次のエントリが見つかります。

```
ra=0x8000000|pa=0x78000000|size=1073741824
```

この場合、0x78000000 <= 0x7e816000 <= (0x78000000 + 1073741824 - 1)、つまり、*pa* <= PA <= (*pa* + *size* - 1) となります。したがって、PA はドメイン ldg2 にあり、対応する実アドレスは 0x8000000 + (0x7e816000 - 0x78000000) = 0xe816000 です。

例 **11–1** 論理ドメイン構成の解析可能な長形式のリスト

```
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
```

例 11–1 論理ドメイン構成の解析可能な長形式のリスト (続き)

```
|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
...
```

◆ ◆ ◆ 第 12 章

Logical Domains Manager での XML インタフェースの使用

この章では、外部ユーザープログラムが Logical Domains ソフトウェアとやり取り可能な eXtensible Markup Language (XML) の通信機構について説明します。ここで取り上げる基本事項は、次のとおりです。

Logical Domains Manager で使用する各種スキーマの詳細は、付録 A 「XML スキーマ」 を参照してください

XML トランスポート

外部プログラムは、eXtensible Messaging and Presence Protocol (XMPP – RFC 3920) を使用して、Logical Domains Manager と通信できます。XMPP は、ローカル接続とリモート接続の両方でサポートされており、デフォルトで有効です。リモート接続を切断するには、`ldmd/xmpp_enabled` SMF プロパティーを `false` に設定し、Logical Domains Manager を再起動します。

```
# svccfg -s ldom/ldmd setprop ldmd/xmpp_enabled=false
# svcadm refresh ldmd
# svcadm restart ldmd
```

XMPP サーバー

Logical Domains Manager は、数多くの利用可能な XMPP クライアントアプリケーションおよびライブラリと通信できる XMPP サーバーを実装しています。LDoms Manager は次のセキュリティー機構を使用しています。

- クライアントと LDoms Manager 自身の間の通信チャネルをセキュリティー保護するための Transport Layer Security (TLS)。
- 認証用の Simple Authentication and Security Layer (SASL)。唯一サポートされている SASL 機構は `PLAIN` です。監視操作や管理操作を可能にするには、サーバーが承認できるようにユーザー名およびパスワードをサーバーに送信する必要があります。

ローカル接続

LDoms Manager は、ユーザークライアントが LDoms Manager 自身と同じドメインで動作しているかどうかを検出し、同じドメインである場合はこのクライアントとの間で最小限の XMPP ハンドシェークを行います。具体的には、TLS を介したセキュアチャネルの設定後の SASL 認証手順がスキップされます。認証および承認は、クライアントインタフェースを実装しているプロセスの資格に基づいて行われます。

クライアントは、フル XMPP クライアントを実装することも、単に `libxml2` Simple API for XML (SAX) パーサーなどのストリーミング XML パーサーを実行することも選択できます。いずれの場合も、クライアントは XMPP ハンドシェークを TLS ネゴシエーションまで処理する必要があります。必要な手順については、XMPP の仕様を参照してください。

XML プロトコル

通信の初期化が完了すると、次に LDoms 定義の XML メッセージが送信されます。XML メッセージには、次の 2 つの一般的なタイプがあります。

- `<LDM_interface>` タグを使用する要求メッセージと応答メッセージ。このタイプの XML メッセージは、コマンドの伝達と、LDoms Manager からの結果の取得に使用されます。これはコマンド行インタフェース (CLI) を使用したコマンドの実行に類似しています。このタグは、イベントの登録および登録解除にも使用されます。
- `<LDM_event>` タグを使用するイベントメッセージ。このタイプの XML メッセージは、LDoms Manager によって送信されたイベントを非同期に報告するために使用されます。

要求メッセージと応答メッセージ

LDoms の XML インタフェースには、次の異なる 2 つの形式があります。

- LDoms Manager にコマンドを送信するための形式。
- 受信メッセージの状態およびこのメッセージ内で要求されている処理に基づいて LDoms Manager が応答するための形式。

この 2 つの形式の XML 構造の多くは共通していますが、両者の違いを理解しやすくするために、ここでは別々に取り扱います。また、このドキュメントには、受信 XML と送信 XML の組み合わせを詳しく記述した XML スキーマも記載します (215 ページの「LDM_Event XML スキーマ」 を参照)。

要求メッセージ

LDoms Manager への受信 XML 要求には、もっとも基本的なレベルで、1 つのオブジェクトで動作する 1 つのコマンドの記述が含まれています。要求が複雑になると、1 つのコマンドで複数のコマンドと複数のオブジェクトを処理できます。基本的な XML コマンドの構造は次のとおりです。

例 12–1　1 つのオブジェクトで動作する 1 つのコマンドの形式

```
<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> タグ

LDoms Manager に送信するすべてのコマンドは、<LDM_interface> タグで始まる必要があります。LDoms Manager に送信するドキュメントでは、ドキュメント内に含まれる <LDM_interface> タグは 1 つのみである必要があります。<LDM_interface> タグには、例 12–1 に示すようなバージョン属性が含まれている必要があります。

<cmd> タグ

ドキュメントでは、<LDM_interface> タグ内に 1 つ以上の <cmd> タグが含まれている必要があります。各 <cmd> セクションには、<action> タグを 1 つのみ含める必要があります。この <action> タグは、実行するコマンドを記述するために使用します。各 <cmd> タグに 1 つ以上の <data> タグを含めて、コマンドの処理対象のオブジェクトを記述する必要があります。

また、<cmd> タグには <option> タグも含めることができます。このタグは、一部のコマンドに関連付けられたオプションおよびフラグを指定するために使用されます。次のコマンドにはオプションが使用されます。

- `remove-domain` コマンドには、`-a` オプションを使用できます。
- `stop-domain` コマンドには、`-f` オプションを使用できます。
- `cancel-operation` コマンドには、`migration` または `reconf` オプションを使用できます。
- `add-spconfig` コマンドには、`-r` *autosave-name* オプションを使用できます。
- `remove-spconfig` コマンドには、`-r` オプションを使用できます。
- `list-spconfig` コマンドには、`-r` [*autosave-name*] オプションを使用できます。

<data> タグ

各 <data> セクションには、指定したコマンドに関連するオブジェクトの記述を含めます。データセクションの形式は、Open Virtualization Format (OVF) ドラフト仕様の XML スキーマ部分に基づいています。このスキーマは、<References> タグ (LDoms では未使用)、<Content> セクション、および <Section> セクションを含む <Envelope> セクションを定義します。

LDoms の場合、<Content> セクションは、特定のドメインを指定および記述するために使用されます。<Content> ノードの id= 属性に指定するドメイン名で、ドメインが識別されます。<Content> セクション内には、特定のコマンドの必要に応じて、ドメインのリソースを記述するための <Section> セクションが 1 つ以上あります。

ドメイン名を指定するだけの場合は、<Section> タグを使用する必要はありません。逆に、コマンドでドメイン識別子が不要な場合は、そのコマンドで必要となるリソースを記述した <Section> セクションを、<Content> セクションの外側で、<Envelope> セクションの内側の位置に指定する必要があります。

オブジェクト情報が推測可能な場合は、<data> セクションに <Envelope> タグを含める必要はありません。この状況は主に、ある処理に該当するすべてのオブジェクトの監視要求、イベントの登録および登録解除の要求に当てはまります。

OVF 仕様のスキーマを使用して、すべてのタイプのオブジェクトを適切に定義できるように、さらに 2 つの OVF タイプが定義されています。

- <gprop:GenericProperty> タグ (236 ページの「GenericProperty XML スキーマ」 を参照。)
- <Binding> タグ (236 ページの「Binding_Type XML スキーマ」 を参照。)

<gprop:GenericProperty> タグは、OVF 仕様には定義がないオブジェクトのプロパティーを取り扱うために定義されました。プロパティー名はノードの key= 属性に定義され、プロパティーの値はノードの内容になります。<binding> タグは、ほかのリソースにバインドされたリソースを定義するために、list-bindings サブコマンド出力で使用されます。

応答メッセージ

送信 XML 応答は、含まれているコマンドおよびオブジェクトに関して受信要求と厳密に一致した構造を持ちますが、そのほかに、指定されている各オブジェクトおよび各コマンド用の <Response> セクションと、要求に対する全体の <Response> セクションが追加されています。<Response> セクションでは、例 12–2 に示すような状態およびメッセージ情報が提供されます。基本的な XML 要求に対する応答の構造は、次のとおりです。

例 12–2　1 つのオブジェクトで動作する 1 つのコマンドに対する応答の形式

```
<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>
```

例 12–2　1 つのオブジェクトで動作する 1 つのコマンドに対する応答の形式　　(続き)

```
          <!-- 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>
```

全体の応答

この <response> セクションは、<LDM_interface> セクションの直下の子であり、要求全体の成功または失敗を示します。受信 XML ドキュメントが不正な形式でないかぎり、<response> セクションには、<status> タグだけが含まれます。この応答状態が成功を示している場合、すべてのオブジェクトに対するすべてのコマンドが成功しています。この応答状態が失敗を示し、<resp_msg> タグがない場合は、元の要求内のコマンドのいずれかが失敗しています。<resp_msg> タグは、XML ドキュメント自体の問題を記述する場合にのみ使用されます。

コマンドの応答

<cmd> セクションの下にある <response> セクションは、特定のコマンドの成功または失敗についてユーザーに通知します。<status> タグは、このコマンドが成功したか失敗したかを示します。全体の応答の場合と同様に、コマンドが失敗した場合で、要求の <cmd> セクションの内容の形式が不正なときは、<response> セクションには <resp_msg> タグのみが含まれます。それ以外の場合の失敗状態は、コマンドが実行されたオブジェクトのいずれかが原因で失敗したことを示しています。

オブジェクトの応答

最後に、<cmd> セクション内の各 <data> セクションにも、<response> セクションがあります。ここでは、この特定のオブジェクトで実行されたコマンドが成功したか失敗したかがわかります。応答の状態が SUCCESS の場合、<response> セクション内に

<resp_msg> タグはありません。状態が FAILURE の場合、そのオブジェクトでのコマンドの実行時に発生したエラーに応じて、<response> フィールドには 1 つ以上の <resp_msg> タグがあります。オブジェクトエラーは、コマンドの実行時に検出された問題、または不正な形式または不明なオブジェクトが原因で発生する可能性があります。

<response> セクションのほかに、<data> セクションにその他の情報が含まれていることがあります。この情報は、受信 <data> フィールドと同じ形式で、失敗の原因となったオブジェクトを記述しています。190 ページの「<data> タグ」 を参照してください。この追加情報は、次の場合に特に有用です。

- コマンドの実行が、特定の <data> セクションに対して失敗したが、別の <data> セクションに対しては成功した場合
- 空の <data> セクションがコマンドに渡されて、一部のドメインでは実行に失敗したが、ほかのドメインでは成功した場合

イベントメッセージ

ポーリングの代わりに、特定の状態変化が発生した場合にイベント通知を受信するように登録できます。個々に、または一括して登録できるイベントのタイプは 3 つあります。詳細は、195 ページの「イベントタイプ」 を参照してください。

登録および登録解除

イベントを登録するには、<LDM_interface> メッセージを使用します。190 ページの「<LDM_interface> タグ」 を参照してください。処理タグには登録または登録解除するイベントのタイプを記述し、<data> セクションは空白のままにしておきます。

例 12–3 イベントの登録要求メッセージの例

```
<LDM_interface version="1.0">
  <cmd>
    <action>reg-domain-events</action>
    <data version="3.0"/>
  </cmd>
</LDM_interface>
```

Logical Domains Manager は、登録または登録解除が成功したかどうかを示す <LDM_interface> 応答メッセージで応答します。

例 12–4 イベントの登録応答メッセージの例

```
<LDM_interface version="1.0">
  <cmd>
```

例 12–4 イベントの登録応答メッセージの例 (続き)

```
    <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>
```

各タイプのイベントの処理文字列は、イベントサブセクションにリストされます。

<LDM_event> メッセージ

イベントメッセージの形式は受信 <LDM_interface> メッセージと同じですが、このメッセージの開始タグは <LDM_event> になる点が異なります。メッセージの処理タグは、イベントをトリガーするために実行された処理です。メッセージのデータセクションにはイベントに関連付けられたオブジェクトが記述されます。詳細は、発生したイベントのタイプによって異なります。

例 12–5 <LDM_event> 通知の例

```
<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>
```

例 12–5 <LDM_event> 通知の例 (続き)

```
  </cmd>
</LDM_event>
```

イベントタイプ

次に、登録できるイベントのタイプを示します。

- ドメインイベント
- ハードウェアイベント
- 進捗イベント
- リソースイベント

これらすべてのイベントは、Logical Domains Manager (`ldm`) サブコマンドに対応しています。

ドメインイベント

ドメインイベントは、ドメインに直接実行できる処理を記述します。次の表に、<LDM_event> メッセージの <action> タグにリストされる可能性のあるドメインイベントを示します。

ドメインイベント	ドメインイベント	ドメインイベント
add-domain	remove-domain	bind-domain
unbind-domain	start-domain	stop-domain
domain-reset	panic-domain	migrate-domain

これらのイベントでは、常に、OVF データセクションにイベントが発生したドメインが記述された <Content> タグのみが含まれます。ドメインイベントを登録するには、<action> タグを **reg-domain-events** に設定した <LDM_interface> メッセージを送信します。これらのイベントの登録を解除するには、処理タグを **unreg-domain-events** に設定した <LDM_interface> メッセージが必要です。

ハードウェアイベント

ハードウェアイベントは、物理的なシステムハードウェアの変更に関係しています。LDoms ソフトウェアの場合、実行できるハードウェア変更は、ユーザーがサービスプロセッサ (SP) 構成の追加、削除、または設定を行う場合の SP への変更だけです。現在、このタイプのイベントは次の 3 つだけです。

- `add-spconfig`
- `set-spconfig`
- `remove-spconfig`

ハードウェアイベントでは、常に、OVF データセクションにイベントが発生している SP 構成が記述された `<Section>` タグのみが含まれます。これらのイベントを登録するには、`<action>` タグを **`reg-hardware-events`** に設定した `<LDM_interface>` メッセージを送信します。これらのイベントの登録を解除するには、`<action>` タグを **`unreg-hardware-events`** に設定した `<LDM_interface>` メッセージが必要です。

進捗イベント

進捗イベントは、ドメインの移行など、長時間にわたって実行されるコマンドに対して発行されます。このイベントは、コマンド実行期間中のそれまでの進捗量を報告します。この時点では、`migration-process` イベントのみが報告されます。

進捗イベントでは、常に、OVF データセクションにイベントの影響を受ける SP 構成が記述された `<Section>` タグのみが含まれます。 これらのイベントを登録するには、`<action>` タグを `reg-hardware-events` に設定した `<LDM_interface>` メッセージを送信します。これらのイベントの登録を解除するには、`<action>` タグを `unreg-hardware-events` に設定した `<LDM_interface>` メッセージが必要です。

進捗イベントの `<data>` セクションは、影響を受けるドメインを記述する `<content>` セクションによって構成されています。この `<content>` セクションでは、`ldom_info` `<Section>` タグを使用して進捗を更新します。次の汎用プロパティーが `ldom_info` セクションに表示されます。

- `--progress` – コマンドの進捗の割合
- `--status` – コマンドのステータス。`ongoing`、`failed`、または `done` のいずれか
- `--source` – 進捗を報告しているマシン

リソースイベント

任意のドメインでリソースを追加、削除、または変更すると、リソースイベントが発生します。これらの一部のイベントのデータセクションには、OVF データセクションにサービス名が示されている `<Section>` タグがある、`<Content>` タグが含まれています。次の表に、`<LDM_event>` メッセージの `<action>` タグにリスト可能なイベントを示します。

リソースイベント	リソースイベント
add-vdiskserverdevice	remove-vdiskserverdevice
set-vdiskserverdevice	remove-vdiskserver
set-vconscon	remove-vconscon
set-vswitch	remove-vswitch
remove-vdpcs	

その他のリソースイベントでは、常に、OVF データセクションにイベントが発生したドメインが記述された <Content> タグのみが含まれます。

リソースイベント	リソースイベント	リソースイベント
add-vcpu	add-crypto	add-memory
add-io	add-variable	add-vconscon
add-vdisk	add-vdiskserver	add-vnet
add-vswitch	add-vdpcs	add-vdpcc
set-vcpu	set-crypto	set-memory
set-variable	set-vnet	set-vconsole
set-vdisk	remove-vcpu	remove-crypto
remove-memory	remove-io	remove-variable
remove-vdisk	remove-vnet	remove-vdpcc

リソースイベントを登録するには、<action> タグを **reg-resource-events** に設定した <LDM_interface> メッセージを送信します。これらのイベントの登録を解除するには、<action> タグを **unreg-resource-events** に設定した <LDM_interface> メッセージが必要です。

すべてのイベント

各イベントを個別に登録しないで、3 つのタイプすべてのイベントを待機するように登録することもできます。3 タイプすべてのイベントを同時に登録するには、<action> タグを **reg-all-events** に設定した <LDM_interface> メッセージを送信します。これらのイベントの登録を解除するには、<action> タグを **unreg-all-events** に設定した <LDM_interface> メッセージが必要です。

Logical Domains Manager の処理

<action> タグに指定するコマンドは、*-*-events コマンドを除いて、LDoms コマンド行インタフェースのコマンドに対応しています。Logical Domains Manager (ldm) サブコマンドの詳細は、ldm(1M) マニュアルページを参照してください。

注 – XML インタフェースは、Logical Domains Manager CLI でサポートされている動詞またはコマンドの別名はサポートしていません。

<action> タグでサポートされている文字列は、次のとおりです。

LDoms の処理	LDoms の処理	LDoms の処理
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
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

LDoms の処理	LDoms の処理	LDoms の処理
cancel-operation	set-domain	

Logical Domains Managerのリソースおよびプロパティー

ここでは、Logical Domains Manager のリソースと、リソースごとに定義できるプロパティーを示します。XML の例では、リソースおよびプロパティーは太字で示されています。これらの例は、バインド出力ではなくリソースを示しています。制約出力は、Logical Domains Manager の処理の入力を作成する場合に使用できます。ただし、ドメイン移行の出力は例外です。210 ページの「ドメインの移行」 を参照してください。各リソースは、<Section> の OVF セクションで定義され、<rasd:OtherResourceType> タグによって指定されます。

論理ドメインの情報 (ldom_info) リソース

例 12–6 ldom_info の XML 出力の例

```
<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リソースは、<Content>セクション内に必ず含まれます。ldom_infoリソース内の次のプロパティーは、省略可能です。

- <rasd:Address>タグ。ドメインに割り当てるMACアドレスを指定します。
- <gprop:GenericPropertykey="failure-policy">タグ。マスタードメインに障害が発生した場合のスレーブドメインの動作を指定します。デフォルト値はignoreです。次に、有効なプロパティー値を示します。
  - ignoreは、マスタードメインの障害を無視します。スレーブドメインは影響を受けません。
  - panicは、マスタードメインに障害が発生した場合、すべてのスレーブドメインにパニックを発生させます。
  - resetは、マスタードメインに障害が発生した場合、すべてのスレーブドメインをリセットします。
  - stopは、マスタードメインに障害が発生した場合、すべてのスレーブドメインを停止します。
- <gprop:GenericPropertykey="hostid">タグ。ドメインに割り当てるホストIDを指定します。
- <gprop:GenericPropertykey="master">タグ。最大4つのマスタードメイン名をコンマで区切って指定します。
- <gprop:GenericPropertykey="progress">タグ。コマンドの進捗の割合を指定します。
- <gprop:GenericPropertykey="source">タグ。コマンドの進捗を報告するマシンを指定します。
- <gprop:GenericPropertykey="status">タグ。コマンドの状態(done、failed、またはongoing)を指定します。

CPU (cpu) リソース

例 12–7　cpuのXMLの例

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

cpu リソースは、<Content> セクション内に必ず含まれます。プロパティーは <rasd:AllocationUnits> タグのみで、仮想 CPU の数を指定します。

MAU (mau) リソース

注 – mau リソースとは、LDoms がサポートするサーバー上で LDoms がサポートする任意の暗号化装置です。現在、モジュラー演算ユニット (MAU) と Control Word Queue (CWQ) の 2 つの暗号化装置がサポートされています。

例 12–8 mau の XML の例

```
<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 リソースは、<Content> セクション内に必ず含まれます。プロパティーは <rasd:AllocationUnits> タグのみで、MAU またはその他の暗号化装置の数を指定します。

メモリー (memory) リソース

例 12–9 memory の XML の例

```
<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>
```

メモリーリソースは、<Content>セクション内に必ず含まれます。プロパティーは<rasd:AllocationUnits>タグのみで、メモリーの量を指定します。

仮想ディスクサーバー(vds)リソース

例 12–10 vdsのXMLの例

```
<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>
```

仮想ディスクサーバー(vds)リソースは、ドメイン記述の一部として<Content>セクションに含まれることも、単独で<Envelope>セクションに記述されることもあります。プロパティーは<gprop:GenericProperty>タグのみです。このタグには、"service_name"というキーがあり、記述されるvdsリソースの名前が含まれています。

仮想ディスクサーバーボリューム(vds_volume)リソース

例 12–11 vds_volumeのXMLの例

```
<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>
```

例 12–11 vds_volume のXMLの例 (続き)

```
    </Section>
</Envelope>
```

vds_volume リソースは、ドメイン記述の一部として <Content> セクションに含まれることも、単独で <Envelope> セクションに記述されることもあります。次のキーを持つ <gprop:GenericProperty> タグが必要です。

- vol_name - ボリュームの名前
- service_name - このボリュームをバインドする仮想ディスクサーバーの名前
- block_dev – このボリュームに関連付けるファイルまたはデバイスの名前

任意で、vds_volume リソースに次のプロパティーも設定できます。

- vol_opts – {ro,slice,excl} のように、これらの項目の 1 つ以上がコンマで区切られて、1 つの文字列となっているもの
- mpgroup – マルチパス (フェイルオーバー) グループの名前

ディスク (disk) リソース

例 12–12 disk のXMLの例

```
<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 リソースは、<Content> セクション内に必ず含まれます。次のキーを持つ <gprop:GenericProperty> タグが必要です。

- vdisk_name - 仮想ディスクの名前
- service_name - この仮想ディスクをバインドする仮想ディスクサーバーの名前
- vol_name - この仮想ディスクを関連付ける仮想ディスクサービスデバイス

任意で、disk リソースに timeout プロパティーも含めることができます。このプロパティーは、仮想ディスククライアント (vdc) と仮想ディスクサーバー (vds) の間に接続を確立するためのタイムアウト値です (秒単位)。複数の仮想ディスク (vdisk) パスがある場合、vdc は、別の vds への接続を試みることができます。また、タイムアウトによって、いずれかの vds への接続が指定の時間内に確実に行われます。

仮想スイッチ (vsw) リソース

例 12–13　vsw の XML の例

```
<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 リソースは、ドメイン記述の一部として <Content> セクションに含まれることも、単独で <Envelope> セクションに記載されることもあります。次のキーを持つ <gprop:GenericProperty> タグが必要です。

- service_name - 仮想スイッチに割り当てる名前。
- linkprop – 仮想デバイスが物理リンクステータスの更新を取得するかどうかを指定します。値が phys-state の場合、仮想デバイスは物理リンクステータスの更新を取得します。値が空白の場合、仮想デバイスは物理リンクステータスの更新を取得しません。デフォルトでは、仮想デバイスは物理リンクステータスの更新を取得しません。
- dev_path – この仮想スイッチに関連付けるネットワークデバイスのパス

任意で、vsw リソースに次のプロパティーも設定できます。

- <rasd:Address> - MAC アドレスを仮想スイッチに割り当てます。
- pvid - ポート仮想ローカルエリアネットワーク (VLAN) 識別子 (ID)。仮想ネットワークをメンバーにする必要のある VLAN をタグなしモードで指定します。

- vid - 仮想ローカルエリアネットワーク (VLAN) 識別子 (ID)。仮想ネットワークおよび仮想スイッチをメンバーにする必要のある VLAN をタグ付きモードで指定します。
- mode - SunCluster のハートビートサポートの場合は sc。

ネットワーク (network) リソース

例 12–14 network の XML の例

```
<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 リソースは、<Content> セクション内に必ず含まれます。次のキーを持つ <gprop:GenericProperty> タグが必要です。

- linkprop – 仮想デバイスが物理リンクステータスの更新を取得するかどうかを指定します。値が phys-state の場合、仮想デバイスは物理リンクステータスの更新を取得します。値が空白の場合、仮想デバイスは物理リンクステータスの更新を取得しません。デフォルトでは、仮想デバイスは物理リンクステータスの更新を取得しません。
- vnet_name - 仮想ネットワーク (vnet) の名前
- service_name - この仮想ネットワークをバインドする仮想スイッチ (vswitch) の名前

任意で、network リソースに次のプロパティーも設定できます。

- <rasd:Address> - MAC アドレスを仮想スイッチに割り当てます。
- pvid - ポート仮想ローカルエリアネットワーク (VLAN) 識別子 (ID)。仮想ネットワークをメンバーにする必要のある VLAN をタグなしモードで指定します。

- vid - 仮想ローカルエリアネットワーク (VLAN) 識別子 (ID)。仮想ネットワークおよび仮想スイッチをメンバーにする必要のある VLAN をタグ付きモードで指定します。
- mode - 仮想ネットワークに対してハイブリッド I/O を有効にする場合は hybrid。

仮想コンソール端末集配信装置 (vcc) リソース

例 12–15 vcc の XML の例

```
<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>
```

vcc リソースは、ドメイン記述の一部として <Content> セクションに含まれることも、単独で <Envelope> セクションに記述されることもあります。次のキーを持つ <gprop:GenericProperty> タグを使用できます。

- service_name - 仮想コンソール端末集配信装置サービスに割り当てる名前
- min_port - この vcc に関連付ける最小ポート番号
- max_port - この vcc に関連付ける最大ポート番号

変数 (var) リソース

例 12–16 var の XML の例

```
<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>
```

例 12–16 varのXMLの例 (続き)

```
    </Section>
  </Content>
</Envelope>
```

varリソースは、<Content>セクション内に必ず含まれます。次のキーを持つ<gprop:GenericProperty>タグを使用できます。

- name - 変数の名前
- value - 変数の値

物理I/Oデバイス(physio_device)リソース

例 12–17 physio_deviceのXMLの例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <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リソースは、<Content>セクション内に必ず含まれます。プロパティーは、<gprop:GenericProperty>タグのみです。このタグには、"name"というキープロパティー値があり、記述されるI/Oデバイスの名前が含まれています。

SP構成(spconfig)リソース

例 12–18 spconfigのXMLの例

```
<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>
```

例 12–18 spconfigのXMLの例 (続き)

```
      </Item>
    </Section>
</Envelope>
```

サービスプロセッサ (SP) 構成 (spconfig) リソースは、必ず単独で <Envelope> セクションに記述されます。次のキーを持つ <gprop:GenericProperty> タグを使用できます。

- spconfig_name - SP に格納されている構成の名前。
- spconfig_status - 特定の SP 構成の現在の状態。このプロパティーは、ldm list-spconfig コマンドの出力で使用されます。

仮想データプレーンチャネルサービス (vdpcs) リソース

例 12–19 vdpcsのXMLの例

```
<Envelope>
  <References/>
  <Content xsi:type="ovf:VirtualSystem_Type" id="ldg1">
    <Section xsi:type="ovf:VirtualHardwareSection_Type">
      <Item>
        <rasd:OtherResourceType>vdpcs</rasd:OtherResourceType>
        <gprop:GenericProperty key="service_name">dg1-vdpcs</gprop:GenericProperty>
      </Item>
    </Section>
  </Content>
</Envelope>
```

このリソースは、Netra DPS 環境でのみ意味を持ちます。vdpcs リソースは、ドメイン記述の一部として <Content> セクションに含まれることも、単独で <Envelope> セクションに記述されることもあります。プロパティーは、<gprop:GenericProperty> タグのみです。このタグには、"service_name" というキープロパティー値があり、記述される仮想データプレーンチャネルサービス (vdpcs) リソースの名前が含まれています。

仮想データプレーンチャネルクライアント(vdpcc)リソース

例 12–20 vdpccのXMLの例

```
<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>
```

このリソースは、Netra DPS環境でのみ意味を持ちます。仮想データプレーンチャネルクライアントリソースは、<Content>セクション内に必ず含まれます。次のキーを持つ<gprop:GenericProperty>タグを使用できます。

- vdpcc_name - 仮想データプレーンチャネルクライアント(vdpcc)の名前
- service_name - このvdpccをバインドする仮想データプレーンチャネルサービス(vdpcs)の名前

コンソール(console)リソース

例 12–21 consoleのXMLの例

```
<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>
      </Item>
    </Section>
  </Content>
</Envelope>
```

consoleリソースは、<Content>セクション内に必ず含まれます。次のキーを持つ<gprop:GenericProperty>タグを使用できます。

- port - この仮想コンソール(console)の変更先のポート
- service_name - このconsoleをバインドする仮想コンソール端末集配信装置(vcc)サービス
- group - このconsoleをバインドするグループの名前

ドメインの移行

次の例は、migrate-domainサブコマンドの<data>セクションの内容を示しています。

例 12–22 migrate-domainの<data>セクションの例

```
<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>
```

各表記の意味は次のとおりです。

- 1番めの<Content>ノード(<ldom_info>セクションなし)は、移行元のソースドメインです。
- 2番めの<Content>ノード(<ldom_info>セクションあり)は、移行先のターゲットドメインです。ソースドメインとターゲットドメインの名前は同じにすることができます。
- ターゲットドメインの<ldom_info>セクションには、移行先のマシンおよびこのマシンへの移行に必要な詳細情報が記述されます。
  - target-hostは、移行先のターゲットマシンです。
  - user-nameは、ターゲットマシンのログインユーザー名です。SASL 64ビットで符号化する必要があります。

- `password`は、ターゲットマシンへのログインに使用するパスワードです。SASL 64 ビットで符号化する必要があります。

注 – Logical Domains Manager では、`sasl_decode64()`を使用してターゲットのユーザー名およびパスワードを復号化し、`sasl_encode64()`を使用してこれらの値を符号化します。SASL 64 符号化は、`base64`符号化に相当します。

◆ ◆ ◆ 付 録 A

XML スキーマ

この付録では、Logical Domains Manager で使用するさまざまな XML スキーマを示します。

この章の内容は次のとおりです。

LDM_interface XML スキーマ

このスキーマは、Open Virtualization Format (OVF) Draft Specification version 0.98 のスナップショットです。

例 A–1 LDM_interface XML スキーマ

```
<?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>
```

例 A–1 LDM_interface XML スキーマ (続き)

```
          <xs:documentation>
               Copyright 2007 Sun Microsystems, Inc.  All rights reserved.
               Use is subject to license terms.
          </xs:documentation>
     </xs:annotation>

<!--
==================
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>
```

例 A–1 LDM_interface XML スキーマ (続き)

```
                <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"/>
  </xs:complexType>
</xs:element> <!-- LDM interface document -->

</xs:schema>
```

LDM_Event XML スキーマ

例 A–2 LDM_Event XML スキーマ

```
<?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 2007 Sun Microsystems, Inc.  All rights reserved.
        Use is subject to license terms.
      </xs:documentation>
    </xs:annotation>

  <!-- LDM interface document -->
  <xs:element name="LDM_event">
    <xs:complexType>
      <xs:sequence>
```

例 A–2 LDM_Event XML スキーマ (続き)

```
      <!-- 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>
 </xs:element> <!-- LDM interface document -->

</xs:schema>
```

ovf-envelope.xsd スキーマ

例 A–3 ovf-envelope.xsd スキーマ

```
<?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"/>
```

例 A–3 ovf-envelope.xsd スキーマ (続き)

```
  <!-- 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>
      <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) -->
```

例 A–3 ovf-envelope.xsd スキーマ (続き)

```
    <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 スキーマ

例 A–4 ovf-section.xsd スキーマ

```
<?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 スキーマ

例 A–5 ovf-core.xsd スキーマ

```
<?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"/>
```

例 A–5 ovf-core.xsd スキーマ (続き)

```
<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>
        <!-- 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">
```

例 A–5 ovf-core.xsd スキーマ (続き)

```
          <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>
      <!-- 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"/>
```

例 A–5 ovf-core.xsd スキーマ (続き)

```
      </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) -->
  <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">
```

例 A–5 ovf-core.xsd スキーマ (続き)

```
          <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>
        <!-- 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>
```

例 A–5 ovf-core.xsd スキーマ (続き)

```
  </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">
```

例 A–5 ovf-core.xsd スキーマ (続き)

```
        <xs:sequence>
          <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:attribute name="transport" type="xs:string"/>
        <xs:anyAttribute namespace="##any"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

ovf-virtualhardware.xsc スキーマ

例 A–6 ovf-virtualhardware.xsc スキーマ

```
<?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>
```

例 A–6 ovf-virtualhardware.xsc スキーマ (続き)

```
      </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"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:schema>
```

cim-rasd.xsd スキーマ

例 A–7 cim-rasd.xsd スキーマ

```
<?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">
```

例 A–7 cim-rasd.xsd スキーマ (続き)

```
    <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 -->
    <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"/>
```

例 A–7 cim-rasd.xsd スキーマ (続き)

```
              </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>
            <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>
```

例 A–7 cim-rasd.xsd スキーマ (続き)

```
    </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"/>
              </xs:restriction>
            </xs:simpleType>
          </xs:union>
        </xs:simpleType>
        <xs:anyAttribute namespace="##any"/>
      </xs:restriction>
```

例 A–7 cim-rasd.xsd スキーマ (続き)

```
      </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 スキーマ

例 A–8 cim-vssd.xsd スキーマ

```
<?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 スキーマ

例 A–9 cim-common.xsd スキーマ

```
<?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"/>

  <!-- 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>
```

例 A–9 cim-common.xsd スキーマ (続き)

```
    </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">
        <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">
```

例 A–9 cim-common.xsd スキーマ (続き)

```
      <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">
        <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>
```

例 A–9 cim-common.xsd スキーマ (続き)

```
  <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>

  <xs:complexType name="qualifierSInt64">
    <xs:simpleContent>
      <xs:extension base="cim:cimLong">
        <xs:attribute ref="cim:qualifier" use="required"/>
```

例 A–9 cim-common.xsd スキーマ (続き)

```
      </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 スキーマ

このスキーマは、Open Virtualization Format (OVF) スキーマに対する拡張です。

例 A–10 GenericProperty XML スキーマ

```
<?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 スキーマ

このスキーマは、Open Virtualization Format (OVF) スキーマに対する拡張です。

例 A–11 Binding_Type XML スキーマ

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/Binding"
```

例 A–11 Binding_Type XML スキーマ (続き)

```
  xmlns:class="/var/opt/SUNWldom/Binding"
  xmlns:rasd="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
  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>
```

◆ ◆ ◆ 付 録 B

Logical Domains Manager の検出

マルチキャストメッセージを使用すると、サブネット上で Logical Domains Manager を検出できます。ldmd デーモンは、ネットワーク上で特定のマルチキャストパケットを待機できます。そのマルチキャストメッセージが特定のタイプの場合、ldmd は呼び出し元に対して応答します。これにより、Logical Domains を実行しているシステム上で ldmd を検出できます。

この付録では、サブネット上のシステムで動作している Logical Domains Manager の検出について説明します。

Logical Domains Manager を実行しているシステムの検出

マルチキャスト通信

この検出メカニズムは、ldmd デーモンによって使用されるものと同じマルチキャストネットワークを使用して、MAC アドレスを自動的に割り当てるときに衝突を検出します。マルチキャストソケットを構成するには、次の情報を指定する必要があります。

```
#define    MAC_MULTI_PORT        64535
#define    MAC_MULTI_GROUP        "239.129.9.27"
```

デフォルトでは、マシンが接続されているサブネット上ではマルチキャストパケットのみを送信できます。この動作をは、ldmd デーモンに ldmd/hops SMF プロパティーを設定することによって変更できます。

メッセージ形式

検出メッセージは、他のメッセージと混同しないように明白にマークされている必要があります。次のマルチキャストメッセージ形式により、検出待機プロセスで検出メッセージを識別できます。

```
#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;
```

▼ サブネット上で動作している Logical Domains Manager を検出する

1 マルチキャストソケットを開きます。

239 ページの「マルチキャスト通信」 に示すポートおよびグループの情報を使用していることを確認してください。

2 ソケット経由で `multicast_msg_t` メッセージを送信します。

メッセージには次の内容を含めるようにしてください。

- `version_no` の有効な値 (`MAC_MULTI_VERSION` によって定義されている `1`)
- `magic_no` の有効な値 (`MAC_MULTI_MAGIC_NO` によって定義されている `92792004`)
- `LDMD_DISC_SEND` の `msg_type`

3 マルチキャストソケットで **Logical Domains Manager** からの応答を待機します。

応答は、次の値が含まれる `multicast_msg_t` メッセージである必要があります。

- `version_no` の有効な値
- `magic_no` の有効な値
- `LDMD_DISC_RESP` に設定された `msg_type`
- 次の情報が含まれる、`ldmd_discovery_t` 構造で構成されたペイロード
  - `ldmd_version` – システム上で動作している Logical Domains Manager のバージョン
  - `hostname` – システムのホスト名
  - `ip_address` – システムの IP アドレス
  - `port_no` – Logical Domains Manager によって通信に使用されているポート番号で、XMPP ポート `6482` にする

Logical Domains Manager からの応答を待機する場合、自動割り当て MAC 衝突検出パケットが破棄されていることを確認してください。

◆ ◆ ◆ 付録 C

Logical Domains Physical-to-Virtual 移行ツール

この付録の内容は次のとおりです。

Logical Domains P2V 移行ツールの概要

Logical Domains Physical-to-Virtual (P2V) 移行ツールは、既存の物理システムを、チップマルチスレッディング (CMT) システム上の論理ドメインで動作する仮想システムに自動的に変換します。ソースシステムは、次のいずれかにすることができます。

- Solaris 8 以降のオペレーティングシステムが動作する `sun4u` SPARC システム
- Solaris 10 OS が動作するが、論理ドメインでは動作していない `sun4v` システム

物理システムから仮想システムへの変換は、次のフェーズで実行されます。

- 収集フェーズ。物理ソースシステムで実行されます。`collect` は、ソースシステムに関して収集した構成情報に基づいて、ソースシステムのファイルシステムイメージを作成します。
- 準備フェーズ。ターゲットシステムの制御ドメインで実行されます。`prepare` は、`collect` フェーズで収集された構成情報に基づいて、ターゲットシステムに論理ドメインを作成します。ファイルシステムイメージは、1 つ以上の仮想ディスクに復元されます。このイメージは、論理ドメインとして動作できるように変更されます。
- 変換フェーズ。ターゲットシステムの制御ドメインで実行されます。`convert` フェーズでは、Solaris の標準アップグレード処理を使用して、作成された論理ドメインが Solaris 10 OS で動作する論理ドメインに変換されます。

P2V 移行ツールの詳細は、`ldmp2v(1M)` マニュアルページを参照してください。

次の節からは、物理システムから仮想システムへの変換が各フェーズで実行される方法について説明します。

収集フェーズ

このフェーズは、変換するシステムで実行されます。一貫性のあるファイルシステムイメージを作成するには、システムの動作を最小限に抑えて、すべてのアプリケーションを停止する必要があります。`ldmp2v` は、マウント済みのすべての UFS ファイルシステムのバックアップを作成します。したがって、論理ドメインに移行するすべてのファイルシステムがマウントされていることを確認してください。`-x` を使用すると、マウント済みのファイルシステムを除外できます。

ソースシステムでの変更は不要です。唯一必要なのは、制御ドメインにインストールされた `ldmp2v` スクリプトです。使用するように選択したアーカイブ方式に応じて、`ufsdump` または `flarcreate` ユーティリティーがソースシステムに存在していることを確認してください。

準備フェーズ

準備フェーズでは、収集フェーズで収集されたデータを使用して、ソースシステムに相当する論理ドメインを作成します。

次のいずれかの方法で `ldmp2v prepare` コマンドを使用できます。

- 自動モード。仮想ディスクを自動的に作成し、ファイルシステムデータを復元します。
  - ソースシステム上にあるものと同じサイズで、論理ドメインと必要な仮想ディスクを作成します。
  - ディスクをパーティションに分割し、ファイルシステムを復元します。

    `/`、`/usr`、および `/var` ファイルシステムの合計サイズが 10G バイト未満の場合、これらのファイルシステムのサイズは、Solaris 10 OS の、より大きなディスク容量要件を満たすように自動的に調整されます。`-x no-auto-adjust-fs` オプションを使用するか、`-m` オプションを使用してファイルシステムのサイズを手動で変更することで、自動サイズ変更を無効にできます。
  - 論理ドメインの OS イメージを変更して、物理ハードウェアへのすべての参照を、論理ドメインに適したバージョンに置き換えます。これにより、Solaris の通常のアップグレード処理を使用して、システムを Solaris 10 OS にアップグレードできます。変更には、`/etc/vfstab` ファイルを更新して新しいディスク名を記述することが含まれます。この処理中に、SVM ミラー化ディスクのカプセル化は解除されます。

- 非自動モード。ユーザーが、仮想ディスクを作成してファイルシステムデータを復元する必要があります。 これにより、ディスクのサイズと数、パーティションの分割、およびファイルシステムのレイアウトを変更できます。このモードの準備フェーズでは、*guest-root* をルートに持つファイルシステムでの論理ドメインの作成と OS イメージの変更のみが実行されます。
- クリーンアップモード。`ldmp2v` で作成された論理ドメインと、その配下にあるすべてのバックエンドデバイスを削除します。

変換フェーズ

変換フェーズでは、Solaris のアップグレード処理を使用して論理ドメインが Solaris 10 OS にアップグレードされます。アップグレード処理は、既存のすべてのパッケージを削除し、Solaris 10 `sun4v` パッケージをインストールします。これにより、`sun4u` から `sun4v` への変換は自動的に実行されます。`convert` フェーズでは、Solaris DVD ISO イメージまたはネットワークインストールイメージを使用できます。Custom JumpStart を使用して、完全に自動化された、操作不要のアップグレード処理を実行することもできます。

Logical Domains P2V 移行ツールのインストール

Logical Domains P2V 移行ツールは、制御ドメインのみでインストールおよび構成されている必要があります。ソースシステムとターゲットシステムで共有されているディレクトリに P2V ツールがインストールされていない場合、`bin/ldmp2v` スクリプトをソースシステムにコピーする必要があります。

必要条件

Logical Domains P2V 移行ツールを実行する前に、次の条件を満たしていることを確認してください。

- ターゲットシステムが、次のシステム上で Logical Domains 1.1 以降を実行している
  - Solaris 10 10/08 OS
  - 適切な Logical Domains 1.1 パッチが適用された Solaris 10 5/08 OS
- ゲストドメインが、Solaris 10 5/08 OS 以降を実行している
- ソースシステムが、Solaris 8 OS 以降を実行している

これらの必要条件のほかに、NFS ファイルシステムがソースシステムとターゲットシステムの両方で共有されるように構成する必要があります。このファイルシステムは、`root` が書き込みできるようにしてください。ただし、共有ファイルシステム

を使用できない場合は、ソースシステムとターゲットシステムの両方でソースシステムのファイルシステムダンプ出力を格納できる大きさのローカルファイルシステムを使用します。

制限事項

Logical Domains P2V 移行ツール Version 1.0 には、次の制限事項があります。

- UFS ファイルシステムのみがサポートされています。
- 各ゲストドメインは、仮想スイッチと仮想ディスクサービスを 1 つしか持てません。
- フラッシュアーカイブ方式は、除外されたファイルシステムを、メッセージを表示せずに無視します。

▼ Logical Domains P2V 移行ツールをインストールする

1. **Logical Domains** のダウンロードページ (`http://www.sun.com/servers/coolthreads/ldoms/get.jsp`) に移動します。

2. **P2V** ソフトウェアパッケージ `SUNWldmp2v` をダウンロードします。

   Logical Domains 1.2 リリース以降では、`SUNWldmp2v` パッケージは Logical Domains zip ファイルに同梱されています。

3. スーパーユーザーになるか、同等の役割を取得します。

   役割には、承認および特権付きコマンドが含まれます。役割の詳細は、『Solaris のシステム管理 (セキュリティサービス)』の「RBAC の構成 (作業マップ)」 を参照してください。

4. `pkgadd` コマンドを使用して、`SUNWldmp2v` パッケージをインストールします。

   ```
   # pkgadd -d . SUNWldmp2v
   ```

5. `/etc/ldmp2v.conf` ファイルを作成して、次のプロパティーを構成します。

   - `VDS` – 仮想ディスクサービスの名前。`VDS="primary-vds0"` など
   - `VSW` – 仮想スイッチの名前。`VSW="primary-vsw0"` など
   - `VCC` – 仮想コンソール端末集配信装置の名前。`VCC="primary-vcc0"` など
   - `BACKEND_TYPE` – バックエンドのタイプ。`zvol` または `file`

- BACKEND_SPARSE – バックエンドデバイスをスパースボリュームまたはスパースファイルとして作成する場合は BACKEND_SPARSE="yes"、スパースでないボリュームまたはファイルとして作成する場合は BACKEND_SPARSE="no"
- BACKEND_PREFIX – 仮想ディスクバックエンドデバイスを作成する場所

  BACKEND_TYPE="zvol" の場合、BACKEND_PREFIX 値を ZFS データセット名として指定します。BACKEND_TYPE="files" の場合、BACKEND_PREFIX 値は、/ からの相対的なディレクトリのパス名として解釈されます。

  たとえば、BACKEND_PREFIX="tank/ldoms" の場合、ZVOL は tank/ldoms/*domain-name* データセット、ファイルは /tank/ldoms/*domain-name* サブディレクトリに作成されます。
- BOOT_TIMEOUT – Solaris OS の起動のタイムアウト時間 (秒)

詳細は、ダウンロード可能なバンドルに含まれている ldmp2v.conf.sample 構成ファイルを参照してください。

ldmp2v コマンドの使用

この節では、3 つのフェーズの例を示します。

例 C–1 収集フェーズの例

ldmp2v collect コマンドの使用方法の例を次に示します。

- **NFS** マウント済みファイルシステムを共有する。次の例は、collect 手順の簡単な実行方法を示しています。この場合、ソースシステムとターゲットシステムは、1 つの NFS マウント済みファイルシステムを共有します。

  スーパーユーザーで、必要なすべての UFS ファイルシステムがマウントされていることを確認してください。

```
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
```

  次の例は、ソースシステムとターゲットシステムが 1 つの NFS マウント済みファイルシステムを共有している場合に収集ツールを実行する方法を示しています。

例 C–1 収集フェーズの例 *(続き)*

```
volumia# ldmp2v collect -d /home/dana/p2v/volumia
Collecting system configuration ...
Archiving file systems ...
  DUMP: Writing 63 Kilobyte records
  DUMP: Date of this level 0 dump: vr 28 nov 2008 15:04:03 MET
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/rdsk/c1t1d0s0 (volumia:/) to /home/dana/p2v/ufsdump.0.
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Estimated 950240 blocks (463,98MB).
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
  DUMP: 950164 blocks (463,95MB) on 1 volume at 6215 KB/sec
  DUMP: DUMP IS DONE
  DUMP: Writing 63 Kilobyte records
  DUMP: Date of this level 0 dump: vr 28 nov 2008 15:05:27 MET
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/rdsk/c1t0d0s0 (volumia:/u01) to /home/dana/p2v/ufsdump.1.
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Estimated 282 blocks (141KB).
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
  DUMP: 250 blocks (125KB) on 1 volume at 8928 KB/sec
  DUMP: DUMP IS DONE
  DUMP: Writing 63 Kilobyte records
  DUMP: Date of this level 0 dump: vr 28 nov 2008 15:05:27 MET
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/rdsk/c1t1d0s3 (volumia:/var) to /home/dana/p2v/ufsdump.2.
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Estimated 13324 blocks (6,51MB).
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
  DUMP: 13228 blocks (6,46MB) on 1 volume at 1146 KB/sec
  DUMP: DUMP IS DONE
```

- **NFS** マウント済みファイルシステムを共有しない。ソースシステムとターゲットシステムが 1 つの NFS マウント済みファイルシステムを共有しない場合、ファイルシステムイメージをローカル記憶領域に書き込んだあとで制御ドメインにコピーできます。`ufsdump` を使用してファイルを除外することはできないため、`ldmp2v` が提供するフラッシュアーカイブ方式を使用します。フラッシュツールは、作成したアーカイブを自動的に除外します。

```
volumia# ldmp2v collect -d /home/dana/p2v/volumia -a flash
Collecting system configuration ...
Archiving file systems ...
```

例 C–1 収集フェーズの例 *(続き)*

```
Determining which filesystems will be included in the archive...
Creating the archive...
895080 blocks
Archive creation complete.
```

- ファイルシステムのバックアップステップをスキップする。NetBackup など、他社のバックアップツールを使用することでシステムのバックアップをすでに利用できる場合は、none アーカイブ方式を使用してファイルシステムのバックアップステップをスキップできます。このオプションを使用する場合、システム構成マニフェストのみが作成されます。

  ```
  volumia# ldmp2v collect -d /home/dana/p2v/volumia -a none
  Collecting system configuration ...
  The following file system(s) must be archived manually: / /u01 /var
  ```

  -d で指定するディレクトリが、ソースシステムとターゲットシステムによって共有されていない場合は、そのディレクトリの内容を制御ドメインにコピーします。準備フェーズを開始する前に、ディレクトリの内容を制御ドメインにコピーする必要があります。

例 C–2 準備フェーズの例

ldmp2v prepare コマンドの使用方法の例を次に示します。

- 次の例は、物理システムの MAC アドレスを維持しながら、/etc/ldmp2v.conf に構成されているデフォルトを使用することで、volumia という論理ドメインを作成します。

  ```
  # 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 ...
  ```

- 次のコマンドは、volumia 論理ドメインに関する情報を表示します。

  ```
  # 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
  ```

例 C–2 準備フェーズの例 *(続き)*

```
DISK
    NAME    DEVICE  TOUT MPGROUP        VOLUME                      SERVER
    disk0                                       volumia-vol0@primary-vds0
    disk1                                       volumia-vol1@primary-vds0
```

- 次の例は、-C オプションを使用して、ドメインとそのバックエンドデバイスを完全に削除できることを示しています。

```
# 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 ...
```

- 次の例は、-m オプションを使用してマウントポイントとその新しいサイズを指定することで、P2V の実行中に 1 つ以上のファイルシステムのサイズを変更できることを示しています。

```
# ldmp2v prepare -d /home/dana/p2v/normaal -m /:8g normaal
Resizing file systems ...
Creating vdisks ...
Creating file systems ...
Populating file systems ...
Modifying guest domain OS image ...
Removing SVM configuration ...
Modifying file systems on SVM devices ...
Unmounting guest file systems ...
Creating domain normaal ...
Attaching vdisks to domain normaal ...
```

例 C–3 変換フェーズの例

ldmp2v convert コマンドの使用方法の例を次に示します。

- ネットワークインストールサーバーを使用する。ldmp2v convert コマンドは、指定した仮想ネットワークインタフェースを使用することによってネットワーク経由で Logical Domains を起動します。インストールサーバーで setup_install_server および add_install_client スクリプトを実行する必要があります。

  Custom JumpStart 機能を使用し、完全に操作不要の変換を実行することもできます。この機能では、JumpStart サーバー上のクライアントに対して適切な sysidcfg およびプロファイルファイルを作成および構成する必要があります。プロファイルには次の行を含めるようにしてください。

例 C–3 変換フェーズの例 (続き)

```
install_type    upgrade
root_device     c0d0s0
```

sysidcfg ファイルは、アップグレード処理にのみ使用されます。したがって、次のような構成で十分であるはずです。

```
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}
```

Custom JumpStart の使用については、『Solaris 10 10/09 インストールガイド (カスタムJumpStart/ 上級編)』を参照してください。

```
# 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 1983-2008 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
onfiguring devices.
Using RPC Bootparams for network configuration information.
Attempting to configure interface vnet0...
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...
```

例 C–3 変換フェーズの例 *(続き)*

```
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
[...]
```

- **ISO** イメージを使用する。`ldmp2v convert` コマンドは、Solaris DVD ISO イメージを論理ドメインに関連付け、そこから起動します。アップグレードを行うには、`sysid` のすべての質問に回答し、「Upgrade」を選択します。

例 C–3 変換フェーズの例 (続き)

注 – sysid の質問への回答は、アップグレード処理時にのみ使用されるため、もっとも単純なオプション (ネットワーク接続なし、ネームサービスなし、など) を選択できます。システムの元の ID は、アップグレードによって維持され、アップグレードの完了後に再起動すると有効になります。アップグレードの実行に必要な時間は、元のシステムにインストールされている Solaris クラスタによって異なります。

```
# 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
```

例 C–3 変換フェーズの例 *(続き)*

```
Please make a choice (0 - 9), or press h or ? for help:
[...]
- Solaris Interactive Installation ----------------------------------------

  This system is upgradable, so there are two ways to install the Solaris
  software.

  The Upgrade option updates the Solaris software to the new release, saving
  as many modifications to the previous version of Solaris software as
  possible.  Back up the system before using the Upgrade option.

  The Initial option overwrites the system disks with the new version of
  Solaris software.  This option allows you to preserve any existing file
  systems.  Back up any modifications made to the previous version of Solaris
  software before starting the Initial option.

  After you select an option and complete the tasks that follow, a summary of
  your actions will be displayed.

--------------------------------------------------------------------------------
      F2_Upgrade    F3_Go Back    F4_Initial    F5_Exit    F6_Help
```

◆ ◆ ◆ 付 録 D

Logical Domains Configuration Assistant

Logical Domains Configuration Assistant を使用すると、基本的なプロパティーを設定することによって論理ドメインの構成手順を実行できます。Logical Domains Configuration Assistant は、Sun CoolThreads サーバーと呼ばれるチップマルチスレッディング (CMT) ベースのシステム上で実行されます。

Configuration Assistant は、構成データを収集したあと、論理ドメインとして起動するのに適した構成を作成します。Configuration Assistant によって選択されるデフォルト値を使用して、有効なシステム構成を作成することもできます。

Configuration Assistant は、グラフィカルユーザーインタフェース (GUI) ツールおよび端末ベースのツールである `ldmconfig` の両方として使用できます。

端末ベースのツールについては、256 ページの「Logical Domains Configuration Assistant (`ldmconfig`) の使用」 および `ldmconfig(1M)` マニュアルページを参照してください。

GUI ツールの起動については、255 ページの「Logical Domains Configuration Assistant (GUI) の使用」 を参照してください。

Logical Domains Configuration Assistant (GUI) の使用

Logical Domains Configuration Assistant GUI は、Logical Domains ZIP バンドルの一部として提供されます。

ターゲットシステムで Logical Domains 1.2 ソフトウェア以上が動作していること、および使用中のシステムで Java™ SE Runtime Environment Version 1.6 以上が動作していることを確認してください。

コマンド行から Configuration Assistant GUI を実行するには、次のとおり入力します。

```
$ java -jar "Configurator.jar"
```

この GUI ツールには、システムの構成を作成するのに役立つオンスクリーンマニュアルが含まれています。

Logical Domains Configuration Assistant (ldmconfig) の使用

端末ベースの Configuration Assistant である ldmconfig では、ユーザーインタフェース画面に対応する一連の操作が実行されます。最終的には、論理ドメインに配備可能な構成が作成されます。

以降の節では、ldmconfig コマンドをインストールする方法および Configuration Assistant ツールのいくつかの機能について説明します。

Logical Domains Configuration Assistant のインストール

Logical Domains Configuration Assistant は、SUNWldm パッケージの一部として提供されます。

SUNWldm パッケージをインストールすると、/usr/sbin ディレクトリに ldmconfig コマンドが格納されます。このコマンドは、旧バージョンでの使用のために、/opt/SUNWldm/bin ディレクトリにもインストールされます。

必要条件

Logical Domains Configuration Assistant をインストールおよび実行する前に、次の条件を満たしていることを確認してください。

- ターゲットシステムで Logical Domains 1.2 ソフトウェア以上が動作している。
- 端末ウィンドウに、1 行あたり 80 文字以上で 24 行表示できる。

制限事項および既知の問題

Logical Domains Configuration Assistant には、次の制限事項があります。

- ldmconfig を使用しながら端末のサイズを変更すると、文字化けが発生することがある
- UFS ディスクファイルは仮想ディスクとしてのみサポートされる
- 既存の論理ドメイン構成が存在しないシステムのみで機能する
- 仮想コンソール端末集配信装置のポートは 5000 ～ 5100
- ゲストドメイン、サービス、およびデバイスに使用されるデフォルトの名前は変更不可

ldmconfig の機能

端末ベースの Configuration Assistant である ldmconfig では、ユーザーインタフェース画面に対応する一連の操作が実行されます。最後の手順に到達するまで、後方 (前の手順) および前方 (次の手順) に移動できます。最後の手順では、構成が生成されます。いつでも Configuration Assistant を終了したり、構成をリセットしてデフォルトを使用できます。最後の画面では、論理ドメインに構成を配備できます。

まず、Configuration Assistant は、システムを自動的に検査し、ベストプラクティスに基づいて最適なデフォルトのプロパティーを判断してから、これらのプロパティーのうち配備の制御に必要なプロパティーを表示します。これは完全なリストではないことに注意してください。他のプロパティーを設定して構成をさらにカスタマイズできます。

ldmconfig ツールの使用法については、ldmconfig(1M) マニュアルページを参照してください。

次のプロパティーを調整できます。

- ゲストドメインの数。作成するアプリケーションのゲストドメインの数を指定します。ゲストドメイン数の最小値は 1 です。最大値は、使用できる VCPU リソースによって決まります。たとえば、64 スレッドの CMT システムで、制御ドメイン用に 4 つのスレッドを予約し、各ゲストドメインに 1 つのスレッドを使用して最大 60 個のゲストドメインを作成できます。ベストプラクティスが選択されている場合、ゲストドメインあたりの VCPU リソースの最小数は、1 コア になります。そのため、1 コアあたり 8 スレッドの 8 コアシステムでベストプラクティスが選択されている場合、それぞれ 1 つのコアが割り当てられた最大 7 個のゲストドメインを作成できます。また、制御ドメインにも 1 つのコアが割り当てられます。

  Configuration Assistant は、そのシステムに構成可能なドメインの最大数を表示します。

  Configuration Assistant は次のタスクを実行し、ドメインを作成します。

  - すべてのドメインに対して実行するタスク
    - 5000 ～ 5100 のポートに仮想端末サービスを作成
    - 仮想ディスクサービスを作成
    - 指定されたネットワークアダプタに仮想ネットワークスイッチを作成
    - 仮想端末サーバーデーモンを有効化
  - 各ドメインに対して実行するタスク
    - 論理ドメインを作成
    - ドメインに割り当てられる VCPU を構成
    - ドメインに割り当てられるメモリーを構成
    - 仮想ディスクとして使用する UFS ディスクファイルを作成

- ディスクファイルの仮想ディスクサーバーデバイス (vdsdev) を作成
- ディスクファイルをドメインの仮想ディスク vdisk0 として割り当て
- 指定されたネットワークアダプタの仮想スイッチに接続された仮想ネットワークアダプタを追加
- OBP プロパティー auto-boot?=true を設定
- OBP プロパティー boot-device=vdisk0 を設定
- ドメインをバインド
- ドメインを起動

- デフォルトのネットワーク。新しいドメインで仮想ネットワークに使用するネットワークアダプタを指定します。このアダプタは、システムに存在する必要があります。Configuration Assistant は、現在システムによってデフォルトアダプタとして使用されているアダプタ、およびリンクステータスがアクティブになっているアダプタ (ケーブル接続されているアダプタ) を強調表示します。
- 仮想ディスクのサイズ。それぞれの新しいドメインに仮想ディスクを作成します。これらの仮想ディスクは、ローカルファイルシステムに存在するディスクファイルに基づいて作成されます。このプロパティーは、各仮想ディスクのサイズを G バイト単位で制御します。最小サイズは 8G バイトです。これは、Solaris 10 OS を格納するために必要なおおよそのサイズに基づきます。最大サイズは 100G バイトです。

  Configuration Assistant がすべてのドメインのディスクファイルを格納するのに十分な領域のあるファイルシステムを検出できない場合、エラー画面が表示されます。この場合、アプリケーションを再実行する前に次の操作が必要になることがあります。

  - 仮想ディスクのサイズを減らす
  - ドメインの数を減らす
  - より容量の大きいファイルシステムを追加する
- 仮想ディスクディレクトリ。新しいドメインの仮想ディスクとして作成されるファイルを格納するのに十分な容量のあるファイルシステムを指定します。このディレクトリは、選択するドメインの数、および仮想ディスクのサイズに基づいて指定します。これらのプロパティーの値が変更された場合は、値を再計算して格納先ディレクトリを選択する必要があります。Configuration Assistant は、十分な領域のあるファイルシステムのリストを表示します。ファイルシステム名を指定すると、このファイルシステムに /ldoms/disks というディレクトリが作成され、このディレクトリにディスクイメージが作成されます。
- ベストプラクティス。プロパティーの値にベストプラクティスを使用するかどうかを指定します。

  - yes という値を選択すると、Configuration Assistant によっていくつかの構成プロパティー値にベストプラクティスが使用されます。ベストプラクティスでは、最小値として、ドメインあたり 1 コアという値が適用されます。これにはシステムドメインも含まれます。その結果、ゲストドメインの最大数は、シス

テムに存在するコアの合計数から、システムドメイン用の 1 コアを引いた数に制限されます。たとえば、それぞれ 8 つのコアが割り当てられた 2 ソケット SPARC Enterprise® T5140 の場合、ゲストドメインの最大数はシステムドメインを除いた 15 個となります。

- no という値を選択すると、Configuration Assistant によって、最少で 1 スレッドが割り当てられたドメインの作成が許可されます。ただし、システムドメインのスレッド数は 4 以上に保持されます。

次に、Configuration Assistant は、作成される配備構成の概略を表示します。これには次の情報が含まれます。

- ドメイン数
- 各ゲストドメインに割り当てられる CPU
- 各ゲストドメインに割り当てられるメモリー
- 仮想ディスクのサイズおよび場所
- ゲストドメインの仮想ネットワークサービスに使用されるネットワークアダプタ
- システムによってサービスに使用される CPU およびメモリーの量
- 有効な Solaris OS DVD が識別されると、これを使用して共有仮想 CD-ROM デバイスが作成され、ゲストドメインに Solaris OS をインストールできるようになります。

最後に、Configuration Assistant は、システムを構成して、指定された論理ドメイン配備を作成します。また、実行される処理についての説明と、システムを構成するために実行するコマンドを表示します。この情報は、システムを構成するために必要な ldm コマンドの使用法を理解するのに役立ちます。

注意 – この構成手順に影響を与えたり、このプロセスを中断したりしないでください。システムの構成が不完全になることがあります。

コマンドが正常に終了したら、変更を有効にするためにシステムを再起動してください。

用語集

この一覧は、Logical Domains のドキュメントで使用される用語、略語、および頭字語を定義したものです。

A

ALOM	Advanced Lights Out Manager
API	Application Programming Interface (アプリケーションプログラミングインタフェース)
`auditreduce`	監査証跡ファイルの監査レコードのマージ選択。`auditreduce(1M)` マニュアルページを参照してください。
監査 **(Auditing)**	Solaris OS BSM を使用して、セキュリティーの変更元を識別すること
承認 **(Authorization)**	Solaris OS RBAC を使用して承認を設定すること

B

`bge`	Broadcom BCM57xx デバイスの Broadcom ギガビット Ethernet ドライバ
BSM	Basic Security Module (基本セキュリティーモジュール)
`bsmconv`	BSM の有効化。`bsmconv(1M)` マニュアルページを参照してください。
`bsmunconv`	BSM の無効化。`bsmunconv(1M)` マニュアルページを参照してください。

C

CD	Compact Disc (コンパクトディスク)

CLI	Command-Line Interface (コマンド行インタフェース)
適合性 **(Compliance)**	システムの構成が事前に定義されたセキュリティープロファイルに適合しているかどうかを確認すること
構成 **(Configuration)**	サービスプロセッサ上に保存されている論理ドメイン構成の名前
CMT	Chip MultiThreading (チップマルチスレッディング)
制約 **(Constraints)**	Logical Domains Manager に対する制約とは、特定のドメインに割り当てられる 1 つ以上のリソースです。使用可能なリソースに応じて、ドメインに追加するように要求したすべてのリソースを受け取るか、まったく受け取らないかのいずれかです。
制御ドメイン **(Control Domain)**	ほかの論理ドメインおよびサービスを作成および管理するドメイン
CPU	Central Processing Unit (中央演算処理装置)
CWQ	Control Word Queue の略で、Sun UltraSPARC T2 ベースのプラットフォーム用の暗号化装置

D

DHCP	Dynamic Host Configuration Protocol (動的ホスト構成プロトコル)
DMA	Direct Memory Access (ダイレクトメモリーアクセス)。CPU を使用せずにメモリーとデバイス (ネットワークカードなど) との間でデータを直接転送する機能です。
DMP	Dynamic MultiPathing (Veritas)
DPS	Data Plane Software
DR	Dynamic Reconfiguration (動的再構成)
`drd`	Logical Domains Manager (Solaris 10 OS) の動的再構成デーモン。`drd(1M)` マニュアルページを参照してください。
DS	Domain Service module (ドメインサービスモジュール)(Solaris 10 OS)
DVD	Digital Versatile Disc (デジタル多用途ディスク)

E

EFI	Extensible Firmware Interface (拡張ファームウェアインタフェース)
ETM	Encoding Table Management (エンコーディングテーブル管理) モジュール (Solaris 10 OS)

F

FC_AL Fiber Channel Arbitrated Loop (ファイバチャネル調停ループ)

FMA Fault Management Architecture (障害管理アーキテクチャー)

`fmd` 障害管理デーモン (Solaris 10 OS)。`fmd(1M)` マニュアルページを参照してください。

`format` ディスクのパーティション分割および保守ユーティリティー。`format(1M)` マニュアルページを参照してください。

`fmthard` ハードディスクのラベルの生成。`fmthard(1M)` マニュアルページを参照してください。

FTP File Transfer Protocol (ファイル転送プロトコル)

G

Gb Gigabit (ギガビット)

ゲストドメイン **(Guest Domain)** I/O ドメインおよびサービスドメインのサービスを使用し、制御ドメインによって管理されます。

GLDv3 Generic LAN Driver version 3 (汎用 LAN ドライバ version 3)

H

強化 **(Hardening)** セキュリティーを向上するために Solaris OS の構成を変更すること

HDD Hard Disk Drive (ハードディスクドライブ)

ハイパーバイザ **(Hypervisor)** オペレーティングシステムとハードウェア層の間に配置されるファームウェア層

I

I/O ドメイン (I/O Domain) 物理 I/O デバイスに対する直接の所有権と直接のアクセス権を持ち、仮想デバイスの形式でほかの論理ドメインとこれらのデバイスを共有するドメイン

IB InfiniBand

IDE Integrated Drive Electronics

IDR Interim Diagnostics Release

ILOM Integrated Lights Out Manager

`io` 内部ディスクおよびPCI-Eコントローラと、それらに接続されたアダプタやデバイスなどのI/Oデバイス

`ioctl` input/output control call (I/O制御コール)

IP Internet Protocol (インターネットプロトコル)

IPMP Internet Protocol Network Multipathing (インターネットプロトコルネットワークマルチパス)

ISO International Organization for Standardization (国際標準化機構)

K

`kaio` Kernel Asynchronous Input/Output (カーネル非同期I/O)

KB KiloByte (Kバイト)

KU Kernel Update (カーネル更新)

L

LAN Local-Area Network (ローカルエリアネットワーク)

LDAP Lightweight Directory Access Protocol

LDC Logical Domain Channel (論理ドメインチャネル)

`ldm` Logical Domain Managerユーティリティー。`ldm(1M)`マニュアルページを参照してください。

`ldmd` Logical Domains Managerデーモン

`lofi` ループバックファイル

論理ドメイン **(Logical Domain)** 1つのコンピュータシステム内で独自のオペレーティングシステムおよびIDを持つ、リソースの個別の論理グループで構成される仮想マシン

Logical Domains (LDoms) Manager 論理ドメインを作成および管理したり、リソースをドメインに割り当てたりするためのCLI

LUN Logical Unit Number (論理ユニット番号)

M

MAC Media Access Control address (メディアアクセス制御アドレス) の略で、Logical Domains によって自動的に割り当てることも、手動で割り当てることも可能

MAU Modular Arithmetic Unit (モジュラー演算ユニット) の略

MB MegaByte (M バイト)

MD サーバーデータベース内のマシン記述

`mem、memory` メモリー単位 - バイト単位でのデフォルトのサイズ。G バイト (G)、K バイト (K)、または M バイト (M) を指定することもできます。ゲストドメインに割り当てることができる、サーバーの仮想化されたメモリーです。

`metadb` SVM メタデバイス状態データベースの複製の作成および削除。`metadb(1M)` マニュアルページを参照してください。

`metaset` ディスクセットの構成。`metaset(1M)` マニュアルページを参照してください。

`mhd` 多重ホストディスク制御操作。`mhd(7i)` マニュアルページを参照してください。

MIB Management Information Base (管理情報ベース)

最小化 **(Minimizing)** 最低限必要な数のコア Solaris OS パッケージをインストールすること

MMF MultiMode Fiber (マルチモードファイバ)

MMU Memory Management Unit (メモリー管理ユニット)

`mpgroup` 仮想ディスクフェイルオーバーのマルチパスグループ名

`mtu` Maximum Transmission Unit (最大転送単位)

N

NAT Network Address Translation (ネットワークアドレス変換)

`ndpsldcc` Netra DPS Logical Domain Channel Client。「`vdpcc`」も参照してください。

`ndpsldcs` Netra DPS Logical Domain Channel Service。「`vdpcs`」も参照してください。

NFS Network File System (ネットワークファイルシステム)

NIS Network Information Service (ネットワーク情報サービス)

NIU Network Interface Unit (ネットワークインタフェースユニット)(Sun SPARC Enterprise T5120 および T5220 サーバー)

NTS Network Terminal Server (ネットワーク端末サーバー)

NVRAM Non-Volatile Random-Access Memory (非揮発性ランダムアクセスメモリー)

`nxge` Sun x8 Express 1/10G Ethernet アダプタ用のドライバ

O

OS Operating System (オペレーティングシステム)

OVF Open Virtualization Format

P

P2V Logical Domains Physical-to-Virtual 移行ツール

PA Physical Address (物理アドレス)

PCI Peripheral Component Interconnect バス

PCI-E PCI Express バス

PCI-X PCI 拡張バス

`pcpu` 物理 CPU

`physio` 物理入出力

PICL Platform Information and Control Library (プラットフォーム情報とコントロールライブラリ)

`picld` PICL デーモン。`picld(1M)` マニュアルページを参照してください。

PM 仮想 CPU の Power Management (電源管理)

`praudit` 監査証跡ファイルの内容の出力。`praudit(1M)` マニュアルページを参照してください。

PRI PRIority (優先度)

R

RA Real Address (実アドレス)

RAID Redundant Array of Inexpensive Disks

RBAC Role-Based Access Control (役割に基づくアクセス制御)

RPC Remote Procedure Call (遠隔手続き呼び出し)

S

SASL Simple Authentication and Security Layer

SAX Simple API for XML パーサー。XML ドキュメントをトラバースします。SAX パーサーはイベントベースで、主にストリーミングデータに使用されます。

システムコントローラ **(System Controller、 SC)** 「サービスプロセッサ」も参照してください。

SCSI Small Computer System Interface

サービスドメイン **(Service Domain)** 仮想スイッチ、仮想コンソールコネクタ、仮想ディスクサーバーなどのデバイスをほかの論理ドメインに提供する論理ドメイン

SMA System Management Agent (システム管理エージェント)

SMF Service Management Facility (サービス管理機能)

SNMP Simple Network Management Protocol (簡易ネットワーク管理プロトコル)

サービスプロセッサ **(Service Processor、 SP)** 「システムコントローラ」も参照してください。

SSH Secure Shell

`ssh` Secure Shell コマンド。`ssh(1)` マニュアルページを参照してください。

`sshd` Secure Shell デーモン。`sshd(1M)` マニュアルページを参照してください。

SunVTS Sun Validation Test Suite

`svcadm` サービスインスタンスの操作。`svcadm(1M)` マニュアルページを参照してください。

SVM Solaris Volume Manager (Solaris ボリュームマネージャー)

T

TCP Transmission Control Protocol (伝送制御プロトコル)

TLS Transport Layer Security

U

UDP User Diagram Protocol (ユーザーダイアグラムプロトコル)

UFS UNIX File System (UNIX ファイルシステム)

ユニキャスト **(Unicast)** 1 つの送信元と 1 つの受信先との間でネットワークを介して行われる通信

USB Universal Serial Bus (ユニバーサルシリアルバス)

`uscsi` ユーザー SCSI コマンドインタフェース。`uscsi(7I)` マニュアルページを参照してください。

UTP Unshielded Twisted Pair (シールドなし・より対線)

V

`var` 変数

VBSC Virtual Blade System Controller (仮想ブレードシステムコントローラ)

`vcc、vconscon` 特定のポート範囲をゲストドメインに割り当てる仮想コンソール端末集配信装置サービス

`vcons、vconsole` システムレベルのメッセージにアクセスするための仮想コンソール。接続は、特定のポートで制御ドメイン上の `vconscon` サービスに接続することによって実現します。

`vcpu` Virtual Central Processing Unit (仮想中央演算処理装置)。サーバーの各コアは、仮想 CPU として表現されます。たとえば、8 コアの Sun Fire T2000 サーバーには、論理ドメイン間で割り当てることができる 32 の仮想 CPU があります。

`vdc` Virtual Disk Client (仮想ディスククライアント)

`vdisk` 仮想ディスク。さまざまな種類の物理デバイス、ボリューム、またはファイルに関連付けられた総称的なブロック型デバイスです。

`vdpcc` Netra DPS 環境における仮想データプレーンチャネルクライアント

`vdpcs` Netra DPS 環境における仮想データプレーンチャネルサービス

`vds、vdiskserver` 仮想ディスクサーバー。これを使用すると、論理ドメインに仮想ディスクをインポートできます。

`vdsdev、vdiskserverdevice` 仮想ディスクサーバーデバイス。仮想ディスクサーバーによってエクスポートされます。このデバイスには、ディスク全体、ディスクのスライス、ファイル、またはディスクボリュームを指定できます。

VLAN Virtual Local Area Network (仮想ローカルエリアネットワーク)

vldc	Virtual Logical Domain Channel Service (仮想論理ドメインチャネルサービス)
vldcc	Virtual Logical Domain Channel Client (仮想論理ドメインチャネルクライアント)
vnet	仮想ネットワークデバイス。仮想 Ethernet デバイスを実装し、仮想ネットワークスイッチ (vswitch) を使用するシステム内のほかの vnet デバイスと通信します。
vntsd	Logical Domains コンソールの仮想ネットワーク端末サーバーデーモン (Solaris 10 OS)。vntsd(1M) マニュアルページを参照してください。
volfs	ボリューム管理ファイルシステム。volfs(7FS) マニュアルページを参照してください。
vsw、vswitch	仮想ネットワークデバイスを外部ネットワークに接続し、仮想ネットワークデバイス間でのパケットの切り替えも行う仮想ネットワークスイッチ。
VTOC	Volume Table Of Contents (ボリューム構成テーブル)
VxDMP	Veritas Dynamic MultiPathing
VxVM	Veritas Volume Manager

W

WAN	Wide-Area Network (広域ネットワーク)

X

XFP	eXtreme Fast Path
XML	eXtensible Markup Language
XMPP	eXtensible Messaging and Presence Protocol

Z

ZFS	Zettabyte File System (Solaris 10 OS)
zpool	ZFS ストレージプール。zpool(1M) マニュアルページを参照してください。
ZVOL	ZFS ボリュームエミュレーションドライバ

索引