Logical Domains (LDoms) 1.1
管理ガイド

Sun Microsystems, Inc.
www.sun.com

Part No. 820-5386-10
2009 年 1 月, Revision A

コメントの送付: http://www.sun.com/hwdocs/feedback

原典: *Logical Domains (LDoms) 1.1 Administration Guide*

Part No: 820-4913-10

Revision A

目次

はじめに　xvii

1. Logical Domains ソフトウェアの概要　1
   ハイパーバイザと論理ドメイン　1
   Logical Domains Manager　3
      論理ドメインの役割　4
      コマンド行インタフェース　4
      仮想入出力　5
         仮想ネットワーク　5
         仮想ストレージ　6
         仮想コンソール　6
      動的再構成　6
      遅延再構成　6
      持続的な構成　8

2. セキュリティー　9
   セキュリティー上の考慮事項　9
   Solaris Security Toolkit および Logical Domains Manager　10
      強化　11
      論理ドメインの最小化　12

承認　13

監査　14

適合性　14

3.　ソフトウェアのインストールおよび有効化　15

Logical Domains をすでに使用しているシステムのアップグレード　16

Solaris OS のアップグレード　16

Logical Domains の制約データベースファイルの保存および復元　16

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　17

Solaris 10 5/08 OS より前の Solaris 10 OS からのアップグレード　17

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　17

▼　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　18

LDoms 1.1 ソフトウェアへのアップグレード　18

▼　LDoms 1.0 ソフトウェアからアップグレードする　18

▼　LDoms 1.0.1、1.0.2、または 1.0.3 からアップグレードする　20

新しいシステムへの Logical Domains ソフトウェアのインストール　21

Solaris OS のアップグレード　21

システムファームウェアのアップグレード　22

▼　システムファームウェアをアップグレードする　22

▼　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　23

Logical Domains Manager および Solaris Security Toolkit のダウンロード　24

▼　ソフトウェアをダウンロードする　24

Logical Domains Manager および Solaris Security Toolkit のインストール　25

Logical Domains Manager および Solaris Security Toolkit ソフトウェアの自動インストール　25

JumpStart を使用した Logical Domains Manager 1.1 および Solaris Security Toolkit 4.2 ソフトウェアのインストール　32

Logical Domains Manager および Solaris Security Toolkit ソフトウェアの手動インストール　35

Logical Domains Manager デーモンの有効化　37

▼ Logical Domains Manager デーモンを有効にする　38

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　38

ユーザー承認の管理　39

ユーザープロファイルの管理　40

ユーザーへの役割の割り当て　40

出荷時デフォルト構成と Logical Domains の無効化　42

▼ すべてのゲスト論理ドメインを削除する　42

▼ 出荷時デフォルト構成を復元する　43

▼ Logical Domains Manager を無効にする　43

▼ Logical Domains Manager の削除　44

▼ システムコントローラから出荷時デフォルト構成を復元する　44

4. サービスと論理ドメインの設定　45

出力メッセージ　45

Sun UltraSPARC T1 プロセッサ　45

Sun UltraSPARC T2 および T2 Plus プロセッサ　46

デフォルトのサービスの作成　46

▼ デフォルトのサービスを作成する　46

制御ドメインの初期構成　48

▼ 制御ドメインを設定する　48

論理ドメインを使用するための再起動　50

▼ 再起動する　50

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　50

▼ 仮想スイッチを主インタフェースとして構成する 51
仮想ネットワーク端末サーバーデーモンの有効化 52
▼ 仮想ネットワーク端末サーバーデーモンを有効にする 52
ゲストドメインの作成と起動 53
▼ ゲストドメインを作成および起動する 53
ゲストドメインへの Solaris OS のインストール 57
▼ DVD からゲストドメインに Solaris OS をインストールする 57
▼ 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 60
▼ ゲストドメインの JumpStart を実行する 62
将来の再構築用の論理ドメイン構成の保存 63
▼ すべての論理ドメイン構成を保存する 63
▼ ゲストドメイン構成を再構築する 63
制御ドメインの再構築 64
論理ドメインの情報 (ldom_info) セクション 66
暗号化 (mau) セクション 67
CPU (cpu) セクション 67
メモリー (memory) セクション 68
物理入出力 (physio_device) セクション 68
仮想スイッチ (vsw) セクション 69
仮想コンソール端末集配信装置 (vcc) セクション 70
仮想ディスクサーバー (vds) セクション 70
仮想ディスクサーバーデバイス (vdsdev) セクション 71

5. Logical Domains ソフトウェアでの PCI バスの使用 73
複数の論理ドメインにまたがる PCI Express バスの構成 73
▼ 分割 PCI 構成を作成する 74
PCI バスでの I/O MMU バイパスモードの有効化 77

6. Logical Domains での仮想ディスクの使用 79

仮想ディスクの概要　79
仮想ディスクの管理　80
▼　仮想ディスクを追加する　80
▼　仮想ディスクバックエンドを複数回エクスポートする　81
▼　仮想ディスクオプションを変更する　82
▼　タイムアウトオプションを変更する　82
▼　仮想ディスクを削除する　82
仮想ディスクの表示　83
フルディスク　83
1 つのスライスディスク　83
仮想ディスクバックエンドオプション　84
読み取り専用 (ro) オプション　84
排他 (excl) オプション　84
スライス (slice) オプション　85
仮想ディスクバックエンド　86
物理ディスクまたはディスクの LUN　86
▼　物理ディスクを仮想ディスクとしてエクスポートする　86
物理ディスクスライス　87
▼　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　87
▼　スライス 2 をエクスポートする　88
ファイルおよびボリューム　88
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　89
▼　ファイルをフルディスクとしてエクスポートする　89
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　90
▼　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　91
ボリュームのエクスポートおよび下位互換性　91
各種のバックエンドのエクスポート方法の概要　93

ガイドライン　93
仮想ディスクマルチパスの構成　94
▼ 仮想ディスクマルチパスを構成する　95
CD、DVD および ISO イメージ　96
▼ 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　97
仮想ディスクのタイムアウト　99
仮想ディスクおよび SCSI　100
仮想ディスクおよび `format`(1M) コマンド　100
仮想ディスクと ZFS の使用　101
サービスドメインでの ZFS プールの構成　101
ZFS を使用したディスクイメージの格納　101
ZFS によるディスクイメージの格納例　102
▼ ZFS ボリュームを使用してディスクイメージを作成する　102
▼ ZFS ファイルを使用してディスクイメージを作成する　102
▼ ZFS ボリュームをエクスポートする　103
▼ ZFS ファイルをエクスポートする　103
▼ 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　103
ディスクイメージのスナップショットの作成　103
▼ ディスクイメージのスナップショットを作成する　104
複製を使用して新規ドメインをプロビジョニングする　104
起動ディスクイメージの複製　104
論理ドメイン環境でのボリュームマネージャーの使用　106
ボリュームマネージャーでの仮想ディスクの使用　106
SVM での仮想ディスクの使用　107
VxVM のインストール時の仮想ディスクの使用　108
仮想ディスクでのボリュームマネージャーの使用　109
仮想ディスクでの ZFS の使用　109

仮想ディスクでの SVM の使用　109
仮想ディスクでの VxVM の使用　109

7. Logical Domains での仮想ネットワークの使用　111
仮想ネットワークの概要　111
仮想スイッチ　111
仮想ネットワークデバイス　112
仮想スイッチの管理　113
▼ 仮想スイッチを追加する　113
▼ 既存の仮想スイッチのオプションを設定する　114
▼ 仮想スイッチを削除する　115
仮想ネットワークデバイスの管理　115
▼ 仮想ネットワークデバイスを追加する　115
▼ 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　116
▼ 仮想ネットワークデバイスを削除する　117
仮想ネットワークデバイスに対応する Solaris ネットワークインタフェース名の判定　117
▼ Solaris OS ネットワークインタフェース名を確認する　117
自動または手動による MAC アドレスの割り当て　118
Logical Domains ソフトウェアに割り当てられる MAC アドレスの範囲　119
自動割り当てのアルゴリズム　120
重複した MAC アドレスの検出　121
解放された MAC アドレス　121
LDoms でのネットワークアダプタの使用　122
▼ ネットワークアダプタが GLDv3 準拠かどうかを判別する　122
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　123
▼ 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　124
論理ドメイン環境での IPMP の構成　125
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　125

▼ ホストルートを構成する 127
サービスドメインでの IPMP の構成と使用 127
Logical Domains ソフトウェアでの VLAN のタグ付けの使用 128
ポート VLAN ID (PVID) 129
VLAN ID (VID) 129
▼ 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 130
NIU ハイブリッド I/O の使用 131
▼ NIU ネットワークデバイスで仮想スイッチを構成する 133
▼ ハイブリッドモードを有効にする 134
▼ ハイブリッドモードを無効にする 134

8. 論理ドメインの移行 135
論理ドメインの移行の概要 135
移行処理の概要 135
ソフトウェアの互換性 136
認証 137
アクティブなドメインの移行 137
CPU 137
メモリー 138
物理入出力 138
仮想入出力 138
NIU ハイブリッド入出力 139
暗号化装置 139
遅延再構成 139
ほかのドメインの操作 139
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 140
CPU 140
仮想入出力 140

予行演習の実行　140
進行中の移行の監視　141
進行中の移行の取り消し　141
移行の失敗からの回復　142
例　142

9. その他の情報とタスク　145
LDoms 1.1 ソフトウェアでの CPU Power Management の使用　145
LDoms 1.1 ソフトウェアでの CPU で電源管理されているストランドの表示　146
▼ CPU で電源管理されているストランドを一覧表示する　146
▼ 電源管理されている CPU を一覧表示する　147
CLI での名前の入力　148
ファイル名 (*file*) と変数名 (*var_name*)　148
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　148
構成名 (*config_name*)　149
その他のすべての名前　149
論理ドメインのリソースの一覧表示　149
マシンが読み取り可能な出力　149
▼ `ldm` サブコマンドの構文の使用法を表示する　149
フラグの定義　153
利用統計情報の定義　154
さまざまなリストの例　154
▼ ソフトウェアのバージョンを表示する (`-V`)　154
▼ 省略形式のリストを生成する　155
▼ 長形式のリストを生成する (`-l`)　155
▼ 拡張リストを生成する (`-e`)　157
▼ 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`)　159
▼ 長形式のリストのサブセットを生成する (`-o` *format*)　159

▼ 変数を一覧表示する 161
▼ バインドを一覧表示する 161
▼ 構成を一覧表示する 162
▼ デバイスを一覧表示する 163
▼ 使用可能なメモリーを一覧表示する 164
▼ サービスを一覧表示する 165
制約の一覧表示 165
▼ 1 つのドメインの制約を一覧表示する 166
▼ 制約を XML 形式で一覧表示する 167
▼ 制約をマシンが読み取り可能な形式で一覧表示する 168
ネットワークを介したゲストコンソールへの接続 168
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 169
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 170
CPU マッピング 170
▼ CPU 番号を確認する 170
メモリーのマッピング 170
▼ 実メモリーアドレスを確認する 171
CPU およびメモリーのマッピングの例 171
コンソールグループの使用 173
▼ 複数のコンソールを 1 つのグループにまとめる 173
論理ドメインを使用した Solaris OS の操作 174
ドメイン化を有効にした場合、Solaris OS の起動後に OpenBoot ファームウェアを使用できない 174
サーバーの電源の再投入 175
▼ 現在の論理ドメイン構成を SC に保存する 175
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 175
Solaris OS のブレークの結果 175
制御ドメインの停止または再起動の結果 176

LDoms と ALOM CMT の使用　177

▼　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　178

BSM 監査の有効化と使用　178

▼　enable-bsm.fin 終了スクリプトを使用する　179

▼　Solaris OS の bsmconv(1M) コマンドを使用する　179

▼　BSM 監査が有効であることを確認する　180

▼　監査を無効にする　180

▼　監査の出力を表示する　180

▼　監査ログをローテーションする　181

10.　Logical Domains Manager での XML インタフェースの使用　183

XML トランスポート　183

XMPP　184

ローカル接続　184

XML プロトコル　184

要求メッセージと応答メッセージ　185

要求　186

応答　188

イベント　190

登録および登録解除　190

<LDM_event> メッセージ　191

イベントタイプ　192

ドメインイベント　192

リソースイベント　193

ハードウェアイベント　194

すべてのイベント　194

Logical Domains Manager の処理　195

Logical Domains Manager のリソースおよびプロパティー　196

論理ドメインの情報 (ldom_info) リソース　196

CPU (cpu) リソース　197
MAU (mau) リソース　197
メモリー (memory) リソース　198
仮想ディスクサーバー (vds) リソース　198
仮想ディスクサーバーボリューム (vds_volume) リソース　199
ディスク (disk) リソース　200
仮想スイッチ (vsw) リソース　201
ネットワーク (network) リソース　202
仮想コンソール端末集配信装置 (vcc) リソース　203
変数 (var) リソース　204
物理 I/O デバイス (physio_device) リソース　204
SP 構成 (spconfig) リソース　205
仮想データプレーンチャネルサービス (vdpcs) リソース　206
仮想データプレーンチャネルクライアント (vdpcc) リソース　207
コンソール (console) リソース　208
ドメインの移行　209

A. XML スキーマ　211
LDM_interface XML スキーマ　211
LDM_Event XML スキーマ　214
ovf-envelope.xsd スキーマ　216
ovf-section.xsd スキーマ　219
ovf-core.xsd スキーマ　220
ovf-virtualhardware.xsc スキーマ　227
cim-rasd.xsd スキーマ　229
cim-vssd.xsd スキーマ　234
cim-common.xsd スキーマ　235
GenericProperty XML スキーマ　239
Binding_Type XML スキーマ　240

用語集　241

はじめに

『Logical Domains (LDoms) 1.1 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 1.1 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 (LDoms) 1.1 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

『Logical Domains (LDoms) 1.1 管理ガイド』および『Logical Domains (LDoms) 1.1 リリースノート』は、次の URL から入手できます。

```
http://docs.sun.com/app/docs/prod/ldoms#hic
```

『Beginners Guide to LDoms: Understanding and Deploying Logical Domains』は、次の Sun BluePrints™ サイトで参照できます。

```
http://www.sun.com/blueprints/0207/820-0832.html
```

使用しているサーバー、ソフトウェア、または Solaris OS に関連するドキュメントは、次の URL で参照できます。

```
http://docs.sun.com
```

必要なドキュメントを検索するには、「検索」ボックスに使用しているサーバー、ソフトウェア、または Solaris OS の名前を入力します。

用途	タイトル	Part No.	形式
LDoms のリリースノート	『Logical Domains (LDoms) 1.1 リリースノート』	820-5392-10	HTML PDF
LDoms の Solaris マニュアルページ	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	なし	HTML
LDoms マニュアルページ	`ldm`(1M) マニュアルページ	なし	SGML
	『Logical Domains (LDoms) Manager 1.1 Man Page Guide』	820-4915-10	PDF
Logical Domains ソフトウェアの基本	『Beginners Guide to LDoms: Understanding and Deploying Logical Domains』	820-0832	PDF
LDoms MIB の管理	『Logical Domains (LDoms) MIB 1.0.1 管理ガイド』	820-3456-10	HTML PDF
LDoms MIB のリリースノート	『Logical Domains (LDoms) MIB 1.0.1 リリースノート』	820-3462-10	HTML PDF
Libvirt for LDoms の管理	『Libvirt for LDoms 1.0.1 管理ガイド』	820-4108-10	HTML PDF
Libvirt for LDoms のリリースノート	『Libvirt for LDoms 1.0.1 リリースノート』	820-4114-10	HTML PDF
Solaris OS のインストール、JumpStart™ の使用、SMF の使用など	Solaris 10 Collection	なし	HTML PDF
セキュリティー	『Solaris Security Toolkit 4.2 管理マニュアル』	819-3789-10	HTML PDF
セキュリティー	『Solaris Security Toolkit 4.2 リファレンスマニュアル』	819-3793-10	HTML PDF
セキュリティー	『Solaris Security Toolkit 4.2 ご使用にあたって』	819-3796-10	HTML PDF
セキュリティー	『Solaris Security Toolkit 4.2 マニュアルページガイド』	819-3794-10	HTML PDF

マニュアル、サポート、およびトレーニング

Sun のサービス	URL
マニュアル	http://docs.sun.com
サポート	http://jp.sun.com/support
トレーニング	http://jp.sun.com/training

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。コメントは下記よりお送りください。

http://www.sun.com/hwdocs/feedback

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 (LDoms) 1.1 管理ガイド』、Part No. 820-5386-10

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 テクノロジを使用するために必要な Solaris OS のすべての機能は、Solaris 10 11/06 release 以上と必須パッチを追加することで使用できるようになります。しかし、論理ドメインを使用するには、システムファームウェアおよび Logical Domains Manager も必要です。詳細は、『Logical Domains (LDoms) 1.1 リリースノート』の「必須および推奨されるソフトウェア」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、SPARC® ハイパーバイザと、SPARC ハイパーバイザがサポートする論理ドメイン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。

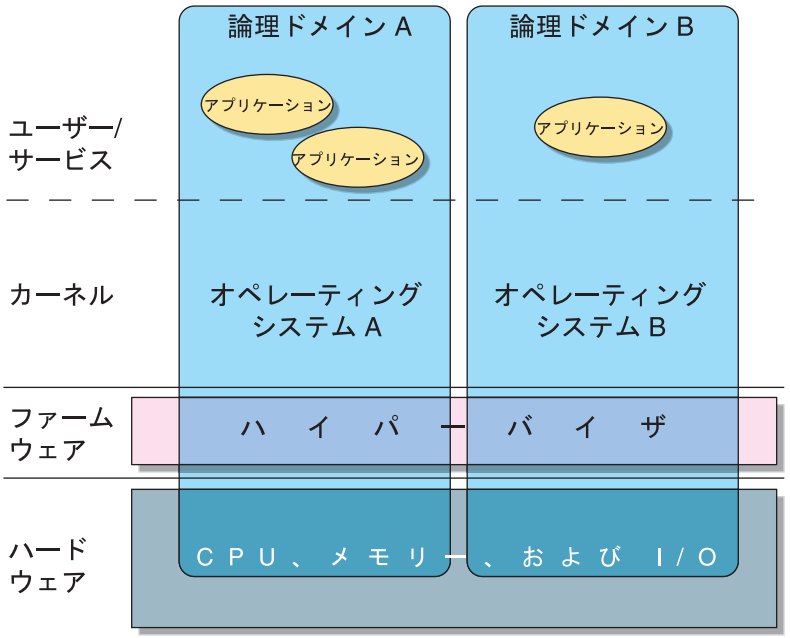
論理ドメインは、独自のオペレーティングシステム、リソース、および単一のコンピュータシステム内での識別情報を持つ個別の論理グループです。各論理ドメインは独立して作成、削除、再構成、および再起動することができ、そのときサーバーの電源の再投入は必要ありません。パフォーマンスおよびセキュリティー上の理由から、さまざまなアプリケーションソフトウェアを異なる論理ドメイン上で動作させて、アプリケーションの独立性を維持することができます。

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。システム管理者は、Logical Domains Manager を使用して、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します。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。これは、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理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。

各仮想マシンは、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

各仮想マシンは、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。また、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。論理ドメインチャネルを使用することで、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、仮想マシンは管理しません。Logical Domains Manager が仮想マシンを実行します。

Logical Domains Manager

Logical Domains Manager は、論理ドメインを作成および管理するために使用します。Logical Domains Manager は、サーバーごとに 1 つだけ存在できます。Logical Domains Manager は、論理ドメインを物理リソースに割り当てます。

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だけが異なります。論理ドメインで実行可能ないくつかの役割を、次に示します。

表 1-1 論理ドメインの役割

ドメインの役割	説明
制御ドメイン	Logical Domains Manager を実行するドメイン。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。Logical Domains ソフトウェアのインストール時に作成される初期ドメインが制御ドメインで、`primary` という名前になります。
サービスドメイン	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するドメイン。
I/O ドメイン	PCI Express コントローラ内のネットワークカードなどの物理 I/O デバイスに対して、直接の所有権を持ち、直接アクセスできるドメイン。I/O ドメインが制御ドメインを兼ねる場合は、デバイスを仮想デバイスの形式でほかのドメインと共有します。設定できる I/O ドメインの数は、使用しているプラットフォームアーキテクチャーによって異なります。たとえば、Sun UltraSPARC® T1 プロセッサを使用している場合、最大 2 つの I/O ドメインを設定できますが、そのうち 1 つは制御ドメインを兼ねる必要があります。
ゲストドメイン	制御ドメインによって管理され、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するドメイン。

既存のシステムがあり、オペレーティングシステムおよびその他のソフトウェアがサーバーですでに実行されている場合、Logical Domains Manager をインストールするとそれが制御ドメインになります。制御ドメインの設定後に一部のアプリケーションを制御ドメインから削除したり、システムの使用効率を最大限にするためにアプリケーションの負荷をドメイン間で分散したりすることができます。

コマンド行インタフェース

Logical Domains Manager では、システム管理者が論理ドメインを作成および構成するためにコマンド行インタフェースが用意されています。CLI には、単一のコマンド `ldm`(1M) と、複数のサブコマンドがあります。

Logical Domains Manager CLI を使用するには、Logical Domains Manager デーモン `ldmd` が実行中である必要があります。`ldm`(1M) コマンドとそのサブコマンドについては、`ldm`(1M) マニュアルページ、および『Logical Domains (LDoms) Manager マニュアルページガイド』で詳しく説明しています。`ldm`(1M) マニュアルページは `SUNWldm` パッケージの一部で、`SUNWldm` パッケージのインストール時にインストールされます。

ldm コマンドを実行するには、使用している UNIX の $PATH 変数に /opt/SUNWldm/bin ディレクトリが指定されている必要があります。ldm(1M) マニュアルページを参照するには、変数 $MANPATH にディレクトリパス /opt/SUNWldm/man を追加します。それぞれ次のようになります。

```
$ PATH=$PATH:/opt/SUNWldm/bin; export PATH (for Bourne or K shell)
$ MANPATH=$MANPATH:/opt/SUNWldm/man; export MANPATH
% set PATH=($PATH /opt/SUNWldm/bin) (for C shell)
% set MANPATH=($MANPATH /opt/SUNWldm/man)
```

仮想入出力

Logical Domains 環境では、管理者は Sun Fire™ または SPARC Enterprise T1000/T2000 サーバー上で、最大 32 のドメインをプロビジョニングすることができます。各ドメインには専用の CPU およびメモリーを割り当てることができますが、それらのシステムにある限られた数の I/O バスや物理 I/O スロットを使用して、ディスクデバイスやネットワークデバイスに対する排他アクセスをすべてのドメインに提供することは不可能です。PCI Express® (PCIe) バスを 2 つに分割することで一部の物理デバイスは共有できますが (73 ページの「複数の論理ドメインにまたがる PCI Express バスの構成」を参照)、排他的なデバイスアクセスをすべてのドメインに提供するには十分ではありません。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。

直接 I/O アクセスを行わないすべての論理ドメインは、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、物理デバイスまたはその機能へのアクセスを提供するサービスを実行します。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。Logical Domains 1.1 ソフトウェアの仮想化 I/O 機能には、仮想のネットワーク、ストレージ、およびコンソールのサポートが含まれます。

仮想ネットワーク

仮想ネットワークのサポートは、仮想ネットワークおよび仮想ネットワークスイッチデバイスという 2 つのコンポーネントを使用して実装されます。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を使用することで、論理ドメインは、クライアントサーバーモデルでは論理ドメインに直接割り当てられないブロックレベルのストレージにアクセスできます。これは、ブロック型デバイスインタフェースとしてエクスポートを行う仮想ディスククライアント (vdc) と、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物理ストレージに送信する仮想ディスクサービス (vds) の 2 つのコンポーネントで構成されます。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、すべてのディスク操作は仮想ディスクサービスを介して物理ディスクに転送されます。

仮想コンソール

Logical Domains 環境では、`primary` ドメインを除くすべてのドメインからのコンソール I/O は、システムコントローラではなく、仮想コンソール端末集配信装置 (vcc) および仮想ネットワーク端末サーバー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。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することができる機能です。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。仮想 CPU の動的再構成は、Solaris 10 OS のすべてのバージョンでサポートされます。仮想 I/O デバイスの動的再構成は、Solaris 10 10/08 以上で動作している論理ドメインでサポートされます。メモリーおよび物理 I/O デバイスの動的再構成はサポートされていません。Logical Domains Manager CLI で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`(1M) が動作している必要があります。

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、OS の次回の再起動後、または OS が動作していない場合は論理ドメインの停止および起動後に有効になります。ドメインで Solaris 10 5/08 以前の OS が動作している場合、`add-vcpu`、`set-vcpu`、および `remove-vcpu` サブコマンドを除く、アクティブな論理ドメインでの追加または削除処理は、遅延再構成処理とみなされます。ドメインで

Solaris 10 10/08 OS が動作している場合、仮想入出力デバイスの追加および削除は遅延構成になりません。ドメインで動作している Solaris OS にかかわらず、アクティブな論理ドメインでの `set-vswitch` サブコマンドは遅延再構成処理とみなされます。

Sun UltraSPARC T1 プロセッサを使用している場合で、Logical Domains Manager が先にインストールされて有効になっているとき、または構成が `factory-default` に復元されているときは、LDoms Manager は構成モードで動作します。このモードでは、再構成要求は受け入れられてキューに入れられますが、処理されません。これにより、実行中のマシンの状態には影響を与えずに新しい構成が生成されて SC に格納されるため、結果として I/O ドメインの遅延再構成や再起動のような制限によって妨げられることがなくなります。

特定の論理ドメインで遅延再構成が処理中になると、その論理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、その論理ドメインに対するその他の再構成要求も延期されます。また、ある論理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。

Solaris 10 5/08 以前の OS が動作している場合、アクティブな論理ドメインで仮想 I/O デバイスを削除する試みは遅延再構成処理として扱われますが、一部の構成変更はドメインで即座に発生します。つまり、関連する Logical Domains Manager CLI 操作が呼び出されるとすぐ、そのデバイスは機能を停止します。ドメインで Solaris 10 10/08 OS が動作している場合には、削除処理全体が仮想 I/O の動的再構成処理の一部として即座に実行されるため、この問題は発生しません。

Logical Domains Manager のサブコマンド `remove-reconf` は、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することができます。遅延再構成機能の使用法の詳細は、`ldm`(1M) マニュアルページまたは『Logical Domains (LDoms) Manager マニュアルページガイド』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 デバイスで実行されている場合は、`ldm remove-reconf` コマンドを使用できません。このような状況では、`ldm remove-reconf` コマンドは失敗します。

持続的な構成

Logical Domains Manager CLI コマンドを使用して、論理ドメインの現在の構成をシステムコントローラ (SC) に格納することができます。構成の追加、使用する構成の指定、構成の削除、およびシステムコントローラ上の構成の表示を行うことができます。`ldm`(1M) マニュアルページまたは『Logical Domains (LDoms) Manager マニュアルページガイド』を参照してください。さらに、起動する構成を選択できる ALOM CMT Version 1.3 コマンドもあります (177 ページの「LDoms と ALOM CMT の使用」を参照)。

第2章

セキュリティー

この章では、Solaris Security Toolkit ソフトウェアの概要と、このソフトウェアを使用して論理ドメインで Solaris OS をセキュリティー保護する方法について説明します。

セキュリティー上の考慮事項

Solaris Security Toolkit ソフトウェアは、非公式には JumpStart™ Architecture and Security Scripts (JASS) ツールキットとも呼ばれ、セキュリティー保護された Solaris OS システムの構築および維持を行うために、自動化され、拡張性の高いスケーラブルな機構を提供します。Solaris Security Toolkit は、Logical Domains Manager の制御ドメインを含む、サーバーの管理には不可欠なデバイスのセキュリティー保護を実現します。

Solaris Security Toolkit 4.2 ソフトウェアパッケージ `SUNWjass` では、`install-ldm` スクリプトを次のように使用することで、制御ドメイン上の Solaris オペレーティングシステムをセキュリティー保護する手段を提供します。

- Logical Domains Manager インストールスクリプト (`install-ldm`) および Logical Domains Manager に固有の制御ドライバ (`ldm_control-secure.driver`) を使用することにより、Solaris Security Toolkit が制御ドメインを自動的に強化します。
- インストールスクリプトの使用時に代替ドライバを選択します。
- インストールスクリプトの使用時にドライバを選択せず、独自の Solaris 強化を適用します。

`SUNWjass` パッケージは、Logical Domains (LDoms) Manager 1.1 ソフトウェアパッケージ `SUNWldm` と一緒に同梱されており、Sun のソフトウェアダウンロード Web サイトから入手できます。Logical Domains Manager 1.1 ソフトウェアをダウンロードしてインストールすると同時に、Solaris Security Toolkit 4.2 ソフトウェアパッケージをダウンロードしてインストールするオプションがあります。Solaris Security

Toolkit 4.2 ソフトウェアパッケージには、Solaris Security Toolkit ソフトウェアを Logical Domains Manager とともに使用できるようにするための必須パッチが含まれています。ソフトウェアがインストールされたら、Solaris Security Toolkit 4.2 ソフトウェアを使用してシステムを強化できます。第 3 章では、Solaris Security Toolkit をインストールおよび構成し、制御ドメインを強化する方法について説明しています。

Solaris Security Toolkit によって提供されるセキュリティー機能のうち、Logical Domains Manager のユーザーが使用可能な機能を次に示します。

- **強化** ― 必須パッチが適用された Solaris Security Toolkit 4.2 ソフトウェアを使用して、Solaris Security Toolkit ソフトウェアを Logical Domains Manager とともに使用できるようにして、システムのセキュリティーを向上するように Solaris OS 構成を変更します。
- **最小化** – LDoms および LDoms Management Information Base (MIB) のサポートに必要な最小限の主要な Solaris OS パッケージをインストールします。
- **承認** – 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) を使用して承認を設定します。
- **監査** – Logical Domains Manager 用に変更された Solaris OS 基本セキュリティーモジュール (BSM) を使用してシステムのセキュリティーの変更元を識別し、何が、いつ、誰によって行われ、どのような影響があるのかを判断します。
- **適合性** – Solaris Security Toolkit の監査機能を使用して、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判断します。

Solaris Security Toolkit および Logical Domains Manager

第 3 章では、Solaris Security Toolkit をインストールして Logical Domains Manager とともに使用する方法について説明しています。Solaris Security Toolkit は、制御ドメインにインストールします。制御ドメインでは Logical Domains Manager が実行されています。また、Solaris Security Toolkit は、ほかの論理ドメインにインストールすることもできます。唯一の違いは、制御ドメインを強化する場合は `ldm_control-secure.driver` ドライバを使用し、ほかの論理ドメインを強化する場合は `secure.driver` などの別のドライバを使用することです。これは、`ldm_control-secure.driver` が制御ドメイン専用であるためです。`ldm_control-secure.driver` は、`secure.driver` をベースにして、Logical Domains Manager で使用できるようにカスタマイズおよびテストしたものです。`secure.driver` の詳細は、『Solaris Security Toolkit 4.2 リファレンスマニュアル』を参照してください。

強化

Solaris Security Toolkit が制御ドメイン上の Solaris OS を強化するために使用するドライバ (`ldm_control-secure.driver`) は、Logical Domains Manager が OS で実行できるように特別な変更を加えたものです。`ldm_control-secure.driver` は、『Solaris Security Toolkit 4.2 リファレンスマニュアル』で説明している `secure.driver` に類似しています。

`ldm_control-secure.driver` は、Logical Domains Manager ソフトウェアを実行しているシステムの制御ドメインに対する基準構成を提供します。これは、Solaris OS ドメインで通常よりも少ないシステムサービスを提供することで、通常の用途ではなく Logical Domains Manager の処理のために制御ドメインを確保することを目的としています。

`install-ldm` スクリプトにより、Logical Domains Manager ソフトウェアがまだインストールされていない場合にはこのソフトウェアがインストールされ、使用可能になります。

`secure.driver` から変更された、その他の重要な事項の概要を次に示します。

- Telnet サーバーは実行できません。代わりに Secure Shell (`ssh`) を使用できます。また、Telnet クライアントを使用して、Logical Domains 仮想ネットワーク端末サーバーデーモン (`vntsd`) によって開始された仮想コンソールにアクセスすることはできます。たとえば、ローカルシステムの TCP ポート 5001 で待機している仮想コンソールが実行中の場合は、次のようにアクセスすることができます。

```
# telnet localhost 5001
```

  `vntsd` を有効にする方法については、37 ページの「Logical Domains Manager デーモンの有効化」を参照してください。これは自動的には有効になりません。

- 次の終了スクリプトが追加されています。これらを使用して、Logical Domains Manager をインストールおよび起動できます。追加されたスクリプトの一部は、カスタマイズしたすべてのドライバに追加する必要がありますが、省略可能なものもあります。スクリプトには、必須であるか任意であるかが示されています。
  - `install-ldm.fin` – `SUNWldm` パッケージをインストールします。(必須)
  - `enable-ldmd.fin` – Logical Domains Manager デーモン (`ldmd`) を有効にします。(必須)
  - `enable-ssh-root-login.fin` – スーパーユーザーが Secure Shell (`ssh`) を使用して直接ログインできるようにします。(任意)
- 次のファイルが変更されました。これらの変更のカスタマイズしたドライバへの組み込みは省略可能であるため、任意として示されています。
  - `/etc/ssh/sshd_config` – root アカウントのアクセスがネットワーク全体で許可されます。このファイルはどのドライバでも使用されません。(任意)
  - `/etc/ipf/ipf.conf` – UDP ポート 161 (SNMP) が開かれます。(任意)

- /etc/host.allow – Secure Shell デーモン (sshd) がローカルサブネットだけでなくネットワーク全体に対してオープンになります。(任意)

■ 次の終了スクリプトが無効 (コメントアウト) になっています。カスタマイズしたすべてのドライバで、disable-rpc.fin スクリプトをコメントにしてください。その他の変更は省略可能です。スクリプトには、必須であるか任意であるかが示されています。

- enable-ipfilter.fin – IP フィルタ (ネットワークパケットフィルタの一種) が有効ではありません。(任意)
- disable-rpc.fin – 遠隔手続き呼び出し (RPC) サービスが有効のままです。RPC サービスは、ネットワーク情報サービス (NIS)、ネットワークファイルシステム (NFS) などのほかの多くのシステムサービスによって使用されます。(必須)
- disable-sma.fin – システム管理エージェント (NET-SNMP) が有効のままです。(任意)
- disable-ssh-root-login.fin – ssh root ログインを無効にできません。
- set-term-type.fin – 不要な旧バージョンのスクリプト。(任意)

論理ドメインの最小化

Solaris OS は、要件に合わせてさまざまな数のパッケージを組み合わせて構成できます。最小化では、このようなパッケージのセットを、必要なアプリケーションを実行するために最低限必要な数にまで減らします。最小化により、セキュリティーに脆弱性があるおそれのあるソフトウェアの数が減り、インストールされたソフトウェアにパッチを正しく適用し続けることに付随する労力の度合いも軽減されるため、最小化は重要です。論理ドメインの最小化処理では、任意のドメインを完全にサポートし続ける、最小化された Solaris OS をインストールするための JumpStart™ サポートが提供されています。

Solaris Security Toolkit では、LDoms の論理ドメインを最小化するために使用する JumpStart プロファイル minimal-ldm_control.profile が用意されています。このプロファイルにより、LDoms および LDoms MIB のサポートに必要なすべての Solaris OS パッケージがインストールされます。LDoms MIB を制御ドメインで使用する場合は、LDoms および Solaris Security Toolkit パッケージのインストール後に、個別にそのパッケージを追加する必要があります。これは、ほかのソフトウェア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 管理ガイド』を参照してください。

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 ー 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 ー 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 2-1 `ldm` サブコマンドおよびユーザー承認

ldm サブコマンド*	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`

* 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。

監査

Logical Domains Manager CLI コマンドの監査は、Solaris OS 基本セキュリティーモジュール (BSM) 監査によって実行されます。Solaris OS 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)』を参照してください。

Logical Domains Manager に対する BSM 監査は、デフォルトでは有効ではありませんが、インフラストラクチャーは用意されています。BSM 監査は、次の 2 つのいずれかの方法で使用できます。

- Solaris Security Toolkit の `enable-bsm.fin` 終了スクリプトを実行します。
- Solaris OS の `bsmconv`(1M) コマンドを使用します。

Logical Domains Manager で BSM 監査を使用する場合の有効化、検証、無効化、出力の表示、およびログの切り替えの詳細は、178 ページの「BSM 監査の有効化と使用」を参照してください。

適合性

Solaris Security Toolkit には、独自の監査機能があります。Solaris Security Toolkit ソフトウェアは、事前に定義されたセキュリティープロファイルと比較して、Solaris OS が動作しているあらゆるシステムのセキュリティー状況を自動的に検証することができます。この適合性機能の詳細は、『Solaris Security Toolkit 4.2 管理マニュアル』の「システムのセキュリティーの監査」を参照してください。

第3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 1.1 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。Logical Domains 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 (LDoms) 1.1 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 (LDoms) 1.1 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 10/08 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。16 ページの「Solaris OS のアップグレード」を参照してください。
- 使用している Sun UltraSPARC T1 プラットフォームのシステムファームウェア version 6.7.*x*、または使用している Sun UltraSPARC T2 または T2 Plus プラットフォーム以上のシステムファームウェア version 7.2.*x*。22 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 LDoms 1.1 ソフトウェア。25 ページの「Logical Domains Manager および Solaris Security Toolkit のインストール」を参照してください。
- (省略可能) Solaris Security Toolkit 4.2 ソフトウェア。25 ページの「Logical Domains Manager および Solaris Security Toolkit のインストール」を参照してください。
- (省略可能) Logical Domains (LDoms) 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 管理ガイド』を参照してください。
- (省略可能) Libvirt for LDoms 1.0.1 ソフトウェアパッケージ。Libvirt for LDoms の使用法の詳細は、『Libvirt for LDoms 1.0.1 管理ガイド』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、16 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、21 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。

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

Solaris OS のアップグレード

使用しているシステムですでに Logical Domain 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。Logical Domains 1.1 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。

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 (LDoms) 1.1 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 10/08 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 をアップグレードする場合、Logical Domains の制約データベースファイルを保存する必要があります。この節では、Logical Domains の制約データベースファイルの保存および復元に関して、知っておく必要のある情報について説明します。

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する必要があります。

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 8/07 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、91 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、LDoms 1.1 ソフトウェアを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 および Solaris Security Toolkit を制御ドメインにダウンロードします。24 ページの「Logical Domains Manager および Solaris Security Toolkit 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
```

LDoms 1.1 ソフトウェアへのアップグレード

この節では、LDoms 1.1 ソフトウェアを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 LDoms 1.1 ソフトウェアで使用する場合は、18 ページの「LDoms 1.0 ソフトウェアから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、LDoms 1.1 ソフトウェアでは機能しません。

LDoms 1.0.1、1.0.2、または 1.0.3 ソフトウェアからアップグレードする場合、20 ページの「LDoms 1.0.1、1.0.2、または 1.0.3 からアップグレードする」に示す手順を実行します。既存の LDoms 1.0.1、1.0.2、および 1.0.3 の設定は、LDoms 1.1 ソフトウェアでも機能します。

▼ LDoms 1.0 ソフトウェアからアップグレードする

既存の LDoms 1.0 の設定は LDoms 1.1 ソフトウェアでは機能しません。そのため、LDoms 1.0 の設定を保存してから、LDoms 1.1 ソフトウェアで使用できるようにその設定をアップグレードする必要があります。次の手順では、`ldm add-domain` コマンドに XML 制約ファイルおよび `-i` オプションを使用して、構成を保存および再構築する方法について説明します。

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。アップグレード後に、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。

この節に示す手順は、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、64 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。

1. 各ドメインで、ドメインの制約を含む XML ファイルを作成します。

```
# ldm ls-constraints -x ldom > ldom.xml
```

2. システムコントローラ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。

```
# ldm ls-config
```

3. システムコントローラに格納されているそれぞれの論理ドメイン構成を削除します。

```
# ldm rm-config config_name
```

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

```
# svcadm disable ldmd
```

5. Logical Domains Manager パッケージ (`SUNWldm`) を削除します。

```
# pkgrm SUNWldm
```

6. Solaris Security Toolkit パッケージ (`SUNWjass`) を使用している場合はこれを削除します。

```
# pkgrm SUNWjass
```

7. 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、22 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 23 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。

8. Solaris Security Toolkit および Logical Domains Manager を再インストールします。25 ページの「Logical Domains Manager および Solaris Security Toolkit のインストール」を参照してください。

9. `primary` ドメインを手動で再構成します。手順については、48 ページの「制御ドメインを設定する」を参照してください。

10. 手順 1 で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
```

▼ LDoms 1.0.1、1.0.2、または 1.0.3 から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、22 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 23 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. Logical Domains Manager デーモン (`ldmd`) を更新します。

```
# svcadm refresh ldmd
```

6. Logical Domains Manager デーモン (ldmd) を有効にします。

```
# svcadm enable ldmd
```

7. 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。

   次のようなメッセージが表示されます。これは、factory-default 構成の場合です。primary ドメインは active になっています。これは、Logical Domains Manager が実行中であることを意味します。

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

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。初期設定では Logical Domains ソフトウェアは有効になっていません。また、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表示されます。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 (LDoms) 1.1 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 (LDoms) 1.1 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 10/08 Release and Installation Collection 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

▼ 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 サイトから入手できます。

`http://sunsolve.sun.com`

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 (LDoms) 1.1 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、システムコントローラで `flashupdate`(1M) 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、23 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. **システムコントローラ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(1M) 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。

```
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、Sun Fire T2000 サーバーの *server-name* は、Sun_Fire_T2000 です。

3. システムコントローラをリセットします。

```
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

システムコントローラ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. システムコントローラの電源を切り、ファームウェアを更新します。

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

4. システムコントローラをリセットして電源を入れます。

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

Logical Domains Manager および Solaris Security Toolkit 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_1.zip`) をダウンロードします。ソフトウェアは、次の Web サイトから入手できます。

   `http://www.sun.com/ldoms`

2. zip ファイルを解凍します。

```
$ unzip LDoms_Manager-1_1.zip
```

   Logical Domains Manager および Solaris Security Toolkit は、同じ zip ファイル内に含まれています。ファイルの構造およびファイルの内容の詳細は、『Logical Domains (LDoms) 1.1 リリースノート』の「Logical Domains 1.1 ソフトウェアの場所」を参照してください。

Logical Domains Manager および Solaris Security Toolkit のインストール

Logical Domains Manager および Solaris Security Toolkit ソフトウェアをインストールするには、次の 3 つの方法があります。

- 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 および Solaris Security Toolkit ソフトウェアの両方が自動的にはインストールされます。25 ページの「Logical Domains Manager および Solaris Security Toolkit ソフトウェアの自動インストール」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。32 ページの「JumpStart を使用した Logical Domains Manager 1.1 および Solaris Security Toolkit 4.2 ソフトウェアのインストール」を参照してください。
- 各パッケージを手動でインストールします。35 ページの「Logical Domains Manager および Solaris Security Toolkit ソフトウェアの手動インストール」を参照してください。

注 – LDoms および Solaris Security Toolkit 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 管理ガイド』を参照してください。

Logical Domains Manager および Solaris Security Toolkit ソフトウェアの自動インストール

`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 OS 10 11/06 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 `SUNWjass/` 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 `SUNWjass` 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。

注 – インストール中に、スクリプトが SUNWjass の以前のバージョンを検出した場合は、これを削除する必要があります。使用している Solaris OS のこれまでの強化を元に戻す必要はありません。

- Logical Domains Manager 1.1 ソフトウェア (SUNWldm パッケージ) をインストールします。
- 必須パッチを含む Solaris Security Toolkit 4.2 ソフトウェア (SUNWjass パッケージ) 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。
- Logical Domains Manager デーモン ldmd を有効にします。
- Solaris Security Toolkit の ldm_control-secure.driver、または -secure.driver で終わるその他のドライバのうち選択したものを使用して、制御ドメインで Solaris OS を強化します。

- **オプション -d を指定して install-ldm スクリプトを使用すると、** -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。このオプションでは、前述の選択肢で示したすべての機能と次の追加オプションを自動的に実行します。
  - Solaris Security Toolkit のカスタマイズしたドライバ (たとえば server-secure-myname.driver) を使用して、制御ドメインで Solaris OS を強化します。
- **オプション -d と none を指定して install-ldm スクリプトを使用すると、** Solaris Security Toolkit を使用して制御ドメインで動作している 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。
- **オプション -p を指定して install-ldm スクリプトを使用すると、**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。32 ページの「Logical Domains Manager デーモンを有効にして Solaris Security Toolkit のみを実行する」を参照してください。

▼ 特別なオプションを指定せずにインストールする

- **オプションを指定せずに install-ldm インストールスクリプトを実行します。**

  インストールスクリプトは、SUNWldm パッケージの一部で、Install サブディレクトリにあります。

```
# Install/install-ldm
```

a. 1 つ以上のパッケージがすでにインストールされている場合は、次のメッセージが表示されます。

```
# Install/install-ldm
ERROR: One or more packages are already installed: SUNWldm SUNWjass.
If packages SUNWldm.v and SUNWjass are factory pre-installed, run
install-ldm -p to perform post-install actions.  Otherwise remove the
package(s) and restart install-ldm.
```

インストール後の処理のみを実行する場合は、32 ページの「Logical Domains Manager デーモンを有効にして Solaris Security Toolkit のみを実行する」に進みます。

b. 処理が正常に実行されると、次の例のようなメッセージが表示されます。

- コード例 3-2 は、次のデフォルトのセキュリティープロファイルを選択した場合に、install-ldm スクリプトが正常に実行されたことを示しています。

  ```
  a) Hardened Solaris configuration for LDoms (recommended)
  ```

- コード例 3-3 は、次のセキュリティープロファイルを選択した場合に、install-ldm スクリプトが正常に実行されたことを示しています。

  ```
  c) Your custom-defined Solaris security configuration
  profile
  ```

  選択肢として表示されるドライバは、名前が -secure.driver で終わるドライバです。名前が -secure.driver で終わらない、カスタマイズしたドライバを書き込む場合は、install-ldm -d オプションでカスタマイズしたドライバを指定する必要があります。30 ページの「カスタマイズされた強化ドライバとともにインストールする」を参照してください。

コード例 3-1　　LDoms 用に強化された Solaris 構成の場合の出力

```
# Install/install-ldm
Welcome to the LDoms installer.

You are about to install the domain manager package that will enable
you to create, destroy and control other domains on your system. Given
the capabilities of the domain manager, you can now change the security
configuration of this Solaris instance using the Solaris Security
Toolkit.

Select a security profile from this list:

a) Hardened Solaris configuration for LDoms (recommended)
b) Standard Solaris configuration
c) Your custom-defined Solaris security configuration profile

Enter a, b, or c [a]: a
The changes made by selecting this option can be undone through the
Solaris Security Toolkit's undo feature. This can be done with the
'/opt/SUNWjass/bin/jass-execute -u' command.

Installing LDoms and Solaris Security Toolkit packages.
pkgadd -n -d "/var/tmp/install/Product/Logical_Domain_Manager" -a pkg_admin
SUNWldm.v
Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Installation of <SUNWldm> was successful.
pkgadd -n -d "/var/tmp/install/Product/Solaris_Security_Toolkit" -a pkg_admin
SUNWjass
Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Installation of <SUNWjass> was successful.

Verifying that all packages are fully installed.  OK.
Enabling services: svc:/ldoms/ldmd:default
Running Solaris Security Toolkit 4.2.0 driver ldm_control-secure.driver.
Please wait. . .
/opt/SUNWjass/bin/jass-execute -q -d ldm_control-secure.driver
Executing driver, ldm_control-secure.driver
Solaris Security Toolkit hardening executed successfully; log file
/var/opt/SUNWjass/run/20070208142843/jass-install-log.txt.  It will not
take effect until the next reboot.  Before rebooting, make sure SSH or
the serial line is setup for use after the reboot.
```

コード例 3-2　　カスタマイズされた構成プロファイルを選択した場合の出力

```
# Install/install-ldm
Welcome to the LDoms installer.

You are about to install the domain manager package that will enable
you to create, destroy and control other domains on your system. Given
the capabilities of the domain manager, you can now change the security
configuration of this Solaris instance using the Solaris Security
Toolkit.

Select a security profile from this list:

a) Hardened Solaris configuration for LDoms (recommended)
b) Standard Solaris configuration
c) Your custom-defined Solaris security configuration profile

Enter a, b, or c [a]: c
Choose a Solaris Security Toolkit .driver configuration profile from
this list
1) ldm_control-secure.driver
2) secure.driver
3) server-secure.driver
4) suncluster3x-secure.driver
5) sunfire_15k_sc-secure.driver

Enter a number 1 to 5: 2
The driver you selected may not perform all the LDoms-specific
operations specified in the LDoms Administration Guide.
Is this OK (yes/no)? [no] y
The changes made by selecting this option can be undone through the
Solaris Security Toolkit's undo feature. This can be done with the
'/opt/SUNWjass/bin/jass-execute -u' command.

Installing LDoms and Solaris Security Toolkit packages.
pkgadd -n -d "/var/tmp/install/Product/Logical_Domain_Manager" -a pkg_admin
SUNWldm.v
Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Installation of <SUNWldm> was successful.
pkgadd -n -d "/var/tmp/install/Product/Solaris_Security_Toolkit" -a pkg_admin
SUNWjass
Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Installation of <SUNWjass> was successful.

Verifying that all packages are fully installed.  OK.
```

コード例 3-2　　カスタマイズされた構成プロファイルを選択した場合の出力 (続き)

```
Enabling services: svc:/ldoms/ldmd:default
Running Solaris Security Toolkit 4.2.0 driver secure.driver.
Please wait. . .
/opt/SUNWjass/bin/jass-execute -q -d secure.driver
Executing driver, secure.driver
Solaris Security Toolkit hardening executed successfully; log file
/var/opt/SUNWjass/run/20070102142843/jass-install-log.txt.  It will not
take effect until the next reboot.  Before rebooting, make sure SSH or
the serial line is setup for use after the reboot.
```

▼ カスタマイズされた強化ドライバとともにインストールする

- Solaris Security Toolkit のカスタマイズされた強化ドライバ (たとえば、server-secure-myname.driver) を指定するには、-d オプションを指定して install-ldm インストールスクリプトを実行します。

  インストールスクリプトは、SUNWldm パッケージの一部で、Install サブディレクトリにあります。

```
# Install/install-ldm -d server-secure-myname.driver
```

処理が正常に実行されると、コード例 3-4 のようなメッセージが表示されます。

コード例 3-3　　install-ldm -d スクリプトが正常に実行された場合の出力

```
# Install/install-ldm -d server-secure.driver
The driver you selected may not perform all the LDoms-specific
operations specified in the LDoms Administration Guide.
Installing LDoms and Solaris Security Toolkit packages.
pkgadd -n -d "/var/tmp/install/Product/Logical_Domain_Manager" -a pkg_admin
SUNWldm.v
Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Installation of <SUNWldm> was successful.
pkgadd -n -d "/var/tmp/install/Product/Solaris_Security_Toolkit" -a pkg_admin
SUNWjass
Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Installation of <SUNWjass> was successful.

Verifying that all packages are fully installed.  OK.
Enabling services: svc:/ldoms/ldmd:default
Running Solaris Security Toolkit 4.2.0 driver server-secure-myname.driver.
Please wait. . .
```

コード例 3-3　install-ldm -d スクリプトが正常に実行された場合の出力 (続き)

```
/opt/SUNWjass/bin/jass-execute -q -d server-secure-myname.driver
Executing driver, server-secure-myname.driver
Solaris Security Toolkit hardening executed successfully; log file
/var/opt/SUNWjass/run/20061114143128/jass-install-log.txt.  It will not
take effect until the next reboot.  Before rebooting, make sure SSH or
the serial line is setup for use after the reboot.
```

▼ インストールし、システムを強化しない

- Solaris Security Toolkit ドライバを使用してシステムを強化しないことを指定するには、`-d none` オプションを指定して `install-ldm` インストールスクリプトを実行します。

  インストールスクリプトは、`SUNWldm` パッケージの一部で、`Install` サブディレクトリにあります。

```
# Install/install-ldm -d none
```

処理が正常に実行されると、コード例 3-5 のようなメッセージが表示されます。

コード例 3-4　install-ldm -d none スクリプトが正常に実行された場合の出力

```
# Install/install-ldm -d none
Installing LDoms and Solaris Security Toolkit packages.
pkgadd -n -d "/var/tmp/install/Product/Logical_Domain_Manager" -a pkg_admin
SUNWldm.v
Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Installation of <SUNWldm> was successful.
pkgadd -n -d "/var/tmp/install/Product/Solaris_Security_Toolkit" -a pkg_admin
SUNWjass
Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Installation of <SUNWjass> was successful.

Verifying that all packages are fully installed.  OK.
Enabling services: svc:/ldoms/ldmd:default
Solaris Security Toolkit was not applied. Bypassing the use of the
Solaris Security Toolkit is not recommended and should only be
performed when alternative hardening steps are to be taken.
```

▼ Logical Domains Manager デーモンを有効にして Solaris Security Toolkit のみを実行する

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おり、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を行う必要がある場合は、このオプションを使用できます。

- **システムを強化するために 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には、-p オプションを指定して install-ldm インストールスクリプトを実行します。**

```
# Install/install-ldm -p
Verifying that all packages are fully installed.  OK.
Enabling services: svc:/ldoms/ldmd:default
Running Solaris Security Toolkit 4.2.0 driver ldm_control-secure.driver.
Please wait. . .
/opt/SUNWjass/bin/jass-execute -q -d ldm_control-secure.driver
Solaris Security Toolkit hardening executed successfully; log file
var/opt/SUNWjass/run/20070515140944/jass-install-log.txt.  It will not
take effect until the next reboot.  Before rebooting, make sure SSH or
the serial line is setup for use after the reboot.
```

JumpStart を使用した Logical Domains Manager 1.1 および Solaris Security Toolkit 4.2 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

- JumpStart サーバーがすでに設定されている場合は、この管理ガイドの 33 ページの「JumpStart ソフトウェアを使用してインストールする」に進んでください。
- JumpStart サーバーがまだ設定されていない場合は、これを設定する必要があります。

  この手順の詳細は、『Solaris 10 10/08 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 10/08 インストールガイド (カスタム JumpStart/上級編)』の「カスタム JumpStart インストールの準備 (作業)」を参照し、次の手順を実行します。**

a. 「作業マップ: カスタム JumpStart インストールの準備」で作業マップを確認します。

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。

Solaris Security Toolkit では、プロファイルおよび終了スクリプトが提供されています。プロファイルおよび終了スクリプトの詳細は、『Solaris Security Toolkit 4.2 リファレンスマニュアル』を参照してください。

▼ JumpStart ソフトウェアを使用してインストールする

1. Solaris Security Toolkit パッケージ (SUNWjass) をダウンロードしたディレクトリに移動します。

```
# cd /path-to-download
```

2. SUNWjass をインストールして、JumpStart (jumpstart) ディレクトリ構造を作成します。

```
# pkgadd -R /jumpstart -d . SUNWjass
```

3. テキストエディタを使用して、ネットワーク環境を反映するように /jumpstart/opt/SUNWjass/Sysidcfg/Solaris_10/sysidcfg ファイルを変更します。

4. /jumpstart/opt/SUNWjass/Drivers/user.init.SAMPLE ファイルを /jumpstart/opt/SUNWjass/Drivers/user.init ファイルにコピーします。

```
# cp user.init.SAMPLE user.init
```

5. パスを反映するように user.init ファイルを編集します。

6. JumpStart のインストール中に Solaris Security Toolkit パッケージ (SUNWjass) を対象のシステムにインストールするには、user.init ファイルで定義した JASS_PACKAGE_MOUNT ディレクトリにこのパッケージを配置する必要があります。次に例を示します。

```
# cp -r /path/to/LDoms_Manager-1_0_2/Product/SUNWjass
/jumpstart/opt/SUNWjass/Packages
```

7. JumpStart のインストール中に Logical Domains Manager パッケージ (SUNWldm.v) を対象のシステムにインストールするには、user.init ファイルで定義した JASS_PACKAGE_MOUNT ディレクトリにダウンロード領域からこのパッケージを配置する必要があります。次に例を示します。

```
# cp -r /path/to/LDoms_Manager-1_0_2/Product/SUNWldm.v
/jumpstart/opt/SUNWjass/Packages
```

8. マルチホーム JumpStart サーバーで問題が発生した場合は、user.init ファイル内の JASS_PACKAGE_MOUNT および JASS_PATCH_MOUNT に関する 2 つのエントリを、JASS_HOME_DIR/Patches および JASS_HOME_DIR/Packages ディレクトリへの正しいパスに変更します。詳細は、user.init.SAMPLE ファイル内のコメントを参照してください。

9. Logical Domains Manager 制御ドメインの基本ドライバとして ldm_control-secure.driver を使用します。

   使用するドライバを変更する方法の詳細は、『Solaris Security Toolkit 4.2 リファレンスマニュアル』の第 4 章を参照してください。ldm_control-secure.driver に対応する Solaris Security Toolkit のメインドライバは、secure.driver です。

10. ldm_control-secure.driver への変更が完了したら、rules ファイルに適切なエントリを作成します。

    - LDoms 制御ドメインを最小化する場合は、rules ファイル内の minimal-ldm-control.profile を次のように指定します。

```
hostname imbulu - Profiles/minimal-ldm_control.profile Drivers/ldm_control-
secure-abc.driver
```

注 – LDoms および Solaris Security Toolkit 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および Libvirt for LDoms パッケージを手動でインストールする必要があります。これらは、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 制御ドメインを最小化しない場合は、エントリは次のようになるはずです。

```
hostname imbulu - Profiles/oem.profile Drivers/ldm_control-secure-abc.driver
```

11. JumpStart のインストール中の強化を取り消すには、次の SMF コマンドを実行して Logical Domains Manager を再起動する必要があります。

```
# svcadm enable svc:/ldoms/ldmd:default
```

Logical Domains Manager および Solaris Security Toolkit ソフトウェアの手動インストール

Logical Domains Manager および Solaris Security Toolkit ソフトウェアを手動でインストールするには、次の手順を実行します。

- 35 ページの「Logical Domains Manager (LDoms) 1.1 ソフトウェアを手動でインストールする」
- 35 ページの「(省略可能) Solaris Security Toolkit 4.2 ソフトウェアを手動でインストールする」
- 36 ページの「(省略可能) 制御ドメインを手動で強化する」

▼ Logical Domains Manager (LDoms) 1.1 ソフトウェアを手動でインストールする

Sun のソフトウェアダウンロードサイトから、Logical Domains Manager 1.1 ソフトウェアの SUNWldm 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、24 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(1M) コマンドを使用して、SUNWldm.v パッケージをインストールします。-G オプションを使用して大域ゾーンのみにパッケージをインストールするよう指定し、-d オプションを使用して SUNWldm.v パッケージを含むディレクトリのパスを指定します。

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

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(1) コマンドを使用して、Logical Domains Manager 1.1 ソフトウェア用の SUNWldm 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。

   バージョン (REV) 情報の例を次に示します。

```
# pkginfo -l SUNWldm | grep VERSION
VERSION=1.1,REV=2007.08.23.10.20
```

▼ (省略可能) Solaris Security Toolkit 4.2 ソフトウェアを手動でインストールする

システムをセキュリティー保護するには、SUNWjass パッケージをダウンロードしてインストールします。必須パッチ (122608-03 および 125672-01) は、SUNWjass パッケージに含まれています。ソフトウェアのダウンロードに関する詳細は、24 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。

Logical Domains Manager ソフトウェアを使用する場合のセキュリティーに関する考慮事項の詳細は、このドキュメントの第 2 章を参照してください。さらに詳細を確認するには、次の URL で Solaris Security Toolkit 4.2 のドキュメントを参照できます。

`http://docs.sun.com`

1. `pkgadd`(1M) コマンドを使用して、`SUNWjass` パッケージをインストールします。

```
# pkgadd -d . SUNWjass
```

2. `pkginfo`(1) コマンドを使用して、Solaris Security Toolkit 4.2 ソフトウェアの `SUNWjass` 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。

```
# pkginfo -l SUNWjass | grep VERSION
VERSION: 4.2.0
```

▼ (省略可能) 制御ドメインを手動で強化する

Solaris Security Toolkit 4.2 パッケージがすでにインストールされている場合にかぎり、この手順を実行してください。

注 – Solaris Security Toolkit を使用して制御ドメインを強化すると、多くのシステムサービスが無効になり、ネットワークアクセスに一定の制限が生じます。詳細は、このドキュメントの xvii ページの「関連マニュアル」を参照して、Solaris Security Toolkit 4.2 のドキュメントで確認してください。

---

1. `ldm_control-secure.driver` を使用して強化します。

```
# /opt/SUNWjass/bin/jass-execute -d ldm_control-secure.driver
```

システムを強化するために、ほかのドライバを使用できます。また、ドライバをカスタマイズして、使用している環境のセキュリティーを調整することもできます。ドライバとそのカスタマイズ方法の詳細は、『Solaris Security Toolkit 4.2 リファレンスマニュアル』を参照してください。

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. 強化を有効にするため、サーバーを停止してから再起動します。

```
# /usr/sbin/shutdown -y -g0 -i6
```

▼ 強化の妥当性検査を行う

- Logical Domains 強化ドライバ (`ldom_control-secure.driver`) によって、強化が適切に適用されたかどうかを確認します。

  別のドライバについて確認する場合は、次のコマンド例のドライバ名を置き換えてください。

```
# /opt/SUNWjass/bin/jass-execute -a ldom_control-secure.driver
```

▼ 強化を取り消す

1. Solaris Security Toolkit によって適用された構成の変更を取り消します。

```
# /opt/SUNWjass/bin/jass-execute -u
```

   Solaris Security Toolkit によって、どの強化の実行を取り消すかが尋ねられます。

2. 取り消す強化の実行を選択します。
3. 構成の強化の取り消しが行われるように、システムを再起動します。

```
# /usr/sbin/shutdown -y -g0 -i6
```

注 – JumpStart のインストール中に実行された強化を取り消すには、次の SMF コマンドを実行して Logical Domains Manager デーモン (`ldmd`) および仮想ネットワーク端末サーバーデーモン (`vntsd`) を再起動する必要があります。

```
# svcadm enable svc:/ldoms/ldmd:default
```

Logical Domains Manager デーモンの有効化

インストールスクリプト `install-ldm` 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。Logical Domains Manager ソフトウェアが手動でインストールされた場合は、Logical Domains Manager デーモンの `ldmd` を有効にする必要があります。これにより、論理ドメインの作成、変更、および制御が可能になります。

▼ Logical Domains Manager デーモンを有効にする

1. svcadm(1M) 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 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。

   次のようなメッセージが表示されます。これは、factory-default 構成の場合です。primary ドメインは active になっています。これは、Logical Domains Manager が実行中であることを意味します。

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

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 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 ー 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 ー 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 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 2-1 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>)
```

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. システムコントローラのすべての論理ドメイン構成を一覧表示します。

```
primary# ldm ls-config
```

2. 以前システムコントローラ (SC) 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 SC 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、すべてのゲストドメインを停止します。

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

4. すべてのゲスト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 構成が再読み込みされるように、システムコントローラ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 の削除

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
```

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、システムコントローラから出荷時デフォルト構成を復元できます。

▼ システムコントローラ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、システムコントローラから出荷時デフォルト構成を復元できます。

1. システムコントローラから出荷時デフォルト構成を復元します。

```
sc> 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。

出力メッセージ

デフォルトのサービスの作成や制御 (primary) ドメインの設定に使用するコマンドで表示される出力メッセージは、プラットフォームによって異なります。

- Sun UltraSPARC T1 プロセッサ
- Sun UltraSPARC T2 および T2 Plus プロセッサ

Sun UltraSPARC T1 プロセッサ

Sun UltraSPARC T1 プロセッサを搭載したサーバーを使用している場合は、primary ドメインの設定コマンドのあとで、次のような通知が表示されます。

```
Notice: the LDom Manager is running in configuration mode. Any
configuration changes made will only take effect after the machine
configuration is downloaded to the system controller and the host
is reset.
```

Sun UltraSPARC T2 および T2 Plus プロセッサ

Sun UltraSPARC T2 または T2 Plus プロセッサを搭載したサーバーを使用している場合は、primary ドメインのデバイスまたはサービスで、動的には実行できない最初の操作を実行したあとで、次のようなメッセージが表示されます。

```
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.
```

Sun UltraSPARC T2 または T2 Plus プロセッサを搭載したサーバーを使用している場合は、再起動するまで primary ドメインに対して設定コマンドを実行するごとに、次のような通知が表示されます。

```
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 まで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 e1000g0 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=e1000g0 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。50 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=e1000g0 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 e1000g0   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. 暗号化リソースを制御ドメインに割り当てます。

注 – 制御ドメインに暗号化デバイスが割り当てられている場合は、CPU を動的に再構成することはできません。そのため、暗号化デバイスを使用していない場合は、set-mau を 0 にします。

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
```

2. 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
```

3. 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
```

4. システムコントローラ (SC) に論理ドメインの機械構成を追加します。

   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
```

5. 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、制御またはサービスドメインを再起動する必要があります。

▼ 再起動する

- **primary ドメインを停止して再起動します。この例では、primary はサービスドメインでもあります。**

```
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 e1000g0) 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、次の構成手順は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 e1000g0 です。

```
primary# ifconfig e1000g0 down unplumb
```

5. 物理ネットワークデバイス (e1000g0) 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 e1000g0 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_e1000g0 netmask netmask_of_e1000g0 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.e1000g0 /etc/hostname.vsw0
primary# mv /etc/dhcp.e1000g0 /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については、123 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、Solaris 10 Reference Manual Collection または 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 が作成されていることを確認してください。詳細は、46 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。

```
# svcadm enable vntsd
```

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

```
# svcs -l vntsd
fmri         svc:/ldoms/vntsd:default
enabled      true
state        online
next_state   none
state_time   Sat Jan 27 03:14:17 2007
```

```
logfile      /var/svc/log/ldoms-vntsd:default.log
restarter    svc:/system/svc/restarter:default
contract_id  93
dependency   optional_all/error svc:/milestone/network (online)
dependency   optional_all/none svc:/system/system-log (online)
```

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。現在は、Solaris 10 11/06 以上の OS である必要があります。必要になる可能性がある特定のパッチについては、『Logical Domains (LDoms) 1.1 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できる方法については、91 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 101 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/c0t0d0s2 vol1@primary-vds0
```

   各表記の意味は次のとおりです。

   - `/dev/dsk/c0t0d0s2` 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、またはファイルで構成される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    -----  5001    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 5001 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 5001
```

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、Sun SPARC Enterprise T5220 システムなどの 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. `primary` ドメインで、`vold`(1M) によって DVD ディスクが正常にマウントされていることを確認します。
4. 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
```

5. 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。

   仮想ディスクのリストに示されている TOUT は、ディスクの追加時に設定されたタイムアウト (設定されている場合) を意味します。この例の仮想ディスク vdisk_cd_media は、仮想ディスクサーバーへの接続の試行時に、タイムアウトしてエラーメッセージを送信するまで 60 秒間待機します。

```
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/c1t1d0s2
   iso_vol                                          /export/solarisdvd.iso
   dvd_vol                                          /dev/dsk/c0t0d0s2
   install_vol                                      /export/install_disk
....
------------------------------------------------------------------------------
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
   vdisk_cd_media   dvd_vol@primary-vds0           60
   vdisk_install    install_vol@primary-vds0
....
```

6. 別のディスク (install_disk) を作成して、Solaris OS のインストール先に追加します。

   これは、既存のファイルシステムのディスク領域を使用して OS をインストールする場合の例です。ゲストドメインに対して物理ディスクがすでに定義されている場合、その物理ディスクを使用して OS をインストールすることもできます。

```
primary# mkfile -n 20g /export/install_disk
primary# ldm add-vdsdev /export/install_disk install_vol@primary-vds0
primary# ldm add-vdisk vdisk_install install_vol@primary-vds0 ldg1
```

7. 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 ..
```

8. 必要なデバイス別名を作成します。この例では、ディスク 2 の別名を作成しています。

```
ok show-disks
a) /virtual-devices@100/channel-devices@200/disk@3
b) /virtual-devices@100/channel-devices@200/disk@2
c) /virtual-devices@100/channel-devices@200/disk@1
d) /virtual-devices@100/channel-devices@200/disk@0
q) NO SELECTION
Enter Selection, q to quit: b
```

9. 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 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_install   /virtual-devices@100/channel-devices@200/disk@3
vdisk_cd_media  /virtual-devices@100/channel-devices@200/disk@2
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
net             /virtual-devices@100/channel-devices@200/network@0
disk            /virtual-devices@100/channel-devices@200/disk@0
virtual-console /virtual-devices/console@1
name            aliases
```

10. ゲストドメインのコンソールで、スライス f の disk@2 から起動します。

```
ok boot /virtual-devices@100/channel0-devices@200/disk@2:f -v
Boot device: /virtual-devices@100/channel-devices@200/disk@2:f  File and args:
-s
```

```
SunOS Release 5.10 Version Generic_137137-09 32-bit
Copyright 1983-2008 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.
```

11. 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 (ldg1) のバインドを解除します。

```
primary# ldm unbind ldg1
```

2. Solaris ISO ファイル (solarisdvd.iso) を、たとえば二次ボリューム (iso_vol@primary-vds0) および仮想ディスク (vdisk_iso) として追加します。

```
primary# ldm add-vdsdev /export/solarisdvd.iso  iso_vol@primary-vds0
primary# 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。

   仮想ディスクのリストに示されている TOUT は、ディスクの追加時に設定されたタイムアウト (設定されている場合) を意味します。仮想ディスク vdisk_iso にはタイムアウト時間が指定されていません。

```
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/c1t1d0s2
   iso_vol                                          /export/solarisdvd.iso
   dvd_vol                                          /dev/dsk/c0t0d0s2
   install_vol                                      /export/install_disk
....
------------------------------------------------------------------------------
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
   vdisk_cd_media   dvd_vol@primary-vds0        60
   vdisk_install    install_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_install   /virtual-devices@100/channel-devices@200/disk@3
vdisk_cd_media  /virtual-devices@100/channel-devices@200/disk@2
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
net             /virtual-devices@100/channel-devices@200/network@0
disk            /virtual-devices@100/channel-devices@200/disk@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 disk@2 から起動します。

```
ok boot /virtual-devices@100/channel0-devices@200/disk@1:f -v
Boot device: /virtual-devices@100/channel-devices@200/disk@1:f  File and args:
-s
SunOS Release 5.10 Version Generic_137137-09 32-bit
Copyright 1983-2008 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 LDoms 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*s*N* という形式になります。ここで、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。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。

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。

63 ページの「ゲストドメイン構成を再構築する」の内容は、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、64 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 ls-constraints -x ldom > ldom.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 マニュアルページまたは『Logical Domains (LDoms) Manager 1.1 Man Page Guide』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。

コード例 4-1　　list-constraints サブコマンドの XML 出力のサンプル

```
<?xml version="1.0"?>
<LDM_interface version="1.1" 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>
            <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>
```

コード例 4-1　　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>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">e1000g0</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>
        <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>
```

コード例 4-1　　list-constraints サブコマンドの XML 出力のサンプル (続き)

```
            <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。

コード例 4-2　　LDoms 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。

コード例 4-3 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。

コード例 4-4 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。

コード例 4-5 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。

コード例 4-6 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
....
```

コード例 4-6 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">e1000g0</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=e1000g) 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=e1000g 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>
```

各表記の意味は次のとおりです。

- 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>
```

各表記の意味は次のとおりです。

- 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
```

第5章

Logical Domains ソフトウェアでの PCI バスの使用

この章では、複数の論理ドメインにまたがって PCI Express バスを構成する方法、および PCI バスで I/O MMU バイパスモードを有効にする方法について説明します。

複数の論理ドメインにまたがる PCI Express バスの構成

注 – Sun SPARC Enterprise T5120 および T5220 サーバーなどの Sun UltraSPARC T-2 ベースのサーバーの場合は、この手順を使用せずに、論理ドメインにはネットワークインタフェースユニット (NIU) を割り当てます。

Sun UltraSPARC T1 ベースのサーバーの PCI Express (PCIe) バスは、さまざまなリーフデバイスが接続される 2 つのポートで構成されます。これらは、`pci@780 (bus_a)` および `pci@7c0 (bus_b)` という名前でサーバーで識別されます。マルチドメイン環境では、Logical Domains Manager を使用して各リーフに個別のドメインを割り当てるように、PCIe バスをプログラムすることができます。つまり、I/O の仮想化を使用する代わりに、複数のドメインが物理デバイスへ直接アクセスできるようにすることができます。

Logical Domains システムに電源が入ると、制御 (`primary`) ドメインはすべての物理デバイスリソースを使用します。このため、primary ドメインが PCIe バスの両方のリーフを所有しています。

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 つのリーフに接続されています。制御ドメインが内部ディスクから起動する場合は、ドメインからそのリーフを削除しないでください。また、主ネットワークのポートを持つリーフを削除していないことを確認してください。制御ドメインまたはサービスドメインから誤ったリーフを削除すると、そのドメインは必要なデバイスにアクセスできず、使用不可になります。主ネットワークのポートがシステムディスク以外の異なるバスにある場合は、ネットワークケーブルをボード上のネットワークポートに移動し、Logical Domains Manager を使用して仮想スイッチ (vsw) を再構成してこの変更を反映してください。

▼ 分割 PCI 構成を作成する

ここで示す例は、Sun Fire T2000 サーバーの場合です。この手順は、Sun Fire T1000 サーバーおよび Netra T2000 サーバーなどの Sun UltraSPARC T1 ベース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。ほとんどの場合、起動ディスクを持つリーフを保持したまま、その他のリーフを primary ドメインから削除してほかのドメインに割り当てる必要があります。

1. `primary` ドメインが PCI Express バスの両方のリーフ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@780         bus_a
    pci@7c0         bus_b
...
```

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。

```
primary# df /
/              (/dev/dsk/c1t0d0s0 ): 1309384 blocks   457028 files
```

3. ブロック型デバイス c1t0d0s0 が接続されている物理デバイスを確認します。

```
primary# ls -l /dev/dsk/c1t0d0s0
lrwxrwxrwx   1 root     root          65 Feb  2 17:19 /dev/dsk/c1t0d0s0 -> ../
../devices/pci@7c0/pci@0/pci@1/pci@0,2/LSILogic,sas@2/sd@0,0:a
```

この例では、ドメイン primary の起動ディスクに対する物理デバイスは、前述の bus_b に対応する、リーフ pci@7c0 の下にあります。つまり、PCIe バスの bus_a (pci@780) を別のドメインに割り当てることができます。

4. /etc/path_to_inst を確認して、ボード上のネットワークポートの物理パスを見つけます。

```
primary# grep e1000g /etc/path_to_inst
```

5. primary ドメインから起動ディスク (この例では pci@780) を含まないリーフを削除します。

```
primary# ldm remove-io pci@780 primary
```

6. この分割 PCI 構成 (この例では split-cfg) をシステムコントローラに追加します。

```
primary# ldm add-config split-cfg
```

また、この構成 (split-cfg) は、再起動後に使用される次の構成として設定されます。

注 – 現在、SC 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。

---

7. 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
```

8. 直接のアクセスが必要なドメイン (この例では ldg1) にリーフ (この例では pci@780) を追加します。

```
primary# ldm add-io pci@780 ldg1
Notice: the LDom Manager is running in configuration mode. Any
configuration changes made will only take effect after the machine
configuration is downloaded to the system controller and the
host is reset.
```

Infiniband カードが構成されていると、pci@780 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、77 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。

9. ドメイン ldg1 を再起動して、変更を有効にします。

再起動する場合は、すべてのドメインをアクティブでない状態にする必要があります。このドメインをはじめて構成する場合、ドメインはアクティブではありません。

```
ldg1# shutdown -i6 -g0 -y
```

10. 適切なリーフが primary ドメインに割り当てられたままで、適切なリーフがドメイン ldg1 に割り当てられていることを確認します。

```
primary# ldm list-bindings primary
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active   -n-cv   SP      4     4G       0.4%  18h 25m
...
IO
    DEVICE           PSEUDONYM         OPTIONS
    pci@7c0          bus_b
...
------------------------------------------------------------------------
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg1             active   -n---   5000    4     2G       10%   35m
...
IO
    DEVICE           PSEUDONYM         OPTIONS
    pci@780          bus_a
...
```

この出力では、PCIe リーフ bus_b とその配下のデバイスがドメイン primary に割り当てられており、bus_a 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 PCIe 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 PCIe 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 PCIe 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@780 ldg1
```

出力では、OPTIONS の下に bypass=on が表示されます。

第6章

Logical Domains での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。

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。

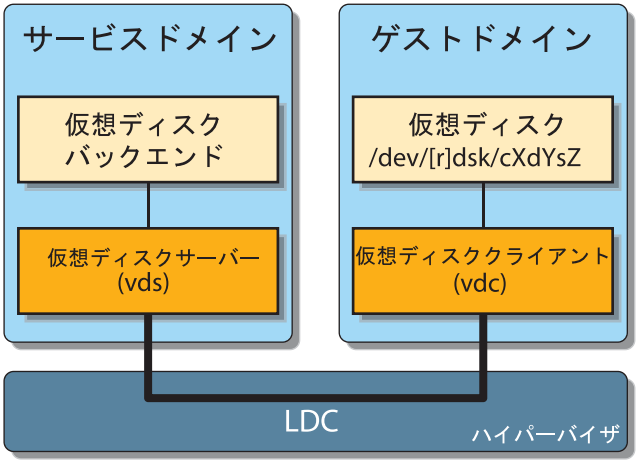
仮想ディスクバックエンドは、物理的でも論理的でもかまいません。物理デバイスには、次のものを含めることができます。

- 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、84 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、99 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] disk_name volume_name@service_name ldom
```

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 [options={ro,slice}] backend volume2@service_name
```

`add-vdsdev` サブコマンドは、次の警告を表示して、バックエンドが複数回エクスポートされていることを示しています。

```
Warning: "backend" is already in use by one or more servers in
guest "ldom"
```

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 番目のスライス (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、107 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、108 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、81 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、86 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. **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、次のようにそのディスクのスライス 2 (`c1t48d0s2`) をサービスドメインからエクスポートする必要があります。**

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

2. **たとえば、サービスドメインから、ディスク (`pdisk`) をゲストドメイン `ldg1` に割り当てます。**

```
service# 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、ディスク (c0d1 など) 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 (8 つのスライスを持つ通常のディスク) であることを確認できます。

```
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. たとえば、物理ディスク c1t57d0 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サービスドメインから次のようにエクスポートする必要があります。

```
service# 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. たとえば、サービスドメインから、ディスク (pslice) をゲストドメイン ldg1 に割り当てます。

```
service# 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 番目のスライス (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. **サービスドメインから、ファイルを仮想ディスクとしてエクスポートします。**

```
service# 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. たとえば、サービスドメインから、ディスク (fdisk) をゲストドメイン ldg1 に割り当てます。

```
service# 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、ディスク (c0d5 など) を表示して、そのディスクがアクセス可能で、フルディスク (8 つのスライスを持つ通常のディスク) であることを確認できます。

```
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. サービスドメインから、ZFS ボリューム (zdisk0 など) を作成して、1 つのスライスディスクとして使用します。

```
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. サービスドメインから、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。

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

3. たとえば、サービスドメインから、ボリューム (zdisk0) をゲストドメイン ldg1 に割り当てます。

```
service# 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。

- LDoms 1.1 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` マニュアルページまたは『Logical Domains (LDoms) Manager 1.1 Man Page Guide』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 つのスライスディスク ‡
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 †	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
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

* ディスク全体をエクスポートします。

† 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。

‡ スライス 2 のみをエクスポートします。

ガイドライン

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。88 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。

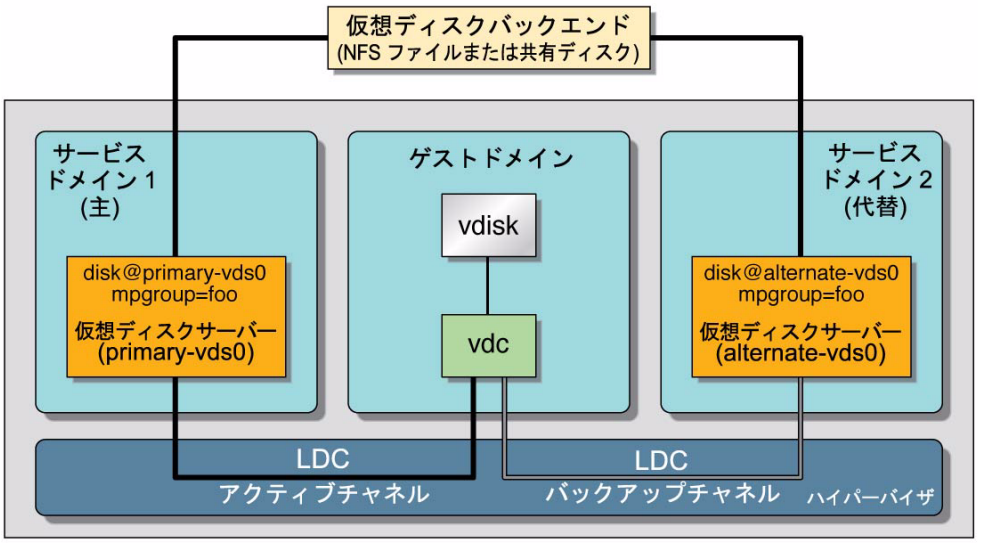
仮想ディスクマルチパスの構成

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。

図 6-2 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。

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、97 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. CD または DVD ドライブに CD または DVD を挿入します。
2. 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
```

3. 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 5 に進みます。

- 手順 2 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....
```

詳細は、`vold.conf`(1M) マニュアルページを参照してください。

b. 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
```

4. 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。

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

5. サービスドメインから、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。

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

6. サービスドメインから、エクスポートされた CD または DVD をゲストドメイン (この例では `ldg1`) に割り当てます。

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

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、81 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。

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、ファイルシステムまたはアプリケーションが入出力処理をブロックしない場合がありますが、そのような場合でもサービスドメインの停止状態が長すぎる場合は失敗し、エラーが報告されます。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`(7D) 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、フルディスクとしてエクスポートされる仮想ディスクを使用するゲストドメインで機能します。1 つのスライスディスクは、`format`(1M) コマンドでは表示されません。また、このようなディスクのパーティション分割を変更することはできません。

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、一部の `format`(1M) サブコマンド (`repair`、`defect` など) 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 10 System Administrator Collection の『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

次の例では、ゲストドメイン `ldg1` にディスクイメージを作成します。このためには、このゲストドメイン用に ZFS 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こ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 ー 約 6G バイト
- スワップパーティション ー 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 ー 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。

▼ 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(1M) コマンドを実行します。
3. sys-unconfig(1M) 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. たとえば、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。

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

6. 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。このイメージを複製して新規ドメインを作成することができます。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。

   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。

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。たとえば、サービスドメインが SVM ボリューム `/dev/md/dsk/d0` を `domain1` に 1 つのスライスディスクとしてエクスポートし、`domain1` では仮想ディスクが `/dev/dsk/c0d2*` として認識されている場合、`domain1` には `s0` デバイス、つまり `/dev/dsk/c0d2s0` のみが存在します。

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
```

注 – 1 つのスライスディスクは format(1M) コマンドでは認識できず、パーティションに分割できません。また、Solaris OS のインストールディスクとしても使用できません。この項目の詳細は、83 ページの「仮想ディスクの表示」を参照してください。

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/c10t600C0FF000000000015153295A4B100d0s1  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%5F30323339483031443 04E0000
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

Logical Domains での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。

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。*n* は、仮想スイッチのインスタンスに対応する数字で、サービスドメインの 1 番めの仮想スイッチは `vsw0` になります。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、50 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。*n* は、仮想ネットワークデバイスのインスタンスに対応する数字です。仮想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

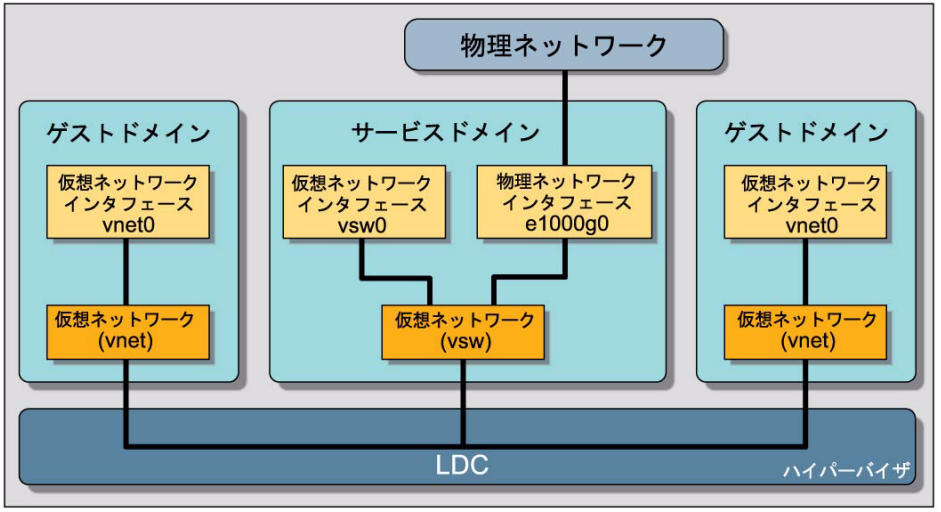


図 7-1 の例の説明は、次のとおりです。

- 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 `e1000g0` 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,...]
[mac-addr=num] [net-dev=device] [mode=sc] vswitch_name ldom
```

各表記の意味は次のとおりです。

- `default-vlan-id=`*vlan-id* には、仮想スイッチとそれに関連付けられた仮想ネットワークデバイスが暗黙に属しているデフォルトの仮想ローカルエリアネットワーク (VLAN) を、タグなしモードで指定します。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、128 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、128 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* には、仮想スイッチをメンバーにする必要のある 1 つ以上の VLAN をタグ付きモードで指定します。詳細は、128 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。
- `mac-addr=`*num* は、このスイッチが使用する MAC アドレスです。数字は、80:00:33:55:22:66 のように標準の 8 ビット表記にする必要があります。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、118 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、このスイッチが処理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 ソフトウェアを動作させない場合は、このオプションを設定しないでください。仮想ネットワークのパフォーマンスに影響することがあります。
- *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]
[net-dev=device] [mode=[sc]] 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、113 ページの「仮想スイッチを追加する」の説明と同じです。

▼ 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]
[vid=vlan-id1,vlan-id2,...] 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、118 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、131 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、128 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* 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 1 つ以上の VLAN をタグ付きモードで指定します。詳細は、128 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。
- *if_name* は、後続の set-vnet または rm-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。
- *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] [vid=vlan-id1,vlan-id2,...] 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* は、設定する仮想ネットワークデバイスに割り当てられる一意のインタフェース名です。
- *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、115 ページの「仮想ネットワークデバイスを追加する」の説明と同じです。

▼ 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* は、削除する仮想ネットワークデバイスに割り当てられる一意のインタフェース名です。
- *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。

仮想ネットワークデバイスに対応する Solaris ネットワークインタフェース名の判定

`ldm list-*` コマンドによって提供される出力で、特定の仮想デバイスに対応するゲストの Solaris OS ネットワークインタフェース名を直接判定する方法はありません。ただし、`ldm list -l` コマンドの出力と、Solaris OS ゲストの `/devices` 配下のエントリを組み合わせて使用すると、これを判定することができます。

▼ 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。

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インスタンスを探します。

```
# 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インスタンスは network@2 です。

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 配下でこのインスタンス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。

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 (121 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。使用されるアルゴリズムは、121 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、123 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。

▼ 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、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。

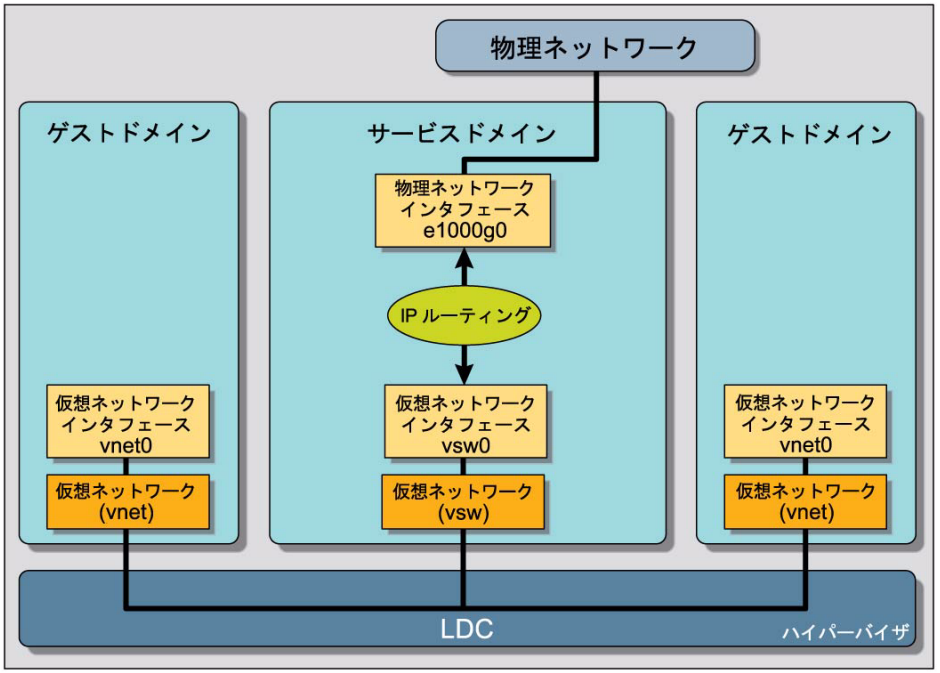
NAT およびルーティング用の仮想スイッチおよびサービスドメインの構成

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、51 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、51 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 Express System Administrator Collection の『System Administration Guide: IP Services』の第 5 章「Configuring TCP/IP Network Services and Ipv4 Addressing (Tasks)」の「Packet Forwarding and Routing on Ipv4 Networks」の節を参照してください。

論理ドメイン環境での IPMP の構成

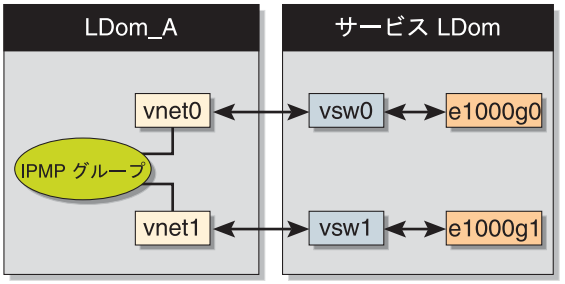
インターネットプロトコルネットワークマルチパス (IPMP) は、複数のネットワークインタフェースカード間の耐障害性と負荷分散を提供します。IPMP を使用すると、1 つ以上のインタフェースを IP マルチパスグループとして構成できます。IPMP を構成すると、システムは IPMP グループ内のインタフェースで障害が発生していないかを自動的に監視します。グループ内のインタフェースに障害が発生したり、保守のために削除されたりすると、IPMP は障害の発生したインタフェースの IP アドレスを自動的に移行して、フェイルオーバーを行います。論理ドメイン環境では、物理ネットワークインタフェースまたは仮想ネットワークインタフェースのいずれかで IPMP を使用したフェイルオーバーを構成できます。

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

IPMP グループに仮想ネットワークデバイスを構成することで、論理ドメインに耐障害性を持たせるように構成できます。アクティブ/スタンバイ構成で、仮想ネットワークデバイスを使用して IPMP グループを設定する場合は、グループでプローブベースの検出を使用するようにグループを設定します。Logical Domains 1.1 ソフトウェアでは、現在、仮想ネットワークデバイスに対するリンクベースの検出とフェイルオーバーはサポートされていません。

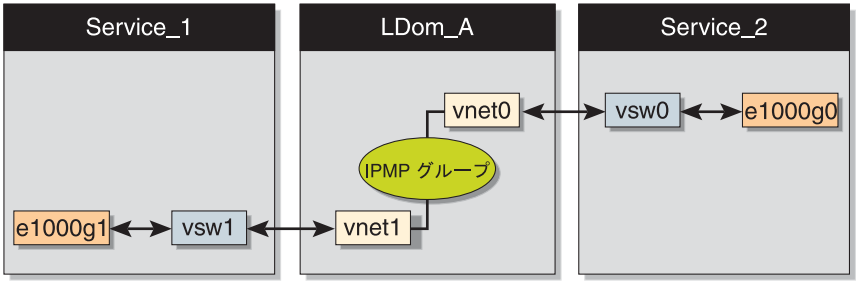
次の図に、サービスドメインで個別の仮想スイッチインスタンス (vsw0 および vsw1) に接続された 2 つの仮想ネットワーク (vnet0 および vnet1) を示します。これらは、同様に、2 つの異なる物理インタフェース (e1000g0 および e1000g1) を使用します。物理インタフェースに障害が発生した場合、LDom_A の IP 層が、プローブベースの検出を使用して対応する vnet の障害と接続の損失を検出し、vnet の二次デバイスに自動的にフェールオーバーします。

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。仮想スイッチインスタンス (vsw0 および vsw1) が構成された 2 つのサービスドメイン (Service_1 および Service_2) は、分割 PCI 構成を使用して設定できます。この場合、ネットワークハードウェアの障害に加えて、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。

▼ ホストルートを構成する

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 10 の『Solaris のシステム管理 (IP サービス)』のパート VI「IPMP」の第 31 章「IPMP の管理 (手順)」の「ターゲットシステムの構成」を参照してください。

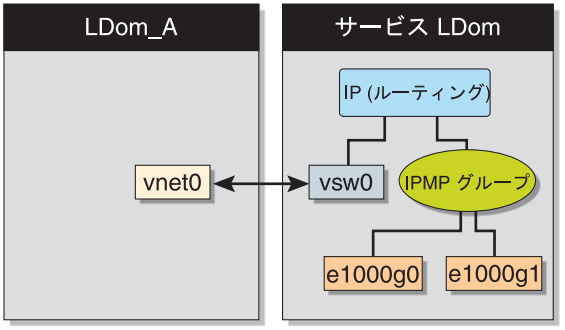
サービスドメインでの IPMP の構成と使用

サービスドメインの物理インタフェースを IPMP グループとして構成すると、論理ドメイン環境でのネットワーク障害の検出と復旧を設定することもできます。これを行うには、サービスドメインの仮想スイッチをネットワークデバイスとして構成し、サービスドメイン自体を IP ルーターとして動作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、物理デバイスを経由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 (`e1000g0` および `e1000g1`) 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 7-5　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



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) マニュアルページまたは『Logical Domains (LDoms) Manager 1.1 Man Page Guide』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. たとえば、仮想スイッチ (vsw) を 2 つの VLAN に割り当てます。VLAN 21 をタグなし、VLAN 20 をタグ付きとして構成します。たとえば、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=e1000g0 pvid=21 vid=20 primary-vsw0 primary
# ldm add-vnet vnet01 primary-vsw0 pvid=20 vid=21,22 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)
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
```

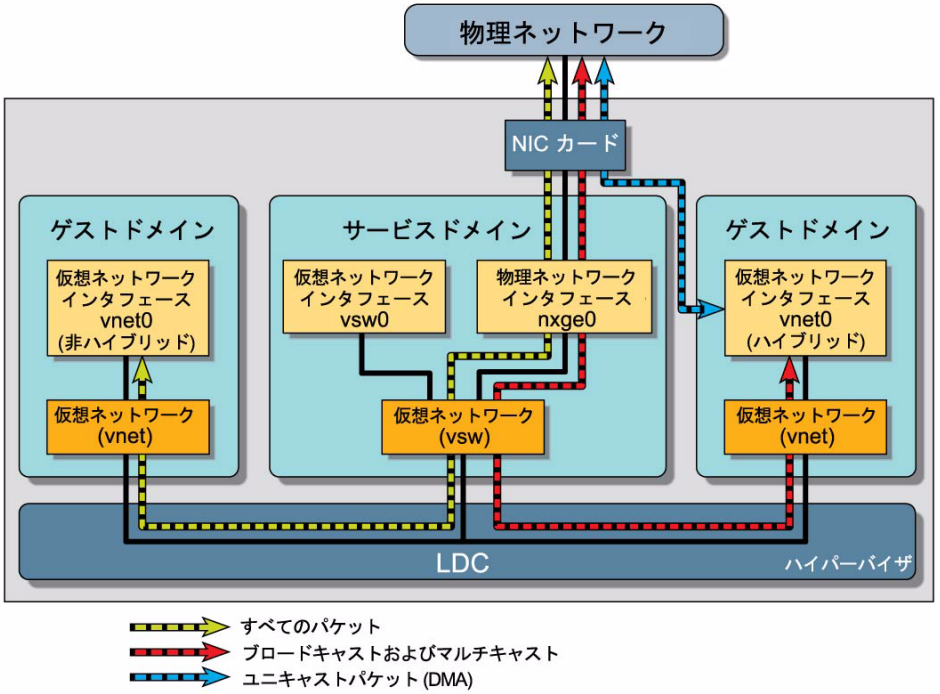
   Solaris OS で VLAN インタフェースを構成する方法の詳細は、『Solaris のシステム管理 (IP サービス)』の「仮想ローカルエリアネットワークの管理」を参照してください。

NIU ハイブリッド I/O の使用

LDoms 1.1 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。

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、Sun UltraSPARC T2 ベースのプラットフォームで NIU ハイブリッド I/O がサポートされます。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。同じシステム内の他のゲストドメインへのブロード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 7-6　　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。134 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、『Logical Domains (LDoms) Manager 1.1 Man Page Guide』または ldm 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
```

第8章

論理ドメインの移行

この章では、今回のリリースの LDoms 1.1 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。

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は、最新バージョンの LDoms 1.1 ファームウェアをサポートしている必要があります。

  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 – 今回が移行機能の最初のリリースであるため、両方のマシンで LDoms 1.1 ソフトウェアおよび最新のファームウェアが動作している必要があります。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 (LDoms) 1.1 リリースノート』を参照してください。

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、ユーザーは、両方のマシンで `solaris.ldoms.write` 承認を受ける必要があります。

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、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

LDoms 1.1 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。

CPU

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。

メモリー

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。

物理入出力

移行される論理ドメインに物理 I/O デバイスを含めないでください。ドメインに物理 I/O デバイスが含まれていると、移行は失敗します。

仮想入出力

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。

NIU ハイブリッド入出力

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。

暗号化装置

暗号化装置をバインドした論理ドメインを移行することはできません。このようなドメインを移行する試みは失敗します。

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、起動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。

CPU

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。

仮想入出力

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。特に、省略形式の状態出力では、移行中のドメインの状態を示す新しいフラグが表示されます。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。

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。

コード例 8-1　　進行中の移行の監視

```
# ldm ls -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 コマンドから cancel-operation サブ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、ソースドメインはマスタードメインとして再開されます。

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。

例

コード例 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。

コード例 8-2　　ゲストドメインの移行

```
# ldm migrate-domain ldg1 t5440-sys-2
Target Password:
#
```

コード例 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。

コード例 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:
#
```

コード例 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。

コード例 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).
```

コード例 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。

コード例 8-5 ターゲットドメインの状態の取得

```
# ldm ls -o status ldg-tgt
NAME
ldg-tgt

STATUS
    OPERATION    PROGRESS    SOURCE
    migration    55%         t5440-sys-1
```

コード例 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。

コード例 8-6 ソースドメインの解析可能な状態の取得

```
# ldm ls -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

その他の情報とタスク

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。

LDoms 1.1 ソフトウェアでの CPU Power Management の使用

LDoms 1.1 ソフトウェアで CPU Power Management (PM) 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 を使用するために必要な情報の概要を示します。詳細は、『Sun Integration Lights Out Management (ILOM) 3.0 CLI Procedures Guide』の「Monitoring Power Consumption」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 1.1) 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 Integration Lights Out Management (ILOM) 3.0 CLI Procedures Guide』の「Monitoring Power Consumption」を参照してください。

LDoms 1.1 ソフトウェアでの CPU で電源管理されているストランドの表示

この節では、LDoms 1.1 ソフトウェアを使用して、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

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 ls -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=
```

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

- 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*)

システムコントローラ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*)、および仮想ディスク名 (*disk_name*) などのその他の名前は、次のような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。

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、159 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。**

コード例 9-1　`ldm` のすべてのサブコマンドの構文の使用法

```
primary# ldm --help

Usage:
      ldm [--help] command [options] [properties] operands
      ldm -V
```

コード例 9-1 ldm のすべてのサブコマンドの構文の使用法 (続き)

```
Options:
     -V         Display version information

Command(s) for each resource (aliases in parens):

     bindings
         list-bindings [-e] [-p] [<ldom>...]

     services
         list-services [-e] [-p] [<ldom>...]

     constraints
         list-constraints ([-x] | [-e] [-p]) [<ldom>...]

     devices
         list-devices [-a] [-p] [cpu] [crypto|mau] [memory] [io]

     domain      ( dom )
         add-domain (-i <file> | mac-addr=<num] [hostid=<num>] <ldom> |
             <ldom>...)
         remove-domain (-a | <ldom>...)
         list-domain [-e] [-l] [-o <format>] [-p] [<ldom>...]
            'format' is one or more of:
            console,cpu,crypto,disk,domain,memory,network,physio,serial,status
         start-domain (-a | -i <file> | <ldom>...)
         stop-domain [-f] (-a | <ldom>...)
         bind-domain (-i <file> | <ldom>)
         unbind-domain <ldom>
         panic-domain <ldom>
         migrate-domain [-n|--dry-run] <source_ldom>
             [<user>@]<target_host>[:<target_ldom>]

     io
         add-io [bypass=on] <bus> <ldom>
         remove-io <bus> <ldom>

     crypto       ( mau )
         add-crypto <number> <ldom>
         set-crypto <number> <ldom>
         remove-crypto <number> <ldom>

     memory       ( mem )
         add-memory <number>[GMK] <ldom>
         set-memory <number>[GMK] <ldom>
         remove-memory <number>[GMK] <ldom>
```

コード例 9-1　　ldm のすべてのサブコマンドの構文の使用法 (続き)

```
operation
    cancel-operation (migration | reconf) <ldom>

reconf
    cancel-reconf <ldom>

spconfig      ( config )
    add-spconfig <config_name>
    set-spconfig <config_name>
    remove-spconfig <config_name>
    list-spconfig

variable    ( var )
    add-variable <var_name>=<value>... <ldom>
    set-variable <var_name>=<value>... <ldom>
    remove-variable <var_name>... <ldom>
    list-variable [<var_name>...] <ldom>

vconscon    ( vcc )
    add-vconscon port-range=<x>-<y> <vcc_name> <ldom>
    set-vconscon port-range=<x>-<y> <vcc_name>
    remove-vconscon [-f] <vcc_name>

vconsole    ( vcons )
    set-vcons [port=[<port-num>]] [group=<group>] [service=<vcc_server>]
        <ldom>

vcpu
    add-vcpu <number> <ldom>
    set-vcpu <number> <ldom>
    remove-vcpu <number> <ldom>

vdisk
    add-vdisk [timeout=<seconds>] <disk_name>
         <volume_name>@<service_name> <ldom>
    set-vdisk [timeout=<seconds>] [volume=<volume_name>@<service_name>]
        <disk_name> <ldom>
    remove-vdisk [-f] <disk_name> <ldom>

vdiskserver ( vds )
    add-vdiskserver <service_name> <ldom>
    remove-vdiskserver [-f] <service_name>

vdpcc       ( ndpsldcc )
    add-vdpcc <vdpcc_name> <service_name> <ldom>
    remove-vdpcc [-f] <vdpcc_name> <ldom>
```

コード例 9-1 ldm のすべてのサブコマンドの構文の使用法 (続き)

```
      vdpcs        ( ndpsldcs )
          add-vdpcs <vdpcs_name> <ldom>
          remove-vdpcs [-f] <vdpcs_name>

      vdiskserverdevice   ( vdsdev )
          add-vdiskserverdevice [options={ro,slice,excl}] [mpgroup=<mpgroup>]
              <backend>  <volume_name>@<service_name>
          set-vdiskserverdevice options=[{ro,slice,excl}] [mpgroup=<mpgroup>]
              <volume_name>@<service_name>
          remove-vdiskserverdevice [-f] <volume_name>@<service_name>

      vnet
          add-vnet [mac-addr=<num>] [mode=hybrid] [pvid=<pvid>]
              [vid=<vid1,vid2,...>] <if_name> <vswitch_name> <ldom>
          set-vnet [mac-addr=<num>] [mode=[hybrid]] [pvid=[<pvid>]]
              [vid=[<vid1,vid2,...>]] [vswitch=<vswitch_name>] <if_name>  <ldom>
          remove-vnet [-f] <if_name> <ldom>

      vswitch     ( vsw )
          add-vswitch [default-vlan-id=<vid>] [pvid=<pvid>]
               [vid=<vid1,vid2,...>] [mac-addr=<num>] [net-dev=<device>]
               [mode=<mode>] <vswitch_name> <ldom>
          set-vswitch [pvid=[<pvid>]] [vid=[<vid1,vid2,...>]] [mac-addr=<num>]
               [net-dev=<device>] [mode=<mode>] <vswitch_name>
          remove-vswitch [-f] <vswitch_name>

Verb aliases:
          Alias          Verb
          -----          -------
          rm             remove
          ls             list

Command aliases:
          Alias          Command
          -----          -------
          cancel-op      cancel-operation
          create         add-domain
          destroy        remove-domain
          remove-reconf  cancel-reconf
          start          start-domain
          stop           stop-domain
          bind           bind-domain
          unbind         unbind-domain
          panic          panic-domain
          migrate        migrate-domain
```

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例

注 – 実際の出力は、ここに示す出力とは少し異なる場合があります。

▼ ソフトウェアのバージョンを表示する (-V)

- **現在インストールされているソフトウェアのバージョンを表示すると、例に示すようなリストが出力されます。**

コード例 9-2 インストールされているソフトウェアのバージョン

```
primary$ ldm -V

Logical Domain Manager (v 1.1)
   Hypervisor control protocol v 1.3
   Using Hypervisor MD v 0.1

System PROM:
   Hypervisor  v. 1.7.0.    @(#)Hypervisor 1.7.0. 2008/11/19 10:20
   OpenBoot    v. 4.30.0.   @(#)OBP 4.30.0. 2008/11/18 13:44
```

▼ 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。

コード例 9-3　　すべてのドメインの省略形式のリスト

```
primary$ ldm list
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active   -t-cv           4     1G       0.5%  3d 21h 7m
ldg1             active   -t---   5000    8     1G        23%  2m
```

▼ 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。

コード例 9-4　　すべてのドメインの長形式のリスト

```
primary$ ldm list -l
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active   -t-cv           1     768M     0.0%  0s

VCPU
    VID    PID    UTIL STRAND
    0      0      0.0%   100%

MEMORY
    RA               PA               SIZE
    0x4000000        0x4000000        768M

IO
    DEVICE           PSEUDONYM        OPTIONS
    pci@780          bus_a
    pci@7c0          bus_b            bypass=on

VCC
    NAME             PORT-RANGE
    vcc0             5000-5100

VSW
    NAME             MAC               NET-DEV   DEVICE    MODE
    vsw0             08:00:20:aa:bb:e0 e1000g0   switch@0  prog,promisc
    vsw1             08:00:20:aa:bb:e1                       routed

VDS
    NAME             VOLUME         OPTIONS          DEVICE
    vds0             myvol-a        slice            /disk/a
                     myvol-b                         /disk/b
                     myvol-c        ro,slice,excl    /disk/c
```

コード例 9-4　　すべてのドメインの長形式のリスト (続き)

```
    vds1             myvol-d                          /disk/d

VDPCS
    NAME
    vdpcs0
    vdpcs1

------------------------------------------------------------------------------
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg1             bound    -----   5000    1     512M

VCPU
    VID    PID    UTIL STRAND
    0      1              100%

MEMORY
    RA               PA               SIZE
    0x4000000        0x34000000       512M

NETWORK
    NAME         SERVICE                     DEVICE       MAC
    mynet-b      vsw0@primary                network@0    08:00:20:ab:9a:12
    mynet-a      vsw0@primary                network@1    08:00:20:ab:9a:11

DISK
    NAME             VOLUME                      DEVICE     SERVER
    mydisk-a         myvol-a@vds0                disk@0     primary
    mydisk-b         myvol-b@vds0                disk@1     primary

VDPCC
    NAME             SERVICE
    myvdpcc-a        vdpcs0@primary
    myvdpcc-b        vdpcs0@primary

VCONS
    NAME             SERVICE                     PORT
    mygroup          vcc0@primary                5000
```

▼ 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。

コード例 9-5　　すべてのドメインの拡張リスト

```
primary$ ldm list -e
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
primary          active   -t-cv           1     768M     0.0%  0s

VCPU
    VID    PID    UTIL STRAND
    0      0      0.0%   100%

MEMORY
    RA               PA               SIZE
    0x4000000        0x4000000        768M

IO
    DEVICE           PSEUDONYM        OPTIONS
    pci@780          bus_a
    pci@7c0          bus_b            bypass=on

VLDC
    NAME
    primary

VCC
    NAME             PORT-RANGE
    vcc0             5000-5100

VSW
    NAME             MAC               NET-DEV   DEVICE    MODE
    vsw0             08:00:20:aa:bb:e0 e1000g0   switch@0  prog,promisc
    vsw1             08:00:20:aa:bb:e1                     routed

VDS
    NAME             VOLUME         OPTIONS          DEVICE
    vds0             myvol-a        slice            /disk/a
                     myvol-b                         /disk/b
                     myvol-c        ro,slice,excl    /disk/c
    vds1             myvol-d                         /disk/d

VDPCS
    NAME
    vdpcs0
    vdpcs1

VLDCC
```

コード例 9-5 すべてのドメインの拡張リスト (続き)

```
    NAME             SERVICE                     DESC
    hvctl            primary@primary             hvctl
    vldcc0           primary@primary             ds

------------------------------------------------------------------------------
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME
ldg1             bound    -----   5000    1     512M

VCPU
    VID    PID    UTIL STRAND
    0      1              100%

MEMORY
    RA               PA               SIZE
    0x4000000        0x34000000       512M

VLDCC
 NAME             SERVICE                     DESC
 vldcc0           primary@primary             ds

NETWORK
    NAME         SERVICE                     DEVICE       MAC
    mynet-b      vsw0@primary                network@0    08:00:20:ab:9a:12
    mynet-a      vsw0@primary                network@1    08:00:20:ab:9a:11

DISK
    NAME             VOLUME                      DEVICE     SERVER
    mydisk-a         myvol-a@vds0                disk@0     primary
    mydisk-b         myvol-b@vds0                disk@1     primary

VDPCC
    NAME             SERVICE
    myvdpcc-a        vdpcs0@primary
    myvdpcc-b        vdpcs0@primary

VCONS
    NAME             SERVICE                     PORT
    mygroup          vcc0@primary                5000
```

▼ 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。**

コード例 9-6　　マシンが読み取り可能なリスト

```
primary$ ldm list -p
VERSION 1.0
DOMAIN|name=primary|state=active|flags=-t-cv|cons=|ncpu=1|mem=805306368|util=
0.0|uptime=0
DOMAIN|name=ldg1|state=bound|flags=-----|cons=5000|ncpu=1|mem=536870912|util=
|uptime=
```

▼ 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。**
  - `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。
  - `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。
  - `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。

コード例 9-7　制御ドメインの CPU 情報のリスト

```
# ldm ls -o cpu primary
NAME
primary

VCPU
   VID    PID    UTIL STRAND
   0      0      1.0%   100%
   1      1      0.6%   100%
   2      2      0.2%   100%
   3      3      0.5%   100%
```

コード例 9-8　ゲストドメインのドメイン情報のリスト

```
# ldm ls -o domain ldm2
NAME             STATE    FLAGS
ldm2             active   -t---

SOFTSTATE
Openboot initializing

VARIABLES
   auto-boot?=false
   boot-device=/virtual-devices@100/channel-devices@200/disk@0
```

コード例 9-9　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 ls -o network,memory ldm1
NAME
ldm1

MAC
   00:14:4f:f9:dd:ae

MEMORY
   RA               PA               SIZE
   0x6800000        0x46800000       1500M

NETWORK
NAME           SERVICE               DEVICE     MAC               MODE PVID
VID
ldm1-network0  primary-vsw0@primary network@0  00:14:4f:fb:21:0f    1
```

▼ 変数を一覧表示する

- ドメイン (ldg1 など) の変数 (boot-device など) を一覧表示します。

コード例 9-10　　ドメインの変数のリスト

```
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 (ldg1 など) を一覧表示します。

コード例 9-11　　ドメインのバインドのリスト

```
primary$ ldm list-bindings ldg1
NAME             STATE    FLAGS   CONS    VCPU  MEMORY   UTIL  UPTIME

ldg1             bound    -----   5000    1     512M

VCPU
    VID    PID    UTIL STRAND
    0      1              100%

MEMORY
    RA               PA               SIZE
    0x4000000        0x34000000       512M

NETWORK
    NAME             SERVICE                     DEVICE     MAC
    mynet-b          vsw0@primary                network@0  08:00:20:ab:9a:12
        PEER                         MAC
        vsw0@primary                 08:00:20:aa:bb:e0
        mynet-a@ldg1                 08:00:20:ab:9a:11
        mynet-c@ldg2                 08:00:20:ab:9a:22
    NAME             SERVICE                     DEVICE     MAC
    mynet-a          vsw0@primary                network@1  08:00:20:ab:9a:11
        PEER                         MAC
        vsw0@primary                 08:00:20:aa:bb:e0
        mynet-b@ldg1                 08:00:20:ab:9a:12
        mynet-c@ldg2                 08:00:20:ab:9a:22

DISK
    NAME             VOLUME                      DEVICE     SERVER
    mydisk-a         myvol-a@vds0                disk@0     primary
    mydisk-b         myvol-b@vds0                disk@1     primary

VDPCC
    NAME             SERVICE
```

コード例 9-11　　ドメインのバインドのリスト (続き)

```
    myvdpcc-a         vdpcs0@primary
    myvdpcc-b         vdpcs0@primary

VCONS
    NAME              SERVICE                     PORT
    mygroup           vcc0@primary                5000
```

▼ 構成を一覧表示する

- SC に格納されている論理ドメイン構成を一覧表示します。

コード例 9-12　　構成のリスト

```
primary$ ldm list-config
factory-default
3guests
foo [next poweron]
primary
reconfig-primary
```

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。再構成後、このラベルは [next poweron] に変わります。
- [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。

コード例 9-13　　すべてのサーバーリソースのリスト

```
primary$ ldm list-devices -a
VCPU
    PID  %FREE     PM
    0       0      NO
    1       0      YES
    2       0      YES
    3       0      YES
    4       100    ---
    5       100    ---
    6       100    ---
    7       100    ---
    8       100    ---
    9       100    ---
    10      100    ---
    11      100    ---
    12      100    ---
    13      100    ---
    14      100    ---
    15      100    ---
    16      100    ---
    17      100    ---
    18      100    ---
    19      100    ---
    20      100    ---
    21      100    ---
    22      100    ---
    23      100    ---
    24      100    ---
    25      100    ---
    26      100    ---
    27      100    ---
    28      100    ---
    29      100    ---
    30      100    ---
    31      100    ---

MAU
    CPUSET                                      BOUND
    (0, 1, 2, 3)                                ldg2
    (4, 5, 6, 7)
    (8, 9, 10, 11)
    (12, 13, 14, 15)
```

コード例 9-13　すべてのサーバーリソースのリスト (続き)

```
    (16, 17, 18, 19)
    (20, 21, 22, 23)
    (24, 25, 26, 27)
    (28, 29, 30, 31)

MEMORY
    PA                  SIZE            BOUND
    0x0                 512K            _sys_
    0x80000             1536K           _sys_
    0x200000            62M             _sys_
    0x4000000           768M            primary
    0x34000000          512M            ldg1
    0x54000000          8M              _sys_
    0x54800000          2G              ldg2
    0xd4800000          29368M

IO
    DEVICE          PSEUDONYM       BOUND   OPTIONS
    pci@780         bus_a           yes
    pci@7c0         bus_b           yes     bypass=on
```

▼ 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。**

コード例 9-14　　サービスのリスト

```
primary$ ldm list-services
VDS
    NAME             VOLUME         OPTIONS          DEVICE
    primary-vds0
VCC
    NAME             PORT-RANGE
    primary-vcc0     5000-5100
VSW
   NAME          MAC               NET-DEV  DEVICE   MODE
   primary-vsw0 00:14:4f:f9:68:d0 e1000g0  switch@0 prog,promisc
```

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 (`ldg1` など) の制約を一覧表示します。

コード例 9-15　1 つのドメインの制約のリスト

```
primary$ ldm list-constraints ldg1
DOMAIN
ldg1

VCPU
    COUNT
    1

MEMORY
    SIZE
    512M

NETWORK
    NAME          SERVICE                     DEVICE       MAC
    mynet-b       vsw0                        network@0    08:00:20:ab:9a:12
    mynet-b       vsw0                        network@0    08:00:20:ab:9a:12

DISK
    NAME              VOLUME
    mydisk-a          myvol-a@vds0
    mydisk-b          myvol-b@vds0

VDPCC
    NAME              SERVICE
    myvdpcc-a         vdpcs0@primary
    myvdpcc-b         vdpcs0@primary

VCONS
    NAME              SERVICE
    mygroup           vcc0
```

▼ 制約を XML 形式で一覧表示する

- 特定のドメイン (`ldg1` など) の制約を XML 形式で一覧表示します。

コード例 9-16　　ドメインの XML 形式の制約

```
primary$ ldm list-constraints -x ldg1
<?xml version="1.0"?>
<LDM_interface version="1.0">
  <data version="2.0">
    <ldom>
      <ldom_info>
        <ldom_name>ldg1</ldom_name>
      </ldom_info>
      <cpu>
        <number>8</number>
      </cpu>
      <memory>
        <size>1G</size>
      </memory>
      <network>
        <vnet_name>vnet0</vnet_name>
        <service_name>primary-vsw0</service_name>
        <mac_address>01:14:4f:fa:0f:55</mac_address>
      </network>
      <disk>
        <vdisk_name>vdisk0</vdisk_name>
        <service_name>primary-vds0</service_name>
        <vol_name>vol0</vol_name>
      </disk>
      <var>
        <name>boot-device</name>
        <value>/virtual-devices@100/channel-devices@200/disk@0:a</value>
      </var>
      <var>
        <name>nvramrc</name>
        <value>devalias vnet0 /virtual-devices@100/channel-devices@200/
network@0</value>
      </var>
      <var>
        <name>use-nvramrc?</name>
        <value>true</value>
      </var>
    </ldom>
  </data>
</LDM_interface>
```

▼ 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。

コード例 9-17　マシンが読み取り可能な形式のすべてのドメインの制約

```
primary$ ldm list-constraints -p
VERSION 1.0
DOMAIN|name=primary
MAC|mac-addr=00:03:ba:d8:b1:46
VCPU|count=4
MEMORY|size=805306368
IO
|dev=pci@780|alias=
|dev=pci@7c0|alias=
VDS|name=primary-vds0
|vol=disk-ldg2|opts=|dev=/ldoms/nv72-ldg2/disk
|vol=vol0|opts=|dev=/ldoms/nv72-ldg1/disk
VCC|name=primary-vcc0|port-range=5000-5100
VSW|name=primary-vsw0|mac-addr=|net-dev=e1000g0|dev=switch@0
DOMAIN|name=ldg1
VCPU|count=8
MEMORY|size=1073741824
VARIABLES
|boot-device=/virtual-devices@100/channel-devices@200/disk@0:a
|nvramrc=devalias vnet0 /virtual-devices@100/channel-devices@200/network@0
|use-nvramrc?=true
VNET|name=vnet0|dev=network@0|service=primary-vsw0|mac-addr=01:14:4f:fa:0f:55
VDISK|name=vdisk0|vol=vol0@primary-vds0
```

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 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。

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
```

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

Solaris の障害管理アーキテクチャー (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 ls -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 ls -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

コード例 9-18 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。

コード例 9-18　論理ドメイン構成の解析可能な長形式のリスト

```
primary$ ldm ls -l -p
VERSION 1.0
DOMAIN|name=primary|state=active|flags=normal,control,vio-service|cons=
SP|ncpu=4|mem=1073741824|util=0.6|uptime=64801|softstate=Solaris running
VCPU
|vid=0|pid=0|util=0.9|strand=100
|vid=1|pid=1|util=0.5|strand=100
|vid=2|pid=2|util=0.6|strand=100
|vid=3|pid=3|util=0.6|strand=100
MEMORY
|ra=0x8000000|pa=0x8000000|size=1073741824
IO
|dev=pci@780|alias=bus_a
|dev=pci@7c0|alias=bus_b
VDS|name=primary-vds0|nclients=2
|vol=disk-ldg1|opts=|dev=/opt/ldoms/testdisk.1
|vol=disk-ldg2|opts=|dev=/opt/ldoms/testdisk.2
VCC|name=primary-vcc0|nclients=2|port-range=5000-5100
VSW|name=primary-vsw0|nclients=2|mac-addr=00:14:4f:fb:42:5c|net-dev=
e1000g0|dev=switch@0|mode=prog,promisc
VCONS|type=SP
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
VARIABLES
|auto-boot?=true
|boot-device=/virtual-devices@100/channel-devices@200/disk@0
VNET|name=net|dev=network@0|service=primary-vsw0@primary|mac-addr=
00:14:4f:f9:8f:e6
VDISK|name=vdisk-1|vol=disk-ldg1@primary-vds0|dev=disk@0|server=primary
VCONS|group=group1|service=primary-vcc0@primary|port=5000
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
VARIABLES
```

コード例 9-18　論理ドメイン構成の解析可能な長形式のリスト (続き)

```
|auto-boot?=true
|boot-device=/virtual-devices@100/channel-devices@200/disk@0
VNET|name=net|dev=network@0|service=primary-vsw0@primary|mac-addr=
00:14:4f:f9:8f:e7
VDISK|name=vdisk-2|vol=disk-ldg2@primary-vds0|dev=disk@0|server=primary
VCONS|group=group2|service=primary-vcc0@primary|port=5000
```

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。

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、つまり、ドメイン化が有効になるときに発生する、Solaris OS を使用した場合の動作の変更について説明します。

注 – ドメイン化が有効かどうかに関する説明は、Sun UltraSPARC T1 ベースのプラットフォームにのみ関連するものです。それ以外のプラットフォームでは、ドメイン化は常に有効になっています。

ドメイン化を有効にした場合、Solaris OS の起動後に OpenBoot ファームウェアを使用できない

Logical Domains Manager によって作成された論理ドメイン構成がインスタンス化されると、ドメイン化は有効になります。ドメイン化が有効な場合には、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 SC に保存する必要があります。

▼ 現在の論理ドメイン構成を SC 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。この事項は、プラットフォームで電源管理がサポートされている場合にのみ適用されます。

Solaris OS のブレークの結果

ドメイン化が有効でない場合にブレークが実行されると、通常、Solaris OS は OpenBoot プロンプトに移行します。この節で説明する動作は、次の 2 つの状況で発生します。

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。

ドメイン化が有効な場合は、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)eboot, 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。

注 – 表 9-1 のドメイン化が有効かどうかに関する質問は、Sun UltraSPARC T1 プロセッサにのみ関連するものです。それ以外のプラットフォームでは、ドメイン化は常に有効になっています。

表 9-1 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	ドメイン化が有効か	他のドメインが構成されているか	動作
halt	無効	なし	**Sun UltraSPARC T1 プロセッサの場合:** ok プロンプトに移行します。
	有効	いいえ	**Sun UltraSPARC T1 プロセッサの場合:** システムは、リセットして OpenBoot ok プロンプトに進むか、または次のプロンプトに進みます。 `r)eboot, o)k prompt, or h)alt?` **Sun UltraSPARC T2 プロセッサの場合:** ホストの電源が切断され、SC で電源が投入されるまで切断されたままです。
	有効	はい	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	無効	なし	**Sun UltraSPARC T1 プロセッサの場合:** ホストの電源が切断され、再投入されます。
	有効	いいえ	**Sun UltraSPARC T1 プロセッサの場合:** ホストの電源が切断され、再投入されます。 **Sun UltraSPARC T2 プロセッサの場合:** ホストを再起動し、電源は切断されません。
	有効	はい	**Sun UltraSPARC T1 プロセッサの場合:** ホストの電源が切断され、再投入されます。 **Sun UltraSPARC T2 プロセッサの場合:** ホストを再起動し、電源は切断されません。

表 9-1　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 (続き)

コマンド	ドメイン化が有効か	他のドメインが構成されているか	動作
shutdown -i 5	無効	なし	Sun UltraSPARC T1 プロセッサの場合: ホストの電源が切断されます。
	有効	いいえ	ホストの電源が切断され、SC で電源が投入されるまで切断されたままです。
	有効	はい	ソフトリセットが行われて再起動します。

LDoms と ALOM CMT の使用

この節では、Advanced Lights Out Manager (ALOM) チップマルチスレッディング (CMT) 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。ALOM CMT ソフトウェアの使用については、『Advanced Lights Out Management (ALOM) CMT v1.3 ガイド』を参照してください。

注意 – ALOM CMT のドキュメントでは 1 つのドメインについて説明しているため、Logical Domains Manager では複数のドメインを導入していることに注意する必要があります。論理ドメインが再起動されると、ゲストドメインの I/O サービスは、制御ドメインが再起動されるまで使用できなくなる場合があります。これは、Logical Domains Manager 1.1 ソフトウェアでは制御ドメインがサービスドメインとして機能するためです。再起動処理の間は、ゲストドメインが動かなくなっているように見えます。制御ドメインが完全に再起動すると、ゲストドメインは通常の操作を再開します。サーバー全体の電源が切断される場合は、ゲストドメインの停止のみが必要になります。

既存の ALOM CMT コマンドでは、追加オプションが使用可能です。

```
bootmode [normal|reset_nvram|bootscript=strong|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 `factory-default` 出荷時構成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

- ALOM CMT ソフトウェアでこ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。

```
sc> 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、システムコントローラ (SC) 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 コマンドで指定した名前を使用して、ALOM CMT の bootmode コマンドでその構成を選択できます。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。

```
sc> bootmode config="ldm-config1"
```

ldm add-config コマンドの詳細は、ldm(1M) マニュアルページまたは『Logical Domains (LDoms) Manager 1.1 Man Page Guide』を参照してください。

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。

この節では、この監査機能を使用する場合に、有効化、検証、無効化、出力の表示、および監査ログの切り替えを行う方法について説明します。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 監査は、次の 2 つのいずれかの方法で有効にできます。監査を無効にする場合は、有効にしたときと同じ方法を使用してください。2 つの方法は次のとおりです。

- Solaris Security Toolkit の enable-bsm.fin 終了スクリプトを使用します。

  enable-bsm.fin スクリプトは、デフォルトでは ldm_control-secure.driver では使用されません。選択したドライバで終了スクリプトを有効にする必要があります。

- Solaris OS の bsmconv(1M) コマンドを使用します。

ここでは、両方の方法についての手順を示します。

▼ enable-bsm.fin 終了スクリプトを使用する

1. ldm_control-secure.driver を *my-ldm.driver* にコピーします。ここで *my-ldm.driver* は ldm_control-secure.driver のコピーの名前です。
2. ldm_control-config.driver を *my-ldm-config.driver* にコピーします。ここで *my-ldm-config.driver* は ldm_control-config.driver のコピーの名前です。
3. ldm_control-hardening.driver を *my-ldm-hardening.driver* にコピーします。ここで *my-ldm-hardening.driver* は ldm_control-hardening.driver のコピーの名前です。
4. *my-ldm.driver* を編集して、新しい構成と強化ドライバを、それぞれ *my-ldm-control.driver* と *my-ldm-hardening.driver* に変更します。
5. *my-ldm-hardening.driver* を編集して、ドライバの次の行の先頭にあるハッシュ記号 (#) を削除します。

```
enable-bsm.fin
```

6. *my-ldm.driver* を実行します。

```
# /opt/SUNWjass/bin/jass-execute -d my-ldm.driver
```

7. Solaris OS を再起動して、監査を有効にします。

▼ Solaris OS の bsmconv(1M) コマンドを使用する

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、Solaris 10 Reference Manual Collection またはマニュアルページを参照してください。

3. Solaris オペレーティングシステムを再起動して、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。

▼ 監査を無効にする

監査を有効にした方法に応じて、次の 2 つのいずれかの方法で監査を無効にすることができます。178 ページの「BSM 監査の有効化と使用」を参照してください。

1. 次のいずれかを実行します。
   - BSM 監査を有効にした Solaris Security Toolkit による強化の実行を取り消します。

```
# /opt/SUNWjass/bin/jass-execute -u
```

   - Solaris OS の `bsmunconv`(1M) コマンドを使用します。

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

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。
  - たとえば、Solaris OS コマンドの `auditreduce`(1M) と `praudit`(1M) 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
```

  - Solaris OS の `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

- Solaris OS の `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 10 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。

- トランスポート ― 外部プログラムと Logical Domains (LDoms) Manager の間で通信を開始する方法。
- プロトコル ― Logical Domains Manager との間で送受信される XML メッセージの形式。
- イベント ― Logical Domains Manager の警告通知。

Logical Domains Manager で使用する各種のスキーマについては、付録 A 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 と通信します。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_enable` SMF プロパティーを `false` に設定し、Logical Domains Manager を再起動します。

```
# svcadm disable ldmd
# svccfg -s ldom/ldmd setprop ldmd/xmpp_enabled=false
# svcadm refresh ldmd
# svcadm enable ldmd
```

XMPP

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 スキーマも記載されています(214 ページの「LDM_Event XML スキーマ」を参照)。

要求

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。

コード例 10-1　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>
    <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、コード例 10-1 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。

<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 (239 ページの「GenericProperty XML スキーマ」を参照)
- <Binding> タグ (240 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

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、コード例 10-2 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。

コード例 10-2 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>
         <!-- 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>
```

コード例 10-2　　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)

```
      <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。187 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

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、192 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。186 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。

コード例 10-3　　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。

コード例 10-4　　イベントの登録応答メッセージの例

```
<LDM_interface version="1.0">
  <cmd>
    <action>reg-domain-events</action>
    <data version="3.0"/>
      <response>
        <status>success</status>
      </response>
    </data>
    <response>
      <status>success</status>
    </response>
  </cmd>
  <response>
    <status>success</status>
  </response>
</LDM_interface>
```

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。

コード例 10-5 <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>
  </cmd>
</LDM_event>
```

イベントタイプ

登録できるイベントには、次の 3 つのタイプがあります。

- ドメインイベント
- リソースイベント
- ハードウェアイベント

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
`add-domain`	`remove-domain`
`bind-domain`	`unbind-domain`
`start-domain`	`stop-domain`
`domain-reset`	`panic-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。

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。

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。

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、『Logical Domains (LDoms) Manager 1.1 Man Page Guide』または ldm 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

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。209 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

コード例 10-6　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>
      </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 2 つのプロパティーは、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="hostid"> タグ。ドメインに割り当てるホスト ID を指定します。

CPU (cpu) リソース

コード例 10-7　　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。

コード例 10-8　　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

コード例 10-9 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

コード例 10-10 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

コード例 10-11　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>
    </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

コード例 10-12　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 リソースに次のプロパティーも設定できます。

- timeout ― 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 (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

コード例 10-13　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>
        <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
- 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

コード例 10-14　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="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。

- 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

コード例 10-15　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

コード例 10-16 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>
    </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

コード例 10-17 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、次のキープロパティー値を持つ <gprop:GenericProperty> タグのみです。

- name ― 記述する I/O デバイスの名前

SP 構成 (spconfig) リソース

コード例 10-18　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>
      </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

コード例 10-19　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、次のキープロパティー値を持つ <gprop:GenericProperty> タグのみです。

- service_name ― 記述す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

仮想データプレーンチャネルクライアント (vdpcc) リソース

コード例 10-20　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

コード例 10-21　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。

コード例 10-22 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 スキーマについて説明します。

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。

コード例 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>
          <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">
```

コード例 A-1　　LDM_interface XML スキーマ (続き)

```
      <xs:enumeration value="success"/>
      <xs:enumeration value="failure"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="responseType">
    <xs:sequence>
      <xs:element name="status" type="statusStringType"/>
      <xs:element name="resp_msg" type="xs:string"
        minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- LDM interface document -->
  <xs:element name="LDM_interface">
    <xs:complexType>
      <xs:sequence>

        <!-- START cmd -->
        <xs:element name="cmd" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="action" type="xs:string"
                minOccurs="0"/>

              <!-- START data -->
              <xs:element name="data" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:choice minOccurs="1" maxOccurs="unbounded">

                    <!--OVF Evelope Version 0.9 -->
                    <xs:element name="Envelope" type="ovf:Envelope_Type"/>
                    <!-- DATA response -->
                    <xs:element name="response" type="responseType"
                      minOccurs="0" maxOccurs="1"/>
                  </xs:choice>
                  <xs:attribute name="version" type="xs:string" use="required"/>
                </xs:complexType>
              </xs:element> <!-- END data -->

              <!-- CMD response -->
              <xs:element name="response" type="responseType"
                minOccurs="0" maxOccurs="1"/>

            </xs:sequence>
          </xs:complexType>
        </xs:element> <!-- END cmd -->
```

コード例 A-1　　LDM_interface XML スキーマ (続き)

```
        <!-- 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

コード例 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>

       <!-- 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>
```

コード例 A-2　　LDM_Event XML スキーマ (続き)

```
          </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

コード例 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"/>

  <!-- 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"/>
```

コード例 A-3　　ovf-envelope.xsd スキーマ (続き)

```
      <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) -->
    <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>
```

コード例 A-3　　ovf-envelope.xsd スキーマ (続き)

```
      <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

コード例 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

コード例 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"/>
  <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"/>
```

コード例 A-5　　ovf-core.xsd スキーマ (続き)

```
          <xs:any namespace="##targetNamespace" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
          <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Configuration parameters that can be passed to the virtual machine for
application-level configuration -->
  <xs:complexType name="PropertySection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Section_Type">
        <xs:sequence>
          <xs:element name="Property" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Description" type="ovf:Info_Type" minOccurs=
"0" maxOccurs="unbounded"/>
                <xs:any namespace="##targetNamespace" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
                <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
              </xs:sequence>
              <xs:attribute name="key" type="xs:string"/>
              <xs:attribute name="type" type="xs:string"/>
              <xs:attribute name="configurableByUser" type="xs:boolean" use=
"optional"/>
              <xs:attribute name="configurableAtRuntime" type="xs:boolean" use=
"optional"/>
              <xs:attribute name="defaultValue" type="xs:string" use=
"optional"/>
              <xs:anyAttribute namespace="##any"/>
            </xs:complexType>
          </xs:element>
          <xs:any namespace="##targetNamespace" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
          <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <!-- 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>
```

コード例 A-5　　ovf-core.xsd スキーマ (続き)

```
    </xs:complexContent>
  </xs:complexType>

  <!-- Provides descriptions for the logical networks used within the package.
These descriptions are typically used as an aid when the package is deployed. -->
  <xs:complexType name="NetworkSection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Section_Type">
        <xs:sequence>
          <xs:element name="Network" maxOccurs="unbounded">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="Description"
type="ovf:Info_Type" minOccurs="0" maxOccurs="unbounded"/>
                <xs:any namespace="##targetNamespace" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
                <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
              </xs:sequence>
              <xs:attribute name="name" type="xs:string" use="required"/>
              <xs:anyAttribute namespace="##any"/>
            </xs:complexType>
          </xs:element>
          <xs:any namespace="##targetNamespace" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
          <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Provides meta-information description of the virtual disks in the package
-->
  <xs:complexType name="DiskSection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Section_Type">
        <xs:sequence>
          <xs:element name="Disk" type="ovf:VirtualDiskDesc_Type" minOccurs="0"
maxOccurs="unbounded"/>
          <xs:any namespace="##targetNamespace" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
          <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##any"/>
```

コード例 A-5　　ovf-core.xsd スキーマ (続き)

```
      </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">
            <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>
```

コード例 A-5　　ovf-core.xsd スキーマ (続き)

```
  </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>
    </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">
```

コード例 A-5　　ovf-core.xsd スキーマ (続き)

```
            <xs:complexType>
              <!-- Id of entity in collection -->
              <xs:attribute name="id" type="xs:string"/>
              <!-- Startup order. Entities are started up starting with lower-
numbers first. Items with same order identifier may be started up concurrently
or in any order. The order is reversed for shutdown. -->
              <xs:attribute name="order" type="xs:int"/>
              <!-- Delay in seconds to wait for the power on to complete -->
              <xs:attribute name="startDelay" type="xs:int"/>
              <!-- Whether to resume power-on sequence, once the guest reports
ok. -->
              <xs:attribute name="waitingForGuest" type="xs:boolean"/>
              <!-- Delay in seconds to wait for the power on to complete -->
              <xs:attribute name="stopDelay" type="xs:int"/>
              <!-- Stop action to use. Valid values are: 'powerOn? (default),
'none?. -->
              <xs:attribute name="startAction" type="xs:string"/>
              <!-- Stop action to use. Valid values are: 'powerOff' (default),
'guestShutdown', 'suspend'. -->
              <xs:attribute name="stopAction" type="xs:string"/>
              <xs:anyAttribute namespace="##any"/>
            </xs:complexType>
          </xs:element>
          <xs:any namespace="##targetNamespace" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
          <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <!-- A comma-separated list of transports that the virtual machine
supports to provide feedback. -->
        <xs:anyAttribute namespace="##any"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- If this section is present, it indicates that the virtual machine needs
to be initially booted to install and configure the software. -->
  <xs:complexType name="InstallSection_Type">
    <xs:complexContent>
      <xs:extension base="ovf:Section_Type">
        <xs:sequence>
          <xs:any namespace="##targetNamespace" processContents="lax"
minOccurs="0" maxOccurs="unbounded"/>
          <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
```

コード例 A-5　　ovf-core.xsd スキーマ (続き)

```
        <!-- 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

コード例 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>
      </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"/>
```

コード例 A-6　　ovf-virtualhardware.xsc スキーマ (続き)

```
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

cim-rasd.xsd スキーマ

コード例 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">
                  <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 -->
```

コード例 A-7　　cim-rasd.xsd スキーマ (続き)

```
                    <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"/>
                  </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>
```

コード例 A-7　　cim-rasd.xsd スキーマ (続き)

```
        <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>
      </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>
```

コード例 A-7　　cim-rasd.xsd スキーマ (続き)

```
      <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>
      </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"/>
```

コード例 A-7　　cim-rasd.xsd スキーマ (続き)

```
      <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

コード例 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

コード例 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">
```

コード例 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="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>
```

コード例 A-9　　cim-common.xsd スキーマ (続き)

```
      <xs:extension base="xs:string">
        <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="cimBoolean">
    <xs:simpleContent>
      <xs:extension base="xs:boolean">
        <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="cimFloat">
    <xs:simpleContent>
      <xs:extension base="xs:float">
        <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="cimDouble">
    <xs:simpleContent>
      <xs:extension base="xs:double">
        <xs:anyAttribute namespace="##any" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="cimChar16">
    <xs:simpleContent>
        <xs:restriction base="cim:cimString">
          <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>
```

コード例 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"/>
```

コード例 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。

コード例 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。

コード例 A-11 Binding_Type XML スキーマ

```
<?xml version='1.0' encoding='utf-8'?>
<xs:schema
  targetNamespace="/var/opt/SUNWldom/Binding"
  xmlns:class="/var/opt/SUNWldom/Binding"
xmlns:rasd="/var/opt/SUNWldom/CIM_ResourceAllocationSettingData"
  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>
```

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 CMT Advanced Lights Out Manager Chip MultiThreading (Advanced Lights Out Manager チップマルチスレッディング)。システムコントローラ上で動作し、CMT サーバーを監視および制御できます。

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`(1M) 監査証跡ファイルの監査レコードのマージおよび選択

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`(1M) BSM の有効化

`bsmunconv`(1M) BSM の無効化

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)

CMT Chip MultiThreading (チップマルチスレッディング)

`config` システムコントローラに保存されている論理ドメイン構成の名前

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`(1M) Logical Domains Manager 用の動的再構成デーモン (Solaris 10 OS)

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

`e1000g` ネットワークインタフェースコントローラの Intel PRO/1000 ギガビットファミリ用のドライバ

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`(1M) 障害管理デーモン (Solaris 10 OS)

`fmthard`(1M) ハードディスクのラベルの生成

`format`(1M) 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

FTP File Transfer Protocol (ファイル転送プロトコル)

G

Gb Gigabit (ギガビット)

GLDv3 Generic LAN Driver version 3 (汎用 LAN ドライバ version 3)

H

HDD Hard Disk Drive (ハードディスクドライブ)

I

I/O ドメイン 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 PCIe 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(1M) Logical Domain Manager ユーティリティー

ldmd Logical Domains Manager デーモン

lofi ループバックファイル

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 を提供する

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、LDoms 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 の略で、Sun UltraSPARC T1 ベースのプラットフォーム用の暗号化装置

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 ー 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(1M)　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

metaset(1M)　ディスクセットの構成

mhd(7I)　多重ホストディスク制御操作

MIB　Management Information Base (管理情報ベース)

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

PA　Physical Address (物理アドレス)

PCI　Peripheral Component Interconnect バス

PCI-X　PCI 拡張バス

PCIe　PCI Express 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(1M)　PICL デーモン

PM　仮想 CPU の Power Management (電源管理)

praudit(1M)　監査証跡ファイルの内容の出力

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。

SC System Controller (システムコントローラ)。サービスプロセッサとも呼ばれます。

SCSI Small Computer System Interface

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)

SP Service Processor (サービスプロセッサ)。システムコントローラとも呼ばれます。

SSH Secure Shell

`ssh`(1) Secure Shell コマンド

`sshd`(1M) Secure Shell デーモン

SunVTS Sun Validation Test Suite

`svcadm`(1M) サービスインスタンスの操作

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)

USB Universal Serial Bus (ユニバーサルシリアルバス)

`uscsi`(7D) ユーザー SCSI コマンドインタフェース

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、またはファイルで構成される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(1M) 論理ドメインコンソール用の仮想ネットワーク端末サーバーデーモン (Solaris 10 OS)

volfs(7FS) ボリューム管理ファイルシステム

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`(1M)　ZFS ストレージプール

ZVOL　ZFS ボリュームエミュレーションドライバ

か

監査　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

き

強化　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

け

ゲストドメイン　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。

さ

サービスドメイン 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

最小化 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

し

承認 Solaris OS RBAC を使用して承認を設定すること

せ

制御ドメイン 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

制約 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 つ以上のリソースのこと。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。

て

適合性 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

は

ハイパーバイザ 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

ゆ

ユニキャスト　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

ろ

論理ドメイン　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、リソース、および識別情報を持つ個別の論理グループ