Oracle® Solaris の管理: Oracle Solaris コンテナ - リソース管理と Oracle Solaris ゾーン

Part No: 819–0385–22
2013 年 3 月

ORACLE®

130327@25097

目次

はじめに

このマニュアルは、Oracle Solaris オペレーティングシステムの管理の情報の大部分について説明する、数冊からなるマニュアルの一部です。このマニュアルでは、オペレーティングシステムがすでにインストールされており、使用する予定のネットワークソフトウェアが設定済みであることを前提としています。

注 – この Oracle Solaris のリリースでは、SPARC および x86 系列のプロセッサアーキテクチャーを使用するシステムをサポートしています。サポートされるシステムは、*http://www.oracle.com/webfolder/technetwork/hcl/index.html* の『Oracle Solaris OS: Hardware Compatibility List』に記載されています。このドキュメントでは、プラットフォームにより実装が異なる場合は、それを特記します。

このドキュメントの x86 に関連する用語については、次を参照してください。

- 「x86」は、64 ビットおよび 32 ビットの x86 互換製品系列を指します。
- 「x64」は、具体的には 64 ビット x86 互換 CPU を指します。
- 「32 ビット x86」は、x86 をベースとするシステムに関する 32 ビット特有の情報を指します。

Oracle Solaris コンテナについて

Oracle Solaris コンテナは、アプリケーション用の完全な実行時環境で、Oracle Solaris ゾーンとも呼ばれます。Oracle Solaris 10 リソースマネージャーと Oracle Solaris ゾーンソフトウェア区分技術は、どちらもこのコンテナの一部です。ゾーンは、プラットフォームのリソースにアプリケーションを仮想的に割り当てます。ゾーンを使用すると、Oracle Solaris オペレーティングシステムの単一のインスタンスを複数のゾーンで共有しているにもかかわらず、アプリケーションコンポーネントを互いに隔離できます。リソース管理機能では、作業負荷に与えるリソースの量を割り当てることができます。

ゾーンは、CPU などのリソースの消費量に制限を設けます。ゾーン内で実行されるアプリケーションの処理要件の変化に応じて、これらの制限を拡張することもできます。

Solaris 10 8/07: Linux アプリケーション用 Oracle Solaris コンテナについて

Linux アプリケーション用 Solaris コンテナは、Oracle の BrandZ 技術を使用して、Linux アプリケーションを Oracle Solaris 10 オペレーティングシステム上で実行します。Linux アプリケーションを変更することなく、非大域ゾーン機能によって提供される安全な環境で実行できます。これにより、Oracle Solaris システムを使用して Linux アプリケーションの開発、テスト、および配備を行うことができます。

この機能を使用するには、パート III「lx ブランドゾーン」を参照してください。

Oracle Solaris 10 11/06: Solaris Trusted Extensions システムでのゾーンの使用について

Trusted Extensions システムでゾーンを使用する方法については、『Oracle Solaris Trusted Extensions 管理の手順』の第 10 章「Trusted Extensions でのゾーンの管理 (手順)」を参照してください。

対象読者

このマニュアルは、Oracle Solaris 10 リリースが稼働しているシステムの管理者を対象としています。このマニュアルを活用するには、少なくとも 1、2 年程度の UNIX システムの管理経験が必要です。

Solaris システム管理マニュアルセットの構成

システム管理ガイドセットに含まれる各ガイドとその内容は、次のとおりです。

ドキュメントのタイトル	トピック
『Solaris のシステム管理 (基本編)』	ユーザーアカウントとグループ、サーバーとクライアントのサポート、システムのシャットダウンとブート、管理サービス、およびソフトウェアの管理 (パッケージとパッチ)
『Solaris のシステム管理 (上級編)』	端末とモデムの設定、システムリソースの管理 (ディスク割り当て、アカウンティング、および crontab ファイルの管理)、システムプロセスの管理、および Oracle Solaris ソフトウェアの障害追跡
『Solaris のシステム管理 (デバイスとファイルシステム)』	リムーバブルメディア、ディスクとデバイス、ファイルシステム、およびデータのバックアップと復元

ドキュメントのタイトル	トピック
『Solaris のシステム管理 (IP サービス) 』	TCP/IP ネットワークの管理、IPv4 および IPv6 アドレスの管理、DHCP、IP セキュリティー、IKE、IP フィルタ、モバイル IP、IP ネットワークのマルチパス化 (IPMP)、および IPQoS
『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』	DNS、NIS、および LDAP のネーミングとディレクトリサービス (NIS から LDAP への移行、および NIS+ から LDAP への移行を含む)
『Solaris のシステム管理 (ネーミングとディレクトリサービス : NIS+ 編)』	NIS+ のネーミングとディレクトリサービス
『Solaris のシステム管理 (ネットワークサービス) 』	Web キャッシュサーバー、時間関連サービス、ネットワークファイルシステム (NFS と Autofs)、メール、SLP、および PPP
『Solaris のシステム管理 (印刷)』	Oracle Solaris の印刷に関するトピックとタスク、印刷サービスやプリンタを設定して管理するためのサービス、ツール、プロトコル、およびテクノロジの使用方法
『Solaris のシステム管理 (セキュリティサービス)』	監査、デバイス管理、ファイルセキュリティー、BART、Kerberos サービス、PAM、Oracle Solaris 暗号化フレームワーク、特権、RBAC、SASL、および Oracle Solaris Secure Shell
『Oracle Solaris の管理: Oracle Solaris コンテナ - リソース管理と Oracle Solaris ゾーン』	リソース管理に関するトピック、プロジェクトとタスク、拡張アカウンティング、リソース制御、公平配分スケジューラ (FSS)、リソース上限デーモン (rcapd) を使った物理メモリー制御、リソースプール、およびゾーンソフトウェア区分技術を使った仮想化
『Oracle Solaris ZFS 管理ガイド』	ZFS ストレージプールおよびファイルシステムの作成と管理、スナップショット、クローン、バックアップ、アクセス制御リスト (ACL) を使用した ZFS ファイルの保護、ゾーンがインストールされた Oracle Solaris システム上での Oracle Solaris ZFS の使用、エミュレートされたボリューム、およびトラブルシューティングとデータ回復
『Oracle Solaris Trusted Extensions 管理の手順』	Trusted Extensions システム固有のシステム管理
『Oracle Solaris Trusted Extensions 構成ガイド』	Oracle Solaris 10 5/08 リリース以降での、Trusted Extensions の計画、有効化、および初期構成の方法

関連ドキュメント

『Solaris Containers: Resource Management and Solaris Zones Developer's Guide』では、システムリソースの区分化と管理を行うアプリケーションの作成方法、および使用する API について説明しています。また、プログラミング例、およびアプリケーションの作成時に考慮するべきプログラミングの問題についても説明しています。

関連するサードパーティーの Web サイト情報

このドキュメントにはオラクル社およびその関連会社が所有または管理しない Web サイトへのリンクが含まれている場合があります。

注 – オラクル社およびその関連会社は、それらの Web サイトのアクセシビリティに関しての評価や言及は行っておりません。このソフトウェアまたはハードウェア、そしてドキュメントは、第三者のコンテンツ、製品、サービスへのアクセス、あるいはそれらに関する情報を提供することがあります。オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスに関して一切の責任を負わず、いかなる保証もいたしません。オラクル社およびその関連会社は、第三者のコンテンツ、製品、サービスへのアクセスまたは使用によって損失、費用、あるいは損害が発生しても一切の責任を負いかねます。

Oracle サポートへのアクセス

Oracle のお客様は、My Oracle Support を通じて電子的なサポートを利用することができます。詳細は、`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` を参照してください。聴覚に障害をお持ちの場合は、`http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` を参照してください。

表記上の規則

次の表では、このマニュアルで使用される表記上の規則について説明します。

表 P–1　表記上の規則

字体	説明	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `machine_name% you have mail.`

表 P–1 表記上の規則 (続き)

字体	説明	例
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: 実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、`rm` *filename* と入力します。
AaBbCc123	書名、新しい単語、および強調する単語を示します。	『ユーザーズガイド』の第 6 章を参照してください。 キャッシュは、ローカルに格納されるコピーです。 ファイルを保存しないでください。 注: いくつかの強調された項目は、オンラインでは太字で表示されます。

コマンド例のシェルプロンプト

Oracle Solaris OS に含まれるシェルで使用する、UNIX のデフォルトのシステムプロンプトとスーパーユーザープロンプトを次に示します。コマンド例に示されるデフォルトのシステムプロンプトは、Oracle Solaris のリリースによって異なります。

表 P–2 シェルプロンプト

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

パ ー ト I

リソース管理

このパートでは、Solaris 10 リソース管理について紹介します。Solaris 10 リソース管理を使用すると、利用可能なシステムリソースをアプリケーションでどのように使用するかを制御できます。

◆ ◆ ◆ 第 1 章

1 Solaris 10 リソース管理の紹介

リソース管理機能は、Solaris コンテナ環境のコンポーネントです。リソース管理を利用すると、アプリケーションが利用可能なシステムリソースをどのように使用するかを制御できます。次のような制御が可能になります。

- プロセッサ時間などのコンピュータリソースを割り当てます
- 割り当てたリソースの使用状況を監視し、必要に応じて調整します
- 解析、課金、および容量計画のために拡張アカウンティング情報を生成します

この章の内容は次のとおりです。

リソース管理の概要

今日のコンピューティング環境では、システム上で実行されるアプリケーションによって生成されるさまざまな作業負荷に柔軟に対応できる能力が求められます。「作業負荷」とは、1 つのアプリケーションまたは一連のアプリケーションが持つすべてのプロセスの集まりです。リソース管理機能を使用しない場合、Solaris オペレーティングシステムは、新しいアプリケーションの要求に動的に適応することによって作業負荷の要求に対応しています。このデフォルトの動作は、通常、システムのすべてのアクティビティーに対してリソースへのアクセスを同等に与えることを意味します。Solaris のリソース管理機能を使用すると、作業負荷を個別に扱うことができるようになります。次のような制御が可能になります。

- 特定のリソースへのアクセスを制限します
- 優先順位に基づいて作業負荷にリソースを提供します
- 作業負荷を互いに分離します

作業負荷が相互に影響し合うことによる性能の低下を最小限に抑える機能と、リソースの使用状況や使用効率を監視する機能を総称して「リソース管理機能」とい

います。リソース管理機能は、いくつかのアルゴリズムの集合として実装されます。これらのアルゴリズムは、アプリケーションがその実行過程で発行する一連のリソース要求を処理します。

リソース管理機能を使用すると、さまざまな作業負荷に対してオペレーティングシステムのデフォルトの動作を変更できます。この場合の「動作」とは、主に、アプリケーションが 1 つ以上のリソース要求をシステムに発行したときに、オペレーティングシステムのアルゴリズムが行う一連の決定処理のことです。リソース管理機能は、次の目的で使用できます。

- あるアプリケーションに対して、リソースの割り当てを拒否したり、ほかのアプリケーションに許可されているよりも大きい割り当てを与えたりします
- 特定の割り当てを個別にではなく一括して行います

リソース管理機能を使用できるようにシステムを構成すると、いくつかの目的が達成できます。次のような制御が可能になります。

- アプリケーションが際限なくリソースを浪費するのを防止します
- 外部イベントに基づいてアプリケーションの優先順位を変更します
- 一連のアプリケーションにリソースの使用を保証する一方で、システムの使用効率を最大限に高めます

リソース管理機能をシステム構成に組み込むときは、次のような作業が事前に必要です。

- システム上で競合する作業負荷の特定
- 競合しない作業負荷と、主要な作業負荷に影響を与えるような性能要件を伴った作業負荷との区別

競合しない作業負荷と競合する作業負荷を特定したら、システムの能力が許す範囲内で、業務サービスへの影響を最小限に抑えたリソース構成を構築できます。

効率のよいリソース管理を実現するために、Solaris システムには制御メカニズム、通知メカニズム、および監視メカニズムが用意されています。これらの機能の多くは、proc(4) ファイルシステム、プロセッサセット、スケジューリングクラスなどの既存メカニズムの拡張機能として提供されます。その他の機能はリソース管理に固有です。これらの機能については、以降の章で説明します。

リソースの分類

リソースは、アプリケーションの動作を変更する目的で操作されるコンピューティングシステムのあらゆる側面を意味します。つまり、リソースは、アプリケーションが暗黙的または明示的に要求する機能です。この機能が拒否または制限された場合は、堅固に作成されているアプリケーションの実行速度が低下します。

リソースの分類は、リソースの識別とは対照的に、多くの基準に基づいて行うことができます。たとえば、リソースは、暗黙的な要求か明示的な要求か、時間 (CPU 時間など) に基づいた要求か時間に無関係な要求 (割り当てられた CPU 配分など) か、などを基準に行うことができます。

通常、スケジューラに基づいたリソース管理は、アプリケーションが暗黙的に要求するリソースに適用されます。たとえば、実行を継続するときは、アプリケーションは暗黙的に追加の CPU 時間を要求します。ネットワークソケットにデータを書き込むときは、アプリケーションは暗黙的に帯域幅を要求します。暗黙的に要求されるリソースの総使用量に対して制約を設けることができます。

帯域幅または CPU サービスのレベルを明示的に折衝できるように、インタフェースを追加することもできます。追加スレッドの要求のように明示的に要求されるリソースは、制約によって管理できます。

リソース管理の制御メカニズム

Solaris オペレーティングシステムでは、制約、スケジューリング、区分の 3 種類の制御メカニズムを使用できます。

制約メカニズム

制約を使用すると、管理者やアプリケーション開発者は、作業負荷が使用する特定のリソースの消費にいくつかの制限を設定できます。制限を設定すると、リソースの消費シナリオを簡単にモデル化できます。また、制限を設定することにより、無秩序にリソースを要求してシステムの性能や可用性に悪影響を及ぼす可能性がある悪質なアプリケーションを制御できます。

制約は、アプリケーションに制限を課します。アプリケーションとシステムの関係は、アプリケーションが動作できないところまで悪化してしまう可能性があります。そのような事態を回避する方法の 1 つは、リソースに関する動作が不明なアプリケーションに対する制約を徐々に強めていくことです。第 6 章「リソース制御 (概要)」で説明するリソース制御機能が、制約メカニズムを提供します。新たに作成するアプリケーションであれば、リソースの制約をアプリケーションが認識するようにすることもできます。ただし、すべての開発者がこの機能を使用するとは限りません。

スケジューリングメカニズム

スケジューリングとは、一定の間隔で割り当てを決定することです。この決定は、予測可能なアルゴリズムに基づいて行われます。現在割り当てられているリソースを必要としないアプリケーションは、ほかのアプリケーションが使用できるように、そのリソースを解放します。スケジューリングに基づいてリソース管理を行うと、リソースに余裕がある構成の場合は使用効率を最大限にできると同時

に、リソースが限界まで、あるいは過剰に使用されている場合には、割り当てを制御できます。スケジューリングのアルゴリズムにより、「制御」という用語の意味が決まります。場合によっては、スケジューリングアルゴリズムは、すべてのアプリケーションがリソースにある程度アクセスできることを保証します。第 8 章「公平配分スケジューラ (概要)」で説明する公平配分スケジューラ (FSS) は、アプリケーションが制御された方法でCPU リソースにアクセスするように管理します。

区分メカニズム

区分は、作業負荷をシステム上で使用可能なリソースの一部に結合 (バインド) するために使用されます。リソースと結合することにより、作業負荷は常に一定量のリソースを使用できることが保証されます。第 12 章「リソースプール (概要)」で説明するリソースプール機能を使用すれば、作業負荷をマシンの特定の部分に制限できます。

区分を使用する構成では、システム全体が過剰使用されるのを防ぐことができます。ただし、この方法では、高い使用効率の達成は難しくなります。プロセッサなど、予約済みのリソースに結合されている作業負荷がアイドル状態になっている場合でも、別の作業負荷がそのリソースを使用することはできないためです。

リソース管理構成

リソース管理構成の一部をネットワークのネームサービスに置くことができます。この機能により、管理者は、リソース管理制約をマシンごとに排他的に適用するのではなく、複数のマシンに対して一括して適用できます。関連する作業は共通識別子を共有でき、その作業の使用状況はアカウンティングデータに基づいて表形式で表すことができます。

リソース管理構成と作業負荷識別子の詳細については、第 2 章「プロジェクトとタスク (概要)」に記載されています。これらの識別子をアプリケーションのリソース使用状況と結び付ける拡張アカウンティング機能は、第 4 章「拡張アカウンティング (概要)」に記載されています。

Solaris ゾーンとの相互動作

リソース管理機能を Solaris ゾーンと組み合わせて使用すると、アプリケーション環境をさらに細かく調整できます。リソース管理機能とゾーンの相互動作については、このガイドの該当セクションで説明します。

リソース管理機能を使用する場合

リソース管理機能を使用して、アプリケーションが必要な応答時間を確保できるようにします。

また、リソース管理機能により、リソースの使用効率を向上させることができます。使用状況を分類して優先付けすることにより、オフピーク時に余ったリソースを効率よく使用でき、処理能力を追加する必要がなくなります。また、負荷の変動が原因でリソースを無駄にすることもなくなります。

サーバーの統合

リソース管理機能は、多くのアプリケーションを 1 台のサーバー上で統合できる環境で使用するともっとも高い効果を発揮します。

多数のマシンの管理は複雑でコストがかかるため、より大規模で拡張性の高いサーバーにアプリケーションを統合することが望まれます。個々の作業負荷を別々のシステムで実行して、そのシステムのリソースへの完全なアクセスを与える代わりに、リソース管理ソフトウェアを使用すれば、システム内の作業負荷を分離できます。リソース管理機能を使用すると、1 つの Solaris システムで複数の異なるアプリケーションを実行して制御することにより、システムの総保有コスト (TCO) を低減することができます。

インターネットサービスやアプリケーションサービスを提供する場合は、リソース管理を使用すると、次のことが可能になります。

- 1 台のマシンに複数の Web サーバーを配置します。各 Web サイトのリソース消費を制御し、各サイトをほかのサイトで起こる可能性のある過剰使用から保護します。
- 欠陥のある CGI (Common Gateway Interface) スクリプトが CPU リソースを浪費するのを防止します。
- 不正な動作をするアプリケーションによって引き起こされる、仮想メモリーのリークを防止します。
- 顧客のアプリケーションが、同じサイトで実行されている別の顧客のアプリケーションの影響を受けないようにします。
- 同一マシン上で異なるレベルまたはクラスのサービスを提供します。
- 課金目的でアカウンティング情報を取得します。

大規模で多様なユーザーが利用するシステムをサポートする場合

リソース管理機能は、特に、教育機関のように大規模で多様なユーザーが利用するシステムでその効果を発揮します。さまざまな作業負荷が混在している場合は、特定のプロジェクトに高い優先順位を与えるようにソフトウェアを構成できます。

たとえば、大きな証券会社のトレーダは、データベースのクエリーや計算を実行するために、一時的に高速なアクセスが必要になる場合があります。一方、ほかのユーザーの作業負荷は、一定しています。トレーダのプロジェクトに、作業量に応じてより高い処理能力を割り当てれば、トレーダは必要とする応答性を確保できます。

また、リソース管理機能は、シン (thin) クライアントシステムをサポートするのにも適しています。これらのプラットフォームは、フレームバッファーとスマートカードのような入力デバイスを持つステートレスなコンソールを備えています。実際の処理は共有サーバー上で行われるため、タイムシェアリング型の環境とみなすことができます。リソース管理機能を使ってサーバー上のユーザーを分離してください。こうすることで、過剰な負荷を引き起こしたユーザーがハードウェアリソースを占有し、システムを使用するほかのユーザーに重大な影響を与えることがなくなります。

リソース管理の設定 (タスクマップ)

次のタスクマップに、システム上でリソース管理機能を設定する際に必要となるタスクの概要を示します。

タスク	説明	参照先
システム上の作業負荷を特定し、各作業負荷をプロジェクト別に分類します。	`/etc/project` ファイル、NIS マップ、または LDAP ディレクトリサービス内にプロジェクトエントリを作成します。	46 ページの「`project` データベース」
システム上の作業負荷に優先順位を付けます。	どのアプリケーションが重要かを判定します。重要な作業負荷にはリソースへの優先的なアクセスが必要になる場合があります。	サービスの目的を考慮してください。

タスク	説明	参照先
システム上で実際のアクティビティーを監視します。	パフォーマンスツールを使用して、システムで実行されている作業負荷の現在のリソース消費量を表示します。その上で、特定のリソースへのアクセスを制限する必要があるかどうか、あるいは特定の作業負荷をほかの作業負荷から分離する必要があるかどうかを判定できます。	206 ページの「システム単位の監視」および cpustat(1M)、iostat(1M)、mpstat(1M)、prstat(1M)、sar(1)、および vmstat(1M) のマニュアルページ
システムで実行されている作業負荷を一時的に変更します。	変更可能な設定値を決めるには、Solaris システムで使用できるリソース制御を参照します。タスクまたはプロセスが実行している間は、コマンド行から値を更新できます。	87 ページの「使用可能なリソース制御」、93 ページの「リソース制御値に対応付けられた大域アクションと局所アクション」、98 ページの「動作中のシステム上のリソース制御値を一時的に更新する」、および rctladm(1M) と prctl(1) のマニュアルページ
project データベースまたはネームサービスプロジェクトデータベース内のプロジェクトエントリごとにリソース制御とプロジェクト属性を設定します。	/etc/project ファイルまたはネームサービスプロジェクトデータベース内の各プロジェクトエントリには、リソース制御または属性を 1 つ以上含めることができます。これらのリソース制御は、そのプロジェクトに属するタスクとプロセスを制約します。リソース制御で指定する各しきい値に対しては、その値に達したときに行われるアクションを 1 つ以上対応付けることができます。 リソース制御は、コマンド行インタフェースを使って設定できます。一部の構成パラメータは、Solaris 管理コンソールでも設定できます。	46 ページの「project データベース」、48 ページの「ローカルの /etc/project ファイルの形式」、87 ページの「使用可能なリソース制御」、93 ページの「リソース制御値に対応付けられた大域アクションと局所アクション」、および第 8 章「公平配分スケジューラ (概要)」
プロジェクト内のプロセスの集合が消費する物理メモリーの容量に上限を設けます。	リソース上限デーモンは、/etc/project ファイルでプロジェクトの rcap.max-rss 属性に指定されたとおり、物理メモリーのリソース上限を制限します。	46 ページの「project データベース」および第 10 章「リソース上限デーモンによる物理メモリーの制御 (概要)」
リソースプール構成を作成します。	リソースプールは、プロセッサなどのシステムリソースを区分する手段を提供し、リブート時にもそのパーティションを保持します。/etc/project ファイルの各エントリに project.pool 属性を 1 つ追加できます。	46 ページの「project データベース」および第 12 章「リソースプール (概要)」

タスク	説明	参照先
公平配分スケジューラ (FSS) をデフォルトのシステムスケジューラとして設定します。	単一の CPU システムまたはプロセッサセット内のすべてのユーザープロセスが同じスケジューリングクラスに属するようにします。	125 ページの「FSS の構成」および `dispadmin(1M)` のマニュアルページ
拡張アカウンティング機能を起動し、タスクまたはプロセスに基づきリソース消費を監視して記録します。	拡張アカウンティングデータを使って現在のリソース制御を評価し、将来の作業負荷のための容量要件を計画します。システム全体の総使用状況を追跡できます。複数のシステムに渡って相互に関連しあう作業負荷について完全な使用統計を取得するために、プロジェクト名は複数のマシンで共有できます。	76 ページの「プロセス、タスク、およびフローの拡張アカウンティングを起動する方法」および `acctadm(1M)` のマニュアルページ
(オプション) 構成をさらに調整する必要がある場合は、引き続きコマンド行から値を変更できます。値は、タスクまたはプロセスの実行中でも変更できます。	既存のタスクに対しては、プロジェクトを再起動しなくても、変更を一時的に適用できます。満足のいく性能が得られるまで値を調整します。次に、`/etc/project` ファイルまたはネームサービスのプロジェクトデータベースで現在の値を更新します。	98 ページの「動作中のシステム上のリソース制御値を一時的に更新する」および `rctladm(1M)` と `prctl(1)` のマニュアルページ
(オプション) 拡張アカウンティングデータを取得します。	アクティブなプロセスおよびタスクの拡張アカウンティングレコードを書き込みます。作成されるファイルは、計画、チャージバック、および課金のために使用できます。`libexacct` への Perl (Practical Extraction and Report Language) インタフェースを使用して、報告および抽出用のカスタムスクリプトを作成することもできます。	`wracct(1M)` のマニュアルページおよび 71 ページの「`libexacct` に対する Perl インタフェース」

◆ ◆ ◆ 第 2 章

プロジェクトとタスク (概要)

この章では、Solaris のリソース管理機能のうち、プロジェクトおよびタスク機能について説明します。プロジェクトとタスクは、作業負荷にラベル付けを行い、ほかの作業負荷と区別するために使用されます。

この章の内容は次のとおりです。

プロジェクトとタスクの機能の使用方法については、第 3 章「プロジェクトとタスクの管理」を参照してください。

Solaris 10 におけるプロジェクトデータベースとリソース制御コマンドの新機能

Solaris 10 の具体的な拡張内容は次のとおりです。

- リソース制御の値およびコマンドで、倍率値と単位修飾子をサポート
- プロジェクト属性のフィールドの検証と操作性が向上
- `prctl` コマンドおよび `projects` コマンドの出力形式を改訂し、新しいオプションを追加
- `useradd` コマンドを使ってユーザーのデフォルトプロジェクトを設定する機能と、`usermod` コマンドと `passmgmt` コマンドを使って情報を変更する機能

この章と第 6 章「リソース制御 (概要)」に含まれている情報に加え、次のマニュアルページも参照してください。

- `passmgmt`(1M)

- projadd(1M)
- projmod(1M)
- useradd(1M)
- usermod(1M)
- resource_controls(5)

Solaris 10 5/08 の拡張機能では、projmod コマンドに -A オプションが追加されました。52 ページの「プロジェクトとタスクで使用するコマンド」を参照してください。

Solaris 10 の新機能の全一覧および Solaris リリースについての説明は、『Oracle Solaris 10 8/11 の新機能』を参照してください。

プロジェクトとタスクの機能

作業負荷の応答性を最適化するには、まず解析対象のシステム上で実行中の作業負荷を特定できなければなりません。この情報は、プロセス指向の手法とユーザー指向の手法のどちらか一方だけを使用して取得できるものではありません。Solaris システムでは、タスク負荷を区別して特定するための 2 つの追加機能を利用できます。プロジェクトとタスクです。「プロジェクト」は、関連した作業に対してネットワーク全体で適用される管理識別子を与えます。「タスク」は、プロセスのグループを、作業負荷のコンポーネントを表す管理しやすいエンティティーにまとめます。

project ネームサービスデータベースで指定された制御は、プロセス、タスク、およびプロジェクトに対して設定されます。プロセスの制御とタスクの制御は fork および settaskid システムコールを通して継承されるので、これらの制御は同じプロジェクト内に作成されるすべてのプロセスとタスクに継承されます。これらのシステムコールについては、fork(2) および settaskid(2) のマニュアルページを参照してください。

実行中のプロセスは、そのプロセスのプロジェクトメンバーシップまたはタスクメンバーシップに基づいて、Solaris の標準コマンドを使って操作できます。拡張アカウンティング機能は、プロセスとタスクの両方の使用状況について報告を作成し、各レコードに管理用プロジェクト識別子のタグを付けることができます。この処理により、オフラインで行う作業負荷解析作業をオンラインでの監視作業と関連付けることができます。プロジェクト識別子は、project ネームサービスデータベースを介して複数のマシンで共有できます。したがって、最終的には、複数のマシン上で実行される (つまり複数のマシンにわたる) 関連した作業負荷のリソース消費をすべてのマシンについて解析できます。

プロジェクト識別子

プロジェクト識別子は、関連する作業を特定するために使用される管理識別子です。プロジェクト識別子は、ユーザー識別子やグループ識別子と同様な、作業負荷に付けられているタグと考えることができます。ユーザーまたはグループは 1 つ以上のプロジェクトに所属できます。プロジェクトは、ユーザー (またはユーザーグループ) が参加することを許可されている作業負荷を表すために使用します。このメンバーシップは、たとえば、使用状況や初期リソース割り当てに基づく課金の根拠となります。ユーザーにはデフォルトのプロジェクトを割り当てる必要がありますが、ユーザーが起動するプロセスは、ユーザーが属しているプロジェクトであれば、どのプロジェクトにでも関連付けることができます。

ユーザーのデフォルトプロジェクトの判定

システムにログインするには、そのユーザーにデフォルトのプロジェクトが割り当てられている必要があります。ユーザーは、そのプロジェクトで指定されたユーザーリストやグループリストに含まれていない場合でも、自動的にそのデフォルトプロジェクトのメンバーになります。

システム上の各プロセスはプロジェクトのメンバーシップを所有しているため、ログインやその他の初期処理にデフォルトのプロジェクトを割り当てるアルゴリズムが必要です。アルゴリズムについては、`getprojent`(3C) のマニュアルページを参照してください。システムは、手順に従ってデフォルトプロジェクトを判定します。デフォルトプロジェクトが見つからない場合、ユーザーのログインまたはプロセスの開始要求は拒否されます。

システムは、次の手順でユーザーのデフォルトプロジェクトを判定します。

1. ユーザーが、`/etc/user_attr` 拡張ユーザー属性データベースで定義されている `project` 属性のエントリを持っている場合は、その `project` 属性の値がデフォルトプロジェクトになります。`user_attr(4)` のマニュアルページを参照してください。
2. `user.`*user-id* という名前のプロジェクトが `project` データベースにある場合は、そのプロジェクトがデフォルトプロジェクトになります。詳細は、`project(4)` のマニュアルページを参照してください。
3. `group.` *group-name* というプロジェクトが `project` データベースにあり、*group-name* がユーザーのデフォルトのグループ名 (`passwd` ファイルで指定されている) である場合は、そのプロジェクトがデフォルトプロジェクトになります。`passwd` ファイルについては、`passwd(4)` のマニュアルページを参照してください。
4. `project` データベースに `default` という特別なプロジェクトがある場合は、そのプロジェクトがデフォルトプロジェクトになります。

このロジックは getdefaultproj() ライブラリ関数が提供します。詳細は、getprojent(3PROJECT) のマニュアルページを参照してください。

useradd 、usermod、および passmgmt コマンドによるユーザー属性の設定

次のコマンドに -K オプションと *key=value* ペアを付けて使用すると、ローカルファイル内のユーザー属性を設定できます。

passmgmt	ユーザー情報を変更する
useradd	ユーザーのデフォルトプロジェクトを設定する
usermod	ユーザー情報を変更する

ローカルファイルには、次のようなものがあります。

- /etc/group
- /etc/passwd
- /etc/project
- /etc/shadow
- /etc/user_attr

NIS などのネットワークネームサービスを使ってローカルファイルに追加エントリを補足する場合、これらのコマンドでは、ネットワークネームサービスで指定された情報を変更することはできません。ただし、これらのコマンドを使用すると、次の内容を外部の「ネームサービスデータベース」と照合して検証できます。

- ユーザー名 (または役割) の一意性
- ユーザー ID の一意性
- 指定されたグループ名の存在

詳細は、passmgmt(1M)、useradd(1M)、usermod(1M)、および user_attr(4) のマニュアルページを参照してください。

project データベース

プロジェクトのデータは、ローカルファイル、ネットワーク情報サービス (NIS) のプロジェクトマップ、または LDAP (Lightweight Directory Access Protocol) ディレクトリサービスに保存できます。/etc/project ファイルまたはネームサービスは、ログイン時、および PAM (プラグイン可能認証モジュール) によるアカウント管理に対する要求があったときに使用され、ユーザーをデフォルトのプロジェクトに結合します。

注 – プロジェクトデータベース内のエントリに対する更新は、/etc/project ファイルまたはネットワークネームサービスのデータベース表現のどちらに対するものであっても、現在アクティブなプロジェクトには適用されません。更新内容は、login コマンドまたは newtask コマンドが使用されたときに、プロジェクトに新たに参加するタスクに適用されます。詳細は、login(1) および newtask(1) のマニュアルページを参照してください。

PAM サブシステム

アカウント情報の変更や設定を行う操作には、システムへのログイン、rcp または rsh コマンドの呼び出し、ftp の使用、su の使用などがあります。アカウント情報の変更や設定が必要な場合は、設定可能なモジュール群を使用して、認証、アカウント管理、資格管理、セッション管理などを行います。

プロジェクト用のアカウント管理 PAM モジュールについては、pam_projects(5) のマニュアルページを参照してください。PAM の概要については、『Solaris のシステム管理 (セキュリティサービス)』の第 17 章「PAM の使用」を参照してください。

ネームサービス構成

リソース管理は、ネームサービス project データベースをサポートします。project データベースが格納されている場所は、/etc/nsswitch.conf ファイルで定義されます。デフォルトでは files が最初に指定されていますが、ソースは任意の順序で指定できます。

```
project: files [nis] [ldap]
```

プロジェクト情報に対して複数のソースが指定されている場合、nsswitch.conf ファイルによりルーチンは最初に指定されているソースで情報を検索し、次にその後に続くソースを検索します。

/etc/nsswitch.conf ファイルの詳細については、『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』の第 2 章「ネームサービススイッチ (概要)」および nsswitch.conf(4) のマニュアルページを参照してください。

ローカルの /etc/project ファイルの形式

nsswitch.conf ファイルで project データベースのソースとして files を選択した場合、ログインプロセスはプロジェクト情報を /etc/project ファイルで検索します。詳細は、projects(1) および project(4) のマニュアルページを参照してください。

project ファイルには、システムによって認識されるプロジェクトごとにエントリが 1 行ずつ、次の形式で記述されています。

```
projname:projid:comment:user-list:group-list:attributes
```

フィールドの定義は次のとおりです。

projname プロジェクト名。英数字、下線 (_)、ハイフン (-)、およびピリオド (.) から成る文字列を指定します。ピリオドは、オペレーティングシステムに対して特殊な意味を持つプロジェクト用に予約されており、ユーザーのデフォルトプロジェクトの名前にだけ使用できます。*projname* にコロン (:) や改行文字を使用することはできません。

projid システムでプロジェクトに割り当てられる一意の数値 ID (PROJID)。*projid* フィールドの最大値は UID_MAX (2147483647) です。

comment プロジェクトの説明。

user-list プロジェクトへの参加を許可されたユーザーをコンマで区切ったリスト。

このフィールドではワイルドカードを使用できます。アスタリスク (*) を指定した場合、すべてのユーザーにプロジェクトへの参加を許可します。感嘆符に続けてアスタリスクを指定した場合 (!*)、すべてのユーザーのプロジェクトへの参加を拒否します。感嘆符 (!) に続けてユーザー名を指定した場合、指定されたユーザーのプロジェクトへの参加を拒否します。

group-list プロジェクトへの参加を許可されたユーザーのグループをコンマで区切ったリスト。

このフィールドではワイルドカードを使用できます。アスタリスク (*) を指定した場合、すべてのグループにプロジェクトへの参加を許可します。感嘆符に続けてアスタリスクを指定した場合 (!*)、すべてのグループのプロジェクトへの参加を拒否します。感嘆符 (!) に続けてグループ名を指定した場合、指定されたグループのプロジェクトへの参加を拒否します。

attributes リソース制御など、名前と値の対をセミコロンで区切ったリスト (第 6 章「リソース制御 (概要)」を参照)。*name* は、オブジェクトに関連する属性を指定する任意の文字列です。また *value* はその属性のオプション値です。

```
name[=value]
```

名前と値の組で、名前に使用できるのは、文字、数字、下線、ピリオドに制限されます。リソース制御 (rctl) のカテゴリとサブカテゴリの区切り文字にはピリオドを使用します。属性名の最初の文字は英字にする必要があります。名前については大文字と小文字は区別されます。

値を、コンマと括弧を使用して構造化することにより、優先順位を設定できます。

セミコロンは、名前と値の組を区切るために使用されます。よって、値の定義には使用できません。コロンは、プロジェクトフィールドを区切るために使用されます。よって、値の定義には使用できません。

注 – このファイルを読み取るルーチンは、無効なエントリを検出すると停止します。このため、無効なエントリの後に指定されているプロジェクトの割り当てはどれも実行されません。

次に、デフォルトの `/etc/project` ファイルの例を示します。

```
system:0:System:::
user.root:1:Super-User:::
noproject:2:No Project:::
default:3::::
group.staff:10::::
```

次に、デフォルトの `/etc/project` ファイルの最後にプロジェクトエントリを追加した例を示します。

```
system:0:System:::
user.root:1:Super-User:::
noproject:2:No Project:::
default:3::::
group.staff:10::::
user.ml:2424:Lyle Personal:::
booksite:4113:Book Auction Project:ml,mp,jtd,kjh::
```

`/etc/project` ファイルにリソース制御と属性を追加することもできます。

- プロジェクトのリソース制御を追加する方法については、102 ページの「リソース制御の設定」を参照してください。

- リソース上限デーモン (rcapd(1M) を参照) を使用して物理メモリーリソースの上限をプロジェクトに対して定義する方法については、131 ページの「プロジェクトの物理メモリーの使用率を制限する属性」を参照してください。
- プロジェクトのエントリに project.pool 属性を追加する方法については、196 ページの「構成の作成」を参照してください。

NIS のプロジェクト構成

NIS を使用している場合は、NIS プロジェクトマップ内でプロジェクトを検索するように、/etc/nsswitch.conf ファイルで指定できます。

```
project: nis files
```

NIS マップの project.byname と project.bynumber はどちらも、次に示すように /etc/project ファイルと同じ形式です。

```
projname:projid:comment:user-list:group-list:attributes
```

詳細については、『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』の第 4 章「ネットワーク情報サービス (NIS) (概要)」を参照してください。

LDAP のプロジェクト構成

LDAP を使用している場合は、LDAP project データベースでプロジェクトを検索するように、/etc/nsswitch.conf ファイルで指定できます。

```
project: ldap files
```

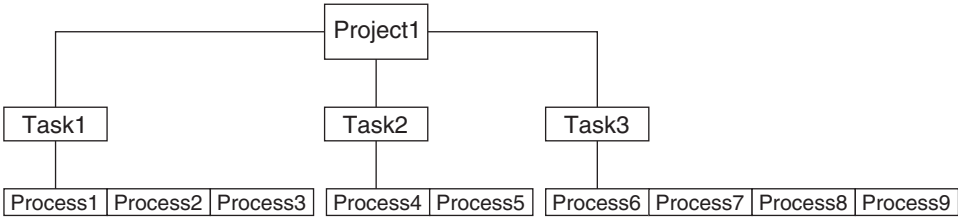
LDAP の詳細については、『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』の第 8 章「LDAP ネームサービスの紹介 (概要/リファレンス)」を参照してください。LDAP データベースにおけるプロジェクトエントリのスキーマの詳細については、『Solaris のシステム管理 (ネーミングとディレクトリサービス : DNS、NIS、LDAP 編)』の「Solaris スキーマ」を参照してください。

タスク識別子

プロジェクトへのログインが成功するたびに、ログインプロセスを含む新しい「タスク」が作成されます。タスクは、時間をかけて行われる一連の作業を表すプロセスです。また、タスクは「作業負荷のコンポーネント」と考えることもできます。各タスクには、自動的にタスク ID が割り当てられます。

各プロセスは、1 つのタスクのメンバーであり、各タスクは 1 つのプロジェクトに関連付けられています。

図 2–1 プロジェクトとタスクのツリー



シグナル送信のようなプロセスグループ上のすべての操作も、タスクでサポートされています。タスクを「プロセッサセット」に結合して、スケジューリングの優先順位とクラスを設定することにより、タスク内の現在のプロセスとそれに続くすべてのプロセスを変更することもできます。

プロジェクトへの参加が発生するたびに、タスクが作成されます。タスクは、次のアクション、コマンド、および関数によって作成されます。

- ログイン
- `cron`
- `newtask`
- `setproject`
- `su`

次のいずれかの方法で、最終的なタスクを作成できます。これ以降は、新しいタスクを作成しようとすると失敗します。

- `newtask` コマンドに `-F` オプションを付けて実行します。
- `project` ネームサービスデータベースで、プロジェクトに `task.final` 属性を設定します。`setproject` によってそのプロジェクト内に作成されるすべてのタスクに、`TASK_FINAL` フラグが設定されます。

詳細は、`login(1)`、`newtask(1)`、`cron(1M)`、`su(1M)`、および `setproject(3PROJECT)` のマニュアルページを参照してください。

拡張アカウンティング機能は、プロセスのアカウンティングデータを提供できます。データはタスクレベルで集計されます。

プロジェクトとタスクで使用するコマンド

次の表に示すコマンドは、プロジェクトとタスクの機能を管理するための主要なインタフェースとなります。

マニュアルページの参照	説明
projects(1)	ユーザーのプロジェクトメンバーシップを表示します。project データベース内のプロジェクトを一覧表示します。特定のプロジェクトについて情報を表示します。プロジェクト名が指定されていない場合は、すべてのプロジェクトについて情報を表示します。projects コマンドに -l オプションを付けて実行すると、詳細情報が出力されます。
newtask(1)	ユーザーのデフォルトのシェルまたは指定されたコマンドを実行し、指定されたプロジェクトが所有する新しいタスクに実行コマンドを配置します。また、newtask は、実行中のプロセスに結合するタスクとプロジェクトを変更するためにも使用できます。-F オプションを付けて実行すると、最終的なタスクを作成できます。
passmgmt(1M)	パスワードファイル内の情報を更新します。-K *key=value* オプションを付けて実行すると、ローカルファイル内のユーザー属性に値を追加したり、ローカルファイル内のユーザー属性を置換したりできます。
projadd(1M)	/etc/project ファイルに新しいプロジェクトエントリを追加します。projadd コマンドは、ローカルシステム上にだけプロジェクトエントリを作成します。projadd は、ネットワークネームサービスから提供される情報は変更できません。 デフォルトファイル /etc/project 以外のプロジェクトファイルを編集する場合に使用できます。project ファイルの構文検査機能を提供します。プロジェクト属性の検証と編集を行います。倍率値をサポートします。

マニュアルページの参照	説明
projmod(1M)	ローカルシステム上のプロジェクトの情報を変更します。projmod は、ネットワークネームサービスから提供される情報は変更できません。ただし、このコマンドは、外部のネームサービスに対してプロジェクト名とプロジェクト ID の一意性を確認します。 デフォルトファイル /etc/project 以外のプロジェクトファイルを編集する場合に使用できます。project ファイルの構文検査機能を提供します。プロジェクト属性の検証と編集を行います。新しい属性の追加、属性の値の追加、または属性の削除に使用できます。倍率値をサポートします。 Solaris 10 5/08 リリース以降は、-A オプションを付けて実行すると、プロジェクトデータベース内にあるリソース制御値をアクティブなプロジェクトに適用できます。prctl コマンドによって手動で設定された値など、project ファイルで定義されている値と一致しない既存の値は削除されます。
projdel(1M)	ローカルシステムからプロジェクトを削除します。projdel は、ネットワークネームサービスから提供される情報は変更できません。
useradd(1M)	デフォルトプロジェクトの定義をローカルファイルに追加します。-K *key*=*value* オプションを付けて実行すると、ユーザー属性を追加したり置換したりできます。
userdel(1M)	ローカルファイルからユーザーのアカウントを削除します。
usermod(1M)	システムにあるユーザーのログイン情報を変更します。-K *key*=*value* オプションを付けて実行すると、ユーザー属性を追加したり置換したりできます。

◆ ◆ ◆ 第 3 章

3 プロジェクトとタスクの管理

この章では、Solaris のリソース管理機能のうち、プロジェクトおよびタスク機能の使用方法について説明します。

この章の内容は次のとおりです。

プロジェクトとタスクの機能の概要については、第 2 章「プロジェクトとタスク (概要)」を参照してください。

注 – これらの機能をゾーンがインストールされている Solaris システムで使用する場合は、非大域ゾーンでこれらのコマンドを実行すると、プロセス ID を受け取るシステムコールインタフェースを通して、同じゾーン内のプロセスだけが認識されます。

プロジェクトとタスクの管理 (タスクマップ)

タスク	説明	参照先
プロジェクトとタスクで使用するコマンドとオプションの例を表示します。	タスク ID とプロジェクト ID を表示し、システムで現在実行されているプロセスとプロジェクトについて各種の統計情報を表示します。	56 ページの「コマンドとコマンドオプションの例」
プロジェクトを定義します。	`/etc/project` ファイルにプロジェクトエントリを追加し、そのエントリの値を変更します。	59 ページの「プロジェクトを定義して現在のプロジェクトを表示する方法」

タスク	説明	参照先
プロジェクトを削除します。	/etc/project ファイルからプロジェクトエントリを削除します。	61 ページの「/etc/project ファイルからプロジェクトを削除する方法」
project ファイルまたはプロジェクトデータベースを検証します。	/etc/project ファイルの構文を検査します。または、外部のネームサービスと照合してプロジェクト名およびプロジェクト ID の一意性を確認します。	62 ページの「/etc/project ファイルの内容を検証する方法」
プロジェクトのメンバーシップ情報を取得します。	起動中のプロセスの現在のプロジェクトメンバーシップを表示します。	63 ページの「プロジェクトのメンバーシップ情報を取得する方法」
新しいタスクを作成します。	newtask コマンドを使用して、特定のプロジェクトに新しいタスクを作成します。	63 ページの「新しいタスクを作成する方法」
実行中のプロセスを別のタスクとプロジェクトに関連付けます。	指定されたプロジェクト内の新しいタスク ID にプロセス番号を関連付けます。	63 ページの「実行中のプロセスを新しいタスクに移動する方法」
プロジェクト属性を追加し、操作します。	プロジェクトデータベースの管理コマンドを使用して、プロジェクト属性の追加、編集、検証、および削除を行います。	64 ページの「プロジェクト属性の編集と検証」

コマンドとコマンドオプションの例

このセクションでは、プロジェクトとタスクで使用するコマンドとオプションの例を示します。

プロジェクトとタスクで使用するコマンドオプション

ps コマンド

タスクおよびプロジェクトの ID を表示するには、ps コマンドに -o オプションを付けて実行します。たとえば、プロジェクト ID を表示するには、次のように入力します。

```
# ps -o user,pid,uid,projid
USER PID   UID  PROJID
jtd  89430 124  4113
```

id コマンド

ユーザーおよびグループ ID に加えて、現在のプロジェクト ID を表示するには、id コマンドに -p オプションを付けて実行します。*user* オペランドを指定した場合、そのユーザーの通常のログインに関連付けられたプロジェクトが表示されます。

```
#  id -p
uid=124(jtd) gid=10(staff) projid=4113(booksite)
```

pgrep コマンドと pkill コマンド

特定のリスト内のプロジェクト ID と一致するプロセスだけを表示するには、pgrep コマンドと pkill コマンドに -J オプションを付けて実行します。

```
# pgrep -J projidlist
# pkill -J projidlist
```

特定のリスト内のタスク ID と一致するプロセスだけを表示するには、pgrep コマンドと pkill コマンドに -T オプションを付けて実行します。

```
# pgrep -T taskidlist
# pkill -T taskidlist
```

prstat コマンド

システムで現在実行中のプロセスとプロジェクトのさまざまな統計情報を表示するには、prstat コマンドに -J オプションを付けて実行します。

```
% prstat -J
      PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 21634 jtd      5512K 4848K cpu0    44    0   0:00.00 0.3% prstat/1
   324 root       29M   75M sleep   59    0   0:08.27 0.2% Xsun/1
 15497 jtd        48M   41M sleep   49    0   0:08.26 0.1% adeptedit/1
   328 root     2856K 2600K sleep   58    0   0:00.00 0.0% mibiisa/11
  1979 jtd      1568K 1352K sleep   49    0   0:00.00 0.0% csh/1
  1977 jtd      7256K 5512K sleep   49    0   0:00.00 0.0% dtterm/1
   192 root     3680K 2856K sleep   58    0   0:00.36 0.0% automountd/5
  1845 jtd        24M   22M sleep   49    0   0:00.29 0.0% dtmail/11
  1009 jtd      9864K 8384K sleep   49    0   0:00.59 0.0% dtwm/8
   114 root     1640K  704K sleep   58    0   0:01.16 0.0% in.routed/1
   180 daemon   2704K 1944K sleep   58    0   0:00.00 0.0% statd/4
   145 root     2120K 1520K sleep   58    0   0:00.00 0.0% ypbind/1
   181 root     1864K 1336K sleep   51    0   0:00.00 0.0% lockd/1
   173 root     2584K 2136K sleep   58    0   0:00.00 0.0% inetd/1
   135 root     2960K 1424K sleep    0    0   0:00.00 0.0% keyserv/4
PROJID    NPROC  SIZE   RSS MEMORY      TIME  CPU PROJECT
    10       52  400M  271M    68%   0:11.45 0.4% booksite
     0       35  113M  129M    32%   0:10.46 0.2% system

Total: 87 processes, 205 lwps, load averages: 0.05, 0.02, 0.02
```

システムで現在実行中のプロセスとタスクのさまざまな統計情報を表示するには、prstat コマンドに -T オプションを付けて実行します。

```
% prstat -T
   PID USERNAME  SIZE   RSS STATE  PRI NICE      TIME  CPU PROCESS/NLWP
 23023 root       26M   20M sleep   59    0   0:03:18 0.6% Xsun/1
 23476 jtd        51M   45M sleep   49    0   0:04:31 0.5% adeptedit/1
 23432 jtd      6928K 5064K sleep   59    0   0:00:00 0.1% dtterm/1
 28959 jtd        26M   18M sleep   49    0   0:00:18 0.0% .netscape.bin/1
 23116 jtd      9232K 8104K sleep   59    0   0:00:27 0.0% dtwm/5
 29010 jtd      5144K 4664K cpu0    59    0   0:00:00 0.0% prstat/1
   200 root     3096K 1024K sleep   59    0   0:00:00 0.0% lpsched/1
   161 root     2120K 1600K sleep   59    0   0:00:00 0.0% lockd/2
   170 root     5888K 4248K sleep   59    0   0:03:10 0.0% automountd/3
   132 root     2120K 1408K sleep   59    0   0:00:00 0.0% ypbind/1
   162 daemon   2504K 1936K sleep   59    0   0:00:00 0.0% statd/2
   146 root     2560K 2008K sleep   59    0   0:00:00 0.0% inetd/1
   122 root     2336K 1264K sleep   59    0   0:00:00 0.0% keyserv/2
   119 root     2336K 1496K sleep   59    0   0:00:02 0.0% rpcbind/1
   104 root     1664K  672K sleep   59    0   0:00:03 0.0% in.rdisc/1
TASKID    NPROC  SIZE   RSS MEMORY      TIME  CPU PROJECT
   222       30  229M  161M    44%   0:05:54 0.6% group.staff
   223        1   26M   20M   5.3%   0:03:18 0.6% group.staff
    12        1   61M   33M   8.9%   0:00:31 0.0% group.staff
     1       33   85M   53M    14%   0:03:33 0.0% system

Total: 65 processes, 154 lwps, load averages: 0.04, 0.05, 0.06
```

注 – -J オプションと -T オプションを一緒に使用することはできません。

プロジェクトとタスクでの cron と su の使用

cron コマンド

cron コマンドは、settaskid を発行し、実行を要求したユーザーの適切なデフォルトプロジェクトを使用して、cron、at、および batch の各ジョブが別のタスクで実行されるようにします。また、at および batch コマンドは、現在のプロジェクト ID を取得して at ジョブを実行するときにプロジェクト ID が復元されるようにします。

su コマンド

su コマンドは、ログインのシミュレーションの一環として、新しいタスクを作成することによってターゲットユーザーのデフォルトプロジェクトに参加します。

su コマンドを使用してユーザーのデフォルトプロジェクトを切り替えるには、次のように入力します。

```
# su user
```

プロジェクトの管理

▼ プロジェクトを定義して現在のプロジェクトを表示する方法

次の例は、`projadd` コマンドを使用してプロジェクトエントリを追加し、`projmod` コマンドを使用してそのエントリを変更する方法を示します。

1 スーパーユーザーになるか、同等の役割を引き受けます。

役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **`projects -l`** を使用して、システムのデフォルトの **`/etc/project`** ファイルを表示します。

```
# projects -l
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

3 *booksite* という名前のプロジェクトを追加します。追加したプロジェクトを *mark* という名前のユーザーにプロジェクト **ID** 番号 *4113* で割り当てます。

```
# projadd -U mark -p 4113 booksite
```

4 再度 **/etc/project** ファイルを表示します。

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
booksite
        projid : 4113
        comment: ""
        users  : mark
        groups : (none)
        attribs:
```

5 **comment** フィールドにプロジェクトを説明するコメントを追加します。

```
# projmod -c 'Book Auction Project' booksite
```

6 **/etc/project** ファイルに加えた変更を確認します。

```
# projects -l
system
        projid : 0
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
```

```
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
booksite
        projid : 4113
        comment: "Book Auction Project"
        users  : mark
        groups : (none)
        attribs:
```

参照　プロジェクト、タスク、およびプロセスをプールに結合する方法については、189 ページの「プール属性の設定とプールへの結合」を参照してください。

▼ /etc/project ファイルからプロジェクトを削除する方法

次の例は、projdel コマンドを使ってプロジェクトを削除する方法を示します。

1. スーパーユーザーになるか、同等の役割を引き受けます。
   役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **projdel** コマンドを使ってプロジェクト *booksite* を削除します。

   ```
   # projdel booksite
   ```

3. **/etc/project** ファイルを表示します。

   ```
   # projects -l
   system
           projid : 0
   ```

```
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
user.root
        projid : 1
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
noproject
        projid : 2
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
default
        projid : 3
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
group.staff
        projid : 10
        comment: ""
        users  : (none)
        groups : (none)
        attribs:
```

4 ユーザー名 *mark* でログインして、**projects** と入力し、このユーザーに割り当てられているプロジェクトを表示します。

```
# su - mark
# projects
default
```

/etc/project ファイルの内容を検証する方法

編集オプションが指定されていない場合、projmod コマンドは project ファイルの内容を検証します。

NIS マップを検証するには、スーパーユーザーとして次のように入力します。

```
# ypcat project | projmod -f –
```

注 – ypcat project | projmod -f – コマンドはまだ実装されていません。

/etc/project ファイルの構文を検査するには、次のように入力します。

```
# projmod -n
```

プロジェクトのメンバーシップ情報を取得する方法

-p フラグを付けて id コマンドを使用し、起動中のプロセスの現在のプロジェクトメンバーシップを表示します。

```
$ id -p
uid=100(mark) gid=1(other) projid=3(default)
```

▼ 新しいタスクを作成する方法

1. 作成先となるプロジェクト *booksite* のメンバーとしてログインします。
2. *booksite* プロジェクト内に新しいタスクを作成します。 それには、システムのタスク ID を取得するための **-v** (冗長) オプションを指定して **newtask** コマンドを実行します。

   ```
   machine% newtask -v -p booksite
   16
   ```

   newtask を実行すると、指定したプロジェクト内に新しいタスクが作成され、そのタスクにユーザーのデフォルトのシェルが置かれます。
3. 起動中のプロセスの現在のプロジェクトメンバーシップを表示します。

   ```
   machine% id -p
   uid=100(mark) gid=1(other) projid=4113(booksite)
   ```

   今度は、プロセスが新しいプロジェクトのメンバーになっています。

▼ 実行中のプロセスを新しいタスクに移動する方法

次の例は、実行中のプロセスを別のタスクと新しいプロジェクトに関連付ける方法を示します。この操作を実行するには、スーパーユーザーでなければなりません。または、プロセスの所有者で、かつ新しいプロジェクトのメンバーでなければなりません。

1. スーパーユーザーになるか、同等の役割を引き受けます。

   役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

注 – プロセスの所有者または新しいプロジェクトのメンバーであれば、この手順は省略できます。

---

2 *book_catalog* プロセスのプロセス ID を取得します。

```
# pgrep book_catalog
    8100
```

3 プロセス *8100* を、新しいタスク ID を使って *booksite* プロジェクトに関連付けます。

```
# newtask -v -p booksite -c 8100
    17
```

`-c` オプションは、`newtask` が指定された既存のプロセスに対して動作することを指定します。

4 タスクとプロセス ID の対応を確認します。

```
# pgrep -T 17
    8100
```

プロジェクト属性の編集と検証

プロジェクトデータベースの管理コマンド `projadd` および `projmod` を使用して、プロジェクト属性を編集できます。

`-K` オプションは、属性の置換リストを指定します。属性はセミコロン (;) で区切られます。`-K` オプションを `-a` オプションとともに使用すると、その属性または属性値が追加されます。`-K` オプションを `-r` オプションとともに使用すると、その属性または属性値が削除されます。`-K` オプションを `-s` オプションとともに使用すると、その属性または属性値が置換されます。

▼ 属性と属性値をプロジェクトに追加する方法

プロジェクトの属性に値を追加するには、`projmod` コマンドに `-a` オプションと `-K` オプションを付けて実行します。属性が存在しない場合は、新たに作成されます。

1 スーパーユーザーになるか、同等の役割を引き受けます。

役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 プロジェクト **myproject** 内に *task.max-lwps* リソース制御属性を値なしで追加します。プロジェクトに加わるタスクでは、その属性にシステム値だけが設定されます。

```
# projmod -a -K task.max-lwps myproject
```

3 その後、プロジェクト **myproject** 内の *task.max-lwps* に値を追加できます。この値は、特権レベル、しきい値、およびしきい値に達したときのアクションから成ります。

```
# projmod -a -K "task.max-lwps=(priv,100,deny)" myproject
```

4 リソース制御は複数の値を持つことができるので、同じオプションを使用して、既存の値リストに別の値を追加できます。

```
# projmod -a -K "task.max-lwps=(priv,1000,signal=KILL)" myproject
```

複数の値はコンマで区切られます。`task.max-lwps` エントリはこの時点で次のようになっています。

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

▼ 属性値をプロジェクトから削除する方法

この手順では次の値を仮定します。

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

1 スーパーユーザーになるか、同等の役割を引き受けます。

役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 プロジェクト *myproject* 内のリソース制御 **task.max-lwps** から属性値を削除するには、**projmod** コマンドに **-r** オプションと **-K** オプションを付けて実行します。

```
# projmod -r -K "task.max-lwps=(priv,100,deny)" myproject
```

`task.max-lwps` が次のように複数の値を持っているとします。

```
task.max-lwps=(priv,100,deny),(priv,1000,signal=KILL)
```

この場合は、最初に一致する値が削除されます。したがって、結果は次のようになります。

```
task.max-lwps=(priv,1000,signal=KILL)
```

▼ リソース制御属性をプロジェクトから削除する方法

リソース制御 `task.max-lwps` をプロジェクト *myproject* から削除するには、`projmod` コマンドに `-r` オプションと `-K` オプションを付けて実行します。

1. スーパーユーザーになるか、同等の役割を引き受けます。
   役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 属性 **task.max-lwps** とそのすべての値を、プロジェクト *myproject* から削除します。

```
# projmod -r -K task.max-lwps myproject
```

▼ プロジェクトの属性と属性値を置換する方法

プロジェクト *myproject* 内のリソース制御 `task.max-lwps` の属性値を別の値で置換するには、`projmod` コマンドに `-s` オプションと `-K` オプションを付けて実行します。属性が存在しない場合は、新たに作成されます。

1. スーパーユーザーになるか、同等の役割を引き受けます。
   役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **task.max-lwps** の現在の値を、次に示す新しい値で置換します。

```
# projmod -s -K "task.max-lwps=(priv,100,none),(priv,120,deny)" myproject
```

   結果は次のようになります。

```
task.max-lwps=(priv,100,none),(priv,120,deny)
```

▼ リソース制御属性の既存の値を削除する方法

1. スーパーユーザーになるか、同等の役割を引き受けます。
   役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **task.max-lwps** の現在の値をプロジェクト *myproject* から削除するには、次のように入力します。

```
# projmod -s -K task.max-lwps myproject
```

◆ ◆ ◆ 第 4 章

拡張アカウンティング (概要)

プロジェクトおよびタスク機能 (第 2 章「プロジェクトとタスク (概要)」で説明) を使って作業負荷にラベル付けを行い、分離することにより、作業負荷ごとのリソース消費を監視できます。「拡張アカウンティング」サブシステムを使用すると、プロセスとタスクの両方について詳細なリソース消費統計情報を取得できます。

この章の内容は次のとおりです。

拡張アカウンティングの使用を開始する場合は、76 ページの「プロセス、タスク、およびフローの拡張アカウンティングを起動する方法」に進んでください。

Oracle Solaris 10 における拡張アカウンティングの新機能

プロセスアカウンティングの `mstate` データを生成できるようになりました。78 ページの「使用可能なアカウンティングリソースを表示する方法」を参照してください。

Oracle Solaris 10 の新機能の一覧および Oracle Solaris リリースについての説明は、『Oracle Solaris 10 8/11 の新機能』を参照してください。

拡張アカウンティングの紹介

拡張アカウンティングサブシステムは、行われた作業の対象プロジェクトの使用状況レコードにラベル付けします。また、拡張アカウンティングを IPQoS (Internet Protocol Quality of Service、IP サービス品質) フローアカウンティングモジュール (『Solaris のシステム管理 (IP サービス)』の第 36 章「フローアカウンティングの使用と統計情報の収集 (手順)」を参照) と組み合わせて、システム上のネットワークフロー情報を取得することもできます。

リソース管理メカニズムを適用する前に、まず、さまざまな作業負荷がシステムに対して行うリソース消費要求の特徴をつかむ必要があります。Solaris オペレーティングシステムの拡張アカウンティング機能を利用すると、タスクベース、プロセスベース、または IPQoS `flowacct` モジュールが提供するセレクタベースで、システムやネットワークのリソース消費を記録する柔軟性が得られます。詳細は、`ipqos`(7IPP) のマニュアルページを参照してください。

システムの使用状況をリアルタイムで計測するオンライン監視ツールとは異なり、拡張アカウンティング機能を使用すると、使用状況の履歴を調べることができます。その上で、将来の作業負荷の容量要件を算定できます。

拡張アカウンティングのデータを使用すれば、リソースの課金、作業負荷の監視、容量計画などの目的でソフトウェアを開発したり購入したりできます。

拡張アカウンティングの動作

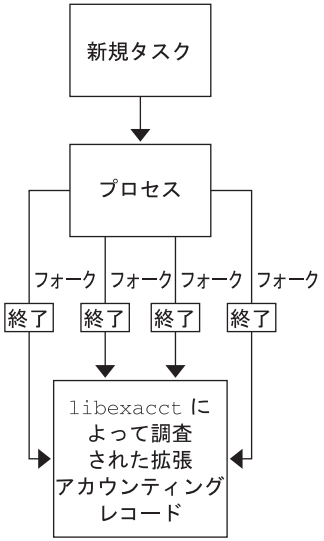
Solaris オペレーティングシステムの拡張アカウンティング機能は、バージョン番号が付けられた拡張ファイル形式を使用してアカウンティングデータを格納します。このデータ形式を使用するファイルは、添付のライブラリ `libexacct` (`libexacct`(3LIB) のマニュアルページを参照) で提供される API を使ってアクセスまたは作成できます。作成されたファイルは、拡張アカウンティング機能を使用できる任意のプラットフォーム上で解析でき、データを容量計画や課金に使用できます。

拡張アカウンティングを起動すると、`libexacct` API で調べることができる統計情報が収集されます。`libexacct` は、`exacct` ファイルを前後どちらの方向からでも検査できます。API は、カーネルが作成するファイルだけでなく、`libexacct` によって生成された他社製のファイルもサポートします。`libexacct` に対する Perl (Practical Extraction and Report Language) インタフェースが用意されています。これを使えば、報告および抽出用のカスタムスクリプトを開発できます。71 ページの「`libexacct` に対する Perl インタフェース」を参照してください。

たとえば、拡張アカウンティングを有効にすると、タスクは、自分のメンバープロセスの総リソース使用状況を追跡します。タスクのアカウンティングレコードは、そのタスクの完了時に書き込まれます。実行中のプロセスやタスクについて中

間レコードを書き込むこともできます。タスクの詳細については、第 2 章「プロジェクトとタスク (概要)」を参照してください。

図 4–1 拡張アカウンティング起動時のタスクの追跡



拡張可能な形式

拡張アカウンティングの形式は、古い SunOS システムのアカウンティングソフトウェアの形式に比べて拡張性があります (『Solaris のシステム管理 (上級編)』の「システムアカウンティング」を参照)。拡張アカウンティングでは、システムアカウンティングメトリックスのシステムへの追加や削除をシステムの解放時またはシステムの操作中に行うことができます。

注 – 拡張アカウンティングと古いシステムのアカウンティングソフトウェアの両方をシステム上で同時に起動できます。

---

exacct レコードとその形式

exacct レコードを作成するルーチンは、次の 2 つの目的で使用できます。

- 他社製の exacct ファイルを作成できるようにします。
- putacct システムコールを使ってカーネルアカウンティングファイルに埋め込むためのタグ付けレコードを作成できるようにします (getacct(2) のマニュアルページを参照)。

注 – Perl インタフェースから putacct システムコールを利用することもできます。

この形式では、すべての変更を明示的なバージョン変更にしなくても、さまざまな形式のアカウンティングレコードを取得できます。アカウンティングデータを使用するアプリケーションは、認識不可能なレコードを無視するように作成する必要があります。

libexacct ライブラリは、ファイルを exacct 形式に変換し、その形式のファイルを生成します。このライブラリは、exacct 形式のファイルに対するインタフェースとしてサポートされている唯一のインタフェースです。

注 – getacct、putacct、wracct の各システムコールは、フローには適用されません。IPQoS フローアカウンティングの構成時には、カーネルによってフローレコードが作成され、ファイルに書き込まれます。

ゾーンがインストールされている Solaris システムでの拡張アカウンティングの使用

拡張アカウンティングサブシステムを大域ゾーンで実行した場合、非大域ゾーンを含むシステム全体の情報が収集および報告されます。大域管理者は、ゾーンごとのリソース消費を決定することもできます。詳細は、396 ページの「ゾーンがインストールされている Oracle Solaris システムでの拡張アカウンティング」を参照してください。

拡張アカウンティング構成

/etc/acctadm.conf ファイルには、現在の拡張アカウンティング構成が含まれます。このファイルは、ユーザーが直接編集するのではなく、acctadm インタフェースを介して編集します。

拡張アカウンティングデータは、デフォルトでは /var/adm/exacct ディレクトリに置かれます。acctadm コマンドを使用すると、プロセスやタスクのアカウンティングデータファイルの格納場所を変更できます。詳細は、acctadm(1M) のマニュアルページを参照してください。

拡張アカウンティングで使用されるコマンド

コマンドのリファレンス	説明
acctadm(1M)	拡張アカウンティング機能のさまざまな属性の変更、拡張アカウンティングの停止と起動を行います。また、プロセス、タスク、およびフローを追跡するためのアカウンティング属性を選択するのに使用します。
wracct(1M)	アクティブなプロセスおよびタスクの拡張アカウンティングアクティビティーを書き込みます。
lastcomm(1)	直前に呼び出されたコマンドを表示します。lastcomm では、標準アカウンティングプロセスのデータまたは拡張アカウンティングプロセスのデータのどちらかを使用できます。

タスクやプロジェクトに関連するコマンドについては、56 ページの「コマンドとコマンドオプションの例」を参照してください。IPQoS フローアカウンティングについては、ipqosconf(1M) のマニュアルページを参照してください。

libexacct に対する Perl インタフェース

Perl インタフェースによって、exacct フレームワークで作成されたアカウンティングファイルを読み取ることのできる、Perl スクリプトを作成できます。exacct ファイルを作成する Perl スクリプトも作成できます。

このインタフェースの機能は、ベースとなる C 言語の API と同様です。可能な場合は、ベースとなる C 言語の API から取得したデータを Perl データタイプとして表示します。この機能によって、データアクセスが簡単になり、バッファーのパック/アンパック操作が不要になります。さらに、あらゆるメモリー管理が Perl ライブラリによって実行されます。

各種のプロジェクト、タスク、exacct 関連機能はいくつかのグループに分けられます。各機能グループは、別々の Perl モジュールに配置されます。各モジュールは、Sun の標準である Sun::Solaris:: Perl パッケージ接頭辞で始まります。Perl exacct ライブラリが提供するクラスはすべて、Sun::Solaris::Exacct モジュールの下にあります。

配下の libexacct(3LIB) ライブラリは、exacct 形式のファイル、カタログタグ、および exacct オブジェクトに対する操作を実行します。exacct オブジェクトは、次の 2 つのタイプに分けられます。

- アイテム (単一データ値 [スカラー])
- グループ (項目のリスト)

次の表に各モジュールの概要を示します。

モジュール (空白文字は使用不可)	説明	参照先
Sun::Solaris::Project	このモジュールは、次のプロジェクト操作関数にアクセスする機能を提供します。getprojid(2)、endprojent(3PROJECT)、fgetprojent(3PROJECT)、getdefaultproj(3PROJECT)、getprojbyid(3PROJECT)、getprojbyname(3PROJECT)、getprojent(3PROJECT)、getprojidbyname(3PROJECT)、inproj(3PROJECT)、project_walk(3PROJECT)、setproject(3PROJECT)、 および setprojent(3PROJECT) です。	Project(3PERL)
Sun::Solaris::Task	このモジュールは、タスク操作関数である gettaskid(2) と settaskid(2) にアクセスする機能を提供します。	Task(3PERL)
Sun::Solaris::Exacct	最上位レベルの exacct モジュール。このモジュールは、exacct 関連のシステムコールである getacct(2)、putacct(2)、および wracct(2) にアクセスする機能を提供します。このモジュールは、libexacct(3LIB) ライブラリ関数である ea_error(3EXACCT) にアクセスする機能も提供します。exacct EO_*、EW_*、EXR_*、P_*、および TASK_* マクロのすべてに対応する定数も、このモジュールで提供されます。	Exacct(3PERL)
Sun::Solaris::Exacct:: Catalog	このモジュールは、exacct カタログタグ内のビットフィールドにアクセスする、オブジェクト指向型メソッドを提供します。このモジュールによって、EXC_*、EXD_*、および EXD_* マクロの定数にもアクセスできます。	Exacct::Catalog(3PERL)
Sun::Solaris::Exacct:: File	このモジュールは、次の libexacct アカウンティングファイル関数にアクセスする、オブジェクト指向型メソッドを提供します。ea_open(3EXACCT)、ea_close(3EXACCT)、ea_get_creator(3EXACCT)、ea_get_hostname(3EXACCT)、ea_next_object(3EXACCT)、ea_previous_object(3EXACCT)、 および ea_write_object(3EXACCT) です。	Exacct::File(3PERL)

モジュール (空白文字は使用不可)	説明	参照先
Sun::Solaris::Exacct:: Object	このモジュールは、個々の exacct アカウンティングファイルオブジェクトにアクセスする、オブジェクト指向型メソッドを提供します。exacct オブジェクトは、該当する Sun::Solaris::Exacct::Object サブクラスに与えられた、隠された参照として表されます。このモジュールはさらに、アイテムかグループかのオブジェクトタイプに分けられます。このレベルで、ea_match_object_catalog(3EXACCT) および ea_attach_to_object(3EXACCT) 関数にアクセスするメソッドがあります。	Exacct::Object(3PERL)
Sun::Solaris::Exacct:: Object::Item	このモジュールは、独立した exacct アカウンティングファイルアイテムにアクセスする、オブジェクト指向型メソッドを提供します。このタイプのオブジェクトは、Sun::Solaris::Exacct::Object から継承します。	Exacct::Object::Item(3PERL)
Sun::Solaris::Exacct:: Object::Group	このモジュールは、独立した exacct アカウンティングファイルグループにアクセスする、オブジェクト指向型メソッドを提供します。このタイプのオブジェクトは、Sun::Solaris::Exacct::Object から継承します。これらのオブジェクトによって、ea_attach_to_group(3EXACCT) 関数にアクセスできます。グループ内のアイテムは Perl 配列として表されます。	Exacct::Object::Group(3PERL)
Sun::Solaris::Kstat	このモジュールは、kstat 機能に対する Perl のタイハッシュインタフェースを提供します。このモジュールの使用例については、Perl で記述された /bin/kstat を参照してください。	Kstat(3PERL)

表で説明したモジュールの使用例については、79 ページの「libexacct に対する Perl インタフェースの使用」を参照してください。

◆ ◆ ◆ 第 5 章

拡張アカウンティングの管理 (タスク)

この章では、拡張アカウンティングサブシステムを管理する方法について説明します。

拡張アカウンティングサブシステムの概要については、第 4 章「拡張アカウンティング (概要)」を参照してください。

拡張アカウンティング機能の管理 (タスクマップ)

タスク	説明	参照先
拡張アカウンティング機能を起動します。	拡張アカウンティングを使用して、システムで実行されている各プロジェクトのリソース消費を監視します。「拡張アカウンティング」サブシステムを使用すると、タスク、プロセス、およびフローの履歴データを取り込むことができます。	76 ページの「プロセス、タスク、およびフローの拡張アカウンティングを起動する方法」、77 ページの「起動スクリプトを使って拡張アカウンティングを起動する方法」
拡張アカウンティングのステータスを表示します。	拡張アカウンティング機能のステータスを調べます。	77 ページの「拡張アカウンティングステータスを表示する方法」
使用可能なアカウンティングリソースを表示します。	システム上の使用可能なアカウンティングリソースを表示します。	78 ページの「使用可能なアカウンティングリソースを表示する方法」
プロセス、タスク、およびフローのアカウンティング機能を停止します。	拡張アカウンティング機能をオフにします。	78 ページの「プロセス、タスク、およびフローアカウンティングを停止する方法」

タスク	説明	参照先
拡張アカウンティング機能に対する Perl インタフェースを使用します。	Perl インタフェースを使用して、報告および抽出用のカスタムスクリプトを作成します。	79 ページの「libexacct に対する Perl インタフェースの使用」

拡張アカウンティング機能の使用

ユーザーは、管理対象の拡張アカウンティングの種類に適した権利プロファイルがあれば、拡張アカウンティングの管理 (アカウンティングの開始、アカウンティングの停止、およびアカウンティング構成パラメータの変更) を行うことができます。

- フロー管理
- プロセス管理
- タスク管理

▼ プロセス、タスク、およびフローの拡張アカウンティングを起動する方法

タスク、プロセス、およびフローの拡張アカウンティング機能を起動するには、`acctadm` コマンドを使用します。`acctadm` の最後に付けられたオプションのパラメータは、このコマンドが、拡張アカウンティング機能のプロセスアカウンティングコンポーネント、システムタスクアカウンティングコンポーネント、フローアカウンティングコンポーネントのいずれに作用するかを示します。

1. スーパーユーザーになるか、同等の役割を引き受けます。

   役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. プロセスの拡張アカウンティングを起動します。

   ```
   # acctadm -e extended -f /var/adm/exacct/proc process
   ```

3. タスクの拡張アカウンティングを起動します。

   ```
   # acctadm -e extended,mstate -f /var/adm/exacct/task task
   ```

4. フローの拡張アカウンティングを起動します。

   ```
   # acctadm -e extended -f /var/adm/exacct/flow flow
   ```

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

起動スクリプトを使って拡張アカウンティングを起動する方法

/etc/init.d/acctadm スクリプトへのリンクを /etc/rc2.d に作成することにより、実行中に拡張アカウンティングを起動できます。

```
# ln -s /etc/init.d/acctadm /etc/rc2.d/Snacctadm
# ln -s /etc/init.d/acctadm /etc/rc2.d/Knacctadm
```

変数 *n* は番号で置き換えられます。

構成を設定するために、少なくとも一度は拡張アカウンティングを手動で起動する必要があります。

アカウンティング構成の詳細については、70 ページの「拡張アカウンティング構成」を参照してください。

拡張アカウンティングステータスを表示する方法

引数なしで acctadm と入力すると、拡張アカウンティング機能の現在のステータスが表示されます。

```
# acctadm
                Task accounting: active
           Task accounting file: /var/adm/exacct/task
         Tracked task resources: extended
       Untracked task resources: none
              Process accounting: active
        Process accounting file: /var/adm/exacct/proc
      Tracked process resources: extended
    Untracked process resources: host
                Flow accounting: active
           Flow accounting file: /var/adm/exacct/flow
         Tracked flow resources: extended
       Untracked flow resources: none
```

この例では、システムタスクアカウンティングが拡張モードと mstate モードで動作しています。プロセスアカウンティングとフローアカウンティングは、拡張モードで動作しています。

注 – 拡張アカウンティングの分野では、マイクロステート (mstate) は、プロセス状態の微小な変化を反映した拡張データを意味し、このデータはプロセス使用状況ファイルで利用できます (proc(4) のマニュアルページを参照)。このデータは、プロセスの活動に関して、基本レコードや拡張レコードよりも詳細な情報を提供します。

使用可能なアカウンティングリソースを表示する方法

使用可能なリソースは、システムやプラットフォームによってさまざまです。acctadm コマンドに -r オプションを付けて実行すると、システム上の使用可能なアカウンティングリソースグループを表示できます。

```
# acctadm -r
process:
extended pid,uid,gid,cpu,time,command,tty,projid,taskid,ancpid,wait-status,zone,flag,
memory,mstate      displays as one line
basic    pid,uid,gid,cpu,time,command,tty,flag
task:
extended taskid,projid,cpu,time,host,mstate,anctaskid,zone
basic    taskid,projid,cpu,time
flow:
extended
saddr,daddr,sport,dport,proto,dsfield,nbytes,npkts,action,ctime,lseen,projid,uid
basic    saddr,daddr,sport,dport,proto,nbytes,npkts,action
```

▼ プロセス、タスク、およびフローアカウンティングを停止する方法

プロセス、タスク、およびフローのアカウンティングを停止するには、それぞれを個別にオフにします。そのためには、acctadm コマンドに -x オプションを付けて実行します。

1. スーパーユーザーになるか、同等の役割を引き受けます。

   役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. プロセスアカウンティングをオフにします。

   ```
   # acctadm -x process
   ```

3. タスクアカウンティングをオフにします。

   ```
   # acctadm -x task
   ```

4. フローアカウンティングをオフにします。

   ```
   # acctadm -x flow
   ```

5 タスクアカウンティング、プロセスアカウンティング、およびフローアカウンティングがオフになったことを確認します。

```
    # acctadm
            Task accounting: inactive
       Task accounting file: none
     Tracked task resources: extended
   Untracked task resources: none
         Process accounting: inactive
    Process accounting file: none
  Tracked process resources: extended
Untracked process resources: host
            Flow accounting: inactive
       Flow accounting file: none
     Tracked flow resources: extended
   Untracked flow resources: none
```

libexacct に対する Perl インタフェースの使用

exacct オブジェクトの内容を再帰的に出力する方法

exacct オブジェクトの内容を再帰的に出力するには、次のコードを使用します。この機能は、ライブラリによって Sun::Solaris::Exacct::Object::dump() 関数として提供されています。ea_dump_object() という簡易関数でこの機能を利用することもできます。

```
sub dump_object
     {
             my ($obj, $indent) = @_;
             my $istr = ’  ’ x $indent;

             #
             # Retrieve the catalog tag.  Because we are
             # doing this in an array context, the
             # catalog tag will be returned as a (type, catalog, id)
             # triplet, where each member of the triplet will behave as
             # an integer or a string, depending on context.
             # If instead this next line provided a scalar context, e.g.
             #    my $cat  = $obj->catalog()->value();
             # then $cat would be set to the integer value of the
             # catalog tag.
             #
             my @cat = $obj->catalog()->value();

             #
             # If the object is a plain item
             #
             if ($obj->type() == &EO_ITEM) {
                     #
```

```
                        # Note: The '%s' formats provide s string context, so
                        # the components of the catalog tag will be displayed
                        # as the symbolic values. If we changed the '%s'
                        # formats to '%d', the numeric value of the components
                        # would be displayed.
                        #
                        printf("%sITEM\n%s  Catalog = %s|%s|%s\n",
                           $istr, $istr, @cat);
                        $indent++;

                        #
                        # Retrieve the value of the item.  If the item contains
                        # in turn a nested exacct object (i.e., an item or
                        # group),then the value method will return a reference
                        # to the appropriate sort of perl object
                        # (Exacct::Object::Item or Exacct::Object::Group).
                        # We could of course figure out that the item contained
                        # a nested item orgroup by examining the catalog tag in
                        # @cat and looking for a type of EXT_EXACCT_OBJECT or
                        # EXT_GROUP.
                        #
                        my $val = $obj->value();
                        if (ref($val)) {
                                # If it is a nested object, recurse to dump it.
                                dump_object($val, $indent);
                        } else {
                                # Otherwise it is just a 'plain' value, so
                                # display it.
                                printf("%s  Value = %s\n", $istr, $val);
                        }

                #
                # Otherwise we know we are dealing with a group.  Groups
                # represent contents as a perl list or array (depending on
                # context), so we can process the contents of the group
                # with a 'foreach' loop, which provides a list context.
                # In a list context the value method returns the content
                # of the group as a perl list, which is the quickest
                # mechanism, but doesn't allow the group to be modified.
                # If we wanted to modify the contents of the group we could
                # do so like this:
                #    my $grp = $obj->value();   # Returns an array reference
                #    $grp->[0] = $newitem;
                # but accessing the group elements this way is much slower.
                #
                } else {
                        printf("%sGROUP\n%s  Catalog = %s|%s|%s\n",
                            $istr, $istr, @cat);
                        $indent++;
                        # 'foreach' provides a list context.
                        foreach my $val ($obj->value()) {
                                dump_object($val, $indent);
                        }
                        printf("%sENDGROUP\n", $istr);
                }
}
```

新しいグループレコードを作成してファイルに書き込む方法

新しいグループレコードを作成して /tmp/exacct というファイルに書き込むには、次のスクリプトを使用します。

```
#!/usr/bin/perl

use strict;
use warnings;
use Sun::Solaris::Exacct qw(:EXACCT_ALL);
# Prototype list of catalog tags and values.
     my @items = (
             [ &EXT_STRING | &EXC_DEFAULT | &EXD_CREATOR      => "me"       ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_PID     => $$         ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_UID     => $<         ],
             [ &EXT_UINT32 | &EXC_DEFAULT | &EXD_PROC_GID     => $(         ],
             [ &EXT_STRING | &EXC_DEFAULT | &EXD_PROC_COMMAND => "/bin/rec" ],
     );

     # Create a new group catalog object.
     my $cat = ea_new_catalog(&EXT_GROUP | &EXC_DEFAULT | &EXD_NONE)

     # Create a new Group object and retrieve its data array.
     my $group = ea_new_group($cat);
     my $ary = $group->value();

     # Push the new Items onto the Group array.
     foreach my $v (@items) {
             push(@$ary, ea_new_item(ea_new_catalog($v->[0]), $v->[1]));
     }

     # Open the exacct file, write the record & close.
     my $f = ea_new_file('/tmp/exacct', &O_RDWR | &O_CREAT | &O_TRUNC)
        || die("create /tmp/exacct failed: ", ea_error_str(), "\n");
     $f->write($group);
     $f = undef;
```

exacct ファイルの内容を出力する方法

exacct ファイルの内容を出力するには、次の Perl スクリプトを使用します。

```
#!/usr/bin/perl

     use strict;
     use warnings;
     use Sun::Solaris::Exacct qw(:EXACCT_ALL);

     die("Usage is dumpexacct <exacct file>\n") unless (@ARGV == 1);

     # Open the exact file and display the header information.
     my $ef = ea_new_file($ARGV[0], &O_RDONLY) || die(error_str());
     printf("Creator:  %s\n", $ef->creator());
```

```
printf("Hostname: %s\n\n", $ef->hostname());

# Dump the file contents
while (my $obj = $ef->get()) {
        ea_dump_object($obj);
}

# Report any errors
if (ea_error() != EXR_OK && ea_error() != EXR_EOF)  {
        printf("\nERROR: %s\n", ea_error_str());
        exit(1);
}
exit(0);
```

Sun::Solaris::Exacct::Object->dump() からの出力例

81 ページの「新しいグループレコードを作成してファイルに書き込む方法」で作成されたファイルに Sun::Solaris::Exacct::Object->dump() を実行した場合の出力例を示します。

```
Creator:  root
Hostname: localhost
GROUP
       Catalog = EXT_GROUP|EXC_DEFAULT|EXD_NONE
       ITEM
         Catalog = EXT_STRING|EXC_DEFAULT|EXD_CREATOR
         Value = me
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_PID
         Value = 845523
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_UID
         Value = 37845
       ITEM
         Catalog = EXT_UINT32|EXC_DEFAULT|EXD_PROC_GID
         Value = 10
       ITEM
         Catalog = EXT_STRING|EXC_DEFAULT|EXD_PROC_COMMAND
         Value = /bin/rec
ENDGROUP
```

◆ ◆ ◆ 第 6 章

リソース制御 (概要)

第 4 章「拡張アカウンティング (概要)」で説明したようにシステム上の作業負荷のリソース消費を判定したら、リソースの使用方法に制限を設けることができます。制限を設けると、作業負荷によるリソースの過剰消費を防ぐことができます。「リソース制御」機能は、この目的に使用される制約メカニズムです。

この章の内容は次のとおりです。

リソース制御を管理する方法については、第 7 章「リソース制御の管理 (タスク)」を参照してください。

Solaris 10 におけるリソース制御の新機能

System V プロセス間通信 (IPC) の `/etc/system` 調整可能パラメータが、次の一連のリソース制御で置き換えられています。

- `project.max-shm-ids`
- `project.max-msg-ids`
- `project.max-sem-ids`
- `project.max-shm-memory`
- `process.max-sem-nsems`
- `process.max-sem-ops`
- `process.max-msg-qbytes`

次のイベントポートリソース制御が追加されています。

- `project.max-device-locked-memory`

- `project.max-port-ids`
- `process.max-port-events`

次の暗号化リソース制御が追加されています。

- `project.max-crypto-memory`

その他、次のリソース制御が追加されています。

- `project.max-lwps`
- `project.max-tasks`
- `project.max-contracts`

詳細は、87 ページの「使用可能なリソース制御」を参照してください。

Solaris 10 の新機能の全一覧および Solaris リリースについての説明は、『Oracle Solaris 10 8/11 の新機能』を参照してください。

リソース制御の概念

Solaris オペレーティングシステムでは、プロセスごとのリソース制限という概念が、第 2 章「プロジェクトとタスク (概要)」で説明したタスクおよびプロジェクトのエンティティーに拡張されています。この拡張機能は、リソース制御 (rctls) 機能によって提供されます。また、割り当ては `/etc/system` 調整可能パラメータを通して設定していましたが、これもリソース制御メカニズムを通して自動的に行われるか、手動で構成するようになりました。

リソース制御には、接頭辞 `zone`、`project`、`task`、または `process` が付きます。リソース制御はシステム全体に適用できます。動作中のシステム上のリソース制御値を更新できます。

このリリースで使用できる標準のリソース制御のリストについては、87 ページの「使用可能なリソース制御」を参照してください。ゾーンごとに使用可能なリソース制御については、252 ページの「リソースタイプのプロパティー」を参照してください。

このリリースで使用できる標準のリソース制御のリストについては、87 ページの「使用可能なリソース制御」を参照してください。

リソース制限とリソース制御

従来から、UNIX システムにはリソース制限機能があります (*rlimit*)。rlimit の機能を使用すると、管理者は、プロセスが消費できるリソースの量に対して 1 つ以上の数値制限を設定できます。この制限には、プロセスごとの CPU 使用時間、プロセスごとのコアファイルサイズ、プロセスごとの最大ヒープサイズが含まれます。「ヒープサイズ」は、プロセスのデータセグメントに割り当てられるスクラッチメモリー領域のサイズです。

リソース制御機能は、リソース制限機能に対する互換性インタフェースを提供します。リソース制限機能を使用する既存のアプリケーションは、変更せずに、引き続き使用できます。また、既存のアプリケーションは、リソース制御機能を利用するように変更されたアプリケーションと同様に監視することができます。

プロセス間通信とリソース制御

プロセスは、数種類のプロセス間通信 (IPC) の 1 つを使用して、互いに通信できます。IPC を使用すると、プロセス間で情報の転送や同期化を行うことができます。Solaris 10 リリースの前は、`/etc/system` ファイルにエントリを追加することで、IPC 調整可能パラメータを設定していました。今後は、リソース制御機能により、カーネルの IPC 機能の動作を定義するリソース制御が提供されるようになりました。これらのリソース制御は、`/etc/system` の調整可能パラメータを置換します。

古いパラメータが、Solaris システムの `/etc/system` ファイルに入っていることがあります。その場合、これらのパラメータは、以前の Solaris リリースの場合と同様に、デフォルトのリソース制御値の初期化に使用されます。ただし、古いパラメータはできるだけ使用しないでください。

どの IPC オブジェクトがプロジェクトの使用状況に影響を与えているかを監視するには、`ipcs` コマンドに `-J` オプションを付けて実行します。表示例については、109 ページの「ipcs を使用する方法」を参照してください。`ipcs` コマンドの詳細については、`ipcs(1)` のマニュアルページを参照してください。

Solaris システムの調整については、『Oracle Solaris カーネルのチューンアップ・リファレンスマニュアル』を参照してください。

リソース制御の制約メカニズム

リソース制御機能は、システムリソースに対する制約メカニズムを提供します。これにより、プロセス、タスク、プロジェクト、およびゾーンが、指定したシステムリソースを過剰消費することを防止できます。このメカニズムは、リソースの過剰消費を防ぐことにより、より管理しやすいシステムを実現します。

制約メカニズムは、容量計画を実施するときにも使用できます。制約を設けることにより、アプリケーションへのリソースの提供を必ずしも拒否することなく、アプリケーションが必要とするリソース量に関する情報を取得できます。

プロジェクトの属性メカニズム

また、リソース制御は、リソース管理機能のための簡単な属性メカニズムとしても利用できます。たとえば、公平配分スケジューラ (FSS) のスケジューリングクラスで動作しているプロジェクトで利用できる CPU の配分は、リソース制御

project.cpu-shares によって定義されます。プロジェクトはリソース制御によって一定の配分を割り当てられるため、制御の超過につながる各種のアクションは許可されません。そのため、リソース制御 project.cpu-shares の現在値は、指定したプロジェクトの属性とみなすことができます。

また、プロジェクト内のプロセスの集合が消費する物理メモリーを規制するには、別の種類のプロジェクト属性が使用されます。これらの属性には、接頭辞 rcap が付きます (たとえば、rcap.max-rss)。リソース制御と同様に、この種類の属性も project データベース中に構成します。リソース制御はカーネルによって同期的に実行されますが、物理メモリーのリソース上限の制限はリソース上限デーモン rcapd によってユーザーレベルで非同期的に強制実行されます。rcapd については、第 10 章「リソース上限デーモンによる物理メモリーの制御 (概要)」および rcapd(1M) のマニュアルページを参照してください。

project.pool 属性は、プロジェクトのプールの結合を指定するために使用されます。リソースプールの詳細については、第 12 章「リソースプール (概要)」を参照してください。

リソース制御と属性の構成

リソース制御機能は、project データベースによって構成されます。第 2 章「プロジェクトとタスク (概要)」を参照してください。リソース制御とその他の属性は、project データベースエントリの最後のフィールドで設定します。各リソース制御に対応付けられる値は、括弧で囲まれ、コンマ区切りのプレーンテキストとして示されます。括弧内の値によって「アクション文節」が構成されます。各アクション文節には、値として特権レベル、しきい値、および特定のしきい値に対応付けられたアクションが含まれます。各リソース制御は複数のアクション文節を持つことができ、各アクション文節もコンマで区切られます。次のエントリは、プロジェクトエンティティーにおけるタスクごとの軽量プロセス (LWP) 制限と、プロセスごとの最長 CPU 時間制限を定義します。process.max-cpu-time は、プロセスの実行時間が合計で 1 時間になるとプロセスに SIGTERM を送信し、1 時間 1 分になると SIGKILL を送信します。表 6–3を参照してください。

```
development:101:Developers:::task.max-lwps=(privileged,10,deny);
  process.max-cpu-time=(basic,3600,signal=TERM),(priv,3660,signal=KILL)
    typed as one line
```

注 – ゾーンが有効になっているシステムの場合、ゾーン規模のリソース制御はゾーン構成で指定されます。その形式は多少異なります。詳細は、247 ページの「ゾーン構成データ」を参照してください。

---

rctladm コマンドを使用すると、リソース制御機能の実行時に問い合わせや制御機能の変更を「大域有効範囲」で行うことができます。prctl コマンドを使用すると、実行時にリソース制御機能の問い合わせや変更を「局所有効範囲」で行うことができます。

詳細については、93 ページの「リソース制御値に対応付けられた大域アクションと局所アクション」、および rctladm(1M) と prctl(1) のマニュアルページを参照してください。

注 – ゾーンがインストールされているシステムでは、非大域ゾーンで rctladm を使用して設定を変更することはできません。各リソース制御の大域ログ状態を表示する場合に、非大域ゾーンで rctladm を使用します。

使用可能なリソース制御

次の表に、このリリースで使用できる標準のリソース制御を示します。

この表では、各制御によって制約されるリソースについて説明し、project データベースにおけるそのリソースのデフォルトの単位を示します。デフォルトの単位には次の 2 種類があります。

- 数量は制限される量を意味します。
- インデックスは最大有効識別子を意味します。

したがって、project.cpu-shares は、プロジェクトで使用することが許可されている配分を示します。一方、process.max-file-descriptor は、open(2) システムコールによってプロセスに割り当てることができる最大ファイル番号を指定します。

表 6–1　標準のリソース制御

制御名	説明	デフォルトの単位
project.cpu-cap	**Solaris 10 8/07:** 1 つのプロジェクトで消費可能な CPU リソース量に対する絶対的な制限。project.cpu-cap 設定と同様、100 の値は 1 つの CPU の 100% を意味します。125 の値は 125% になります。CPU キャップの使用時は、100% がシステム上の 1 つの CPU の上限となります。	数量 (CPU の数)
project.cpu-shares	このプロジェクトに対して、公平配分スケジューラ (FSS(7) のマニュアルページを参照) で使用することが許可されている CPU 配分。	数量 (配分)

表 6–1 標準のリソース制御 (続き)

制御名	説明	デフォルトの単位
project.max-crypto-memory	ハードウェアによる暗号化処理の高速化のために libpkcs11 が使用できるカーネルメモリーの合計量。カーネルバッファーおよびセッション関連の構造体の割り当ては、このリソース制御に対してチャージされます。	サイズ (バイト)
project.max-locked-memory	ロックされる物理メモリーの許容合計量。 priv_proc_lock_memory がユーザーに割り当てられている場合、そのユーザーがすべてのメモリーをロックするのを防ぐため、このリソース制御の設定も検討してください。 **Solaris 10 8/07:** Solaris 10 8/07 リリースでは、project.max-device-locked-memory は削除され、このリソース制御で置き換えられました。	サイズ (バイト)
project.max-port-ids	イベントポートの許容最大数。	数量 (イベントポート数)
project.max-sem-ids	このプロジェクトに許容されるセマフォー ID の最大数。	数量 (セマフォー ID の数)
project.max-shm-ids	このプロジェクトに許容される共有メモリー ID の最大数。	数量 (共有メモリー ID の数)
project.max-msg-ids	このプロジェクトに許容されるメッセージキュー ID の最大数。	数量 (メッセージキュー ID の数)
project.max-shm-memory	このプロジェクトに許容される System V 共有メモリーの合計量。	サイズ (バイト)
project.max-lwps	このプロジェクトで同時に使用できる LWP の最大数。	数量 (LWP 数)
project.max-tasks	このプロジェクトに許容されるタスクの最大数	数量 (タスク数)
project.max-contracts	このプロジェクトに許容される契約の最大数	数量 (契約数)
task.max-cpu-time	このタスクのプロセスで使用できる最長 CPU 時間。	時間 (秒)

表 6–1 標準のリソース制御 (続き)

制御名	説明	デフォルトの単位
task.max-lwps	このタスクのプロセスで同時に使用できる LWP の最大数。	数量 (LWP 数)
process.max-cpu-time	このプロセスで使用できる最長 CPU 時間。	時間 (秒)
process.max-file-descriptor	このプロセスで使用できる最大のファイル記述子インデックス。	インデックス (最大ファイル記述子)
process.max-file-size	このプロセスの書き込みに使用できる最大ファイルオフセット。	サイズ (バイト)
process.max-core-size	このプロセスによって作成されるコアファイルの最大サイズ。	サイズ (バイト)
process.max-data-size	このプロセスで使用できるヒープメモリーの最大サイズ。	サイズ (バイト)
process.max-stack-size	このプロセスに使用できる最大スタックメモリーセグメント。	サイズ (バイト)
process.max-address-space	このプロセスで使用できる、セグメントサイズの総計としての最大アドレス空間。	サイズ (バイト)
process.max-port-events	イベントポートあたりに許容されるイベントの最大数。	数量 (イベント数)
process.max-sem-nsems	セマフォーセットあたりに許容されるセマフォーの最大数。	数量 (セットあたりのセマフォー数)
process.max-sem-ops	1 回の semop コールに許容されるセマフォー操作の最大数 (semget() のコール時にリソース制御からコピーされる値)。	数量 (操作の数)
process.max-msg-qbytes	メッセージキュー内のメッセージの最大バイト数 (msgget() のコール時にリソース制御からコピーされる値)。	サイズ (バイト)
process.max-msg-messages	メッセージキュー内のメッセージの最大数 (msgget() のコール時にリソース制御からコピーされる値)。	数量 (メッセージ数)

リソース制御の設定や変更がまったく行われていないシステム上では、リソース制御のデフォルト値を表示できます。そのようなシステムでは、/etc/system や project データベースにデフォルト以外のエントリが含まれていません。値を表示するには、prctl コマンドを使用します。

ゾーン規模のリソース制御

ゾーン規模のリソース制御は、ゾーン内のすべてのプロセスエンティティーによる総リソース消費を制限します。240 ページの「ゾーン規模のリソース制御の設定」および266 ページの「ゾーンの構成方法」で説明されているとおり、ゾーン規模のリソース制御は、グローバルプロパティー名を使用して設定することもできます。

表 6–2　ゾーン規模のリソース制御

制御名	説明	デフォルトの単位
zone.cpu-cap	**Solaris 10 5/08:** 1 つの非大域ゾーンで消費可能な CPU リソース量に対する絶対的な制限。project.cpu-cap 設定と同様、100 の値は 1 つの CPU の 100% を意味します。125 の値は 125% になります。CPU キャップの使用時は、100% がシステム上の 1 つの CPU の上限となります。	数量 (CPU の数)
zone.cpu-shares	このゾーンに対する公平配分スケジューラ (FSS) の CPU 配分	数量 (配分)
zone.max-locked-memory	ゾーンで使用できるロックされた物理メモリーの合計量。 priv_proc_lock_memory がゾーンに割り当てられている場合、そのゾーンがすべてのメモリーをロックするのを防ぐため、このリソース制御の設定も検討してください。	サイズ (バイト)
zone.max-lwps	このゾーンで同時に使用できる LWP の最大数	数量 (LWP 数)
zone.max-msg-ids	このゾーンに許容されるメッセージキュー ID の最大数	数量 (メッセージキュー ID の数)
zone.max-sem-ids	このゾーンに許容されるセマフォー ID の最大数	数量 (セマフォー ID の数)
zone.max-shm-ids	このゾーンに許容される共有メモリー ID の最大数	数量 (共有メモリー ID の数)
zone.max-shm-memory	このゾーンに許容される System V 共有メモリーの合計量	サイズ (バイト)
zone.max-swap	このゾーンのユーザープロセスのアドレス空間マッピングと tmpfs マウントで消費できるスワップの合計量。	サイズ (バイト)

ゾーン規模のリソース制御の構成方法については、252 ページの「リソースタイプのプロパティー」および 266 ページの「ゾーンの構成方法」を参照してください。lx ブランドゾーンでゾーン規模のリソース制御を使用する方法については、483 ページの「lx ブランドゾーンを構成、検証、および確定する方法」を参照してください。

ゾーン規模のリソース制御を大域ゾーンに適用することも可能です。詳細は、第 17 章「非大域ゾーンの構成 (概要)」および429 ページの「ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用」を参照してください。

単位のサポート

リソース制御の種類を示す大域フラグは、すべてのリソース制御に対して定義されます。これらのフラグは、種類に関する基本情報を prctl などのアプリケーションに伝えるために、システムによって使用されます。アプリケーションはこの情報を使用して、次の内容を判定します。

- 各リソース制御に適した単位の文字列
- 倍率値を解釈するときに使用する正しい倍率

次の大域フラグを使用できます。

大域フラグ	リソース制御の種類ごとの文字列	修飾子	倍率
RCTL_GLOBAL_BYTES	バイト	B	1
		KB	2^{10}
		MB	2^{20}
		GB	2^{30}
		TB	2^{40}
		PB	2^{50}
		EB	2^{60}
RCTL_GLOBAL_SECONDS	秒	s	1
		Ks	10^{3}
		Ms	10^{6}
		Gs	10^{9}
		Ts	10^{12}
		Ps	10^{15}
		Es	10^{18}

大域フラグ	リソース制御の種類ごとの文字列	修飾子	倍率
RCTL_GLOBAL_COUNT	数	なし	1
		K	10^3
		M	10^6
		G	10^9
		T	10^{12}
		P	10^{15}
		E	10^{18}

リソース制御に倍率値を使用できます。次の例は、倍率付きのしきい値を示します。

```
task.max-lwps=(priv,1K,deny)
```

注 – 単位修飾子は、`prctl`、`projadd`、および`projmod`コマンドに使用できます。`project`データベース自体で単位修飾子を使用することはできません。

リソース制御値と特権レベル

リソース制御のしきい値は、局所アクションのトリガーやロギングなどの大域アクションの発生が可能である実行ポイントを設定します。

リソース制御の各しきい値は、特権レベルに対応付ける必要があります。次の 3 種類の特権レベルのいずれかを使用します。

- 基本値 — 呼び出し元プロセスの所有者が変更できます
- 特権値 — 特権を持っている呼び出し元 (スーパーユーザー) だけが変更できます
- システム値 — オペレーティングシステムによる処理が実行されている間は、固定されます

リソース制御は、システムまたはリソースの提供者によって定義されるシステム値を 1 つ持つことが保証されます。システム値は、オペレーティングシステムが提供できるリソースの量を意味します。

特権値はいくつでも定義できます。基本値は 1 つだけ許可されます。特権値を指定しないで実行される操作には、デフォルトで、基本レベルの特権が割り当てられます。

リソース制御値の特権レベルは、リソース制御ブロックの特権フィールドで、RCTL_BASIC、RCTL_PRIVILEGED、または RCTL_SYSTEM のように定義します。詳細は、setrctl(2) のマニュアルページを参照してください。prctl コマンドを使用すると、基本レベルおよび特権レベルに対応付けられている値を変更できます。

リソース制御値に対応付けられた大域アクションと局所アクション

リソース制御値に対応付けられるアクションには、2 種類あります。 大域アクションと局所アクションです。

リソース制御値に対応付けられた大域アクション

大域アクションは、システム上のすべてのリソース制御のリソース制御値に適用されます。rctladm コマンド (rctladm(1M) のマニュアルページを参照) を使用すると、次の動作を実行できます。

- アクティブなシステムリソース制御の大域的状態を表示します
- 大域ログ作成アクションを設定します

リソース制御に対応付けられた大域ログ作成アクションは、無効にしたり有効にしたりできます。syslog アクションの程度を設定するには、重要度を syslog=*level* のように割り当てます。*level* に設定できる値は次のとおりです。

- debug
- info
- notice
- warning
- err
- crit
- alert
- emerg

デフォルトでは、リソース制御の違反は大域ログ作成では記録されません。Solaris 10 5/08 リリースでは、大域アクションを構成できないリソース制御に対して、レベル n/a が追加されました。

リソース制御値に対応付けられた局所アクション

局所アクションは、制御値を超えようとしているプロセスに対して実行されます。リソース制御に設定された各しきい値に対して、1 つ以上のアクションを対応付けることができます。局所アクションには、3 つの種類があります。 none、deny、および signal= です。これら 3 つのアクションは、次のように使用されます。

`none` しきい値を超える量のリソース要求に対して、何のアクションも行いません。このアクションは、アプリケーションの進行に影響を与えることなく、リソースの使用状況を監視するのに役立ちます。プロセスがリソース制御のしきい値を超えたときに大域メッセージを表示することもできます。ただし、このアクションによってプロセスが影響を受けることはありません。

`deny` しきい値を超える量のリソース要求を拒否できます。たとえば、`task.max-lwps` リソース制御に `deny` アクションが指定されている場合、制御値を超えるような新しいプロセスを作成する `fork` システムコールは失敗します。`fork(2)` のマニュアルページを参照してください。

`signal=` リソース制御値を超えたときに大域シグナルメッセージを送信するアクションを有効にすることができます。プロセスがしきい値を超えると、プロセスにシグナルが送信されます。プロセスがさらにリソースを消費しても、追加のシグナルが送信されることはありません。使用できるシグナルの一覧については、表 6–3 を参照してください。

すべてのリソース制御にすべてのアクションを適用できるわけではありません。たとえば、プロセスは、その所属先のプロジェクトに割り当てられている CPU 配分を超えることはできません。したがって、`project.cpu-shares` リソース制御に `deny` アクションを適用することはできません。

実装上の制限により、しきい値に設定できるアクションは、各制御の大域プロパティーによって制限されます。`rctladm(1M)` のマニュアルページを参照してください。次の表に、使用できるシグナルアクションを示します。シグナルの詳細については、`signal(3HEAD)` のマニュアルページを参照してください。

表 6–3 リソース制御値に使用できるシグナル

シグナル	説明	注意事項
SIGABRT	プロセスを終了します。	
SIGHUP	ハングアップシグナルを送信します。開いた回線上でキャリアが検出されなくなったときに発生します。シグナルは、端末を制御しているプロセスグループに送信されます。	
SIGTERM	プロセスを終了します。ソフトウェアによって送信される終了シグナルです。	
SIGKILL	プロセスを終了し、プログラムを強制終了します。	
SIGSTOP	プロセスを停止します。ジョブ制御シグナルです。	

表 6–3 リソース制御値に使用できるシグナル (続き)

シグナル	説明	注意事項
SIGXRES	リソース制御の制限超過です。リソース制御機能によって生成されます。	
SIGXFSZ	プロセスを終了します。ファイルサイズの制限超過です。	RCTL_GLOBAL_FILE_SIZE プロパティー (process.max-file-size) を持つリソース制御だけで使用可能です。詳細は、rctlblk_set_value(3C) のマニュアルページを参照してください。
SIGXCPU	プロセスを終了します。CPU 時間の制限超過です。	RCTL_GLOBAL_CPUTIME プロパティー (process.max-cpu-time) を持つリソース制御だけで使用可能です。詳細は、rctlblk_set_value(3C) のマニュアルページを参照してください。

リソース制御のフラグとプロパティー

システム上のリソース制御には、それぞれ特定のプロパティーセットが対応付けられています。このプロパティーセットは、一連のフラグとして定義されます。これらのフラグは、そのリソースが制御されているすべてのインスタンスに対応付けられます。大域フラグは変更できませんが、rctladm または getrctl システムコールを使って取得できます。

ローカルフラグは、特定のプロセスまたはプロセス集合に対するリソース制御の特定のしきい値について、デフォルトの動作と構成を定義します。あるしきい値のローカルフラグが、同じリソース制御で定義されている別のしきい値の動作に影響することはありません。ただし、大域フラグは、特定の制御に対応付けられているすべての値の動作に影響します。ローカルフラグは、対応する大域フラグによる制約の範囲内で、prctl コマンドまたは setrctl システムコールを使って変更できます。setrctl(2) を参照してください。

ローカルフラグ、大域フラグ、およびそれらの定義の詳細な一覧については、rctlblk_set_value(3C) のマニュアルページを参照してください。

特定のリソース制御がしきい値に達したときのシステムの動作を確認するには、rctladm を使ってそのリソース制御の大域フラグを表示します。たとえば、process.max-cpu-time の値を表示するには、次のように入力します。

```
$ rctladm process.max-cpu-time
    process.max-cpu-time  syslog=off  [ lowerable no-deny cpu-time inf seconds ]
```

大域フラグは、次のことを示します。

lowerable	この制御の特権値を下げるのに、スーパーユーザー特権を必要としません。
no-deny	しきい値を超えても、リソースへのアクセスは拒否されません。
cpu-time	リソースがしきい値に達したとき、SIGXCPU を送信できます。
seconds	リソース制御の時間。
no-basic	特権タイプ basic を持つリソース制御値を設定できません。特権付きリソース制御値だけが許可されます。
no-signal	リソース制御値に対してローカルのシグナルアクションを設定できません。
no-syslog	このリソース制御に対して大域の syslog メッセージアクションを設定できません。
deny	しきい値を超えたときに、必ずリソースの要求を拒否します。
count	リソース制御のカウント (整数) 値。
bytes	リソース制御のサイズの単位。

リソース制御のローカル値とアクションを表示するには、prctl コマンドを使用します。

```
$ prctl -n process.max-cpu-time $$
    process 353939: -ksh
    NAME    PRIVILEGE    VALUE    FLAG   ACTION              RECIPIENT
 process.max-cpu-time
        privileged   18.4Es    inf   signal=XCPU                 -
        system       18.4Es    inf   none
```

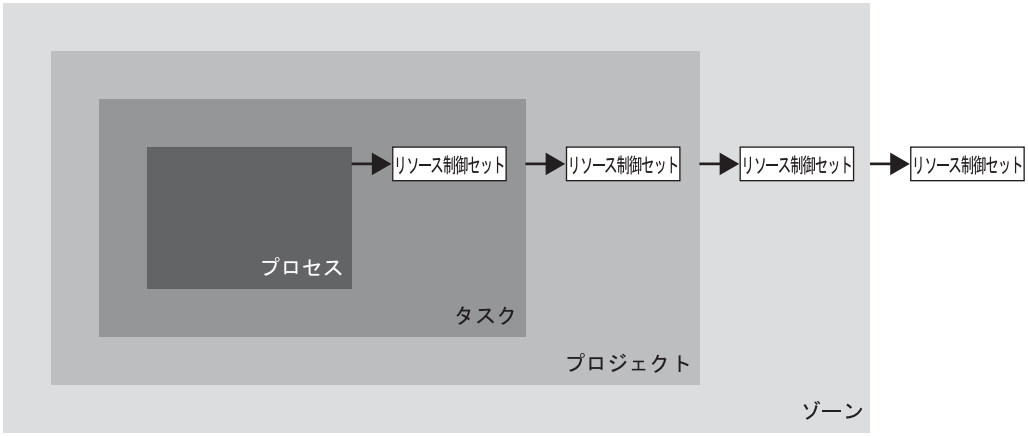
この例では、2 つのしきい値の両方に max (RCTL_LOCAL_MAXIMAL) フラグが設定されており、リソース制御には inf (RCTL_GLOBAL_INFINITE) フラグが設定されています。inf の値は無限大です。この値は制限を与えません。したがって、構成されているように、両方のしきい値は無限大値を意味し、これらの値を上回ることはありません。

リソース制御の実行

1 つのリソースには、複数のリソース制御を設定できます。リソース制御は、プロセスモデルの包含レベルごとに 1 つずつ設定できます。同じリソース上の異なるコンテナレベルでリソース制御がアクティブな場合、まず、もっとも小さいコンテナの

制御が実行されます。したがって、process.max-cpu-time と task.max-cpu-time の両方の制御が同時に検出された場合は、まず process.max-cpu-time に対するアクションが実行されます。

図 6–1 プロセス集合、コンテナの包含関係、およびそのリソース制御セット



リソース制御イベントの大域監視

プロセスのリソース消費が不明な場合がよくあります。リソース消費に関する詳細な情報を入手するには、rctladm コマンドで利用できる大域リソース制御アクションを使用してみてください。rctladm を使用して、リソース制御に syslog アクションを設定します。そのリソース制御が管理するエンティティーでしきい値が検出されると、構成したログレベルでシステムメッセージが記録されます。詳細は、第 7 章「リソース制御の管理 (タスク)」および rctladm(1M) のマニュアルページを参照してください。

リソース制御の適用

表 6–1 に示されている各リソース制御をプロジェクトに割り当てることができるのは、ログイン時、newtask または su が呼び出されたとき、あるいは、at、batch、cron など、プロジェクトを扱うことができる起動ツールが呼び出されたときです。開始される各コマンドは、呼び出し側のユーザーのデフォルトプロジェクトとは異なるタスクで起動されます。詳細は、login(1)、newtask(1)、at(1)、cron(1M)、および su(1M) のマニュアルページを参照してください。

project データベース内のエントリに対する更新は、/etc/project ファイルまたはネットワークネームサービスのデータベース表現のどちらに対するものであっても、現在アクティブなプロジェクトには適用されません。更新内容は、新しいタスクがログインまたは newtask によってプロジェクトに参加したときに適用されます。

動作中のシステム上のリソース制御値を一時的に更新する

project データベースで変更された値は、プロジェクト内で開始される新しいタスクに対してだけ有効になります。ただし、rctladm および prctl コマンドを使用すると、動作中のシステムのリソース制御を更新できます。

ログステータスの更新

rctladm コマンドは、システム全体で、各リソース制御の大域ログ状態に影響を与えます。このコマンドは、大域的状態を表示し、制御の限度を超えたときに syslog が記録するログのレベルを設定できます。

リソース制御の更新

prctl コマンドを使用すると、プロセスごと、タスクごと、またはプロジェクトごとにリソース制御値とアクションを表示したり、一時的に変更したりできます。プロジェクト ID、タスク ID、またはプロセス ID を入力として指定すると、このコマンドは、制御が定義されているレベルでリソース制御に対して動作します。

変更した値とアクションはすぐに適用されます。ただし、これらの変更が適用されるのは、現在のプロセス、タスク、またはプロジェクトだけです。変更内容は、project データベースには記録されません。システムを再起動すると、変更内容は失われます。リソース制御を永続的に変更するには、project データベースで変更を行う必要があります。

project データベースで変更できるリソース制御設定はすべて、prctl コマンドでも変更できます。基本値と特権値はどちらも、追加、削除が可能です。またそれらのアクションも変更できます。デフォルトでは、基本レベルのリソース制御はすべての操作の影響を受けます。スーパーユーザー特権があるプロセスとユーザーは、特権レベルのリソース制御も変更できます。システムリソースの制御は変更できません。

リソース制御で使用するコマンド

リソース制御で使用するコマンドを次の表に示します。

コマンドのリファレンス	説明
ipcs(1)	このコマンドを使用すると、どの IPC オブジェクトがプロジェクトの使用状況に影響を与えているかを監視できます
prctl(1)	このコマンドを使用すると、リソース制御機能の実行時に問い合わせやリソース制御機能の変更をローカルに行うことができます
rctladm(1M)	このコマンドを使用すると、リソース制御機能の実行時に問い合わせやリソース制御機能の変更を大域的に行うことができます

resource_controls(5) のマニュアルページでは、単位や倍率なども含め、プロジェクトデータベース経由で使用可能なリソース制御について説明しています。

◆ ◆ ◆ 第 7 章

7 リソース制御の管理 (タスク)

この章では、リソース制御機能を管理する方法について説明します。

リソース制御機能の概要については、第 6 章「リソース制御 (概要)」を参照してください。

リソース制御の管理 (タスクマップ)

タスク	説明	参照先
リソース制御を設定します。	`/etc/project` ファイルでプロジェクトにリソース制御を設定します。	102 ページの「リソース制御の設定」
アクティブなプロセス、タスク、またはプロジェクトについて、リソース制御値をローカルに取得または変更します。	システム上のアクティブなプロセス、タスク、またはプロジェクトに関連付けられているリソース制御に対し、実行時に問い合わせや変更を行います。	104 ページの「`prctl` コマンドの使用」
稼働中のシステムのリソース制御の大域的状態を表示または更新します。	システム全体の各リソース制御の大域ログ状態を表示します。また、制御値を超えたときに `syslog` で記録するログのレベルを設定します。	108 ページの「`rctladm` の使用」
アクティブなプロセス間通信 (IPC) 機能のステータスを報告します。	アクティブなプロセス間通信 (IPC) 機能について情報を表示します。どの IPC オブジェクトがプロジェクトの使用状況に影響を与えているかを監視します。	109 ページの「`ipcs` の使用」

タスク	説明	参照先
Web サーバーに十分な CPU 容量が割り当てられているかどうかを判定します。	リソース制御に大域アクションを設定します。このアクションを設定すると、リソース制御値を超えたエンティティーについて通知を受け取ることができ、リソース制御値が低すぎるかどうかを判定できます。	110 ページの「Web サーバーに十分な CPU 容量が割り当てられているかどうかを判定する方法」

リソース制御の設定

▼ プロジェクト内の各タスクの最大 LWP 数を設定する方法

この手順では、`x-files` というプロジェクトを `/etc/project` ファイルに追加し、このプロジェクト内に作成されるタスクに適用する LWP の最大数を設定します。

1. スーパーユーザーになるか、同等の役割を引き受けます。

   役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **projadd** コマンドに **-K** オプションを付けて実行して、**x-files** というプロジェクトを作成します。このプロジェクト内に作成される各タスクの **LWP** の最大数を **3** に設定します。

   ```
   # projadd -K 'task.max-lwps=(privileged,3,deny)' x-files
   ```

3. **/etc/project** ファイル内のエントリを表示します。それには、次のいずれかの方法を使用します。

   - 次のように入力します。

     ```
     # projects -l
     system
             projid : 0
             comment: ""
             users  : (none)
             groups : (none)
             attribs:
     .
     .
     .
     x-files
             projid : 100
             comment: ""
     ```

```
        users  : (none)
        groups : (none)
        attribs: task.max-lwps=(privileged,3,deny)
```

- 次のように入力します。

```
# cat /etc/project
system:0:System:::
.
.
.
x-files:100::::task.max-lwps=(privileged,3,deny)
```

例 **7–1** セッション例

上記の手順を実行したあと、プロジェクト `x-files` に `newtask` を使って参加することで新しいタスクを作成したスーパーユーザーは、そのタスクの実行中、LWP を 3 つまでしか作成できません。次の注釈付きのセッション例を参照してください。

```
# newtask -p x-files csh

# prctl -n task.max-lwps $$
process: 111107: csh
NAME    PRIVILEGE    VALUE    FLAG   ACTION            RECIPIENT
task.max-lwps
        privileged       3       -   deny                      -
        system       2.15G     max   deny                      -
# id -p
uid=0(root) gid=1(other) projid=100(x-files)

# ps -o project,taskid -p $$
 PROJECT TASKID
 x-files    73

# csh        /* creates second LWP */

# csh        /* creates third LWP */

# csh        /* cannot create more LWPs */
Vfork failed
#
```

▼ プロジェクトに複数の制御を設定する方法

`/etc/project` ファイルには、各プロジェクトごとに複数のリソース制御設定を記述でき、さらに各リソース制御ごとに複数のしきい値を記述できます。しきい値はアクション文節で定義されます。複数の値はコンマで区切られます。

1 スーパーユーザーになるか、同等の役割を引き受けます。

役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **projmod** コマンドに **-s** オプションと **-K** オプションを付けて実行することで、プロジェクト **x-files** にリソース制御を設定します。

```
# projmod -s -K 'task.max-lwps=(basic,10,none),(privileged,500,deny);
process.max-file-descriptor=(basic,128,deny)' x-files      one line in file
```

次の制御が設定されます。

- タスクごとの LWP 最大数について、アクションなしの basic 制御。
- タスクごとの LWP 最大数について、特権レベルの deny 制御。この制御により、102 ページの「プロジェクト内の各タスクの最大 LWP 数を設定する方法」の例のように、最大数を超える数の LWP を作成しようとすると失敗します。
- プロセスごとの最大ファイル記述子は basic レベルに制限されており、最大値を超える open コールはすべて失敗します。

3. 次のいずれかの方法で、ファイル内のエントリを表示します。
   - 次のように入力します。

```
# projects -l
.
.
.
x-files
        projid : 100
        comment: ""
        users  : (none)
        groups : (none)
        attribs: process.max-file-descriptor=(basic,128,deny)
                 task.max-lwps=(basic,10,none),(privileged,500,deny)      one line in file
```

   - 次のように入力します。

```
# cat etc/project
.
.
.
x-files:100::::process.max-file-descriptor=(basic,128,deny);
task.max-lwps=(basic,10,none),(privileged,500,deny)      one line in file
```

prctl コマンドの使用

prctl コマンドを使用すると、システム上のアクティブなプロセス、タスク、またはプロジェクトに関連付けられているリソース制御に対し、実行時に問い合わせや変更を行うことができます。詳細は、prctl(1) のマニュアルページを参照してください。

▼ prctl コマンドを使ってデフォルトのリソース制御値を表示する方法

この手順は、リソース制御の設定や変更がまったく行われていないシステム上で実行する必要があります。/etc/system ファイルまたは project データベース内には、デフォルト以外のエントリしか記述できません。

- 実行中の現在のシェルなど、任意のプロセスに対して **prctl** コマンドを実行します。

```
# prctl $$
process: 100337: -sh
NAME    PRIVILEGE       VALUE    FLAG   ACTION                       RECIPIENT
process.max-port-events
        privileged      65.5K       -   deny                                 -
        system          2.15G     max   deny                                 -
process.crypto-buffer-limit
        system          16.0EB    max   deny                                 -
process.max-crypto-sessions
        system          18.4E     max   deny                                 -
process.add-crypto-sessions
        privileged        100       -   deny                                 -
        system          18.4E     max   deny                                 -
process.min-crypto-sessions
        privileged         20       -   deny                                 -
        system          18.4E     max   deny                                 -
process.max-msg-messages
        privileged      8.19K       -   deny                                 -
        system          4.29G     max   deny                                 -
process.max-msg-qbytes
        privileged      64.0KB      -   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-sem-ops
        privileged        512       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-sem-nsems
        privileged        512       -   deny                                 -
        system          32.8K     max   deny                                 -
process.max-address-space
        privileged      16.0EB    max   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-file-descriptor
        basic             256       -   deny                            100337
        privileged      65.5K       -   deny                                 -
        system          2.15G     max   deny                                 -
process.max-core-size
        privileged      8.00EB    max   deny                                 -
        system          8.00EB    max   deny                                 -
process.max-stack-size
        basic           8.00MB      -   deny                            100337
        privileged      8.00EB      -   deny                                 -
        system          8.00EB    max   deny                                 -
process.max-data-size
        privileged      16.0EB    max   deny                                 -
        system          16.0EB    max   deny                                 -
process.max-file-size
        privileged      8.00EB    max   deny,signal=XFSZ                     -
```

```
        system          8.00EB    max   deny                                 -
process.max-cpu-time
        privileged      18.4Es    inf   signal=XCPU                          -
        system          18.4Es    inf   none                                 -
task.max-cpu-time
        system          18.4Es    inf   none                                 -
task.max-lwps
        system          2.15G     max   deny                                 -
project.max-contracts
        privileged      10.0K       -   deny                                 -
        system          2.15G     max   deny                                 -
project.max-device-locked-memory
        privileged       499MB      -   deny                                 -
        system          16.0EB    max   deny                                 -
project.max-port-ids
        privileged      8.19K       -   deny                                 -
        system          65.5K     max   deny                                 -
project.max-shm-memory
        privileged      1.95GB      -   deny                                 -
        system          16.0EB    max   deny                                 -
project.max-shm-ids
        privileged        128       -   deny                                 -
        system          16.8M     max   deny                                 -
project.max-msg-ids
        privileged        128       -   deny                                 -
        system          16.8M     max   deny                                 -
project.max-sem-ids
        privileged        128       -   deny                                 -
        system          16.8M     max   deny                                 -
project.max-tasks
        system          2.15G     max   deny                                 -
project.max-lwps
        system          2.15G     max   deny                                 -
project.cpu-shares
        privileged          1       -   none                                 -
        system          65.5K     max   none                                 -
zone.max-lwps
        system          2.15G     max   deny                                 -
zone.cpu-shares
        privileged          1       -   none                                 -
        system          65.5K     max   none                                 -
```

▼ prctl コマンドを使って特定のリソース制御の情報を表示する方法

- 実行中の現在のシェルの最大ファイル記述子を表示します。

```
# prctl -n process.max-file-descriptor $$
process: 110453: -sh
NAME    PRIVILEGE       VALUE    FLAG   ACTION       RECIPIENT
process.max-file-descriptor
        basic             256       -   deny             110453
        privileged      65.5K       -   deny                  -
        system          2.15G     max   deny
```

▼ prctl を使って値を一時的に変更する方法

次の手順では、prctl コマンドを使用して x-files プロジェクトに新しい特権値を一時的に追加し、プロジェクトあたり 4 つ以上の LWP を使用することを拒否します。その結果は、102 ページの「プロジェクト内の各タスクの最大 LWP 数を設定する方法」の結果と同等になります。

1 スーパーユーザーになるか、同等の役割を引き受けます。

役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **newtask** を使って **x-files** プロジェクトに参加します。

```
# newtask -p x-files
```

3 **id** コマンドに **-p** オプションを付けて実行し、正しいプロジェクトに参加できたことを確認します。

```
# id -p
uid=0(root) gid=1(other) projid=101(x-files)
```

4 **project.max-lwps** に新しい特権値を追加して、**LWP** の数を **3** つまでに制限します。

```
# prctl -n project.max-lwps -t privileged -v 3 -e deny -i project x-files
```

5 結果を確認します。

```
# prctl -n project.max-lwps -i project x-files
process: 111108: csh
NAME    PRIVILEGE    VALUE    FLAG   ACTION            RECIPIENT
project.max-lwps
        privileged       3       -   deny                      -
        system       2.15G     max   deny                      -
```

▼ prctl を使ってリソース制御値を下げる方法

1 スーパーユーザーになるか、同等の役割を引き受けます。

役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **prctl** コマンドに **-r** オプションを付けて実行し、**process.max-file-descriptor** リソース制御の最小値を変更します。

```
# prctl -n process.max-file-descriptor -r -v 128 $$
```

▼ prctl を使ってプロジェクトの制御値を表示、置換、確認する方法

1 スーパーユーザーになるか、同等の役割を引き受けます。

役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 プロジェクト *group.staff* の **project.cpu-shares** の値を表示します。

```
# prctl -n project.cpu-shares -i project group.staff
project: 2: group.staff
NAME    PRIVILEGE       VALUE    FLAG   ACTION     RECIPIENT
project.cpu-shares
        privileged          1       -   none               -
        system          65.5K     max   none
```

3 **project.cpu-shares** の現在の値 **1** を値 **10** で置換します。

```
# prctl -n project.cpu-shares -v 10 -r -i project group.staff
```

4 プロジェクト **group.staff** の **project.cpu-shares** の値を表示します。

```
# prctl -n project.cpu-shares -i project group.staff
project: 2: group.staff
NAME    PRIVILEGE       VALUE    FLAG   ACTION     RECIPIENT
project.cpu-shares
        privileged         10       -   none               -
        system          65.5K     max   none
```

rctladm の使用

rctladm を使用する方法

rctladm コマンドを使用すると、リソース制御機能の大域的状態を実行時に問い合わせたり変更したりできます。詳細は、rctladm(1M) のマニュアルページを参照してください。

たとえば、rctladm に -e オプションを付けて実行して、リソース制御の大域属性 syslog を有効にすることができます。制御が限界を超えたとき、指定された syslog レベルで通知が記録されます。process.max-file-descriptor の大域属性 syslog を有効にするには、次のように入力します。

```
# rctladm -e syslog process.max-file-descriptor
```

rctladm コマンドを引数なしで使用すると、各リソース制御の大域フラグが大域タイプフラグも含めて表示されます。

```
# rctladm
process.max-port-events     syslog=off  [ deny count ]
process.max-msg-messages    syslog=off  [ deny count ]
process.max-msg-qbytes      syslog=off  [ deny bytes ]
process.max-sem-ops         syslog=off  [ deny count ]
process.max-sem-nsems       syslog=off  [ deny count ]
process.max-address-space   syslog=off  [ lowerable deny no-signal bytes ]
process.max-file-descriptor syslog=off  [ lowerable deny count ]
process.max-core-size       syslog=off  [ lowerable deny no-signal bytes ]
process.max-stack-size      syslog=off  [ lowerable deny no-signal bytes ]
.
.
.
```

ipcs の使用

ipcs を使用する方法

ipcs ユーティリティーを使用すると、アクティブなプロセス間通信 (IPC) 機能について情報を表示できます。詳細は、ipcs(1) のマニュアルページを参照してください。

ipcs に -J オプションを付けて実行すると、どのプロジェクトの制限に対して IPC オブジェクトが割り当てられているかを確認できます。

```
# ipcs -J
    IPC status from <running system> as of Wed Mar 26 18:53:15 PDT 2003
T         ID      KEY        MODE       OWNER    GROUP    PROJECT
Message Queues:
Shared Memory:
m       3600      0       --rw-rw-rw-   uname    staff    x-files
m        201      0       --rw-rw-rw-   uname    staff    x-files
m       1802      0       --rw-rw-rw-   uname    staff    x-files
m        503      0       --rw-rw-rw-   uname    staff    x-files
m        304      0       --rw-rw-rw-   uname    staff    x-files
m        605      0       --rw-rw-rw-   uname    staff    x-files
m          6      0       --rw-rw-rw-   uname    staff    x-files
m        107      0       --rw-rw-rw-   uname    staff    x-files
Semaphores:
s          0      0       --rw-rw-rw-   uname    staff    x-files
```

容量に関する警告

リソース制御に対して大域アクションを設定すると、リソース制御値を超えたエンティティーに関する通知を受け取ることができ、リソース制御値が低すぎるかどうかを判断できます。

たとえば、一般的な作業負荷のための十分な CPU リソースが Web サーバーに割り当てられているかどうかを確認する場合を考えます。この容量は、sar データで CPU

のアイドル時間と平均負荷率を解析すれば判定できます。また、拡張アカウンティングデータを調べて、Web サーバープロセスで同時に実行しているプロセス数を確認することもできます。

より簡単な方法は、Web サーバーをタスクに配置することです。その上で、syslog を使って大域アクションを設定すると、タスクがマシンの性能に適した LWP の計画数を上回ったときに、警告が通知されます。

詳細は、sar(1) のマニュアルページを参照してください。

▼ Web サーバーに十分な CPU 容量が割り当てられているかどうかを判定する方法

1. **prctl** コマンドを使用して、**httpd** プロセスを含むタスクにスーパーユーザーが所有する特権レベルのリソース制御を設定します。各タスクの **LWP** の総数を **40** に制限し、すべての局所アクションを無効にします。

   ```
   # prctl -n task.max-lwps -v 40 -t privileged -d all ‘pgrep httpd‘
   ```

2. リソース制御 **task.max-lwps** で、システムログの大域アクションを有効にします。

   ```
   # rctladm -e syslog task.max-lwps
   ```

3. 作業負荷がリソース制御を超えるかどうかを監視します。

   作業負荷がリソース制御を超えると、次のような内容が /var/adm/messages に記録されます。

   ```
   Jan  8 10:15:15 testmachine unix: [ID 859581 kern.notice]
   NOTICE: privileged rctl task.max-lwps exceeded by task 19
   ```

◆ ◆ ◆ 第 8 章

公平配分スケジューラ (概要)

作業負荷データを解析することによって、特定の作業負荷または作業負荷のグループが CPU リソースを占有しているかどうかを判定できます。作業負荷が CPU 使用量の制限を超えていない場合は、システム上での CPU 時間の割り当て方針を変更することができます。この章で説明する公平配分スケジューリングクラスを使用すると、タイムシェアリング (TS) スケジューリングクラスの優先順位スキームではなく、配分に基づいて CPU 時間を割り当てることができます。

この章の内容は次のとおりです。

公平配分スケジューラの使用を開始する方法については、第 9 章「公平配分スケジューラの管理 (タスク)」を参照してください。

スケジューラの紹介

オペレーティングシステムの基本的な仕事は、どのプロセスがシステムリソースへのアクセスを取得できるようにするか調整することです。プロセススケジューラ (別名、ディスパッチャー) は、カーネルの一部であり、プロセスへの CPU の割り当てを制御します。スケジューラには、スケジューリングクラスという概念があります。各スケジューリングクラスでは、クラス内のプロセスのスケジューリングに使用するスケジューリング方針を定義します。TS スケジューラは Solaris オペレーティングシステムにおけるデフォルトのスケジューラであり、使用可能な CPU へのアクセスをすべてのプロセスに相対的に等しく与えます。ただし、特定のプロセスにより多くのリソースを与えたい場合もあります。

公平配分スケジューラ (FSS) では、各作業負荷に対する使用可能な CPU リソースの割り当てを、その作業負荷の重要性に基づいて制御します。この重要性は、各作業負荷に割り当てる CPU リソースの「配分」で表します。

各プロジェクトに CPU 配分を与えて、CPU リソースに対するプロジェクトの使用権を制御します。FSS では、プロジェクトに属するプロセス数ではなく、割り当てられた配分に基づいて、プロジェクト間に CPU リソースが公平に配分されることが保証されています。FSS は、ほかのプロジェクトとの比較に基づいて、CPU リソースを多く使用するプロジェクトの CPU 使用権を減らし、CPU リソースの使用が少ないプロジェクトの CPU 使用権を増やすことで公平さを実現します。

FSS は、カーネルスケジューリングクラスモジュールとクラス固有のバージョンの `dispadmin(1M)` および `priocntl(1)` コマンドから構成されます。FSS が使用するプロジェクト配分は、`project(4)` データベース内の `project.cpu-shares` プロパティーで指定します。

注 – ゾーンがインストールされているシステムで `project.cpu-shares` リソース制御を使用する場合は、247 ページの「ゾーン構成データ」、395 ページの「非大域ゾーンで使用されるリソース制御」、および429 ページの「ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用」を参照してください。

CPU 配分の定義

「配分」という用語は、プロジェクトに割り当てられる CPU リソースの配分を定義するために使用されます。プロジェクトに割り当てる CPU 配分をほかのプロジェクトよりも多くすると、そのプロジェクトが公平配分スケジューラから受け取る CPU リソースも多くなります。

CPU 配分は、CPU リソースの比率ではありません。配分は、ほかの作業負荷との比較に基づいた作業負荷の相対的な重要性を定義します。プロジェクトに CPU 配分を割り当てる場合に重要なことは、プロジェクトが持つ配分自体ではありません。ほ

かのプロジェクトと比較して、そのプロジェクトが配分をいくつ持っているかを把握することが重要です。また、そのプロジェクトが CPU リソースについて、ほかのいくつのプロジェクトと競合しているかということも考慮に入れる必要があります。

注 – 配分がゼロのプロジェクトに属するプロセスは、常に最下位のシステム優先順位 (0) で実行されます。このようなプロセスが実行されるのは、配分がゼロでないプロジェクトが CPU リソースを使用していないときだけです。

CPU 配分とプロセスの状態

Solaris システムでは、プロジェクトの作業負荷は一般に複数のプロセスから構成されます。公平配分スケジューラの観点からは、各プロジェクトの作業負荷は、「アイドル」状態か「アクティブ」状態のどちらかです。プロジェクトのどのプロセスも CPU リソースを使用していないとき、プロジェクトはアイドル状態であるといいます。このような場合、プロセスは一般にスリープ (入出力の完了を待機している状態) または停止状態にあります。プロジェクトの 1 つ以上のプロセスが CPU リソースを使用しているとき、プロジェクトはアクティブ状態であるといいます。すべてのアクティブなプロジェクトが持つ配分の合計が、プロジェクトに割り当てられる CPU リソースの配分の計算に使用されます。

アクティブなプロジェクトが増えると、各プロジェクトの CPU 割り当ては減りますが、プロジェクト間の CPU 割り当て比率は変わりません。

CPU 配分と使用効率

配分割り当ては、使用効率とは異なります。CPU リソースの 50% が割り当てられているプロジェクトの CPU 使用率は、平均するとわずか 20% ほどです。その上、配分が CPU 使用量を制限するのは、ほかのプロジェクトと競合するときだけです。プロジェクトに対する割り当てが低い場合でも、そのプロジェクトがシステムで単独に実行されているときは、常に 100% の処理能力を CPU から受け取ります。使用可能な CPU サイクルが浪費されることはありません。つまり、使用可能な CPU サイクルはプロジェクト間に配分されます。

動作中の作業負荷に小さい配分を割り当てると、性能が低下します。ただし、システムが過負荷にならないかぎり、配分割り当て数が原因で作業が完了しないことはありません。

CPU 配分の例

2 つの CPU を搭載したシステムがあり、それらの CPU は CPU にバインドされた 2 つの作業負荷 *A* および *B* を並列に実行しているとします。各作業負荷は別個のプロジェクトとして実行されています。各プロジェクトは、プロジェクト *A* に S_A 配分が割り当てられ、プロジェクト *B* に S_B 配分が割り当てられるように構成されています。

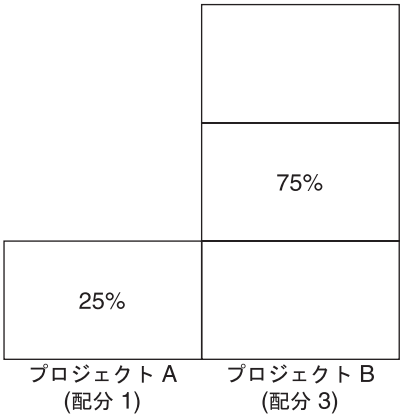
従来の TS スケジューラを使用した場合、システムで実行されている各作業負荷には、平均して同じ量の CPU リソースが与えられます。つまり、各作業負荷にはシステム容量の 50% が割り当てられます。

FSS スケジューラの制御で実行する場合でも、$S_A = S_B$ の配分を割り当てると、各プロジェクトにほぼ等量の CPU リソースが与えられます。これに対して、プロジェクトに異なる配分を与えた場合、CPU リソースの割り当て量は異なります。

次に示す 3 つの例は、さまざまな構成での配分の働きを示しています。これらの例に示されているとおり、配分は、要求が使用可能なリソース量と同じまたはそれを超えている場合にのみ使用量を数学的に正確に表します。

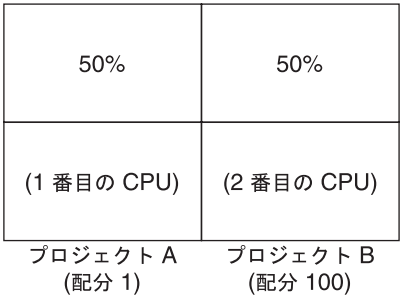
例 1: CPU にバインドされた 2 つのプロセスが各プロジェクトに存在する場合

プロジェクト *A* と *B* がそれぞれ CPU に結合されたプロセスを 2 つ持ち、かつ $S_A = 1$、$S_B = 3$ である場合、配分の合計数は 1 + 3 = 4 になります。この構成で、十分な数の CPU 要求があると、*A* と *B* には、それぞれ CPU リソースの 25%、75% が割り当てられます。



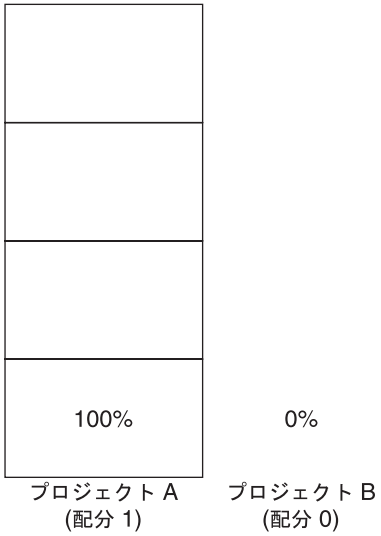
例 2: プロジェクト間に競合がない場合

プロジェクト *A* と *B* がそれぞれ CPU に結合されたプロセスを「1 つ」だけ持ち、かつ $S_A = 1$、$S_B = 100$ である場合、配分の合計数は 101 になります。各プロジェクトは、実行中のプロセスを 1 つしか持たないため、CPU を 1 つしか使用できません。この構成では、CPU リソースを得るための競合がプロジェクト間に存在しないので、プロジェクト *A* および *B* には、それぞれ全 CPU リソースの 50% が割り当てられます。この構成の場合、CPU 配分は CPU リソースの割り当てに影響しません。プロジェクトへの割り当ては同じ (50/50) になります。これは、両方のプロジェクトに割り当てられる配分がゼロの場合でも同様です。



例 3: 一方のプロジェクトが実行されない場合

プロジェクト *A* と *B* がそれぞれ CPU に結合されたプロセスを 2 つ持ち、かつ *A* に 1 配分、*B* に 0 配分が与えられている場合、プロジェクト *B* には CPU リソースがまったく割り当てられず、プロジェクト *A* にすべての CPU リソースが割り当てられます。プロジェクト *B* のプロセスは常にシステム優先順位 0 で実行されるため、実行される可能性はまったくありません。これは、プロジェクト *A* のプロセスの方が常に高い優先順位を持っているためです。



FSS の構成

プロジェクトとユーザー

プロジェクトは、FSS スケジューラの作業負荷コンテナです。プロジェクトに割り当てられているユーザーのグループは、個別の管理可能なブロックとして扱われます。個人ユーザー用に独自の配分を持つプロジェクトを作成できます。

ユーザーは、異なる配分が割り当てられているさまざまなプロジェクトのメンバーになることができます。プロセスをあるプロジェクトから別のプロジェクトに移動すると、プロセスに割り当てられる CPU リソース量は変化します。

project(4) データベースとネームサービスの詳細については、46 ページの「project データベース」を参照してください。

CPU 配分の構成

CPU 配分の構成は project データベースのプロパティーとして、ネームサービスによって管理されます。

プロジェクトに関連付けられている最初のタスクまたはプロセスが setproject(3PROJECT) ライブラリ関数を使って作成されると、project データベース内でリソース制御 project.cpu-shares として定義されている CPU 配分がカーネルに渡されます。リソース制御 project.cpu-shares が定義されていないプロジェクトには、1 配分が割り当てられます。

次の例では、/etc/project ファイル内のこのエントリでプロジェクト *x-files* の配分に *5* が設定されています。

```
x-files:100::::project.cpu-shares=(privileged,5,none)
```

プロジェクトに割り当てられている CPU 配分を、プロセスの実行中にデータベースで変更しても、プロジェクトの配分は、その時点では変更されません。変更内容を有効にするには、プロジェクトを再起動する必要があります。

project データベース内のプロジェクトの属性を変更することなくプロジェクトに割り当てられている配分を一時的に変更するには、prctl コマンドを使用します。たとえば、*x-files* プロジェクトに関連付けられているプロセスの実行中に、そのプロジェクトのリソース制御 project.cpu-shares の値を *3* に変更するには、次のように入力します。

```
# prctl -r -n project.cpu-shares -v 3 -i project x-files
```

詳細は、prctl(1) のマニュアルページを参照してください。

-r　　指定されたリソース制御の現在の値を置き換えます。

-n *name*　　リソース制御の名前を指定します。

-v *val*　　リソース制御の値を指定します。

-i *idtype*　　ID タイプを指定します。

x-files　　変更対象を指定します。この例では、プロジェクト *x-files* が変更対象です。

プロジェクト ID 0 のプロジェクト system には、起動時の初期化スクリプトで起動されるすべてのシステムデーモンが含まれます。system は、無制限の配分を持つプロジェクトとしてみなされます。したがって、プロジェクト system は、ほかのプロジェクトに与えられている配分とは関係なく、常に最初にスケジュールされます。プロジェクト system に無制限の配分を割り当てない場合は、project データベースでこのプロジェクトの配分を変更します。

前述のように、配分がゼロのプロジェクトに属するプロセスには、常にシステム優先順位 0 が与えられます。配分が 1 以上のプロジェクトは、優先順位 1 以上で実行されます。したがって、配分がゼロのプロジェクトは、ゼロ以外の配分を持つプロジェクトが CPU リソースを要求していないときにだけスケジュールされます。

1 つのプロジェクトに割り当てられる配分の最大数は 65535 です。

FSS とプロセッサセット

プロセッサセットに FSS を連携して使用すると、連携させない場合よりも、各プロセッサセット上で実行するプロジェクト間の CPU リソースの割り当てをよりきめ細かく制御できます。FSS スケジューラは、プロセッサセットを完全に独立したパーティションとして処理します。つまり、各プロセッサセットは、CPU 割り当てについてそれぞれ個別に制御されます。

1 つのプロセッサセットで実行されるプロジェクトの CPU 割り当てが、別のプロセッサセットで実行されるプロジェクトの CPU 配分や動作によって影響を受けることはありません。なぜなら、異なるプロセッサセットで実行されるプロジェクトが同じリソースについて競合することはないからです。競合が発生するのは、プロジェクトが同じプロセッサセット内で実行されている場合だけです。

プロジェクトに割り当てられている配分はシステム全体に適用されます。どのプロセッサセットで実行されようと、プロジェクトの各部分には同じ配分が与えられます。

プロセッサセットが使用されている場合、プロジェクトの CPU 割り当ては、各プロセッサセット内で実行されるアクティブなプロジェクトに対して算出されます。

異なるプロセッサセット内で実行されるプロジェクトのパーティションは、異なる CPU 割り当てを持つことになります。1 つのプロセッサセット内の各プロジェクトパーティションに対する CPU 割り当ては、同じプロセッサセット内で実行されるほかのプロジェクトの割り当てにだけ依存します。

プロセッサセット境界内で実行されるアプリケーションの性能と可用性が、新しいプロセッサセットの導入によって影響を受けることはありません。また、ほかのプロセッサセットで実行されるプロジェクトの配分割り当ての変更によって、アプリケーションが影響を受けることもありません。

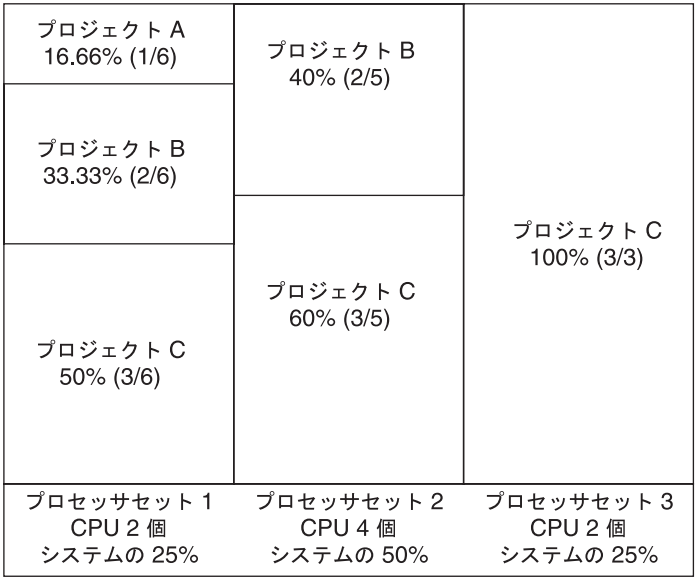
空のプロセッサセット (プロセッサが存在しないセット) や、プロセッサセットにバインドされたプロセスを持たないプロセッサセットは、FSS スケジューラの動作にまったく影響を与えません。

FSS とプロセッサセットの例

8 つの CPU を持つサーバーがプロジェクト *A*、*B*、および *C* 内で CPU にバインドされたアプリケーションをいくつか実行しているものとします。プロジェクト *A* には 1 配分、プロジェクト *B* には 2 配分、プロジェクト *C* には 3 配分がそれぞれ割り当てられています。

プロジェクト *A* は、プロセッサセット 1 だけで実行されています。プロジェクト *B* は、プロセッサセット 1 および 2 で実行されています。プロジェクト *C* は、プロ

セッサセット 1、2、および 3 で実行されています。各プロジェクトには、使用可能なすべての CPU 処理能力を利用するだけの十分な数のプロセスが存在しているものとします。したがって、CPU リソースを得るための競合が各プロセッサセットで常に発生します。



このようなシステムでは、システム全体でのプロジェクトの CPU 割り当ての合計は次のようになります。(pset = プロセッサセット)

プロジェクト	割り当て
プロジェクト A	$4\% = (1/6 \text{ X } 2/8)_{pset1}$
プロジェクト B	$28\% = (2/6 \text{ X } 2/8)_{pset1} + (2/5 * 4/8)_{pset2}$
プロジェクト C	$67\% = (3/6 \text{ X } 2/8)_{pset1} + (3/5 \text{ X } 4/8)_{pset2} + (3/3 \text{ X } 2/8)_{pset3}$

これらの割合は、プロジェクトに与えられている CPU 配分値とは一致しません。ただし、各プロセッサセット内では、プロジェクトごとの CPU 割り当て比率はプロジェクトのそれぞれの配分に比例します。

このシステム上にプロセッサセットが存在しない場合、CPU リソースの配分は、次に示すように、異なったものになります。

プロジェクト	割り当て
プロジェクト A	16.66% = (1/6)
プロジェクト B	33.33% = (2/6)
プロジェクト C	50% = (3/6)

FSS とほかのスケジューリングクラスの併用

デフォルトでは、FSS スケジューリングクラスは、タイムシェアリング (TS)、対話型 (IA)、および固定優先順位 (FX) の各スケジューリングクラスと同じ範囲の優先順位 (0 から 59) を使用します。そのため、これらのスケジューリングクラスのプロセスが同じプロセッサセットを共有しないようにする必要があります。FSS、TS、IA、および FX の各クラスにプロセスが混在すると、予期せぬスケジューリング処理が実行される場合があります。

プロセッサセットを使用する場合は、1 つのシステム内で TS、IA、および FX を FSS と混在させることができます。ただし、各プロセッサセットで実行されるすべてのプロセスは、「1 つの」スケジューリングクラスに所属している必要があります。このようにすると、これらのプロセスが同じ CPU について競合することはありません。プロセッサセットを使用しない場合は、特に FX スケジューラを FSS スケジューリングクラスと併用しないようにしてください。これにより、FX クラスのアプリケーションが高い優先順位を使用して、FSS クラスのアプリケーションの実行を妨げることはありません。

TS クラスと IA クラスのプロセスは、同じプロセッサセット内で、またはプロセッサセットが存在しない同じシステム内で混在させることができます。

Solaris システムでは、スーパーユーザー権限を持つユーザーは、リアルタイム (RT) スケジューラも利用できます。デフォルトでは、RT スケジューリングクラスは FSS とは異なる範囲のシステム優先順位 (通常は 100 から 159) を使用します。RT と FSS は「互いに素」な範囲 (重複しない範囲) の優先順位を使用しているので、FSS は同じプロセッサセット内の RT スケジューリングクラスと共存できます。ただし、FSS スケジューリングクラスは、RT クラスで実行するプロセスを制御することはできません。

たとえば、4 つのプロセッサから構成されるシステムで、CPU に結合されているシングルスレッドの RT プロセスは 1 つのプロセッサを占有できます。システムが FSS も実行している場合、通常のユーザープロセスは、RT プロセスが使用していない残りの 3 つの CPU について競合します。RT プロセスは CPU を使い続けることはありません。RT プロセスがアイドル状態になったとき、FSS は 4 つのプロセッサをすべて使用します。

次のコマンドを入力して、プロセッサセットが実行しているスケジューリングクラスを特定し、各プロセッサセットが TS、IA、FX、または FSS のプロセスのいずれかを実行するように構成されていることを確認します。

```
$ ps -ef -o pset,class | grep -v CLS | sort | uniq
1 FSS
1 SYS
2 TS
2 RT
3 FX
```

システムのスケジューリングクラスの設定

システムにデフォルトのスケジューリングクラスを設定するには、125 ページの「FSS をデフォルトのスケジューラクラスにする方法」、235 ページの「ゾーンのスケジューリングクラス」、および dispadmin(1M) を参照してください。実行中のプロセスを別のスケジューリングクラスに移動するには、125 ページの「FSS の構成」と priocntl(1) を参照してください。

ゾーンがインストールされているシステムでのスケジューリングクラス

非大域ゾーンでは、システムのデフォルトのスケジューリングクラスが使用されます。システムのデフォルトスケジューリングクラスの設定が更新された場合、非大域ゾーンでは、ブート時またはリブート時にこの新しい設定が取得されます。

この場合に望ましい FSS の使用方法は、dispadmin コマンドを使用して、FSS をシステムのデフォルトのスケジューリングクラスに設定する方法です。このようにすると、すべてのゾーンがシステムの CPU リソースの公平配分を受けることができます。ゾーンが使用されている場合のスケジューリングクラスの詳細については、235 ページの「ゾーンのスケジューリングクラス」を参照してください。

デフォルトのスケジューリングクラスの変更やリブートを行うことなく、実行中のプロセスを別のスケジューリングクラスに移動する方法については、表 27–5 および priocntl(1) のマニュアルページを参照してください。

FSS で使用するコマンド

次の表に示すコマンドは、公平配分スケジューラを管理するためのプライマリ管理インタフェースとなります。

コマンドのリファレンス	説明
priocntl(1)	指定されたプロセスのスケジューリングパラメータを表示または設定します。実行中のプロセスを別のスケジューリングクラスに移動します。

コマンドのリファレンス	説明
ps(1)	実行中のプロセスに関する情報を一覧表示します。プロセッサセットがどのスケジューリングクラスで実行されているかを示します。
dispadmin(1M)	システムのデフォルトのスケジューラを設定します。FSS スケジューラのタイムクォンタム (time quantum) 値を調べ、調整する場合にも使用されます。
FSS(7)	公平配分スケジューラ (FSS) を記述します。

◆ ◆ ◆ 第 9 章

9 公平配分スケジューラの管理 (タスク)

この章では、公平配分スケジューラ (FSS) の使用方法について説明します。

FSS の概要については、第 8 章「公平配分スケジューラ (概要)」を参照してください。ゾーンが使用されている場合のスケジューリングクラスの詳細については、235 ページの「ゾーンのスケジューリングクラス」を参照してください。

公平配分スケジューラの管理 (タスクマップ)

タスク	説明	参照先
CPU 使用量を監視します。	プロジェクトの CPU 使用量およびプロセッサセット内のプロジェクトの CPU 使用量を監視します。	124 ページの「FSS の監視」
デフォルトのスケジューラクラスを設定します。	FSS などのスケジューラをシステムのデフォルトスケジューラとして設定します。	125 ページの「FSS をデフォルトのスケジューラクラスにする方法」
実行中のプロセスを、あるスケジューリングクラスから FSS クラスなどの別のスケジューリングクラスに移動します。	デフォルトのスケジューリングクラスの変更やリブートを行うことなく、あるスケジューリングクラスから別のスケジューリングクラスにプロセスを手動で移動します。	125 ページの「プロセスを TS クラスから FSS クラスに手動で移動する方法」
すべての実行中のプロセスを、FSS クラスなどの別のスケジューリングクラスに移動します。	デフォルトのスケジューリングクラスの変更やリブートを行うことなく、すべてのスケジューリングクラスから別のスケジューリングクラスにプロセスを手動で移動します。	126 ページの「プロセスをすべてのユーザークラスから FSS クラスに手動で移動する方法」

タスク	説明	参照先
プロジェクトのプロセスを、FSS クラスなどの別のスケジューリングクラスに移動します。	プロジェクトのプロセスを、現在のスケジューリングクラスから別のスケジューリングクラスに手動で移動します。	127 ページの「プロジェクトのプロセスを FSS クラスに手動で移動する方法」
FSS パラメータを調べ、調整します。	スケジューラのタイムクォンタムの値を調整します。「タイムクォンタム」とは、スレッドがプロセッサ上で実行を開始してからそのプロセッサを放棄するまでの時間量のことです。	127 ページの「スケジューラのパラメータを調整する方法」

FSS の監視

prstat コマンド (prstat(1M) のマニュアルページを参照) を使用すると、アクティブなプロジェクトごとの CPU 使用量を監視できます。

タスク用の拡張アカウンティングデータを使用して、長期間使用される CPU リソースの合計量について、プロジェクトごとの統計情報を取得できます。詳細は、第 4 章「拡張アカウンティング (概要)」を参照してください。

▼ システムの CPU 使用量をプロジェクトごとに監視する方法

- システム上で実行されているプロジェクトの CPU 使用量を監視するには、**prstat** コマンドに **-J** オプションを付けて実行します。

```
% prstat -J
```

▼ プロセッサセット内の CPU 使用量をプロジェクトごとに監視する方法

- 1 つまたは複数のプロセッサセットについて、プロジェクトの CPU 使用量を監視するには、次のように入力します。

```
% prstat -J -C pset-list
```

ここで *pset-list* は、プロセッサセット ID をコンマで区切って並べたリストです。

FSS の構成

Solaris システムのほかのスケジューリングクラスで使用するものと同じコマンドを、FSS でも使用できます。スケジューラクラス、スケジューラの調整可能パラメータ、および個々のプロセスのプロパティーを構成できます。

`svcadm restart` を使用すると、スケジューラサービスを再起動することができます。詳細は、`svcadm(1M)` のマニュアルページを参照してください。

▼ FSS をデフォルトのスケジューラクラスにする方法

CPU 配分割り当てを有効にするには、FSS をシステムのデフォルトのスケジューラにする必要があります。

`priocntl` と `dispadmin` コマンドを組み合わせて使用することにより、FSS はただちにデフォルトのスケジューラになり、この設定はリブート後も有効です。

1. スーパーユーザーになるか、同等の役割を引き受けます。

   役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. システムのデフォルトのスケジューラが **FSS** になるように設定します。

   ```
   # dispadmin -d FSS
   ```

   この変更指定は次のリブートで有効になります。リブート後は、システムのすべてのプロセスが FSS スケジューリングクラスで実行されます。

3. リブートを行わずに、この構成をただちに有効にします。

   ```
   # priocntl -s -c FSS -i all
   ```

▼ プロセスを TS クラスから FSS クラスに手動で移動する方法

デフォルトのスケジューリングクラスを変更した後でリブートしなくても、あるスケジューリングクラスから別のスケジューリングクラスにプロセスを手動で移動できます。次の手順は、TS スケジューリングクラスから FSS スケジューリングクラスにプロセスを手動で移動する方法を示しています。

1. スーパーユーザーになるか、同等の役割を引き受けます。

   役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **init** プロセス (**pid 1**) を **FSS** スケジューリングクラスに移動します。

   ```
   # priocntl -s -c FSS -i pid 1
   ```

3. すべてのプロセスを **TS** スケジューリングクラスから **FSS** スケジューリングクラスに移動します。

   ```
   # priocntl -s -c FSS -i class TS
   ```

注 – すべてのプロセスは、リブート後には再び TS スケジューリングクラスで実行されます。

▼ プロセスをすべてのユーザークラスから **FSS** クラスに手動で移動する方法

TS 以外のデフォルトのクラスを使用している場合、たとえば、デフォルトで IA クラスを使用するウィンドウ環境がシステムで実行されている場合があります。デフォルトのスケジューリングクラスを変更した後でリブートしなくても、すべてのプロセスを FSS スケジューリングクラスに手動で移動できます。

1. スーパーユーザーになるか、同等の役割を引き受けます。

   役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **init** プロセス (**pid 1**) を **FSS** スケジューリングクラスに移動します。

   ```
   # priocntl -s -c FSS -i pid 1
   ```

3. すべてのプロセスを現在のスケジューリングクラスから **FSS** スケジューリングクラスに移動します。

   ```
   # priocntl -s -c FSS -i all
   ```

注 – すべてのプロセスは、リブート後には再びデフォルトのスケジューリングクラスで実行されます。

▼ プロジェクトのプロセスを FSS クラスに手動で移動する方法

プロジェクトのプロセスを、現在のスケジューリングクラスから FSS スケジューリングクラスに手動で移動できます。

1 スーパーユーザーになるか、同等の役割を引き受けます。

役割には、認証と特権コマンドが含まれます。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 プロジェクト ID *10* で実行するプロセスを FSS スケジューリングクラスに移動します。

```
# priocntl -s -c FSS -i projid 10
```

プロジェクトのプロセスは、リブート後には再びデフォルトのスケジューリングクラスで実行されます。

スケジューラのパラメータを調整する方法

dispadmin コマンドを使用すると、システムの稼働中にプロセススケジューラパラメータを表示または変更できます。たとえば、dispadmin コマンドを使用して、FSS スケジューラのタイムクォンタム (time quantum) 値を調べ、調整できます。「タイムクォンタム」とは、スレッドがプロセッサ上で実行を開始してからそのプロセッサを放棄するまでの時間量のことです。

システムの稼働中に FSS スケジューラの現在のタイムクォンタムを表示するには、次のように入力します。

```
$ dispadmin -c FSS -g
#
# Fair Share Scheduler Configuration
#
RES=1000
#
# Time Quantum
#
QUANTUM=110
```

-g オプションを使用するときに、同時に -r オプションも指定すると、タイムクォンタム値の表示に使用する最小単位を指定できます。最小単位を指定しないと、タイムクォンタム値はデフォルトのミリ秒で表示されます。

```
$ dispadmin -c FSS -g -r 100
#
# Fair Share Scheduler Configuration
```

```
#
RES=100
#
# Time Quantum
#
QUANTUM=11
```

FSS スケジューリングクラスにスケジューリングパラメータを設定するには、`dispadmin -s` を使用します。*file* 内の値は、`-g` オプションで得られる出力と同じ形式で指定する必要があります。これらの値は、カーネル内の現在の値を上書きします。次の行を入力します。

```
$ dispadmin -c FSS -s file
```

◆ ◆ ◆ 第 10 章

10 リソース上限デーモンによる物理メモリーの制御 (概要)

リソース上限デーモン rcapd を使用すると、リソース上限が定義されたプロジェクト内で動作するプロセスが消費する物理メモリーを規制できます。

Solaris 10 8/07: システムでゾーンを実行している場合は、大域ゾーンから rcapd を使用して、非大域ゾーンでの物理メモリーの消費を規制できます。詳細は、第 18 章「非大域ゾーンの計画と構成 (タスク)」を参照してください。

この章の内容は次のとおりです。

rcapd 機能の使用手順については、第 11 章「リソース上限デーモンの管理 (タスク)」を参照してください。

リソース上限デーモンによる物理メモリー制御の新機能

Solaris 10: projmod コマンドを使用して、/etc/project ファイル内の rcap.max-rss 属性を設定できるようになりました。

Solaris 10 11/06: Solaris サービス管理機能 (SMF) のサービスとしてリソース上限デーモンを有効化/無効化することに関する情報が、追加されました。

Solaris 10 の新機能の全一覧および Solaris リリースについての説明は、『Oracle Solaris 10 8/11 の新機能』を参照してください。

リソース上限デーモンの紹介

リソース「上限」とは、物理メモリーなどのリソース消費量の上限のことです。物理メモリーの上限はプロジェクトごとに設定します。

リソース上限デーモンとその関連ユーティリティーは、物理メモリーのリソース上限を制限および管理するメカニズムを提供します。

リソース制御と同様に、リソース上限を定義するには、`project` データベース内にあるプロジェクトエントリの属性を使用します。リソース制御はカーネルによって同期的に実行されますが、物理メモリーのリソース上限の制限はリソース上限デーモンによってユーザーレベルで非同期的に実行されます。このリソース上限の制限は非同期的に実行されるため、リソース上限デーモンがサンプリングを行う間隔の分だけ、短時間の遅延が発生します。

`rcapd` については、`rcapd(1M)` のマニュアルページを参照してください。プロジェクトと `project` データベースについては、第 2 章「プロジェクトとタスク (概要)」 および `project(4)` のマニュアルページを参照してください。リソース制御については、第 6 章「リソース制御 (概要)」 を参照してください。

リソース上限制御のしくみ

リソース上限デーモンは、物理メモリー上限が定義されたプロジェクトのリソース使用効率を定期的にサンプリングします。このサンプリング間隔は管理者が指定できます。詳細は、135 ページの「サンプリング間隔の決定」 を参照してください。システムの物理メモリー使用効率が上限実行しきい値を超え、ほかの条件にも適合する場合、リソース上限デーモンは、物理メモリー上限が定義されたプロジェクトのリソース消費をその上限以下に減らします。

仮想メモリーシステムは物理メモリーを複数の「ページ」に分割します。「ページ」は、Solaris メモリー管理サブシステムにおける物理メモリーの基礎となる単位です。データをファイルからメモリーに読み取るとき、仮想メモリーシステムは 1 度に 1 ページずつ読み取ります。この動作のことを、ファイルの「ページイン」と呼びます。リソース消費を減らすとき、リソース上限デーモンは、あまり使用されていないページをスワップデバイス (物理メモリーの外にある領域) に再配置します。この動作のことを「ページアウト」と呼びます。

物理メモリーを管理するために、リソース上限デーモンは、プロジェクトの作業負荷の常駐セットのサイズを、作業セットのサイズに対して調節します。常駐セットとは、物理メモリーに常駐するページのことです。作業セットとは、作業負荷がその処理サイクル中にアクティブに使用するページのことです。作業セットは、プロセスが動作するモードや処理するデータの種類に応じて、そのときどきで変化します。すべての作業負荷がその作業セットの常駐に十分な物理メモリーにアクセスできるのが理想です。しかし、作業セットは、物理メモリーのほか、セカンダリディスク記憶装置にも格納することが可能です。

rcapd は 1 度に 1 つのインスタンスしか実行できません。

プロジェクトの物理メモリーの使用率を制限する属性

物理メモリーリソースの上限をプロジェクトに対して定義するには、project データベースエントリにこの属性を追加して、常駐セットサイズ (RSS) の上限を設定します。

rcap.max-rss　プロジェクト内のプロセスが利用できる物理メモリーの合計 (バイト数)。

たとえば、/etc/project ファイル内の次の行は、10G バイトの RSS 上限を db というプロジェクトに対して設定します。

```
db:100::db,root::rcap.max-rss=10737418240
```

注 – 指定した上限値はシステムによってページのサイズに丸められることがあります。

---

projmod コマンドを使用すると、/etc/project ファイル内の rcap.max-rss 属性を設定できます。

```
# projmod -s -K rcap.max-rss=10GB db
```

/etc/project ファイルには、次の行が含まれるようになります。

```
db:100::db,root::rcap.max-rss=10737418240
```

rcapd の構成

リソース上限デーモンを構成するには、rcapadm コマンドを使用します。次のような作業が可能です。

- 上限実行しきい値を設定します
- rcapd が実行する動作間隔を設定します
- リソース上限制御を有効または無効にします
- 構成済みのリソース上限デーモンの現在のステータスを表示します

資源上限デーモンを構成するには、スーパーユーザーの特権を持っているか、プロファイルの一覧内に Process Management プロファイルが含まれている必要があります。Process Management 役割と System Administrator 役割には、どちらにも Process Management プロファイルが含まれています。

構成の変更を rcapd に適用するには、構成間隔に従うか (134 ページの「rcapd の動作間隔」を参照)、または必要に応じて SIGHUP を送信します (kill(1) のマニュアルページを参照)。

引数なしで使用した場合、rcapadm はリソース上限デーモンの現在のステータスを表示します (構成されている場合のみ)。

次の項では、上限の制限、上限値、および rcapd の動作間隔について説明します。

ゾーン環境がインストールされているシステムでのリソース上限デーモンの使用

ゾーンを構成するときに capped-memory リソースを設定することにより、ゾーンの常駐セットサイズ (RSS) 使用量を制御できます。詳細は、236 ページの「Solaris 10 8/07: 物理メモリーの制御と capped-memory リソース」を参照してください。大域ゾーンも含め、ゾーン内で rcapd を実行すると、そのゾーン内のプロジェクトにメモリー上限を適用できます。

特定のゾーンで消費可能なメモリー量に、次のリブートまで一時的な上限を設定することができます。142 ページの「ゾーンに一時的なリソース上限を指定する方法」を参照してください。

rcapd をゾーン内で使用して、リソース上限が定義されたプロジェクト内で動作するプロセスが消費する物理メモリーを規制する場合は、そのゾーン内でデーモンを構成する必要があります。

別のゾーン内にあるアプリケーションのメモリー上限を選択する場合、通常は、そのアプリケーションが別のゾーン内にあることを考慮する必要はありません。例外は、ゾーン別のサービスの場合です。ゾーン別のサービスは、メモリーを消費します。システムの物理メモリー量およびメモリー上限を決定する際には、このメモリー消費を考慮してください。

注 – rcapd を lx ブランドゾーンで実行することはできません。ただし、デーモンを大域ゾーンから使用してブランドゾーンのメモリー上限を設定することはできます。

メモリー上限実行しきい値

「メモリー上限実行しきい値」とは、上限制限を引き起こす、システム上の物理メモリーの使用効率 (パーセンテージ) のことです。システムの物理メモリー使用率がこのしきい値を超えたとき、メモリー上限が制限されます。この使用率には、アプ

リケーションやカーネルが使用する物理メモリーも含まれます。この使用効率は、メモリー上限が制限される方法を決定します。

上限が制限されると、プロジェクトの作業負荷からメモリーがページアウトされることがあります。

- メモリーをページアウトすることによって、作業負荷に定義された上限を超えた分のメモリーのサイズを小さくすることができます。
- メモリーをページアウトすることによって、システム上のメモリー上限実行しきい値を超えた物理メモリーの使用率を下げることができます。

作業負荷は、定義された上限までの物理メモリーを使用することが許可されます。システムのメモリー使用率がメモリー上限実行しきい値を下回っている場合、作業負荷は上限より多くのメモリーを使用できます。

上限実行しきい値を設定する方法については、140 ページの「メモリー上限実行しきい値を設定する方法」を参照してください。

上限値の決定

プロジェクトの上限の設定が低すぎると、通常の状態でも、作業負荷が効率的に機能するだけのメモリーを使用できない可能性があります。作業負荷がより多くのメモリーを要求するためページングが発生し、システムの性能に悪影響がでます。

プロジェクトの上限の設定が高すぎると、上限に達する前に、利用可能な物理メモリーを使い果たす可能性があります。この場合、物理メモリーは、`rcapd` ではなくカーネルによって効率的に管理されます。

プロジェクトの上限を決定するときには、次の要素を考慮します。

入出力システムへの影響
: サンプリングした使用率がプロジェクトの上限を超えている場合、リソース上限デーモンはプロジェクトの作業負荷の物理メモリー使用率を減らそうとします。上限が制限されている間は、作業負荷がマッピングしているファイルには、スワップなどのデバイスが使用されます。上限を頻繁に超える作業負荷の場合、その性能は、スワップデバイスの性能に大きく左右されます。このような作業負荷を実行することは、作業負荷の上限と同じサイズの物理メモリーを持つマシン上で作業負荷を実行することと似ています。

CPU 使用率への影響
: リソース上限デーモンの CPU 使用率は、このデーモンが上限を制限するプロジェクトの作業負荷内のプロセスの数と、作業負荷のアドレス空間のサイズによって変化します。

リソース上限デーモンの CPU 時間の一部は、作業負荷の使用率のサンプリングに費やされます。作業負荷にプロセスを追加すると、使用率のサンプリングにかかる時間が増えます。

上限値を超えると上限が制限され、リソース上限デーモンの CPU 時間がさらに消費されます。消費される CPU 時間は仮想メモリーの量に比例します。消費される CPU 時間は、作業負荷のアドレス空間の合計サイズの変化によって増減します。この情報は、rcapstat の出力の vm 列に報告されます。詳細は、136 ページの「rcapstat によるリソース使用効率の監視」および rcapstat(1) のマニュアルページを参照してください。

共有メモリーの報告

rcapd デーモンは、ほかのプロセスと共有されているメモリーページ、あるいは、同じプロセス内で複数回マッピングされているメモリーページについて、かなり正確な RSS の見積もりを報告します。異なるプロジェクトのプロセスが同じメモリーを共有している場合、そのメモリーは、そのメモリーを共有しているすべてのプロジェクトの RSS の合計に含められます。

この見積もりは、データベースのように共有メモリーを多用する作業負荷に使用できます。データベースの作業負荷では、次のようにプロジェクトの通常の使用率をサンプリングすることによって、適切な初期上限値を決定することもできます。prstat コマンドに -J または -Z オプションを付けて実行し、その出力を使用します。詳細は、prstat(1M) のマニュアルページを参照してください。

rcapd の動作間隔

rcapd を定期的に実行するように、rcapd の動作間隔を設定できます。

すべての間隔は秒単位で指定します。次の表で、rcapd の動作とそのデフォルトの間隔値について説明します。

動作	デフォルトの間隔値 (秒)	説明
scan	15	プロジェクトの作業負荷内で動作しているプロセスを走査する間隔の秒数。最小値は 1 秒です。
sample	5	常駐セットサイズのサンプリングから、その後に上限を制限するまでの間の秒数。最小値は 1 秒です。
report	5	ページング統計を更新する間隔の秒数。0 に設定すると、ページング統計は更新されず、rcapstat からの出力も最新の状態を示さなくなります。
config	60	再構成する間隔の秒数。再構成イベントでは、rcapadm は構成ファイルを更新用に読み取って、project データベースに新しいまたは改訂されたプロジェクト上限があるかどうかを走査します。SIGHUP を rcapd に送信すると、再構成が即座に行われます。

間隔を調節する方法については、141 ページの「動作間隔を設定する方法」を参照してください。

rcapd 走査間隔の決定

走査間隔とは、rcapd が新しいプロセスを探す頻度のことです。多くのプロセスが動作しているシステムでは、一覧の走査に時間がかかるため、走査間隔を長くして、消費される CPU 時間の合計を減らしたほうがよい場合もあります。しかし、走査間隔は、あるプロセスの存在が上限が定義されている作業負荷に属するとみなされるまでに最低限必要な時間も意味します。生存期間が短いプロセスを数多く実行する作業負荷の場合、走査間隔が長いと、rcapd はそれらのプロセスが作業負荷に属さないものとみなす可能性があります。

サンプリング間隔の決定

rcapadm で構成したサンプリング間隔は、作業負荷の使用率をサンプリングして上限を超えていた場合に、rcapd が上限を制限するまでの、最短時間です。サンプリング間隔を短くすると、ほとんどの場合、rcapd が上限を頻繁に制限するため、ページングによる入出力が増えます。しかし、サンプリング間隔を短くすると、特定の作業負荷の物理メモリー使用率が急増した場合に、ほかの作業負荷への影響を抑えるこ

とにもなります。サンプリングの合間には、この作業負荷は自由にメモリーを消費でき、上限が定義されているほかの作業負荷のメモリーすらも利用できますが、この合間が狭められることになるのです。

rcapstat に指定したサンプリング間隔が、rcapadm で rcapd に指定したサンプリング間隔よりも短い場合、いくつかの間隔に対する出力がゼロになることがあります。この状況が発生するのは、rcapd が統計を更新する間隔が、rcapadm で指定した間隔よりも長いためです。rcapadm で指定した間隔は、rcapstat で使用されるサンプリング間隔から独立しています。

rcapstat によるリソース使用効率の監視

上限が定義されたプロジェクトのリソース使用率を監視するには、rcapstat を使用します。rcapstat の報告例については、143 ページの「rcapstat による報告の生成」を参照してください。

報告用のサンプリング間隔、および統計を繰り返す回数を設定できます。

interval サンプリング間隔を指定します (秒)。デフォルトの間隔は 5 秒です。

count 統計を繰り返す回数を指定します。デフォルトでは、終了シグナルを受信するまで、あるいは、rcapd プロセスが終了するまで、rcapstat は統計を報告し続けます。

rcapstat が最初に発行する報告では、ページング統計はデーモンの起動以降の活動を示します。以後の報告では、前回の報告以降の活動を示します。

次の表に、rcapstat の報告の列見出しを定義します。

rcapstat の列見出し	説明
id	上限が定義されたプロジェクトの ID。
プロジェクト	プロジェクト名。
nproc	プロジェクト内のプロセス数。
vm	プロジェクト内のプロセスが使用する仮想メモリーサイズの合計。マップされたファイルおよびデバイスもすべて含みます。単位は、K バイト (K)、M バイト (M)、または G バイト (G) です。
rss	プロジェクト内のプロセスが使用すると推定される常駐セットサイズ (RSS) の合計。単位は、K バイト (K)、M バイト (M)、または G バイト (G) です。共有されるページは考慮されません。

rcapstat の列見出し	説明
cap	プロジェクトに定義された RSS 上限。メモリー上限を指定する方法については、131 ページの「プロジェクトの物理メモリーの使用率を制限する属性」または rcapd(1M) のマニュアルページを参照してください。
at	前回の rcapstat のサンプリング以降、 rcapd がページアウトしようとしたメモリーの合計。
avgat	前回の rcapstat のサンプリング以降に発生した各サンプリングサイクル中、rcapd がページアウトしようとしたメモリーの平均。rcapd がコレクション RSS をサンプリングする頻度は、rcapadm で設定できます。134 ページの「rcapd の動作間隔」を参照してください。
pg	前回の rcapstat のサンプリング以降、 rcapd が正常にページアウトしたメモリーの合計。
avgpg	前回の rcapstat のサンプリング以降に発生した各サンプリングサイクル中、rcapd が正常にページアウトしたと推定されるメモリーの平均。rcapd がプロセス RSS をサンプリングする頻度は rcapadm で設定できます。134 ページの「rcapd の動作間隔」を参照してください。

rcapd とともに使用されるコマンド

コマンドのリファレンス	説明
rcapstat(1)	上限が定義されたプロジェクトのリソース使用効率を監視します。
rcapadm(1M)	リソース上限デーモンを構成します。構成済みのリソース上限デーモンの現在のステータスを表示します。リソース上限制御を有効または無効にします。
rcapd(1M)	リソース上限デーモン。

◆ ◆ ◆ 第 11 章

11 リソース上限デーモンの管理 (タスク)

この章では、リソース上限デーモン rcapd を構成して使用する手順について説明します。

rcapd の概要については、第 10 章「リソース上限デーモンによる物理メモリーの制御 (概要)」を参照してください。

リソース上限デーモンの構成と使用 (タスクマップ)

タスク	説明	参照先
メモリー上限実行しきい値を設定します。	プロセスが利用できる物理メモリーが少なくなったときに制限する上限を構成します。	140 ページの「メモリー上限実行しきい値を設定する方法」
動作間隔を設定します。	この間隔は、リソース上限デーモンが行う定期的な動作に適用されます。	141 ページの「動作間隔を設定する方法」
リソース上限制御を有効にします。	システムでリソース上限制御を起動します。	141 ページの「リソース上限制御を有効にする方法」
リソース上限制御を無効にします。	システムでリソース上限制御を停止します。	142 ページの「リソース上限制御を無効にする方法」
上限とプロジェクトの情報を報告します。	報告を生成するためのコマンド例を表示します。	143 ページの「上限とプロジェクトの情報の報告」
プロジェクトの常駐セットサイズを監視します。	プロジェクトの常駐セットサイズについて報告を生成します。	144 ページの「プロジェクトの RSS の監視」
プロジェクトの作業セットサイズを決定します。	プロジェクトの作業セットサイズについて報告を生成します。	144 ページの「プロジェクトの作業セットサイズの決定」

タスク	説明	参照先
メモリー使用効率とメモリー上限を報告します。	各サンプリング間隔に対応する報告の最後に、メモリー使用効率と上限実行しきい値を示す行を出力します。	145 ページの「メモリー使用効率とメモリー上限実行しきい値の報告」

rcapadm によるリソース上限デーモンの管理

このセクションでは、rcapadm コマンドを使用してリソース上限デーモンを構成する手順について説明します。詳細は、131 ページの「rcapd の構成」および rcapadm(1M) のマニュアルページを参照してください。rcapadm を使用してゾーンに一時的なリソース上限を指定する方法についても説明します。

引数なしで使用した場合、rcapadm はリソース上限デーモンの現在のステータスを表示します (構成されている場合のみ)。

▼ メモリー上限実行しきい値を設定する方法

上限は、プロセスが利用できる物理メモリーが少なくなるまで制限されないように構成できます。詳細は、132 ページの「メモリー上限実行しきい値」を参照してください。

最小値 (デフォルト) は 0 です。これは、メモリー上限が常に制限されることを意味します。最小値を変更するには、次の手順に従います。

1. スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

   System Administrator 役割には Process Management プロファイルが含まれています。役割を作成してユーザーに割り当てる方法については、『Solaris のシステム管理ガイド (セキュリティサービス)』の「RBAC の管理 (タスクマップ)」を参照してください。

2. **rcapadm** の **-c** オプションを使用することで、メモリー上限を制限するときの物理メモリー使用効率を設定します。

   ```
   # rcapadm -c percent
   ```

   percent は 0 から 100 までの値です。この値を大きくするほど、規制が小さくなります。つまり、上限が定義されたプロジェクトの作業負荷は、システムのメモリー使用効率がこのしきい値を超えない限り、上限を適用されることなく実行できます。

参照 現在の物理メモリーの使用効率と上限実行しきい値を表示する方法については、145 ページの「メモリー使用効率とメモリー上限実行しきい値の報告」を参照してください。

▼ 動作間隔を設定する方法

134 ページの「rcapd の動作間隔」では、rcapd が行う定期的な動作の間隔について説明しています。rcapadm を使用して動作間隔を設定するには、次の手順に従います。

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割を作成してユーザーに割り当てる方法については、『Solaris のシステム管理ガイド (セキュリティサービス)』の「RBAC の管理 (タスクマップ)」を参照してください。

2 **-i** オプションを使用して、動作間隔の値を設定します。

```
# rcapadm -i interval=value,...,interval=value
```

注 – すべての動作間隔の値の単位は秒です。

▼ リソース上限制御を有効にする方法

リソース上限制御をシステムで有効にする方法は 3 つあります。リソース上限制御を有効にすると、さらに /etc/rcap.conf ファイルがデフォルト値で設定されます。

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割を作成してユーザーに割り当てる方法については、『Solaris のシステム管理ガイド (セキュリティサービス)』の「RBAC の管理 (タスクマップ)」を参照してください。

2 次のどちらかの方法でリソース上限デーモンを有効にします。

- svcadm コマンドを使って、リソース上限制御を有効にします。

  ```
  # svcadm enable rcap
  ```

- リソース上限デーモンを有効にして、ただちにブートし、かつ、システムをブートするたびにブートするようにします。次のように入力します。

  ```
  # rcapadm -E
  ```

- リソース上限デーモンをただちには起動せず、ブート時に有効にするには、-n オプションも指定します。

  ```
  # rcapadm -n -E
  ```

▼ リソース上限制御を無効にする方法

リソース上限制御をシステムで無効にする方法は 3 つあります。

1. スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

   System Administrator 役割には Process Management プロファイルが含まれています。役割を作成してユーザーに割り当てる方法については、『Solaris のシステム管理ガイド (セキュリティサービス)』の「RBAC の管理 (タスクマップ)」を参照してください。

2. 次のどちらかの方法でリソース上限デーモンを無効にします。

   - **svcadm** コマンドを使用して、リソース上限制御をオフにします。

     ```
     # svcadm disable rcap
     ```

   - リソース上限デーモンを無効にして、ただちに停止し、かつ、システムをブートしても起動しないようにするには、次のように入力します。

     ```
     # rcapadm -D
     ```

   - リソース上限デーモンを停止せずに無効にするには、**-n** オプションも指定します。

     ```
     # rcapadm -n -D
     ```

ヒント – リソース上限デーモンの安全な無効化

---

rcapd を安全に無効にするには、svcadm コマンド、または -rcapadm コマンドを D オプションとともに使用します。リソース上限デーモンを強制終了すると (kill(1) のマニュアルページを参照)、プロセスが停止状態のままになり、手動で再起動しなければならない場合があります。プロセスの実行を再開するには、prun コマンドを使用します。詳細は、prun(1) のマニュアルページを参照してください。

▼ ゾーンに一時的なリソース上限を指定する方法

この手順は、特定のゾーンで消費可能な最大のメモリー量を割り当てる場合に使用します。この値は、次のリブートまでに限り有効です。持続的な上限を設定するには、zonecfg コマンドを使用します。

1. スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

   System Administrator 役割には Process Management プロファイルが含まれています。

2 ゾーン **my-zone** に最大メモリーの値として **512M** バイトを設定します。

```
# rcapadm -z testzone -m 512M
```

rcapstat による報告の生成

リソース上限制御の統計を報告するには、rcapstat を使用します。136 ページの「rcapstat によるリソース使用効率の監視」 コマンドを使って報告を生成する方法については、Monitoring Resource Utilization With rcapstatを参照してください。このセクションでは、報告の列見出しについても説明します。rcapstat(1) のマニュアルページにも、この情報が含まれています。

このセクションでは、さまざまな目的の報告を生成する方法を、例を使用しながら説明します。

上限とプロジェクトの情報の報告

この例では、2 人のユーザーに関連付けられた 2 つのプロジェクトに、上限が定義されています。user1 の上限は 50M バイト、user2 の上限は 10M バイトです。

次のコマンドは、5 つの報告を 5 秒間のサンプリング間隔で生成します。

```
user1machine% rcapstat 5 5
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M   50M    0K 3312K    0K
 78194   user2      1  2368K  1856K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1856K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
    id project  nproc     vm    rss   cap    at avgat    pg avgpg
112270   user1     24   123M    35M   50M    0K    0K    0K    0K
 78194   user2      1  2368K  1928K   10M    0K    0K    0K    0K
```

出力の最初の 3 行は 1 回目の報告です。ここには、2 つのプロジェクトに関する上限とプロジェクトの情報、および rcapd 起動以降のページング統計が記載されています。at と pg の列において、user1 にはゼロより大きな値が入っており、user2 にはゼロが入っています。これは、1 回目の報告の期間中、user1 は上限を超えたが、user2 は超えなかったことを意味します。

2 回目以降の報告では、目立った活動はありません。

プロジェクトの RSS の監視

この例では、プロジェクト user1 の RSS が上限を超えています。

次のコマンドは、5 つの報告を 5 秒間のサンプリング間隔で生成します。

```
user1machine% rcapstat 5 5

   id project  nproc    vm   rss   cap    at avgat     pg  avgpg
376565   user1      3 6249M 6144M 6144M  690M  220M  5528K  2764K
376565   user1      3 6249M 6144M 6144M    0M  131M  4912K  1637K
376565   user1      3 6249M 6171M 6144M   27M  147M  6048K  2016K
376565   user1      3 6249M 6146M 6144M 4872M  174M  4368K  1456K
376565   user1      3 6249M 6156M 6144M   12M  161M  3376K  1125K
```

プロジェクト user1 は 3 つのプロセスを持っており、それぞれがアクティブに物理メモリーを使用しています。pg 列の正の値が示しているとおり、rcapd は、このプロジェクトのプロセスの物理メモリーの使用率を上限に合わせて下げようと、メモリーをページアウトし続けています。しかし、rcapd は RSS を上限値未満に保つことに成功していません。これは、rss 値が相応の減少を示していないことでわかります。作業負荷はメモリーをページアウトするとすぐにまたメモリーを使用しており、その結果、RSS の値がまた上がってしまいます。これは、プロジェクトの常駐メモリーがすべてアクティブに使用されており、かつ、作業セットサイズ (*WSS*) が上限よりも大きいことを意味します。そのため、rcapd は、上限に合わせて作業セットの一部をページアウトせざるを得ません。この状況では、次のうちのいずれかが起きるまで、システムのページフォルト率および関連する入出力は高いままです。

- WSS が小さくなる。
- 上限を上げる。
- アプリケーションのメモリーアクセスパターンが変わる。

この状況では、サンプリング間隔を短くすることで、RSS 値と上限値の差を小さくできる可能性があります。rcapd が作業負荷をサンプリングして上限を制限する頻度が上がるからです。

注 – ページフォルトが発生するのは、新しいページを作成する必要があるとき、あるいは、システムがスワップデバイスからページをコピーする必要があるときです。

プロジェクトの作業セットサイズの決定

この例では、前の例と同じプロジェクトを使用します。

前の例では、プロジェクト user1 が自分に許可された上限よりも多くの物理メモリーを使用しているところを示しました。この例では、どれくらいのメモリーをプロジェクトの作業負荷が必要とするのかを示します。

```
user1machine% rcapstat 5 5
    id project  nproc    vm   rss   cap    at avgat     pg  avgpg
376565   user1      3 6249M 6144M 6144M  690M    0K   689M     0K
376565   user1      3 6249M 6144M 6144M    0K    0K     0K     0K
376565   user1      3 6249M 6171M 6144M   27M    0K    27M     0K
376565   user1      3 6249M 6146M 6144M 4872K    0K  4816K     0K
376565   user1      3 6249M 6156M 6144M   12M    0K    12M     0K
376565   user1      3 6249M 6150M 6144M 5848K    0K  5816K     0K
376565   user1      3 6249M 6155M 6144M   11M    0K    11M     0K
376565   user1      3 6249M 6150M   10G   32K    0K    32K     0K
376565   user1      3 6249M 6214M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
376565   user1      3 6249M 6247M   10G    0K    0K     0K     0K
```

サイクルの中ほどで、プロジェクト user1 の上限を 6G バイトから 10G バイトに上げました。値を上げることによって上限の制限が止まり、常駐セットサイズが上昇しました。常駐セットサイズを規制するのは、ほかのプロセスと、マシンのメモリー容量だけになりました。rss 列が、プロジェクトの作業セットサイズ (この例では 6247M バイト) を反映して安定する場合があります。この値が、このプロジェクトのプロセスがページフォルトを頻繁に起こさずに動作できる、上限の最小値です。

user1 の上限が 6G バイトであるとき、サンプリング間隔である 5 秒ごとに、rcapd が作業負荷のメモリーの一部をページアウトするにつれて、RSS は減少して入出力は増加します。ページアウトの直後、これらのページを必要とする作業負荷は、(動作し続ける限り) メモリーをページインします。このサイクルは、例の中ほどで上限を10G バイトに上げるまで繰り返されます。その後、RSS は 6.1G バイトで安定します。作業負荷の RSS が上限より低くなったため、これ以後ページングは発生しません。また、ページングに関連する入出力は止まります。このため、観察時にプロジェクトが行なっていた作業の実行には、6.1G バイトが必要であったことがわかります。

vmstat(1M) および iostat(1M) のマニュアルページも参照してください。

メモリー使用効率とメモリー上限実行しきい値の報告

rcapstat の -g オプションを使用すると、次のことを報告できます。

- 現在の物理メモリーの使用効率 (システムにインストールされている物理メモリーに占めるパーセンテージ)
- rcapadm で設定されたシステムメモリー上限実行しきい値

-g オプションを使用すると、各サンプリング間隔に対応する報告の最後に、メモリー使用効率と上限実行しきい値を示す行が出力されます。

```
# rcapstat -g
    id project   nproc    vm   rss   cap    at avgat   pg  avgpg
376565    rcap       0    0K    0K   10G    0K    0K   0K     0K
physical memory utilization: 55%   cap enforcement threshold: 0%
    id project   nproc    vm   rss   cap    at avgat   pg  avgpg
376565    rcap       0    0K    0K   10G    0K    0K   0K     0K
physical memory utilization: 55%   cap enforcement threshold: 0%
```

◆ ◆ ◆ 第 12 章

12 リソースプール (概要)

この章では、次の機能について説明します。

- リソースプール。マシンリソースの区分に使用されます
- 動的リソースプール (DRP)。設定されているシステムの目標に合うように、各リソースプールのリソース割り当てを動的に調整します

Solaris 10 11/06 リリースから、リソースプールおよび動的リソースプールが、Solaris サービス管理機能 (SMF) 内のサービスになりました。これらの各サービスは、別個に有効にできます。

この章の内容は次のとおりです。

この機能の使用手順については、第 13 章「リソースプールの作成と管理 (タスク)」を参照してください。

リソースプールと動的リソースプールの新機能

Solaris 10: システムイベントやアプリケーションの負荷変化に応じて各プールのリソース割り当てを調整するメカニズムがリソースプールに追加されました。動的リソースプールによって管理者が決定を行いやすくなり、決定を行う回数も減少します。管理者がシステム性能の目標を指定すると、それを維持するように自動的に調整が行われます。

`projmod` コマンドを使用して、`/etc/project` ファイル内の `project.pool` 属性を設定できるようになりました。

Solaris 10 の新機能の全一覧および Solaris リリースについての説明は、『Oracle Solaris 10 8/11 の新機能』を参照してください。

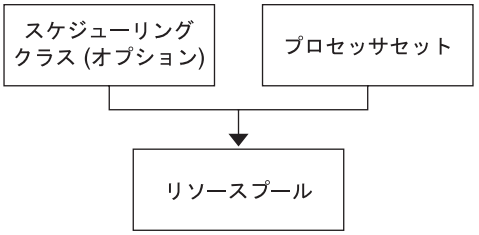
Solaris 10 11/06: リソースプールと動的リソースプールが、SMF サービスになりました。

リソースプールの紹介

「リソースプール」を使用すると、作業負荷によって特定のリソースが重複して消費されないように、作業負荷を分離することができます。このような方法でリソースを確保すると、さまざまな作業負荷が混在するシステム上で予測どおりの性能を得ることができます。

リソースプールは、プロセッサセット (pset) の構成やスケジューリングクラスの割り当て (任意) に対して一貫した構成メカニズムを提供します。

図 12–1 リソースプールのフレームワーク



プールは、システムで使用可能なさまざまなリソースセットを結合した特定のものと考えることができます。リソースのさまざまな組み合わせを表す各種のプールを作成できます。

```
pool1: pset_default
pool2: pset1
pool3: pset1, pool.scheduler="FSS"
```

リソースプールは、複数のパーティションをグループ化することにより、ラベル付けされている作業負荷とハンドルを対応付けることができます。/etc/project ファイル内の各プロジェクトエントリには単一のプールを関連付けることができます。それらのプールを指定するには、project.pool 属性を使用します。

プールが有効になっている場合、基本構成は「デフォルトプール」と「デフォルトプロセッサセット」から成ります。ユーザー定義のプールやプロセッサセットを作成し、構成に追加することもできます。CPU は 1 つのプロセッサセットだけに所属できます。ユーザー定義のプールやプロセッサセットは破棄できます。デフォルトプールとデフォルトプロセッサセットは破棄できません。

デフォルトプールの pool.default プロパティーは true に設定されます。デフォルトプロセッサセットの pset.default プロパティーは true に設定されます。したがって、デフォルトプールとデフォルトプロセッサセットはどちらも、名前が変更された場合でも識別できます。

ユーザー定義プールメカニズムは主に、5 つ以上の CPU を搭載する大規模なマシンで使用されます。ただし、小規模なマシンでもこの機能を活用することができます。小規模なマシンでは、重要でないリソースパーティションを共有するプールを作成できます。重要なリソースにだけ、専用のプールが使用されます。

動的リソースプールの紹介

動的リソースプールは、システムイベントやアプリケーション負荷の変化に対応して各プールのリソース割り当てを動的に調整するメカニズムを提供します。動的リソースプールによって管理者が決定を行いやすくなり、決定を行う回数も減少します。管理者がシステム性能の目標を指定すると、それを維持するように自動的に調整が行われます。構成に変更を加えると、変更内容がログに記録されます。これらの機能は、主にリソースコントローラ poold によって実行されます。動的リソース割り当てが必要なときは、常にこのシステムデーモンをアクティブにしておく必要があります。poold は定期的にシステムの負荷を調べ、システムの性能を最適に保つために、リソース消費に関する調整が必要かどうかを判定します。poold の構成は libpool の構成に保存されています。poold の詳細については、poold(1M) のマニュアルページを参照してください。

リソースプールと動的リソースプールの有効化/無効化について

リソースプールおよび動的リソースプールを有効化/無効化する方法については、173 ページの「プール機能の有効化と無効化」を参照してください。

ゾーンで使用されるリソースプール

ヒント – **Solaris 10 8/07:** システム上で構成済みのリソースプールにゾーンを関連付ける代わりに、`zonecfg` コマンドを使用して、ゾーンの稼働中に有効になる一時プールを作成することもできます。詳細は、234 ページの「Solaris 10 8/07: `dedicated-cpu` リソース」を参照してください。

ゾーンが有効になっているシステムの場合、1 つの非大域ゾーンにはリソースプールを 1 つだけ関連付けることができますが、特定のゾーンに割り当てたプールをそのゾーン専用にする必要はありません。また、大域ゾーンから `poolbind` コマンドを使用して、非大域ゾーンの個々のプロセスを別のプールに結合することもできません。非大域ゾーンをプールに関連付ける方法については、265 ページの「ゾーンを構成、検証、および確定する」を参照してください。

プールに対してスケジューリングクラスを設定した場合は、そのプールに非大域ゾーンを関連付けると、そのゾーンではそのスケジューリングクラスがデフォルトで使用されます。

動的リソースプールを使用している場合、実行中の `poold` インスタンスの有効範囲は大域ゾーンに制限されます。

`poolstat` ユーティリティーを非大域ゾーンで実行すると、そのゾーンに関連付けられているプールの情報だけが表示されます。非大域ゾーンで引数なしで `pooladm` コマンドを実行すると、そのゾーンに関連付けられているプールの情報だけが表示されます。

リソースプールのコマンドについては、169 ページの「リソースプール機能で使用するコマンド」を参照してください。

リソースプールを使用する場合

リソースプールは、多くの管理作業に適用できる汎用メカニズムを提供します。

バッチ処理サーバー
プールの機能を使用して、1 つのサーバーを 2 つのプールに分割します。一方のプールは、ログインセッションとタイムシェアリングユーザーによる対話型作業に使用されます。もう一方のプールは、バッチシステムを介して投入されるジョブに使用されます。

アプリケーションサーバーまたはデータベースサーバー
アプリケーションの要件に基づいて、対話型アプリケーション用のリソースを区分します。

アプリケーションの段階的な調整
ユーザーが期待するサービスレベルを設定します。

最初は、目標とする最終的なサービスの一部だけを実行するマシンを導入することがあります。マシンをオンラインにしたときに、予約方式のリソース管理メカニズムが確立していなければ、ユーザーがサービスに不満を持つ可能性があります。

たとえば、公平配分スケジューラは CPU の使用効率を最適化します。1 つしかアプリケーションを実行していないマシンの応答時間は速く感じられます。実際には、複数のアプリケーションがロードされると、このような応答時間は得られません。アプリケーションごとに個別のプールを用意することにより、各アプリケーションで使用可能な CPU 数の上限をあらかじめ設定してから、すべてのアプリケーションを運用することができます。

複雑なタイムシェアリングサーバー
多数のユーザーをサポートするサーバーを区分します。サーバーの区分によって、ユーザーごとの応答が時間をより確実に予測できる分離メカニズムが提供されます。

ユーザーをグループに分割して個別のプールに結合し、公平配分スケジューラ (FSS) 機能を使用すれば、CPU 割り当てを調整して、優先順位を持つユーザーグループをサポートできます。このような割り当ては、ユーザーの役割や課金などに基づいて行えます。

定期的に変動する作業負荷
リソースプールを使用して、変動する作業負荷に対応します。

サイトでの作業負荷の変動が月次、四半期、年次などの周期で予想できる場合があります。サイトでこのような変動が予想できる場合は、`cron` ジョブで `pooladm` を実行して、複数のプール構成を使い分けることができます。(152 ページの「リソースプールのフレームワーク」を参照)。

リアルタイムアプリケーション
RT スケジューラと専用のプロセッサリソースを使用して、リアルタイムプールを作成します。

システムの使用効率
システムの目標を設定して適用します。

自動プールデーモンの機能を使用して、使用可能なリソースを特定してから作業負荷を監視すると、指定した目標がいつ満たされなくなるかを検出できます。可能な場合はデーモンで修正操作を実行したり、状況をログに記録したりできます。

リソースプールのフレームワーク

`/etc/pooladm.conf` 構成ファイルには、静的なプール構成が記述されます。静的構成では、リソースプール機能に関連して管理者がシステムをどのように構成するかを記述できます。別のファイル名を指定することもできます。

サービス管理機能 (SMF) または `pooladm -e` コマンドを使ってリソースプールフレームワークを有効にする場合で、`/etc/pooladm.conf` ファイルが存在するときは、このファイル内の構成がシステムに適用されます。

リソースプールフレームワーク内でのリソースの処置に関する情報は、カーネルで保持されます。これは動的構成と呼ばれ、特定のシステムの、ある時点でのリソースプール機能を表します。動的構成を表示するには、`pooladm` コマンドを使用します。プールやリソースセットについて表示されるプロパティーの順序は、場合によって異なります。動的構成に対する変更は、次の方法で行われます。

- 静的構成ファイルを適用する、間接的な方法
- `poolcfg` コマンドに `-d` オプションを使用する、直接的な方法

場合に応じて起動できるように、複数の静的プール構成ファイルを作成しておくことができます。`cron` ジョブで `pooladm` を起動して、複数のプール構成を使い分けることができます。`cron` ユーティリティーの詳細は、`cron(1M)` のマニュアルページを参照してください。

デフォルトでは、リソースプールフレームワークは無効になっています。動的構成を作成したり変更したりするには、リソースプールが有効になっている必要があります。リソースプールフレームワークが無効になっている場合でも、`poolcfg` コマンドまたは `libpool` コマンドを使って静的構成ファイルを操作することはできます。プール機能が無効になっている場合、静的構成ファイルを作成することはできません。構成ファイルの詳細については、155 ページの「プール構成の作成」を参照してください。

リソースプールおよび `poold` システムデーモンで使用するコマンドについては、次のマニュアルページに記載されています。

- `pooladm(1M)`
- `poolbind(1M)`
- `poolcfg(1m)`
- `poold(1M)`
- `poolstat(1M)`
- `libpool(3LIB)`

/etc/pooladm.conf の内容

次の要素は、動的構成も含め、すべてのリソースプール構成に使用できます。

system　システムの全体的な動作に影響を与えるプロパティー

プール　リソースプールの定義

pset　プロセッサセットの定義

cpu　プロセッサの定義

これらの要素に含まれているプロパティーを操作することで、リソースプールフレームワークの状態と動作を変更できます。たとえば、プールプロパティー pool.importance は、プールの相対的な重要性を示します。このプロパティーは、リソースの競合が発生した場合の解決に使用されます。詳細は、libpool(3LIB) のマニュアルページを参照してください。

プールのプロパティー

プール機能では、プール、リソース、またはコンポーネントに設定される、名前と型の指定されたプロパティーがサポートされています。管理者は、プールのさまざまな要素に対して、追加のプロパティーを設定することもできます。プロジェクト属性に似たプロパティー名前空間が使用されます。

たとえば、次のコメントは、特定の Datatree データベースに pset が関連付けられていることを示します。

```
Datatree,pset.dbname=warehouse
```

プロパティーの型の詳細については、161 ページの「poold のプロパティー」を参照してください。

注 – いくつかの特殊プロパティーが内部使用のために予約されています。これらを設定したり削除したりすることはできません。詳細は、libpool(3LIB) のマニュアルページを参照してください。

システム上でのプールの実装

ユーザー定義のプールをシステム上に実装するには、次のどちらかの方法を使用します。

- Solaris ソフトウェアがブートすると、init スクリプトは /etc/pooladm.conf ファイルが存在するかどうかを検査します。このファイルが検出され、プールが有効化されると、pooladm が呼び出され、この構成をアクティブなプール構成に

します。システムは、/etc/pooladm.conf で指定されている編成に従って、動的な構成を作成し、マシンのリソースは指定どおりに区分されます。

- Solaris システムが起動しているときは、pooladm コマンドを使用して、プール構成が存在しない場合はプール構成を起動したり、プール構成を変更したりできます。デフォルトでは、pooladm コマンドは /etc/pooladm.conf の内容を使用します。ただし、別のディレクトリとファイル名を指定し、そのファイルを使用してプール構成を変更することもできます。

リソースプールを有効化または無効化する方法については、173 ページの「プール機能の有効化と無効化」を参照してください。ユーザー定義のプールやリソースが使用されている間は、プール機能を無効にすることはできません。

リソースプールを構成するには、スーパーユーザーの特権を持っているか、またはプロファイルの一覧内に Process Management プロファイルが含まれている必要があります。System Administrator 役割には Process Management プロファイルが含まれています。

poold リソースコントローラは、動的リソースプール機能とともに起動されます。

project.pool 属性

/etc/project ファイル内のプロジェクトエントリに project.pool 属性を追加すると、そのエントリに単一のプールを関連付けることができます。プロジェクトで開始される新しい作業は、適切なプールに結合されます。詳細は、第 2 章「プロジェクトとタスク (概要)」を参照してください。

たとえば、projmod コマンドを使用して、/etc/project ファイル内のプロジェクト *sales* に project.pool 属性を設定できます。

```
# projmod -a -K project.pool=mypool sales
```

SPARC: 動的再構成の処理とリソースプール

動的再構成 (DR) を使用すると、システムの実行中にハードウェアを再構成できます。DR 操作により、特定の種類のリソースが増加または減少することもあれば、影響を受けないこともあります。DR は使用可能なリソース量に影響を与えることがあるので、プール機能を DR 操作に含めておく必要があります。DR 処理が開始されると、プールのフレームワークは構成の妥当性を検証します。

現在のプール構成が無効にならないかぎり、DR 処理は、独自の構成ファイルが更新されるまで実行を続けます。無効な構成とは、使用可能なリソースでサポートできない構成のことです。

DR 処理によってプール構成が無効になった場合、操作は失敗し、メッセージログにメッセージが書き込まれます。構成処理を強制的に最後まで実行するには、DR の強制オプションを使用します。強制オプションで処理を続行すると、プール構成は、新しいリソース構成に合うように変更されます。DR 処理と強制オプションについては、使用している Sun ハードウェアの動的再構成に関するユーザーガイドを参照してください。

動的リソースプールを使用している場合、`poold` デーモンがアクティブになっている間に、その制御からパーティションが除外されることがあります。詳細は、166 ページの「リソース不足の特定」を参照してください。

プール構成の作成

構成ファイルには、システム上で作成されるプールに関する記述が含まれます。構成ファイルには、操作可能な構成要素が記述されています。

- システム
- pool
- pset
- cpu

操作可能な構成要素については、`poolcfg(1m)` を参照してください。

プールが有効になっている場合、構造化された `/etc/pooladm.conf` ファイルを次の 2 つの方法で作成できます。

- `pooladm` コマンドに `-s` オプションを付けて実行して、現在のシステム上のリソースを検出し、その結果を構成ファイルに記録します。

  この方法をお勧めします。プール機能で操作できるシステム上のアクティブなリソースとコンポーネントがすべて記録されます。リソースには、既存のプロセッサセットの構成が含まれます。最後に、プロセッサセットの名前を変更したり、必要に応じてプールを作成したりして、構成を変更できます。
- `poolcfg` コマンドに `-c` オプションと `discover` サブコマンドまたは `create system` *name* サブコマンドを付けて実行して、新しいプール構成を作成します。

  これらのオプションは、以前のリリースとの下位互換性を保つために残されています。

`/etc/pooladm.conf` ファイルを変更するには、`poolcfg` または `libpool` を使用します。このファイルを直接編集しないでください。

動的構成の直接操作

poolcfg コマンドに -d オプションを付けて実行すると、動的構成内の CPU リソースタイプを直接操作できます。リソースを転送するには、次の 2 つの方法があります。

- 識別された利用可能なリソースすべてをセット間で転送するように要求します。
- 特定の ID を持つリソースだけをターゲットセットに転送します。リソース構成が変更されたときやシステムのリブート後は、リソースに関連付けられているシステム ID が変わることがあります。

具体例は、187 ページの「リソースの転送」を参照してください。

リソース転送によって poold からアクションが引き起こされることがあります。詳細は、156 ページの「poold の概要」を参照してください。

poold の概要

プールのリソースコントローラ poold は、システムターゲットと観察可能な統計情報を使用して、管理者によって指定されたシステム性能の目標を維持します。動的リソース割り当てが必要なときは、常にこのシステムデーモンをアクティブにしておく必要があります。

poold リソースコントローラは、使用可能なリソースを特定してから作業負荷を監視して、システム使用率に関する目標がいつ満たされなくなるかを検出できます。poold は、これらの目標の観点から別の構成を検討し、改善操作を実行します。可能な場合は、目標を満たすようにリソースを再構成します。この操作が不可能な場合は、ユーザーによって指定された目標が達成不可能になったことをログに記録します。再構成を行なった後、デーモンは作業負荷目標の監視を再開します。

poold では決定の履歴が保持され、確認に使用されます。決定履歴を使用すると、それまでに行なった再構成のうち、改善効果を示さなかったものを削除できます。

作業負荷の目標が変更された場合や、システムで使用可能なリソースが変更された場合は、再構成が非同期に実行されることもあることに注意してください。

動的リソースプールの管理

DRP サービスは、サービス識別子 svc:/system/pools/dynamic 下のサービス管理機能 (SMF) によって管理されます。

有効化、無効化、再起動の要求など、このサービスに関する管理作業は、svcadm コマンドを使用して実行できます。サービスのステータスは、svcs コマンドを使用して照会できます。詳細は、svcs(1) および svcadm(1M) のマニュアルページを参照してください。

DRP の制御方法として望ましいのは SMF インタフェースですが、下位互換性を維持するために、次の方法も使用できます。

- 動的リソース割り当てが不要になった場合は、SIGQUIT シグナルまたは SIGTERM シグナルを使って poold を停止できます。これらのシグナルはどちらも、poold を正常に終了させます。
- poold では、リソースやプールの構成の変更が自動的に検出されます。ただし、SIGHUP シグナルを使用して、再構成を強制的に実行することもできます。

構成の制約と目標

poold は、管理者の指示に基づいて再構成を行います。これらの指示は、一連の制約および目標として指定します。poold はこれらの指定に基づき、可能性のあるさまざまな構成を、既存の構成に対する相対値として決定します。次に poold は、現在の構成のリソース割り当てを変更して、候補となる新しい構成を生成します。

構成の制約

制約は、構成に加えることのできる変更を一部除外することで、作成可能な構成の範囲に影響を与えます。libpool 構成で次の制約を指定できます。

- CPU 割り当ての最小値と最大値
- セットから移動できない固定コンポーネント

プールプロパティーの詳細については、libpool(3LIB) のマニュアルページと153 ページの「プールのプロパティー」を参照してください。

pset.min プロパティーと pset.max プロパティーの制約

これら 2 つのプロパティーは、プロセッサセットに割り当てることのできるプロセッサの最小数と最大数を制限します。これらのプロパティーの詳細については、表 12–1 を参照してください。

同じ Solaris インスタンスのリソースパーティションどうしであれば、これらの制約の範囲内で、あるパーティションから別のパーティションにリソースを割り当てることができます。リソースセットに関連付けられているプールに結合することで、リソースにアクセスできるようになります。この結合はログイン時に実行されるか、または、PRIV_SYS_RES_CONFIG 特権を持っている管理者が手動で行います。

cpu.pinned プロパティーの制約

cpu-pinned プロパティーは、DRP で特定の CPU を現在のプロセッサセットから移動してはならないことを示します。この libpool プロパティーを設定すると、プロセッサセット内で実行されている特定のアプリケーションでのキャッシュ使用効率を最大限に高めることができます。

このプロパティーの詳細については、表 12–1 を参照してください。

pool.importance プロパティーの制約

pool.importance プロパティーは、管理者が定義した、プールの相対的な重要度を示します。

構成の目標

目標は制約と同様の方法で指定されます。目標の全一覧については、表 12–1 を参照してください。

目標には 2 つのカテゴリがあります。

作業負荷に依存する目標
: 作業負荷に依存する目標とは、システムで実行される作業負荷の性質によって変化する目標です。たとえば、utilization 目標などがあります。リソースセットの使用効率の数値は、そのセットでアクティブになっている作業負荷の性質によって変化します。

作業負荷に依存しない目標
: 作業負荷に依存しない目標とは、システムで実行される作業負荷の性質によって変化しない目標です。たとえば、CPU の locality 目標などがあります。リソースセットの近傍性の評価値は、そのセットでアクティブになっている作業負荷の性質によって変化することはありません。

次の 3 種類の目標を定義できます。

名前	有効な要素	演算子	値
wt-load	system	なし	なし
locality	pset	なし	loose|tight|none
utilization	pset	<>~	0–100%

目標は、libpool 構成内のプロパティー文字列に格納されます。プロパティー名は、次のとおりです。

- system.poold.objectives
- pset.poold.objectives

目標の構文は次のとおりです。

- objectives = objective [; objective]*
- objective = [n:] keyword [op] [value]

どの目標にも、オプションで重要性を表す接頭辞を付けることができます。重要性の値は目標に乗算され、この目標が目標関数の評価に与える影響を高めます。指定できる範囲は、0 から INT64_MAX (9223372036854775807) までです。指定されていない場合、重要性のデフォルト値は 1 です。

一部の要素タイプでは、複数の種類の目標がサポートされています。pset などはその一例です。このような要素には、複数の種類の目標を指定できます。また、1 つの pset 要素に複数の使用効率目標を指定することもできます。

使用例については、184 ページの「構成の目標を定義する方法」を参照してください。

wt-load 目標

wt-load 目標では、リソースの使用効率に合わせてリソースを割り当てるような構成が有利に導かれます。この目標が有効になっていると、より多くのリソースを使用するリソースセットには、より多くのリソースが与えられることになります。wt-load は「重み付けされた負荷 (weighted load)」を意味します。

最小値と最大値のプロパティーを使って満足のいく制約を設定したあとで、これらの制約の範囲内でデーモンが自由にリソースを操作できるようにする場合に、この目標を使用してください。

locality 目標

locality 目標は、近傍性グループ (lgroup) データによって測定される近傍性が、選択された構成に対して与える影響を変化させます。近傍性は応答時間と定義することもできます。lgroup は、CPU リソースとメモリーリソースを表します。lgroup は、Solaris システムでリソースどうしの距離を調べるために使用されます。測定単位は時間です。近傍性グループによる抽象化については、『プログラミングインタフェース』の「近傍性グループの概要」を参照してください。

この目標には、次の 3 つの値のいずれかを指定できます。

tight　この値を設定すると、リソースの近傍性を最大にするような構成が有利に導かれます。

loose　この値を設定すると、リソースの近傍性を最小にするような構成が有利に導かれます。

none　この値を設定すると、どのような構成が有利になるかは、リソースの近傍性に依存しません。これは locality 目標のデフォルト値です。

一般に、locality 目標は tight に設定することをお勧めします。ただし、メモリー帯域幅を最大にする場合や、リソースセットに対する DR 操作の影響を最小にする場合は、この目標を loose に設定するか、デフォルト値の none にしておいてください。

utilization 目標

utilization 目標では、指定された使用効率目標を満たしていないパーティションにリソースを割り当てるような構成が有利に導かれます。

この目標は、演算子と値で指定されます。演算子は次のとおりです。

<　「小なり」演算子は、指定された値が最大のターゲット値であることを示します。

>　「大なり」演算子は、指定された値が最小のターゲット値であることを示します。

~　「ほぼ等しい」演算子は、指定された値がターゲット値であり、いくらかの変動が許容されることを示します。

pset には、演算子の種類ごとに使用効率目標を 1 つ設定できます。

- ~ 演算子を設定した場合、< 演算子や > 演算子は設定できません。
- < 演算子や > 演算子を設定した場合、~ 演算子は設定できません。< 演算子の設定と > 演算子の設定が互いに矛盾してはならないことに注意してください。

< 演算子と > 演算子の両方を使用すると、範囲を指定できます。これらの値は、重複しないように検証されます。

構成目標の例

次の例では、pset に対する次のような目標が poold によって評価されます。

- utilization を 30% から 80% の範囲に保つ必要がある。
- プロセッサセットの locality を最大にする必要がある。
- 各目標の重要性はデフォルト値の 1 とする。

例 12–1　poold の目標の例

```
pset.poold.objectives "utilization > 30; utilization < 80; locality tight"
```

その他の使用例については、184 ページの「構成の目標を定義する方法」を参照してください。

poold のプロパティー

プロパティーには 4 つのカテゴリがあります。

- 構成
- 制約
- 目標
- 目標パラメータ

表 12–1 定義済みのプロパティー名

プロパティー名	タイプ	カテゴリ	説明
system.poold.log-level	string	構成	ロギングレベル
system.poold.log-location	string	構成	ログの場所
system.poold.monitor-interval	uint64	構成	監視のサンプリング間隔
system.poold.history-file	string	構成	決定履歴の場所
pset.max	uint64	制約	このプロセッサセットに割り当てられる CPU の最大数
pset.min	uint64	制約	このプロセッサセットに割り当てられる CPU の最小数
cpu.pinned	bool	制約	CPU がこのプロセッサセットに固定されているかどうか
system.poold.objectives	string	目標	poold の目標式の構文に従った書式付き文字列
pset.poold.objectives	string	目標	poold の目標式の構文に従った書式付き文字列
pool.importance	int64	目標パラメータ	ユーザーが割り当てた重要性

poold の構成可能な機能

デーモンの動作の次のような部分は構成可能です。

- 監視の間隔
- ロギングレベル
- ログの場所

これらのオプションは、プール構成で指定します。コマンド行から poold を起動する方法でも、ログレベルを制御できます。

poold の監視間隔

プロパティー名 system.poold.monitor-interval を使用して、値をミリ秒単位で指定します。

poold のログ情報

ログを通じて 3 つのカテゴリの情報が提供されます。これらのカテゴリは、ログで次のように識別されています。

- 構成
- 監視
- 最適化

プロパティー名 system.poold.log-level を使用して、ログパラメータを指定します。このプロパティーが指定されていない場合、ログレベルのデフォルト値は NOTICE です。パラメータのレベルは階層的になっています。ログレベルとして DEBUG を設定すると、poold は、すべての定義済みメッセージをログに記録します。INFO レベルでは、ほとんどの管理者にとって適度な情報が得られます。

コマンド行で poold コマンドに -l オプションとパラメータを付けて実行すると、生成するログ情報のレベルを指定できます。

次のパラメータを使用できます。

- ALERT
- CRIT
- ERR
- WARNING
- NOTICE
- INFO
- DEBUG

これらのパラメータレベルは、syslog に使用される同様のレベルと直接対応づけられます。syslog の使用方法の詳細については、164 ページの「ログの場所」を参照してください。

poold のログ機能を構成する方法の詳細については、186 ページの「poold のログレベルを設定する方法」を参照してください。

情報ログ機能の構成

生成されるメッセージの種類は次のとおりです。

ALERT　libpool 構成へのアクセスに関連する問題など、libpool 機能の基本的な予期せぬ障害。これによってデーモンは終了します。今すぐ管理者が対応する必要があります。

CRIT　予期せぬ障害に起因する問題。これによってデーモンは終了します。今すぐ管理者が対応する必要があります。

ERR　処理を制御するユーザー指定のパラメータに関連する問題。リソースセットの使用効率目標が衝突していて、解決不能な場合などです。管理者が介入して目標を修正する必要があります。poold は、衝突している目標を無視することで、改善操作を試みます。ただし、エラーによっては、デーモンが終了することもあります。

WARNING　構成パラメータの設定に関連する警告。構成パラメータの設定が技術的には正しくても、特定の実行環境には適さない場合などです。たとえば、すべての CPU リソースを固定にすると、poold はプロセッサセット間で CPU リソースを移動することができなくなります。

DEBUG　構成処理をデバッグするときに必要となる詳細情報が含まれたメッセージ。通常は、管理者がこの情報を使用することはありません。

情報ログ機能の監視

生成されるメッセージの種類は次のとおりです。

CRIT　予期せぬ監視障害に起因する問題。これによってデーモンは終了します。今すぐ管理者が対応する必要があります。

ERR　予期せぬ監視エラーに起因する問題。管理者が介入して修正する必要がある場合もあります。

NOTICE　リソース制御の領域移行に関するメッセージ。

INFO　リソース使用効率の統計情報に関するメッセージ。

DEBUG　監視処理をデバッグするときに必要となる詳細情報が含まれたメッセージ。通常は、管理者がこの情報を使用することはありません。

情報ログ機能の最適化

生成されるメッセージの種類は次のとおりです。

WARNING　これらのメッセージは、最適な決定を行う際の問題に関連して表示されることがあります。たとえば、最小値と最大値または固定コンポーネントの数によって、厳しすぎる制約をリソースセットに与えている場合などです。

これらのメッセージは、最適な再割り当てを行う際の、予期せぬ制限に起因する問題について表示されることがあります。たとえば、リソースを消費するプロセスがバインドされているプロセッサセットから最後のプロセッサを削除する場合などです。

NOTICE　これらのメッセージは、使用可能な構成や、決定履歴の上書きに起因して実装されない構成について表示されることがあります。

INFO　これらのメッセージは、考慮される代替構成について表示されることがあります。

DEBUG　最適化処理をデバッグするときに必要となる詳細情報が含まれたメッセージ。通常は、管理者がこの情報を使用することはありません。

ログの場所

system.poold.log-location プロパティーを使用して、poold のログの出力先を指定します。poold の出力先として SYSLOG の場所を指定できます (syslog(3C) のマニュアルページを参照)。

このプロパティーが指定されていない場合、poold ログのデフォルトの出力先は /var/log/pool/poold です。

コマンド行から poold を呼び出す場合、このプロパティーは使用しません。ログエントリは、呼び出しを行なった端末の stderr に出力されます。

logadm によるログ管理

poold がアクティブになっている場合、logadm.conf ファイル内に、デフォルトファイル /var/log/pool/poold を管理するためのエントリが含まれています。このエントリは次のとおりです。

```
/var/log/pool/poold -N -s 512k
```

logadm(1M) および logadm.conf(4) のマニュアルページを参照してください。

動的リソース割り当てのしくみ

このセクションでは、リソースを動的に割り当てるために poold で使用される手順と要因について説明します。

使用可能なリソースについて

poold プロセスの有効範囲内で使用できるすべてのリソースが、使用可能なリソースと見なされます。1 つの Solaris インスタンスが最大の制御範囲になります。

ゾーンが有効になっているシステムの場合、実行中の poold インスタンスの有効範囲は、大域ゾーンに制限されます。

使用可能なリソースの特定

リソースプールには、アプリケーションで消費できるすべてのシステムリソースが含まれます。

実行中の Solaris インスタンスごとに、1 つのパーティションには、CPU など 1 種類のリソースを割り当てる必要があります。1 種類のリソースを 1 つまたはそれ以上のパーティションに割り当ててもかまいません。各パーティションには、一意のリソースセットが含まれます。

たとえば、4 つの CPU と 2 つのプロセッサセットを持つマシンは、次のように設定されます。

pset 0: 0 1

pset 1: 2 3

ここで、コロンのあとの 0、1、2、3 という数字は、CPU の ID を表しています。これら 2 つのプロセッサセットで、4 つの CPU すべてが使用されていることに注目してください。

このマシンで次のような設定は不可能です。

pset 0: 0 1

pset 1: 1 2 3

CPU 1 を同時に割り当てることができる pset は 1 つだけなので、このような設定はできません。

リソースが属しているパーティション以外のパーティションからは、そのリソースにアクセスすることはできません。

使用可能なリソースを発見するために、poold はアクティブなプール構成を調べてパーティションを見つけます。制御対象となっているリソースの種類ごとに、すべてのパーティションに含まれているすべてのリソースが集計され、使用可能なリソースの合計量が求められます。

poold では、このリソース量を基にして処理が行われます。ただし、この基本量に制約が適用され、poold で割り当てを行う必要がある際の柔軟性が制限されることもあります。使用可能な制約については、157 ページの「構成の制約」を参照してください。

リソース不足の特定

poold の制御範囲とは、効率のよい区分と管理を主に poold が担当している、使用可能なリソースセットのことです。ただし、この制御範囲にあるリソースを操作できるメカニズムがほかにもあり、それが構成に影響を与えることもあります。poold がアクティブになっている間に特定のパーティションが制御範囲外になった場合、poold は使用可能なリソースを適切に操作することによってその制御を取り戻そうとします。poold デーモンは、有効範囲内に追加のリソースを見つけられない場合、リソース不足に関する情報をログに記録します。

リソース使用効率の判定

poold は通常、その制御範囲にあるリソースの使用率を監視することに時間の大部分を費やします。この監視は、作業負荷に依存する目標が満たされているかどうかを確認するために実行されます。

たとえば、プロセッサセットの場合、セット内のすべてのプロセッサについてすべての測定が実行されます。リソース使用効率は、サンプリング間隔に対して、リソースが使用された時間の割合を示します。リソース使用効率はパーセンテージで表され、0 - 100 の値です。

制御違反の特定

157 ページの「構成の制約と目標」で説明した指示は、システムが目標を満たさないことを検出するために使用されます。 これらの目標は、作業負荷に直接関連しています。

ユーザーが構成した目標を満たしていないパーティションは、制御違反です。制御違反には、同期と非同期の 2 種類があります。

- 同期目標違反は、デーモンによって作業負荷の監視中に検出されます。
- 非同期目標違反は、デーモンの監視動作とは無関係に発生します。

非同期の目標違反は、次のようなイベントによって引き起こされます。

- 制御範囲に対してリソースが追加または削除された。
- 制御範囲が再構成された。
- poold リソースコントローラが再起動された。

作業負荷に関連しない目標が目標関数の評価に与える影響は、評価のたびに一定であると見なされます。作業負荷に関連しない目標が再評価されるのは、いずれかの非同期違反によって再評価処理が引き起こされたときだけです。

適切な改善操作の決定

リソースコントローラは、リソースを消費するプロセスでリソースが不足していると判定した場合、まずそのリソースを増やして性能を改善しようとします。

制御範囲について構成で指定された目標を満たすように、別の構成が検討され評価されます。

この処理では、リソースの移動結果を監視し、各リソースパーティションの応答性を評価しながら、徐々に細かい調整が行われます。決定履歴を参照して、それまでに行なった再構成のうちで改善効果を示さなかったものが削除されます。履歴データの関連度をより詳しく評価するために、プロセスの名前や数量といったほかの情報も使用されます。

修正操作を実行できない場合、デーモンは状況をログに記録します。詳細は、162 ページの「poold のログ情報」を参照してください。

poolstat によるプール機能とリソース使用効率の監視

システムのプールが有効になっている場合にリソースの使用効率を監視するには、poolstat ユーティリティーを使用します。このユーティリティーは、システム上でアクティブになっているすべてのプールを調べ、選択された出力モードに基づいて統計情報を報告します。poolstat の統計を使用すると、どのリソースパーティションの使用効率が高いかを判定できます。これらの統計情報を解析して、システムで多くのリソースが要求されたときにリソースの再割り当てを行う方法を決定できます。

poolstat ユーティリティーのオプションを使用すると、特定のプールを調べたり、リソースセット固有の統計情報を報告したりできます。

システムにゾーンが実装されている場合は、非大域ゾーンで poolstat を使用すると、そのゾーンのプールに関連付けられているリソースに関する情報が表示されます。

poolstat ユーティリティーの詳細については、poolstat(1M) のマニュアルページを参照してください。poolstat のタスクと使用方法については、192 ページの「poolstat を使ってプールに関連付けられているリソースについて統計情報を報告する」を参照してください。

poolstat の出力

デフォルトの出力形式では、poolstat は、見出し行を出力したあと、プールごとに 1 行ずつ表示されます。各プールの行は、プール ID とプール名で始まります。それに続く列は、プールに接続されているプロセッサセットに関する統計データです。複数のプールに接続されているリソースセットは、各プールで 1 回ずつ表示されるので、複数回表示されます。

列見出しは次のとおりです。

id　プール ID。

pool　プール名。

rid　リソースセット ID。

rset　リソースセット名。

type　リソースセットの種類。

min　リソースセットの最小サイズ。

max　リソースセットの最大サイズ。

size　リソースセットの現在のサイズ。

used　リソースセットのうちで現在使用されている部分のサイズ。

この値は、リソースセットの使用効率にリソースセットのサイズを乗算して計算されます。前回のサンプリング間隔でリソースセットが再構成されている場合、この値は報告されないことがあります。報告されなかった値はハイフン (-) で示されます。

load　リソースセットに加えられている負荷の絶対値。

このプロパティーの詳細については、libpool(3LIB) のマニュアルページを参照してください。

poolstat の出力では、次のことを指定できます。

- 列の順序
- 表示する見出し

poolstat の動作間隔の調整

poolstat で実行する操作をカスタマイズできます。報告用のサンプリング間隔、および統計を繰り返す回数を設定できます。

interval poolstat が実行する定期的な処理の間隔を調整します。すべての間隔は秒単位で指定します。

count 統計を繰り返す回数を指定します。デフォルトでは、poolstat は 1 回だけ統計情報を報告します。

interval と *count* を指定しなかった場合、統計は 1 回だけ報告されます。*interval* を指定し、*count* を指定しなかった場合、統計報告が無限に繰り返されます。

リソースプール機能で使用するコマンド

次の表に記載されているコマンドは、プール機能を管理するための主要なインタフェースとなります。ゾーンが有効になっているシステムでこれらのコマンドを使用する方法については、150 ページの「ゾーンで使用されるリソースプール」を参照してください。

マニュアルページの参照	説明
pooladm(1M)	システムのプール機能を有効または無効にします。特定の構成をアクティブにします。または、現在の構成を削除して、関連付けられているリソースをデフォルトのステータスに戻します。オプションを指定しないで実行すると、pooladm は、現在の動的プール構成を表示します。
poolbind(1M)	プロジェクト、タスク、およびプロセスを手動でリソースプールに結合できます。

マニュアルページの参照	説明
poolcfg(1m)	プールやセットに対する構成操作を実行します。このツールを使って作成された構成は、pooladm によってターゲットホスト上でインスタンス化されます。 poolcfg に -c オプションと info サブコマンド引数を付けて実行すると、/etc/pooladm.conf に保存されている静的構成の情報が表示されます。このコマンドにファイル名の引数を追加すると、そのファイルに保存されている静的構成の情報が表示されます。たとえば、poolcfg -c info /tmp/newconfig では、/tmp/newconfig というファイルに保存されている静的構成の情報が表示されます。
poold(1M)	プールシステムデーモン。このデーモンは、システムターゲットと観察可能な統計情報を使用して、管理者によって指定されたシステム性能の目標を維持します。目標が満たされていないときに修正操作を実行できない場合、poold は状況をログに記録します。
poolstat(1M)	プールに関連付けられているリソースについて統計情報を表示します。システム管理者にとって性能解析が簡単になり、リソースを区分または再区分するタスクに役立つ情報が得られます。特定のプールを調べたり、リソースセット固有の統計情報を報告したりするためのオプションも用意されています。

ライブラリの API は、libpool で提供されます (libpool(3LIB) のマニュアルページを参照)。プログラムからプール構成を操作するには、このライブラリを使用します。

◆ ◆ ◆ 第 13 章

リソースプールの作成と管理 (タスク)

この章では、システムのリソースプールを設定し管理する方法について説明します。

リソースプールの概要については、第 12 章「リソースプール (概要)」を参照してください。

動的リソースプールの管理 (タスクマップ)

タスク	説明	参照先
リソースプールを有効または無効にします。	システムのリソースプールをアクティブまたは無効にします。	173 ページの「プール機能の有効化と無効化」
動的リソースプールを有効または無効にします。	システムの動的リソースプール機能をアクティブまたは無効にします。	173 ページの「プール機能の有効化と無効化」
静的なリソースプール構成を作成します。	現在の動的構成と一致する静的構成ファイルを作成します。詳細は、152 ページの「リソースプールのフレームワーク」を参照してください。	178 ページの「静的構成を作成する方法」
リソースプール構成を変更します。	追加のプールを作成するなどして、システムのプール構成を変更します。	179 ページの「構成の変更方法」

タスク	説明	参照先
リソースプールをスケジューリングクラスに対応付けます。	プールをスケジューリングクラスに対応付けることで、プールに結合されているすべてのプロセスが、指定されたスケジューラを使用できるようにします。	181 ページの「プールをスケジューリングクラスに対応付ける方法」
構成の制約を設定し、構成の目標を定義します。	poold に対して、修正操作を実行するときに考慮する目標を指定します。構成の目標の詳細については、156 ページの「poold の概要」を参照してください。	183 ページの「構成の制約を設定する方法」および184 ページの「構成の目標を定義する方法」
ログのレベルを設定します。	poold で生成するログ情報のレベルを指定します。	186 ページの「poold のログレベルを設定する方法」
poolcfg コマンドでテキストファイルを使用します。	poolcfg コマンドにテキストファイルから入力します。	186 ページの「poolcfg でコマンドファイルを使用する方法」
カーネルでリソースを転送します。	カーネルでリソースを転送します。たとえば、特定の ID を持つリソースをターゲットセットに転送します。	187 ページの「リソースの転送」
プール構成を起動します。	デフォルト構成ファイル内の構成を起動します。	188 ページの「プール構成を起動する方法」
プール構成を確定する前に、プール構成を検証します。	検証が実行されるとどうなるかをテストするために、プール構成を検証します。	188 ページの「構成を確定する前に構成を検証する方法」
システムからプール構成を削除します。	プロセッサセットなど、対応付けられているすべてのリソースがデフォルトのステータスに戻ります。	189 ページの「プール構成を削除する方法」
プロセスをプールに結合します。	システムで実行中のプロセスをリソースプールに手動で対応付けます。	190 ページの「プロセスをプールに結合する方法」
タスクやプロジェクトをプールに結合します。	タスクやプロジェクトをリソースプールに対応付けます。	190 ページの「タスクまたはプロジェクトをプールに結合する方法」
新しいプロセスをリソースプールに結合します。	プロジェクト内の新しいプロセスを特定のプールに自動的に結合するには、project データベース内の各エントリに属性を追加します。	191 ページの「プロジェクトの project.pool 属性を設定する方法」

タスク	説明	参照先
project 属性を使ってプロセスを別のプールに結合します。	新たに開始されるプロセスについて、プールとの結合を変更します。	191 ページの「project 属性を使ってプロセスを別のプールに結合する方法」
poolstat ユーティリティーを使って報告を生成します。	指定した間隔で複数の報告を生成します。	192 ページの「特定の間隔で複数の報告を生成する」
リソースセットの統計情報を報告します。	poolstat ユーティリティーを使って pset リソースセットの統計情報を報告します。	192 ページの「リソースセットの統計情報を報告する」

プール機能の有効化と無効化

Solaris 10 11/06 リリースから、svcadm コマンド (svcadm(1M) のマニュアルページを参照) を使って、システム上のリソースプールサービスおよび動的リソースプールサービスを有効または無効に設定できるようになりました。

pooladmコマンド (pooladm(1M) のマニュアルページを参照) を使用すると、次のタスクも実行できます。

- プール機能を有効にして、プールを操作できるようにします
- プール機能を無効にして、プールを操作できないようにします

注 – システムをアップグレードする際に、リソースプールフレームワークが有効で、/etc/pooladm.conf ファイルが存在する場合、プールサービスが有効になり、このファイル内の構成がシステムに適用されます。

▼ Solaris 10 11/06 以降: svcadm を使ってリソースプールサービスを有効にする方法

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 リソースプールサービスを有効にします。

```
# svcadm enable system/pools:default
```

▼ Solaris 10 11/06 以降: svcadm を使ってリソースプールサービスを無効にする方法

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 リソースプールサービスを無効にします。

```
# svcadm disable system/pools:default
```

▼ Solaris 10 11/06 以降: svcadm を使って動的リソースプールサービスを有効にする方法

1 スーパーユーザーになるか、**Service Management** 権利プロファイルが含まれている役割を引き受けます。

役割には、認証と特権コマンドが含まれます。役割の作成方法およびユーザーに役割を割り当てる方法については、『Solaris のシステム管理 (セキュリティサービス)』の「RBAC の構成 (作業マップ)」と『Solaris のシステム管理 (セキュリティサービス)』の「RBAC の管理(作業マップ)」を参照してください。

2 動的リソースプールサービスを有効にします。

```
# svcadm enable system/pools/dynamic:default
```

例 13–1 リソースプールサービスに対する動的リソースプールサービスの依存

この例では、DRP を実行する場合に、最初にリソースプールを有効にする必要があることを示します。

リソースプールと動的リソースプールの間には、依存関係があります。DRP は、リソースプールに依存するサービスになっています。DRP の有効化/無効化は、リソースプールとは関係なく実行できます。

次の例では、リソースプールと動的リソースプールの両方が現在無効に設定されています。

```
# svcs *pool*
STATE          STIME    FMRI
disabled       10:32:26 svc:/system/pools/dynamic:default
disabled       10:32:26 svc:/system/pools:default
```

動的リソースプールを有効にします。

```
# svcadm enable svc:/system/pools/dynamic:default
# svcs -a | grep pool
disabled       10:39:00 svc:/system/pools:default
offline        10:39:12 svc:/system/pools/dynamic:default
```

DRP サービスはまだオフラインです。

svcs コマンドの -x オプションを使って、DRP サービスがオフラインになっている原因を特定します。

```
# svcs -x *pool*
svc:/system/pools:default (resource pools framework)
 State: disabled since Wed 25 Jan 2006 10:39:00 AM GMT
Reason: Disabled by an administrator.
   See: http://sun.com/msg/SMF-8000-05
   See: libpool(3LIB)
   See: pooladm(1M)
   See: poolbind(1M)
   See: poolcfg(1M)
   See: poolstat(1M)
   See: /var/svc/log/system-pools:default.log
Impact: 1 dependent service is not running.  (Use -v for list.)

svc:/system/pools/dynamic:default (dynamic resource pools)
 State: offline since Wed 25 Jan 2006 10:39:12 AM GMT
Reason: Service svc:/system/pools:default is disabled.
   See: http://sun.com/msg/SMF-8000-GE
   See: poold(1M)
   See: /var/svc/log/system-pools-dynamic:default.log
Impact: This service is not running.
```

リソースプールサービスを有効にして、DRP サービスを実行可能にします。

```
# svcadm enable svc:/system/pools:default
```

svcs *pool* コマンドを使用すると、システムによって次の情報が表示されます。

```
# svcs *pool*
STATE          STIME    FMRI
online         10:40:27 svc:/system/pools:default
online         10:40:27 svc:/system/pools/dynamic:default
```

例 13–2 リソースプールサービスが無効な場合の動的リソースプールへの影響

両方のサービスがオンラインで、リソースプールサービスを無効にする場合は、次のコマンドを実行します。

```
# svcadm disable svc:/system/pools:default
```

svcs *pool* コマンドを使用すると、システムによって次の情報が表示されます。

```
# svcs *pool*
STATE          STIME    FMRI
disabled       10:41:05 svc:/system/pools:default
```

```
online         10:40:27 svc:/system/pools/dynamic:default
# svcs *pool*
STATE          STIME    FMRI
disabled       10:41:05 svc:/system/pools:default
online         10:40:27 svc:/system/pools/dynamic:default
```

ただし、リソースプールサービスが無効になるため、結果として DRP サービスが `offline` になります。

```
# svcs *pool*
STATE          STIME    FMRI
disabled       10:41:05 svc:/system/pools:default
offline        10:41:12 svc:/system/pools/dynamic:default
```

DRP サービスがオフラインになっている原因を特定します。

```
# svcs -x *pool*
svc:/system/pools:default (resource pools framework)
 State: disabled since Wed 25 Jan 2006 10:41:05 AM GMT
Reason: Disabled by an administrator.
   See: http://sun.com/msg/SMF-8000-05
   See: libpool(3LIB)
   See: pooladm(1M)
   See: poolbind(1M)
   See: poolcfg(1M)
   See: poolstat(1M)
   See: /var/svc/log/system-pools:default.log
Impact: 1 dependent service is not running.  (Use -v for list.)

svc:/system/pools/dynamic:default (dynamic resource pools)
 State: offline since Wed 25 Jan 2006 10:41:12 AM GMT
Reason: Service svc:/system/pools:default is disabled.
   See: http://sun.com/msg/SMF-8000-GE
   See: poold(1M)
   See: /var/svc/log/system-pools-dynamic:default.log
Impact: This service is not running.
```

DRP が機能するためには、リソースプールを起動する必要があります。たとえば、`pooladm` コマンドと `-e` オプションを使ってリソースプールを起動できます。

```
# pooladm -e
```

その後、`svcs *pool*` コマンドを実行すると、次のように表示されます。

```
# svcs *pool*
STATE          STIME    FMRI
online         10:42:23 svc:/system/pools:default
online         10:42:24 svc:/system/pools/dynamic:default
```

▼ Solaris 10 11/06 以降: svcadm を使って動的リソースプールサービスを無効にする方法

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 動的リソースプールサービスを無効にします。

```
# svcadm disable system/pools/dynamic:default
```

▼ pooladm を使ってリソースプールを有効にする方法

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 プール機能を有効にします。

```
# pooladm -e
```

▼ pooladm を使ってリソースプールを無効にする方法

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 プール機能を無効にします。

```
# pooladm -d
```

プールの構成

▼ 静的構成を作成する方法

-s オプションを付けて /usr/sbin/pooladm を実行して、現在の動的構成と一致する静的構成ファイルを作成します。別のファイル名を指定した場合を除き、デフォルトの場所 /etc/pooladm.conf が使用されます。

pooladm コマンドに -c オプションを付けて実行して、構成を確定します。次に、pooladm コマンドに -s オプションを付けて実行して、動的構成の状態と一致するように静的構成ファイルを更新します。

注 – 動的構成と一致する新しい構成を作成するには、以前の機能 poolcfg -c discover を使用するよりも、新機能 pooladm -s を使用することをお勧めします。

始める前に　使用しているシステムでプールを有効にします。

1. スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

   System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 現在の動的構成と一致するように静的構成ファイルを更新します。

   ```
   # pooladm -s
   ```

3. 構成ファイルの内容を読みやすい形式で表示します。

   構成には、システムによって作成されたデフォルトの要素が含まれています。

   ```
   # poolcfg -c info
   system tester
           string  system.comment
           int     system.version 1
           boolean system.bind-default true
           int     system.poold.pid 177916

           pool pool_default
                   int     pool.sys_id 0
                   boolean pool.active true
                   boolean pool.default true
                   int     pool.importance 1
                   string  pool.comment
                   pset    pset_default

           pset pset_default
                   int     pset.sys_id -1
   ```

```
boolean pset.default true
uint    pset.min 1
uint    pset.max 65536
string  pset.units population
uint    pset.load 10
uint    pset.size 4
string  pset.comment
boolean testnullchanged true

cpu
        int     cpu.sys_id 3
        string  cpu.comment
        string  cpu.status on-line

cpu
        int     cpu.sys_id 2
        string  cpu.comment
        string  cpu.status on-line

cpu
        int     cpu.sys_id 1
        string  cpu.comment
        string  cpu.status on-line

cpu
        int     cpu.sys_id 0
        string  cpu.comment
        string  cpu.status on-line
```

4 **/etc/pooladm.conf** にある構成を確定します。

```
# pooladm -c
```

5 (オプション) **/tmp/backup** という静的構成ファイルに動的構成をコピーするには、次のように入力します。

```
# pooladm -s /tmp/backup
```

▼ 構成の変更方法

構成を拡張するために、`pset_batch` というプロセッサセットと `pool_batch` というプールを作成します。次に、このプールとプロセッサセットを結合によって対応付けます。

サブコマンドの引数に空白が含まれている場合は、引用符で囲むことを忘れないでください。

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **pset_batch** というプロセッサセットを作成します。

```
# poolcfg -c 'create pset pset_batch (uint pset.min = 2; uint pset.max = 10)'
```

3 **pool_batch** というプールを作成します。

```
# poolcfg -c 'create pool pool_batch'
```

4 このプールとプロセッサセットを結合によって対応付けます。

```
# poolcfg -c 'associate pool pool_batch (pset pset_batch)'
```

5 対応付けた後の構成を表示します。

```
# poolcfg -c info
system tester
        string  system.comment kernel state
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line
```

```
pool pool_batch
        boolean pool.default false
        boolean pool.active true
        int pool.importance 1
        string pool.comment
        pset pset_batch

pset pset_batch
        int pset.sys_id -2
        string pset.units population
        boolean pset.default true
        uint pset.max 10
        uint pset.min 2
        string pset.comment
        boolean pset.escapable false
        uint pset.load 0
        uint pset.size 0

        cpu
                int     cpu.sys_id 5
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 4
                string  cpu.comment
                string  cpu.status on-line
```

6 **/etc/pooladm.conf** にある構成を確定します。

```
# pooladm -c
```

7 (オプション) **/tmp/backup** という静的構成ファイルに動的構成をコピーするには、次のように入力します。

```
# pooladm -s /tmp/backup
```

▼ プールをスケジューリングクラスに対応付ける方法

プールをスケジューリングクラスに対応付けることで、プールに結合されているすべてのプロセスがこのスケジューラを使用できるようになります。このためには、`pool.scheduler` プロパティーにスケジューラの名前を設定します。次の例では、プール `pool_batch` を公平配分スケジューラ (FSS) に対応付けます。

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割を作成してユーザーに割り当てる方法については、『Solaris のシステム管理ガイド (セキュリティーサービス)』の「RBAC の管理 (タスクマップ)」を参照してください。

2 **pool_batch** プールを変更して、**FSS** に対応付けます。

```
# poolcfg -c 'modify pool pool_batch (string pool.scheduler="FSS")'
```

3 対応付けた後の構成を表示します。

```
# poolcfg -c info
system tester
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916

        pool pool_default
                int     pool.sys_id 0
                boolean pool.active true
                boolean pool.default true
                int     pool.importance 1
                string  pool.comment
                pset    pset_default

        pset pset_default
                int     pset.sys_id -1
                boolean pset.default true
                uint    pset.min 1
                uint    pset.max 65536
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line

        pool pool_batch
                boolean pool.default false
                boolean pool.active true
                int pool.importance 1
                string pool.comment
                string pool.scheduler FSS
                pset batch
```

```
pset pset_batch
        int pset.sys_id -2
        string pset.units population
        boolean pset.default true
        uint pset.max 10
        uint pset.min 2
        string pset.comment
        boolean pset.escapable false
        uint pset.load 0
        uint pset.size 0

        cpu
                int     cpu.sys_id 5
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 4
                string  cpu.comment
                string  cpu.status on-line
```

4 **/etc/pooladm.conf** にある構成を確定します。

```
# pooladm -c
```

5 (オプション) **/tmp/backup** という静的構成ファイルに動的構成をコピーするには、次のように入力します。

```
# pooladm -s /tmp/backup
```

▼ 構成の制約を設定する方法

制約は、構成に加えることのできる変更を一部除外することで、作成可能な構成の範囲に影響を与えます。ここでは、cpu.pinned プロパティーを設定する手続きを示します。

次の例では、cpuid は整数です。

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 静的構成または動的構成内の **cpu.pinned** プロパティーを変更します。

- ブート時の (静的) 構成を変更します。

```
# poolcfg -c 'modify cpu <cpuid> (boolean cpu.pinned = true)'
```

- ブート時の構成を変更せずに、実行中の (動的) 構成を変更します。

  ```
  # poolcfg -dc 'modify cpu <cpuid> (boolean cpu.pinned = true)'
  ```

▼ 構成の目標を定義する方法

poold に対して、修正操作を実行するときに考慮する目標を指定できます。

次の手順では、wt-load 目標を設定して、poold がリソースの使用効率に合わせてリソースを割り当てるようにします。この構成目標を達成しやすくするために、locality 目標は無効にします。

1. スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

   System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. システム **tester** を変更して、**wt-load** 目標を優先するようにします。

   ```
   # poolcfg -c 'modify system tester (string system.poold.objectives="wt-load")'
   ```

3. デフォルトのプロセッサセットの **locality** 目標を無効にします。

   ```
   # poolcfg -c 'modify pset pset_default (string pset.poold.objectives="locality none")'
   ```

4. **pset_batch** プロセッサセットの **locality** 目標を無効にします。

   ```
   # poolcfg -c 'modify pset pset_batch (string pset.poold.objectives="locality none")'
   ```

5. 対応付けた後の構成を表示します。

   ```
   # poolcfg -c info
   system tester
           string  system.comment
           int     system.version 1
           boolean system.bind-default true
           int     system.poold.pid 177916
           string  system.poold.objectives wt-load

           pool pool_default
                   int     pool.sys_id 0
                   boolean pool.active true
                   boolean pool.default true
                   int     pool.importance 1
                   string  pool.comment
                   pset    pset_default

           pset pset_default
                   int     pset.sys_id -1
                   boolean pset.default true
                   uint    pset.min 1
                   uint    pset.max 65536
   ```

```
                string  pset.units population
                uint    pset.load 10
                uint    pset.size 4
                string  pset.comment
                boolean testnullchanged true
                string  pset.poold.objectives locality none

                cpu
                        int     cpu.sys_id 3
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 2
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 1
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 0
                        string  cpu.comment
                        string  cpu.status on-line

        pool pool_batch
                boolean pool.default false
                boolean pool.active true
                int pool.importance 1
                string pool.comment
                string pool.scheduler FSS
                pset batch

        pset pset_batch
                int pset.sys_id -2
                string pset.units population
                boolean pset.default true
                uint pset.max 10
                uint pset.min 2
                string pset.comment
                boolean pset.escapable false
                uint pset.load 0
                uint pset.size 0
                string  pset.poold.objectives locality none

                cpu
                        int     cpu.sys_id 5
                        string  cpu.comment
                        string  cpu.status on-line

                cpu
                        int     cpu.sys_id 4
                        string  cpu.comment
                        string  cpu.status on-line
```

6 **/etc/pooladm.conf** にある構成を確定します。

```
# pooladm -c
```

7 (オプション) **/tmp/backup** という静的構成ファイルに動的構成をコピーするには、次のように入力します。

```
# pooladm -s /tmp/backup
```

▼ poold のログレベルを設定する方法

poold が生成するログ情報のレベルを指定するには、poold 構成の system.poold.log-level プロパティーを設定します。poold の構成は libpool の構成に保存されています。詳細は、162 ページの「poold のログ情報」および poolcfg(1m) と libpool(3LIB) のマニュアルページを参照してください。

コマンド行で poold コマンドを使用する方法でも、poold で生成するログ情報のレベルを指定できます。

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **poold** コマンドに **-l** オプションとパラメータ (**INFO** など) を付けて実行することで、ログのレベルを設定します。

```
# /usr/lib/pool/poold -l INFO
```

使用可能なパラメータについては、162 ページの「poold のログ情報」を参照してください。デフォルトのログレベルは NOTICE です。

▼ poolcfg でコマンドファイルを使用する方法

poolcfg コマンドに -f オプションを付けて使用すると、poolcfg コマンドの -c オプションに指定する引数をテキストファイルから入力できます。この方法は、一連の操作を実行する場合に使用します。複数のコマンドを処理した場合でも、それらのコマンドがすべて正常に終了するまで、構成は更新されません。特に大規模な構成や複雑な構成の場合は、この手法を使用した方が、個々のサブコマンドを起動するよりも便利です。

コマンドファイルでは、# という文字はコメント記号として機能し、その行の残り部分がコメントと見なされます。

1 入力ファイル **poolcmds.txt** を作成します。

```
$ cat > poolcmds.txt
create system tester
create pset pset_batch (uint pset.min = 2; uint pset.max = 10)
create pool pool_batch
associate pool pool_batch (pset pset_batch)
```

2 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割を作成してユーザーに割り当てる方法については、『Solaris のシステム管理ガイド (セキュリティーサービス)』の「RBAC の管理 (作業マップ)」を参照してください。

3 コマンドを実行します。

```
# /usr/sbin/poolcfg -f poolcmds.txt
```

リソースの転送

`-d` オプションを付けた `poolcfg` に `-c` オプションの `transfer` サブコマンド引数を付けて実行すると、カーネルでリソースを転送できます。`-d` オプションは、コマンドにファイルから入力するのではなく、直接カーネル上で実行することを示します。

次の手順では、2 つの CPU をプロセッサセット `pset1` からプロセッサセット `pset2` にカーネルで移動します。

▼ プロセッサセット間で CPU を移動する方法

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **2 つの CPU を `pset1` から `pset2` に移動します。**

`from` 文節と `to` 文節は、どの順序で使用してもかまいません。`to` 文節と `from` 文節は、1 つのコマンドにそれぞれ 1 つだけ使用できます。

```
# poolcfg -dc 'transfer 2 from pset pset1 to pset2'
```

例 13–3 プロセッサセット間で CPU を移動する別の方法

あるリソースタイプの特定のリソースの ID を指定して転送する場合は、別の構文が用意されています。たとえば、次のコマンドは、ID が `0` と `2` の 2 つの CPU を `pset_large` プロセッサセットに割り当てます。

```
# poolcfg -dc "transfer to pset pset_large (cpu 0; cpu 2)"
```

参考 トラブルシューティング

要求を満たすための十分なリソースがない場合や、指定された ID が見つからない場合は、転送は失敗し、エラーメッセージが表示されます。

プール構成の起動と削除

`pooladm` コマンドを使用すると、特定のプール構成をアクティブにしたり、現在アクティブになっているプール構成を削除したりできます。このコマンドの詳細については、`pooladm(1M)` のマニュアルページを参照してください。

▼ プール構成を起動する方法

デフォルト構成ファイル `/etc/pooladm.conf` に保存されている構成を起動するには、`pooladm` に `-c` オプション (構成の確定) を付けて実行します。

1. スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

   System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **`/etc/pooladm.conf`** にある構成を確定します。

   ```
   # pooladm -c
   ```

3. (オプション) たとえば **`/tmp/backup`** という静的構成ファイルに動的構成をコピーするには、次のように入力します。

   ```
   # pooladm -s /tmp/backup
   ```

▼ 構成を確定する前に構成を検証する方法

`-n` オプションと `-c` オプションをともに使用すると、検証が実行されるとどうなるかをテストできます。構成が実際に確定されることはありません。

次のコマンドは、`/home/admin/newconfig` に保存されている構成を検証します。検出されたエラー条件が表示されますが、構成自体は変更されません。

1. スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

   System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 構成を確定する前に構成を検証します。

```
# pooladm -n -c /home/admin/newconfig
```

▼ プール構成を削除する方法

現在アクティブになっている構成を削除して、プロセッサセットなどの関連付けられているすべてのリソースをデフォルトのステータスに戻すには、-x オプション (構成の削除) を使用します。

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 現在アクティブになっている構成を削除します。

```
# pooladm -x
```

-x オプションを付けて pooladm を実行すると、ユーザーが定義したすべての要素が動的構成から削除されます。すべてのリソースがデフォルトの状態に戻り、プールとの結合もすべてデフォルトプールとの結合で置換されます。

参考 プロセッサセット内におけるスケジューリングクラスの混在

TS クラスのプロセスと IA クラスのプロセスを同一プロセッサセット内で混在させても問題はありません。1 つのプロセッサセット内でその他のスケジューリングクラスを混在させると、予期できない結果が生じる可能性があります。pooladm -x を使用した結果、1 つのプロセッサセット内にスケジューリングクラスが混在している場合は、priocntl コマンドを使用して、実行中のプロセスを別のスケジューリングクラスに移動してください。125 ページの「プロセスを TS クラスから FSS クラスに手動で移動する方法」を参照してください。priocntl(1) のマニュアルページも参照してください。

プール属性の設定とプールへの結合

リソースプールをプロジェクトに関連付けるために、project.pool 属性を設定できます。

実行中のプロセスをプールに結合するには、次の 2 つの方法を使用できます。

- poolbind コマンド (poolbind(1M) のマニュアルページを参照) を使用して、特定のプロセスを指定されたリソースプールに結合します。

- project データベース内の project.pool 属性を使用して、新しいログインセッションや newtask コマンドで起動されるタスクを結合するプールを指定します。newtask(1)、projmod(1M)、および project(4) のマニュアルページを参照してください。

▼ プロセスをプールに結合する方法

次の手順では、poolbind コマンドに -p オプションを付けて実行して、プロセス (この例では、現在のシェル) を ohare というプールに手動で結合します。

1. スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

   System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. プロセスをプールに手動で結合します。

   ```
   # poolbind -p ohare $$
   ```

3. **poolbind** に **-q** オプションを付けて実行することで、プロセスとプールの結合を確認します。

   ```
   $ poolbind -q $$
   155509 ohare
   ```

   プロセス ID とプールへの結合が表示されます。

▼ タスクまたはプロジェクトをプールに結合する方法

タスクまたはプロジェクトをプールに結合するには、poolbind コマンドに -i オプションを指定します。次の例では、airmiles プロジェクト内のすべてのプロセスを laguardia プールに結合します。

1. スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

   System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **airmiles** プロジェクト内のすべてのプロセスを **laguardia** プールに結合します。

   ```
   # poolbind -i project -p laguardia airmiles
   ```

▼ プロジェクトの project.pool 属性を設定する方法

プロジェクトのプロセスをリソースプールに結合するために、project.pool 属性を設定できます。

1 スーパーユーザーになるか、**Process Management** プロファイルが含まれている役割を引き受けます。

System Administrator 役割には Process Management プロファイルが含まれています。役割の詳細については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **project** データベース内の各エントリに **project.pool** 属性を追加します。

```
# projmod -a -K project.pool=poolname  project
```

▼ project 属性を使ってプロセスを別のプールに結合する方法

studio と backstage という 2 つのプールを持つ構成が存在するものとします。/etc/project ファイルの内容は、次のとおりです。

```
user.paul:1024::::project.pool=studio
user.george:1024::::project.pool=studio
user.ringo:1024::::project.pool=backstage
passes:1027::paul::project.pool=backstage
```

この構成の場合、ユーザー paul によって起動されるプロセスは、デフォルトで studio プールに結合されます。

ユーザー paul は、起動するプロセスのプール結合を変更できます。paul は、newtask を使用して (この場合は passes プロジェクト内で起動することで)、作業を backstage プールに結合することもできます。

1 **passes** プロジェクトでプロセスを起動します。

```
$ newtask -l -p passes
```

2 **poolbind** コマンドに **-q** オプションを付けて実行し、プロセスとプールの結合を確認します。また、二重ドル記号 **($$)** を使用して親シェルのプロセス番号をコマンドに渡します。

```
$ poolbind -q $$
6384  pool backstage
```

プロセス ID とプールへの結合が表示されます。

poolstat を使ってプールに関連付けられているリソースについて統計情報を報告する

poolstat コマンドを使用すると、プールに関連付けられているリソースについて統計情報を表示できます。詳細は、167 ページの「poolstat によるプール機能とリソース使用効率の監視」および poolstat(1M) のマニュアルページを参照してください。

このセクションでは、さまざまな目的の報告を生成する方法を、例を使用しながら説明します。

poolstat のデフォルトの出力を表示する

引数なしで poolstat と入力すると、見出し行に続いて、1 行に 1 つずつプールが表示されます。情報の行には、プール ID、プール名、およびプールに接続されているプロセッサセットに関するリソース統計が表示されます。

```
machine% poolstat
                           pset
       id pool           size used load
        0 pool_default      4  3.6  6.2
        1 pool_sales        4  3.3  8.4
```

特定の間隔で複数の報告を生成する

次のコマンドは、3 つの報告を 5 秒間のサンプリング間隔で生成します。

```
machine% poolstat 5 3
                            pset
 id pool                  size used load
 46 pool_sales               2  1.2  8.3
  0 pool_default             2  0.4  5.2
                            pset
 id pool                  size used load
 46 pool_sales               2  1.4  8.4
  0 pool_default             2  1.9  2.0
                            pset
 id pool                  size used load
 46 pool_sales               2  1.1  8.0
  0 pool_default             2  0.3  5.0
```

リソースセットの統計情報を報告する

次の例では、poolstat コマンドに -r オプションを付けて実行し、プロセッサセットのリソースセットの統計情報を報告します。リソースセット pset_default は複数のプールに接続されているので、このプロセッサセットは各プールで 1 回ずつ表示されます。

```
machine% poolstat -r pset
      id pool          type rid rset          min  max size used load
       0 pool_default  pset  -1 pset_default    1  65K    2  1.2  8.3
       6 pool_sales    pset   1 pset_sales      1  65K    2  1.2  8.3
       2 pool_other    pset  -1 pset_default    1  10K    2  0.4  5.2
```

◆ ◆ ◆ 第 14 章

リソース管理の構成例

この章では、リソース管理のフレームワークについて考察し、仮想的なサーバー統合プロジェクトについて説明します。

この章の内容は次のとおりです。

統合前の構成

この例では、5 つのアプリケーションを 1 つのシステムに統合します。対象となるアプリケーションは、それぞれリソース要件、ユーザー数、およびアーキテクチャーが異なります。現在、各アプリケーションは、それぞれの要件を満たす専用サーバーに置かれています。次の表にアプリケーションとその特性を示します。

アプリケーションの説明	特性
アプリケーションサーバー	2 CPU を超えるとスケーラビリティーが低くなります
アプリケーションサーバー用のデータベースインスタンス	負荷の高いトランザクション処理
テストおよび開発環境用のアプリケーションサーバー	GUI に基づいたコードテスト
トランザクション処理サーバー	応答時間を保証します
スタンドアロンのデータベースインスタンス	大量のトランザクションを処理し、複数のタイムゾーンに対してサービスを提供します

統合後の構成

次の構成を使用して、アプリケーションを 1 つのシステムに統合します。

- アプリケーションサーバーは、2 つの CPU から構成されるプロセッサセットを持ちます。
- アプリケーションサーバーのデータベースインスタンスとスタンドアロンのデータベースインスタンスは、4 つ以上の CPU から構成される 1 つのプロセッサセットに統合されます。スタンドアロンのデータベースインスタンスはそのリソースの 75% が保証されます。
- テストおよび開発用のアプリケーションサーバーには IA スケジューリングクラスを適用して、UI の応答性を保証します。メモリーを制約して、不正なコードによる影響を低減します。
- トランザクション処理サーバーには 2 つ以上の CPU から構成される専用のプロセッサセットを割り当てて、応答時間を短縮します。

この構成の対象には、各リソースセットのプロセッササイクルを消費している実行中の既知のアプリケーションすべてが含まれます。したがって、プロセッサリソースを必要としているセットにプロセッサリソースをセット間で転送できるように、次のような制約を設定します。

- `wt-load` 目標を設定して、使用効率の高いリソースセットに、より多くのリソースを割り当てるようにします。
- `locality` 目標を `tight` に設定して、プロセッサの近傍性が最大になるようにします。

また、どのリソースセットについても使用効率が 80% を超えないようにする制約も適用します。この制約により、アプリケーションは必要なリソースに確実にアクセスできます。さらに、トランザクション処理のプロセッサセットについては、使用効率を 80% 以下に保つという目標の重要性を、ほかの目標の 2 倍にします。この重要性は構成で定義します。

構成の作成

`/etc/project` データベースファイルを編集します。エントリを追加して必要なリソース制御を実装し、ユーザーをリソースプールにマップしたら、ファイルを表示します。

```
# cat /etc/project
.
.
.
user.app_server:2001:Production Application Server:::project.pool=appserver_pool
user.app_db:2002:App Server DB:::project.pool=db_pool;project.cpu-shares=(privileged,1,deny)
```

```
development:2003:Test and development::staff:project.pool=dev_pool;
process.max-address-space=(privileged,536870912,deny)     keep with previous line
user.tp_engine:2004:Transaction Engine:::project.pool=tp_pool
user.geo_db:2005:EDI DB:::project.pool=db_pool;project.cpu-shares=(privileged,3,deny)
.
.
.
```

注 – 開発チームはタスクを開発プロジェクトで実行する必要があります。これは、このプロジェクトへのアクセスをユーザーのグループ ID (GID) で制限しているためです。

pool.host という名前で入力ファイルを作成し、必要なリソースプールの構成に使用します。次に、ファイルを表示します。

```
# cat pool.host
create system host
create pset dev_pset (uint pset.min = 0; uint pset.max = 2)
create pset tp_pset (uint pset.min = 2; uint pset.max=8)
create pset db_pset (uint pset.min = 4; uint pset.max = 6)
create pset app_pset (uint pset.min = 1; uint pset.max = 2)
create pool dev_pool (string pool.scheduler="IA")
create pool appserver_pool (string pool.scheduler="TS")
create pool db_pool (string pool.scheduler="FSS")
create pool tp_pool (string pool.scheduler="TS")
associate pool dev_pool (pset dev_pset)
associate pool appserver_pool (pset app_pset)
associate pool db_pool (pset db_pset)
associate pool tp_pool (pset tp_pset)
modify system tester (string system.poold.objectives="wt-load")
modify pset dev_pset (string pset.poold.objectives="locality tight; utilization < 80")
modify pset tp_pset (string pset.poold.objectives="locality tight; 2: utilization < 80")
modify pset db_pset (string pset.poold.objectives="locality tight;utilization < 80")
modify pset app_pset (string pset.poold.objectives="locality tight; utilization < 80")
```

pool.host 入力ファイルを使って構成を更新します。

```
# poolcfg -f pool.host
```

構成をアクティブにします。

```
# pooladm -c
```

システム上でフレームワークが有効になっています。

構成の表示

フレームワークの構成には、システムによって作成されたデフォルトの要素も含まれています。この構成を表示するには、次のように入力します。

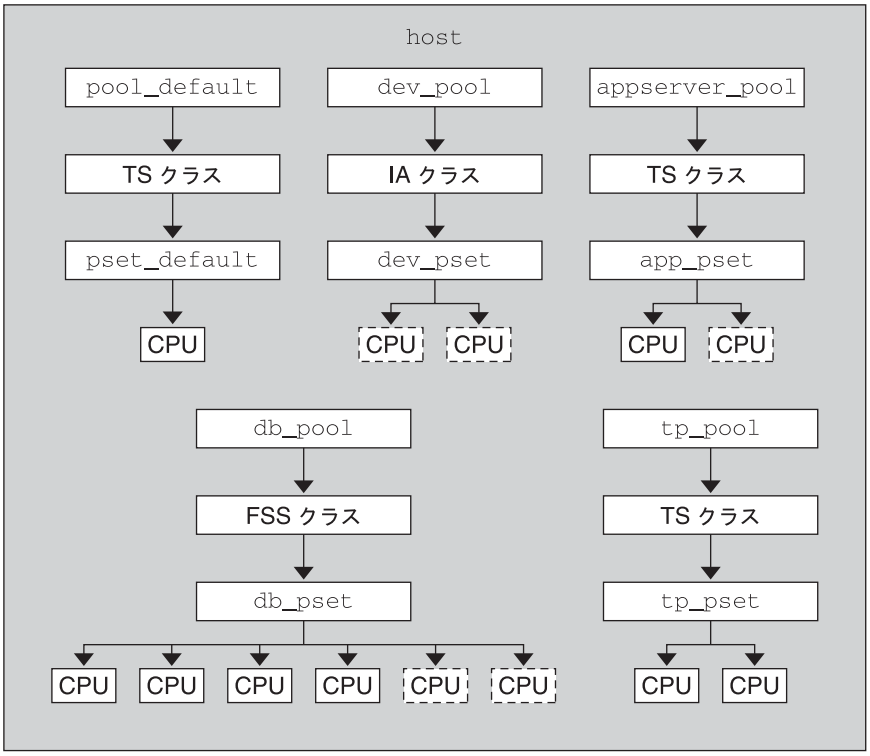
```
# pooladm
system host
        string  system.comment
        int     system.version 1
        boolean system.bind-default true
        int     system.poold.pid 177916
        string  system.poold.objectives wt-load

        pool dev_pool
                int     pool.sys_id 125
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler IA
                pset    dev_pset

        pool appserver_pool
                int     pool.sys_id 124
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler TS
                pset    app_pset

        pool db_pool
                int     pool.sys_id 123
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler FSS
                pset    db_pset

        pool tp_pool
                int     pool.sys_id 122
                boolean pool.default false
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler TS
                pset    tp_pset

        pool pool_default
                int     pool.sys_id 0
                boolean pool.default true
                boolean pool.active true
                int     pool.importance 1
                string  pool.comment
                string  pool.scheduler TS
                pset    pset_default
```

```
pset dev_pset
        int     pset.sys_id 4
        string  pset.units population
        boolean pset.default false
        uint    pset.min 0
        uint    pset.max 2
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80

pset tp_pset
        int     pset.sys_id 3
        string  pset.units population
        boolean pset.default false
        uint    pset.min 2
        uint    pset.max 8
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; 2: utilization < 80

        cpu
                int     cpu.sys_id 1
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 2
                string  cpu.comment
                string  cpu.status on-line

pset db_pset
        int     pset.sys_id 2
        string  pset.units population
        boolean pset.default false
        uint    pset.min 4
        uint    pset.max 6
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80

        cpu
                int     cpu.sys_id 3
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 4
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 5
```

```
                string  cpu.comment
                string  cpu.status on-line

        cpu
                int     cpu.sys_id 6
                string  cpu.comment
                string  cpu.status on-line
pset app_pset
        int     pset.sys_id 1
        string  pset.units population
        boolean pset.default false
        uint    pset.min 1
        uint    pset.max 2
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0
        string  pset.poold.objectives locality tight; utilization < 80
        cpu
                int     cpu.sys_id 7
                string  cpu.comment
                string  cpu.status on-line

pset pset_default
        int     pset.sys_id -1
        string  pset.units population
        boolean pset.default true
        uint    pset.min 1
        uint    pset.max 4294967295
        string  pset.comment
        boolean pset.escapable false
        uint    pset.load 0
        uint    pset.size 0

        cpu
                int     cpu.sys_id 0
                string  cpu.comment
                string  cpu.status on-line
```

フレームワークのグラフィック表示が続きます。

図 14–1 サーバー統合の構成



注 – 上記の図のプール `db_pool` では、スタンドアロンのデータベースインスタンスに CPU リソースの 75% が保証されています。

◆ ◆ ◆ 第 15 章

Solaris 管理コンソールのリソース制御機能

この章では、Solaris 管理コンソールのリソース制御機能と性能監視機能について説明します。このコンソールを使って制御できるのは、リソース管理機能の一部だけです。

このコンソールでは、システムの性能を監視したり、プロジェクト、タスク、およびプロセスにリソース制御の値 (表 15–1 を参照) を入力したりします。このコンソールは、何台ものシステムに渡って構成されている数百の構成パラメータを管理する際に、コマンド行インタフェース (CLI) の代わりとして使用できる便利で安全なツールです。各システムは個別に管理されます。このコンソールのグラフィカルインタフェースは、ユーザーの経験レベルに合った使い方ができます。

この章の内容は次のとおりです。

Solaris 管理コンソールの使用 (タスクマップ)

タスク	説明	参照先
コンソールを使用します	ローカル環境、ネームサービス環境、またはディレクトリサービス環境で Solaris 管理コンソールを起動します。ネームサービス環境では、パフォーマンスツールは使用できません。	『Solaris のシステム管理 (基本編)』の「Solaris 管理コンソールを起動する」および『Solaris のシステム管理 (基本編)』の「ネームサービス環境での Oracle Solaris 管理ツールの使用 (作業マップ)」

タスク	説明	参照先
システムパフォーマンスを監視します	「System Status」の下にあるパフォーマンスツールにアクセスします。	205 ページの「パフォーマンスツールにアクセスする方法」
プロジェクトにリソース制御機能を追加します	「System Configuration」の下にある「リソース制御 (Resource Controls)」タブにアクセスします。	209 ページの「「リソース制御 (Resource Controls)」タブへのアクセス方法」

コンソールの概要

リソース管理機能は、Solaris 管理コンソールのコンポーネントです。このコンソールは、GUI に基づいた管理ツールのためのコンテナです。管理ツールはツールボックスと呼ばれるコレクションに格納されています。コンソールとその使用方法については、『Solaris のシステム管理 (基本編)』の第 2 章「Solaris 管理コンソールの操作 (手順)」を参照してください。

コンソールとそのツール群のドキュメントは、コンソールのオンラインヘルプで参照できます。オンラインヘルプで参照できるドキュメントの内容については、『Solaris のシステム管理 (基本編)』の「Solaris 管理コンソール (概要)」を参照してください。

管理範囲

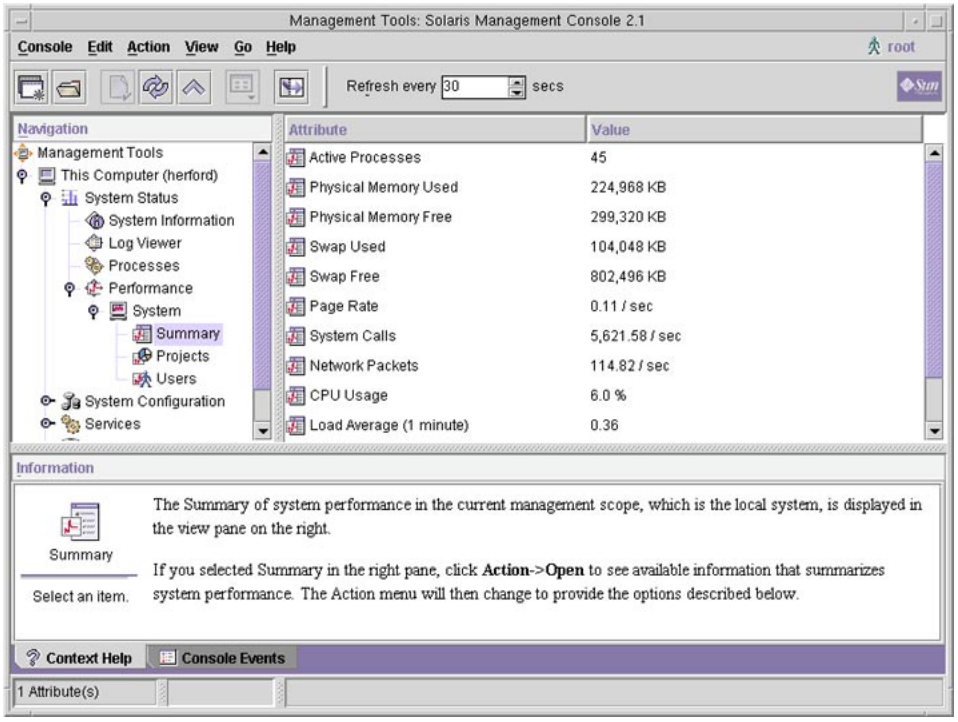
「管理範囲」とは、選択した管理ツールで使用するネームサービス環境のことです。リソース制御機能とパフォーマンスツールを使用するときの管理範囲は、`/etc/project` ローカルファイルまたは NIS から選択します。

コンソールセッションで選択する管理範囲は、`/etc/nsswitch.conf` ファイルで特定されるもっとも優先順位の高いネームサービスと一致するべきです。

パフォーマンスツール

パフォーマンスツールは、リソースの使用状況を監視するために使用します。リソースの使用状況をシステム単位で集計したり、プロジェクト単位または個人ユーザー単位で表示したりできます。

図 15–1　Solaris管理コンソールのパフォーマンスツール



▼ パフォーマンスツールにアクセスする方法

パフォーマンスツールは、ナビゲーションペインの「System Status」の下にあります。パフォーマンスツールにアクセスするには、次の手順に従います。

1. ナビゲーションペインの「**System Status**」コントロール要素をクリックします。
   このコントロール要素は、ナビゲーションペインのメニュー項目を拡張するために使用します。
2. 「パフォーマンス **(Performance)**」コントロール要素をクリックします。
3. 「システム **(System)**」コントロール要素をクリックします。
4. 「サマリー **(Summary)**」、「プロジェクト **(Projects)**」、または「ユーザー **(Users)**」をダブルクリックします。
   何を選択するかは、監視する対象によって異なります。

システム単位の監視

次の属性の値が表示されます。

属性	説明
アクティブプロセス (Active Processes)	システム上でアクティブなプロセス数
物理メモリー使用量 (Physical Memory Used)	使用中のシステムメモリーのサイズ
物理メモリー空き容量 (Physical Memory Free)	使用可能なシステムメモリーのサイズ
スワップ使用量 (Swap Used)	使用中のシステムスワップ領域のサイズ
スワップ空き容量 (Swap Free)	使用可能なシステムスワップ領域のサイズ
ページング頻度 (Page Rate)	システムページングの頻度
システムコール (System Calls)	秒あたりのシステムコール数
ネットワークパケット (Network Packets)	秒あたりに送信されるネットワークのパケット数
CPU 使用量 (CPU Usage)	現在使用中の CPU の比率
平均負荷率 (Load Average)	過去 1 分、5 分、または 15 分の間にシステム実行キューに存在した平均プロセス数

プロジェクト単位またはユーザー単位の監視

次の属性の値が表示されます。

属性	短い名前	説明
入力ブロック (Input Blocks)	`inblk`	読み取られたブロック数
書き込まれたブロック (Blocks Written)	`oublk`	書き込まれたブロック数
読み取られた/書き込まれた文字数 (Chars Read/Written)	`ioch`	読み取りおよび書き込みが行われた文字数
データページフォルトのスリープ時間 (Data Page Fault Sleep Time)	`dftime`	データページフォルトの処理で経過した時間
強制的なコンテキストスイッチ (Involuntary Context Switches)	`ictx`	コンテキストの強制的な切り替え数

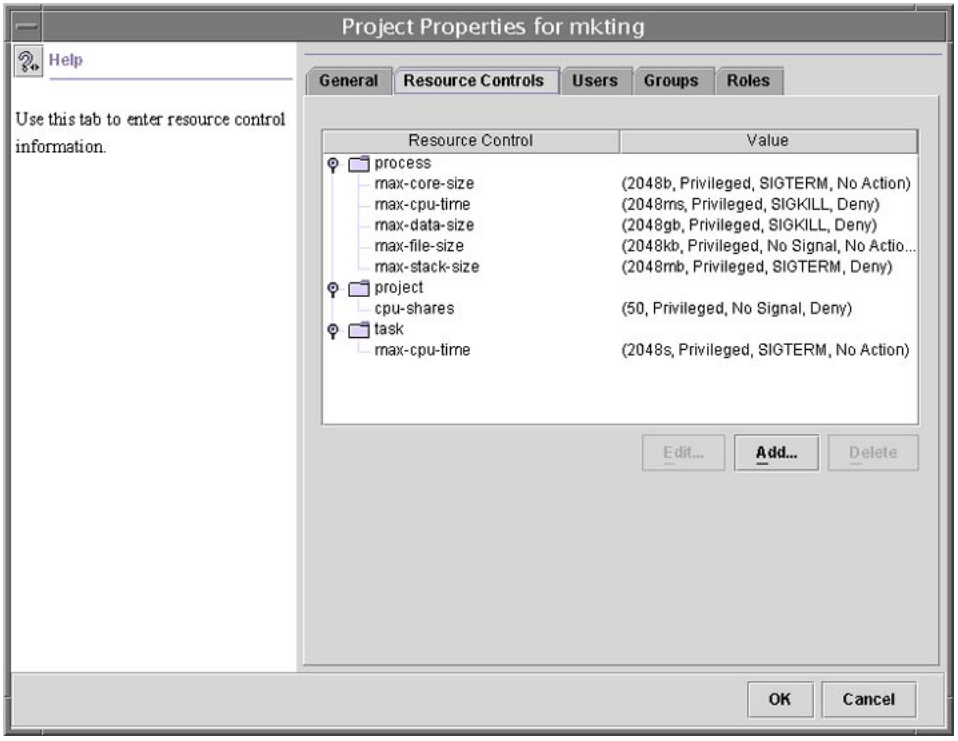
属性	短い名前	説明
システムモード時間 (System Mode Time)	`stime`	カーネルモードで経過した時間
メジャーページフォルト (Major Page Faults)	`majfl`	メジャーページフォルト数
受信したメッセージ (Messages Received)	`mrcv`	受信されたメッセージ数
送信したメッセージ (Messages Sent)	`msend`	送信されたメッセージ数
マイナーページフォルト (Minor Page Faults)	`minf`	マイナーページフォルト数
プロセス数 (Num Processes)	`nprocs`	ユーザーまたはプロジェクトが所有するプロセス数
LWP 数 (Num LWPs)	`count`	軽量プロセスの数
その他のスリープ時間 (Other Sleep Time)	`slptime`	`tftime`、`dftime`、`kftime`、および `ltime` を除いたスリープ時間
CPU 時間 (CPU Time)	`pctcpu`	プロセス、ユーザー、またはプロジェクトが使用した最新の CPU 時間の比率
使用メモリー (Memory Used)	`pctmem`	プロセス、ユーザー、またはプロジェクトが使用したシステムメモリーの比率
ヒープサイズ (Heap Size)	`brksize`	プロセスのデータセグメントに割り当てられているメモリーサイズ
常駐サイズ (Resident Set Size)	`rsssize`	プロセスによって要求されている現在のメモリーサイズ
プロセスイメージサイズ (Process Image Size)	`size`	プロセスイメージのサイズ (K バイト)
受信したシグナル (Signals Received)	`sigs`	受信されたシグナルの数
停止時間 (Stopped Time)	`stoptime`	停止状態で経過した時間
スワップ操作 (Swap Operations)	`swaps`	進行中のスワップ操作の数
実行されたシステムコール (System Calls Made)	`sysc`	設定された (直前の) 時間間隔で実行されたシステムコール数

属性	短い名前	説明
システムページフォルトのスリープ時間 (System Page Fault Sleep Time)	`kftime`	ページフォルトの処理で経過した時間
システムトラップ時間 (System Trap Time)	`ttime`	システムトラップの処理で経過した時間
テキストページフォルトのスリープ時間 (Text Page Fault Sleep Time)	`tftime`	テキストページフォルトの処理で経過した時間
ユーザーロック待機のスリープ時間 (User Lock Wait Sleep Time)	`ltime`	ユーザーロックを待機している間に経過した時間
ユーザーモード時間 (User Mode Time)	`utime`	ユーザーモードで経過した時間
ユーザーおよびシステムモード時間 (User and System Mode Time)	`time`	CPU 実行の累積時間
任意コンテキストスイッチ (Voluntary Context Switches)	`vctx`	コンテキストの自主的な切り替え数
CPU 待機時間 (Wait CPU Time)	`wtime`	CPU を待機している間に経過した時間 (応答時間)

「リソース制御 (Resource Controls)」タブ

リソース制御を使用して、プロジェクトをリソース制約の集合と対応付けることができます。これらの制約によって、プロジェクトのコンテキストで実行するタスクまたはプロセスのリソース許容量が決定されます。

図 15–2　Solaris 管理コンソールの「リソース制御 (Resource Controls)」タブ



▼ 「リソース制御 (Resource Controls)」タブへのアクセス方法

「リソース制御 (Resource Controls)」タブは、ナビゲーションペインの「System Configuration」の下にあります。「リソース制御 (Resource Controls)」にアクセスするには、次の手順に従います。

1. ナビゲーションペインの「**System Configuration**」コントロール要素をクリックします。
2. 「プロジェクト (**Projects**)」をダブルクリックします。
3. コンソールのメインウィンドウにあるプロジェクトをクリックして選択します。
4. 「アクション (**Action**)」メニューから「プロパティ (**Properties**)」を選択します。

5 「リソース制御 (Resource Controls)」タブをクリックします。
プロセス、プロジェクト、およびタスクのリソース制御の値を表示、追加、編集、または削除します。

設定可能なリソース制御

次の表に、コンソールで設定できるリソース制御を示します。この表では、各制御によって制約されるリソースについて説明し、project データベースにおけるそのリソースのデフォルトの単位を示します。デフォルトの単位には次の 2 種類があります。

- 数量は制限される量を意味します。
- インデックスは最大有効識別子を意味します。

したがって、project.cpu-shares は、プロジェクトで使用することが許可されている配分を示します。一方、process.max-file-descriptor は、open(2) システムコールによってプロセスに割り当てることができる最大ファイル番号を指定します。

表 15–1 Solaris 管理コンソールで使用できる標準のリソース制御

制御名	説明	デフォルトの単位
project.cpu-shares	このプロジェクトに対して、公平配分スケジューラ (FSS) で使用することが許可されている CPU 配分 (FSS(7) のマニュアルページを参照)	数量 (配分)
task.max-cpu-time	タスクのプロセスで使用できる最長 CPU 時間	時間 (秒)
task.max-lwps	タスクのプロセスで同時に使用できる LWP の最大数	数量 (LWP 数)
process.max-cpu-time	プロセスで使用できる最長 CPU 時間	時間 (秒)
process.max-file-descriptor	プロセスで使用できる最大のファイル記述子インデックス	インデックス (最大ファイル記述子)
process.max-file-size	プロセスで書き込むことができるファイルオフセットの最大サイズ	サイズ (バイト)
process.max-core-size	プロセスによって作成されるコアファイルの最大サイズ	サイズ (バイト)
process.max-data-size	プロセスで使用できるヒープメモリーの最大サイズ	サイズ (バイト)

表 15–1　Solaris 管理コンソールで使用できる標準のリソース制御　(続き)

制御名	説明	デフォルトの単位
process.max-stack-size	プロセスで使用できるスタックメモリーセグメントの最大サイズ	サイズ (バイト)
process.max-address-space	プロセスで使用できる、セグメントサイズの総計としての最大アドレス空間	サイズ (バイト)

値の設定

プロセス、プロジェクト、およびタスクのリソース制御値を表示、追加、編集、または削除できます。これらの操作は、コンソールのダイアログボックスで実行します。

リソース制御 (Resource Control) と値 (Value) は、コンソールに表形式で表示されます。リソース制御 (Resource Control) の欄には、設定可能なリソース制御の一覧が表示されます。値 (Value) の欄には、各リソース制御に対応付けられているプロパティーが表示されます。表内では、これらの値は括弧で囲まれており、コンマで区切られたプレーンテキストとして表示されます。括弧内の値によって「アクション文節」が構成されます。各アクション文節には、値として、しきい値、特権レベル、1 つのシグナル、および特定のしきい値に対応付けられている 1 つの局所アクションが含まれます。各リソース制御は複数のアクション文節を持つことができ、各アクション文節もコンマで区切られます。

注 – 実行中のシステムでは、コンソールから project データベースに行なった値の変更は、プロジェクトで起動される新しいタスクに対してだけ有効になります。

コンソールのリファレンス

プロジェクトとタスクについては、第 2 章「プロジェクトとタスク (概要)」を参照してください。リソース制御については、第 6 章「リソース制御 (概要)」を参照してください。公平配分スケジューラ (FSS) については、第 8 章「公平配分スケジューラ (概要)」を参照してください。

注 – すべてのリソース制御をコンソールで設定できるわけではありません。コンソールで設定できる資源制御の一覧については、表 15–1 を参照してください。

パート II

ゾーン

このパートでは、Oracle Solaris ゾーン (コンテナ) ソフトウェア区分技術について紹介します。この技術を使用すると、オペレーティングシステムサービスを仮想化し、アプリケーションの実行に適した隔離された環境を実現できます。このように隔離することで、あるゾーンで実行中のプロセスが、ほかのゾーンで実行中のプロセスを監視したり、それらに影響を与えたりすることを防ぐことができます。

◆ ◆ ◆ 第 16 章

Solaris ゾーンの紹介

Solaris オペレーティングシステムの Solaris ゾーン機能を使用すると、アプリケーションの実行に適した隔離された環境をシステム上に実現できます。Solaris ゾーンは、Solaris コンテナ環境のコンポーネントです。

この章で扱う内容は、次のとおりです。

システムにゾーンを作成する準備が完了している場合は、第 17 章「非大域ゾーンの構成 (概要)」に進んでください。

ゾーンの概要

ゾーン区分技術は、オペレーティングシステムサービスを仮想化し、アプリケーションの実行に適した隔離されたセキュアな環境を提供するために使用されます。ゾーンとは、Oracle Solaris システムの 1 つのインスタンス内で作成される、仮想化されたオペレーティングシステム環境です。ゾーンを作成すると、そのアプリケーション実行環境で実行されるプロセスは、システムのほかの部分から隔離されます。この分離を行うことで、1 つのゾーン内で稼働しているプロセスがほかのゾーンで稼働しているプロセスを監視したりそれらのプロセスに影響を及ぼしたりすることが防止されます。スーパーユーザー資格で実行されているプロセスであっても、ほかのゾーンの活動を監視したり操作したりすることはできません。

また、ゾーンにより、アプリケーションを配備するマシンの物理的属性からアプリケーションを分離する抽象層も提供されます。このような属性の例として、物理デバイスパスがあります。

ゾーンは、Oracle Solaris 10 リリース以降を実行しているすべてのマシンで使用できます。システムに作成できるゾーン数の上限は 8192 です。1 つのシステムで効率的にホストできるゾーン数は、すべてのゾーンで実行されるアプリケーションソフトウェアに必要な総リソース量によって決まります。

Solaris 10 リリースでは、非大域ゾーンのルートファイルシステムモデルには 2 種類あります。疎ルートと完全ルートです。「疎ルートゾーン」モデルは、オブジェクトの共有を最適化します。「完全ルートゾーン」モデルは、構成に関して最大の構成可能性を提供します。これらの概念については、第 18 章「非大域ゾーンの計画と構成 (タスク)」を参照してください。

Oracle Solaris 10 コンテナ (非大域ゾーン) では、静的にリンクされたバイナリをサポートしていません。

Solaris 10 9/10: インストールされる製品 (システム資産と呼ばれる) は、自動登録機能によって制御されます。インストール中に、ユーザーは資格を入力するか、または匿名で登録します。システムのリブート時に、新しい製品のサービスタグが `My Oracle Support` サーバーにアップロードされます。この機能は、大域ゾーンでのみ使用できます。詳細は、『Solaris のシステム管理 (基本編)』を参照してください。

ブランドゾーンについて

ブランドゾーン (BrandZ) は、実行時の動作の代替セットを含むコンテナを作成するためのフレームワークを提供します。「ブランド」は、さまざまなオペレーティング環境を指す場合があります。たとえば、非大域ゾーンは、Solaris 8 オペレーティングシステムや Linux などのオペレーティング環境をエミュレートすることができます。

ブランドは、ゾーンにインストールできるオペレーティング環境を定義し、ゾーンにインストールされたソフトウェアが正しく機能するようにゾーン内でのシステムの動作を決定します。また、ゾーンのブランドにより、アプリケーションの起動時に正しいアプリケーションタイプが識別されます。すべてのブランドゾーン管理は、標準のゾーンコマンドの拡張を通して実行されます。管理手順のほとんどはすべてのゾーンで同一です。

Solaris 10 8/07 オペレーティングシステムまたはそれ以降の Solaris 10 リリースを実行する SPARC マシンでは、次の 2 つのブランドがサポートされています。

- 『Solaris のシステム管理: Solaris 8 Containers』に記載されている `solaris8` ブランド、Solaris 8 コンテナ
- 『Solaris のシステム管理: Solaris 9 Containers』に記載されている `solaris9` ブランド、Solaris 9 コンテナ

Solaris 10 OS でサポートされる他のブランドは次のとおりです。

- パート III「lx ブランドゾーン」に記載されている x86 および x64 システム用の Linux lx ブランド
- docs.sun.com の Sun Cluster 3.2 1/09 Software Collection for Solaris OS に記載されている cluster ブランド

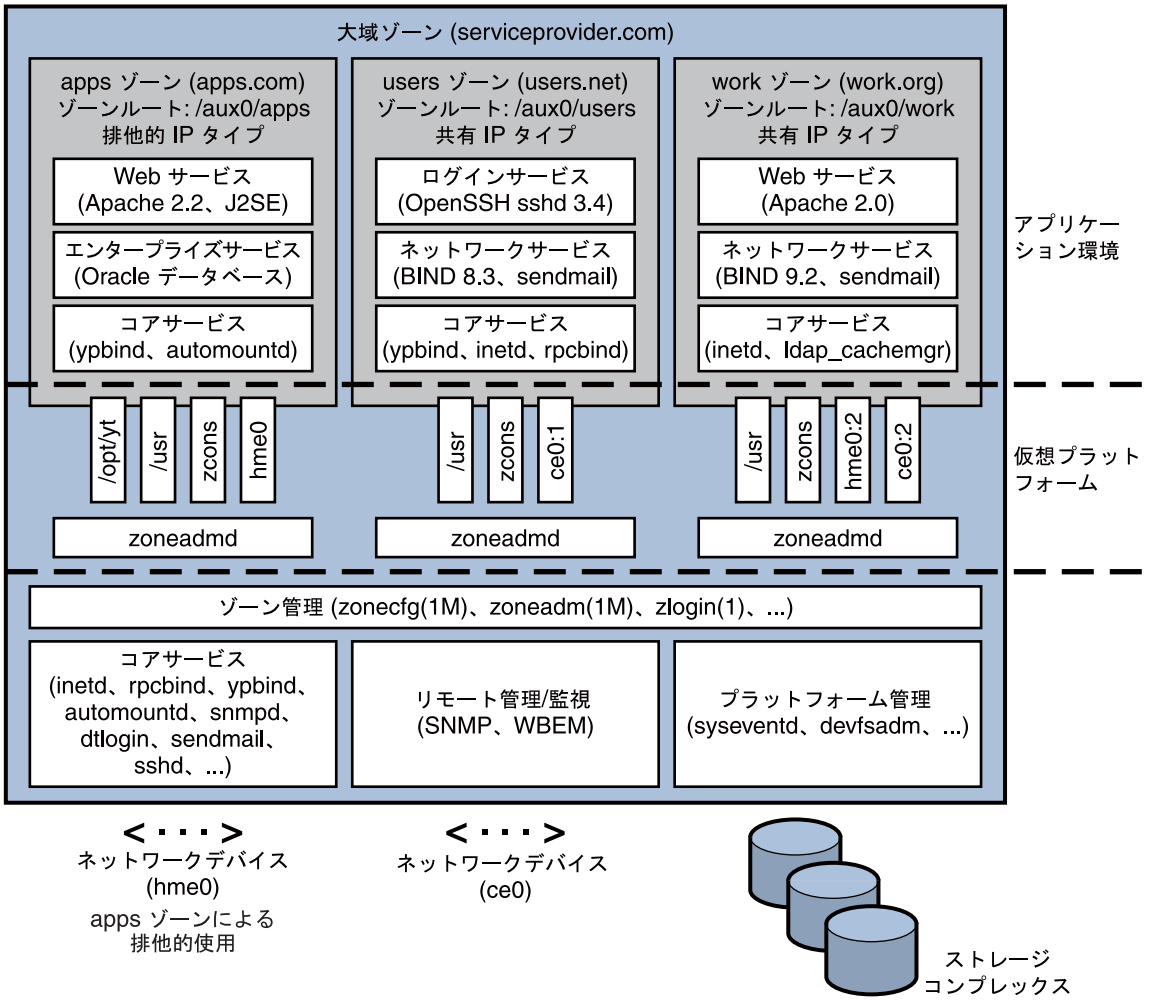
ラベルが有効になっている Trusted Solaris システムにブランドゾーンを構成してインストールすることはできますが、このシステム構成でブランドゾーンをブートすることはできません。

ゾーンを使用する場合

ゾーンは、多くのアプリケーションを 1 台のサーバー上で統合できる環境で使用するともっとも高い効果を発揮します。多数のマシンの管理は複雑でコストがかかるため、より大規模で拡張性の高いサーバーにアプリケーションを統合することが望まれます。

次の図は、4 つのゾーンから成るシステムを示しています。この統合された環境の例では、apps、users、および work の各ゾーンは、ほかのゾーンの作業負荷とは無関係に作業負荷を実行しています。この例は、同じアプリケーションのさまざまなバージョンをそれぞれ異なるゾーンで実行でき、悪影響を引き起こすことなく、統合の要件を満たすことができることを示しています。ゾーンごとにカスタマイズされたサービスを提供できます。

図 16–1 ゾーンによるサーバー統合の例



ゾーンを使用すると、システムのリソースをより効率的に利用できます。リソースの動的再割り当てにより、使用されていないリソースを必要に応じてほかのコンテナに移動できます。障害およびセキュリティーの隔離により、動作状態の悪いアプリケーションのために使用効率の低い専用のシステムを用意する必要がなくなります。ゾーンを使用すると、このようなアプリケーションをほかのアプリケーションと統合できます。

ゾーンを使用すると、総合的なシステムセキュリティーを維持しながら、管理機能の一部を委譲できます。

ゾーンのしくみ

非大域ゾーンは、1 つの箱と考えることができます。この箱の中では、システムのほかの部分と相互に作用することなく、1 つ以上のアプリケーションを実行できます。Solaris ゾーンは、ソフトウェアで定義される柔軟な境界を使用して、ソフトウェアアプリケーションやサービスを隔離します。これにより Solaris オペレーティングシステムの 1 つのインスタンス内で実行されるアプリケーションを互いに独立して管理することができます。したがって、同じアプリケーションのさまざまなバージョンをそれぞれ異なるゾーンで実行でき、構成の要件を満たすことができます。

ゾーンに割り当てられたプロセスは、同じゾーンに割り当てられたほかのプロセスを操作、監視したり、これらのプロセスと直接通信したりできます。システムのほかのゾーンに割り当てられたプロセスや、ゾーンに割り当てられていないプロセスに対しては、このような機能は実行できません。異なるゾーンに割り当てられたプロセスどうしでは、ネットワーク API を介した通信のみ可能です。

Solaris 10 8/07 以降では、ゾーンが独自の排他的 IP インスタンスを持っているか、または IP 層の構成と状態を大域ゾーンと共有しているかに応じて、IP ネットワーク接続を 2 通りの方法で構成できます。ゾーンの IP タイプの詳細については、236 ページの「ゾーンネットワークインタフェース」を参照してください。構成については、266 ページの「ゾーンの構成方法」を参照してください。

Solaris システムごとに 1 つの「大域ゾーン」があります。大域ゾーンは 2 つの機能を持っています。大域ゾーンは、システムのデフォルトのゾーンであり、システム全体の管理に使用されるゾーンでもあります。「大域管理者」が「非大域ゾーン」(単にゾーンとも呼ばれる) を作成した場合を除き、すべてのプロセスが大域ゾーンで実行されます。

非大域ゾーンの構成、インストール、管理、およびアンインストールは、大域ゾーンからのみ行うことができます。システムハードウェアからブートできるのは、大域ゾーンだけです。物理デバイス、共有 IP ゾーンでのルーティング、動的再構成 (DR) といったシステム基盤の管理は、大域ゾーンでのみ行うことができます。大域ゾーンで実行されるプロセスは、適切な権限が付与されていれば、ほかのゾーンに関連付けられているオブジェクトにもアクセスできます。

非大域ゾーンの特権付きプロセスには許可されていない操作を、大域ゾーンの特権のないプロセスが実行できることもあります。たとえば、大域ゾーンのユーザーは、システムのすべてのプロセスに関する情報を表示できます。この機能がサイトで問題になる場合は、大域ゾーンへのアクセスを制限します。

大域ゾーンも含め、各ゾーンにはゾーン名が割り当てられます。大域ゾーンの名前は常に `global` となります。各ゾーンには、一意の数値 ID も与えられます。これは、ゾーンのブート時にシステムによって割り当てられます。大域ゾーンには、常に ID `0` が割り当てられます。ゾーンの名前と数値 ID については、243 ページの「`zonecfg` コマンドの使用」を参照してください。

各ゾーンには、ノード名も割り当てられます。これは、ゾーン名とは完全に独立した名前です。ノード名は、ゾーンの管理者によって割り当てられます。詳細は、380 ページの「非大域ゾーンのノード名」を参照してください。

各ゾーンには、ルートディレクトリのパスが設定されます。これは、大域ゾーンのルートディレクトリに対する相対パスです。詳細は、243 ページの「zonecfg コマンドの使用」を参照してください。

デフォルトでは、非大域ゾーンのスケジューリングクラスは、システムのスケジューリングクラスと同じに設定されます。ゾーンのスケジューリングクラスを設定する方法については、235 ページの「ゾーンのスケジューリングクラス」を参照してください。

`priocntl` (`priocntl(1)` のマニュアルページを参照) を使用すると、デフォルトのスケジューリングクラスの変更やリブートを行うことなく、実行中のプロセスを別のスケジューリングクラスに移動できます。

ゾーン機能のサマリー

次の表に、大域ゾーンと非大域ゾーンの特性をまとめます。

ゾーンの種類	特性
大域	■ システムによって ID `0` が割り当てられます ■ システムでブートされ実行される Solaris カーネルの単一のインスタンスを提供します ■ Oracle Solaris システムソフトウェアパッケージの完全なインストールが含まれています ■ 追加のソフトウェアパッケージや、パッケージを通してインストールされない追加のソフトウェア、ディレクトリ、ファイル、その他のデータが含まれている場合もあります ■ 大域ゾーンにインストールされているすべてのソフトウェアコンポーネントに関する情報を格納した、一貫性のある完全な製品データベースを提供します ■ 大域ゾーンのホスト名やファイルシステムテーブルなど、大域ゾーンのみに固有の構成情報を保持します ■ すべてのデバイスとすべてのファイルシステムが認識される、唯一のゾーンです ■ 非大域ゾーンの存在と構成が認識される、唯一のゾーンです ■ 非大域ゾーンの構成、インストール、管理、またはアンインストールを行うことができる、唯一のゾーンです

ゾーンの種類	特性
非大域	■ ゾーンのブート時にシステムによってゾーン ID が割り当てられます ■ 大域ゾーンからブートされる Solaris カーネルの下で処理を共有します ■ 完全な Oracle Solaris オペレーティングシステムソフトウェアパッケージのインストール済みのサブセットが含まれています ■ 大域ゾーンから共有された Oracle Solaris ソフトウェアパッケージが含まれています ■ 大域ゾーンから共有されたものではない、インストールされた追加のソフトウェアパッケージが含まれていることもあります ■ 追加のソフトウェア、ディレクトリ、ファイル、非大域ゾーンで作成されたその他のデータなど、パッケージを通してインストールされないもの、あるいは、大域ゾーンからの共有でないものが含まれていることもあります ■ ゾーンにインストールされているすべてのソフトウェアコンポーネントに関する情報を格納した、一貫性のある完全な製品データベースを持ちます。非大域ゾーンに置かれているコンポーネントと、読み取り専用モードで大域ゾーンから共有されるコンポーネントがあります ■ ほかのゾーンの存在を認識できません ■ 自身を含め、ゾーンのインストール、管理、アンインストールを行うことはできません ■ 非大域ゾーンのホスト名やファイルシステムテーブルなど、その非大域ゾーンのみに固有の構成情報を保持します ■ 独自のタイムゾーン設定を持つことができます

非大域ゾーンの管理のしくみ

大域管理者は、スーパーユーザー特権または Primary Administrator の役割を持ちます。大域ゾーンにログインすると、大域管理者はシステム全体を監視したり制御したりできます。

非大域ゾーンは「ゾーン管理者」が管理できます。大域管理者が Zone Management プロファイルをゾーン管理者に割り当てます。ゾーン管理者の特権は、非大域ゾーンに対してのみ有効です。

非大域ゾーンの作成のしくみ

大域管理者が zonecfg コマンドを使ってゾーンを構成します。このとき、ゾーンの仮想プラットフォームやアプリケーション環境に応じて、各種のパラメータを指定します。次に、大域管理者がゾーンをインストールします。大域管理者は、ゾーン管理コマンド zoneadm を使用して、ゾーンに対応するファイルシステム階層にソフトウェアをパッケージレベルでインストールします。大域管理者は、zlogin コマンドを使用して、インストールされたゾーンにログインできます。初回ログイン時にゾーンの内部構成が完了します。次に、zoneadm コマンドを使ってゾーンをブートします。

ゾーンの構成については、第 17 章「非大域ゾーンの構成 (概要)」を参照してください。ゾーンのインストールについては、第 19 章「非大域ゾーンのインストール、停止、複製、およびアンインストールについて (概要)」を参照してください。ゾーンへのログインについては、第 21 章「非大域ゾーンへのログイン (概要)」を参照してください。

非大域ゾーンの状態モデル

非大域ゾーンの状態は、次の 6 つのいずれかになります。

構成済み	ゾーンの構成は完了し、安定した記憶領域に確定されています。ただし、ゾーンのアプリケーション環境の要素のうち、最初のブート後に指定する必要のあるものは、まだ含まれていません。
不完全	インストール処理やアンインストール処理の途中は、zoneadm によってターゲットゾーンの状態が「不完全」に設定されます。処理が正常に完了すると、適切な状態に設定されます。
インストール済み	ゾーンの構成はシステム上でインスタンス化されています。zoneadm コマンドにより、指定された Solaris システムでその構成を正常に使用できるかどうかが確認されます。パッケージはゾーンのルートパスにインストールされます。この状態では、ゾーンに関連付けられた仮想プラットフォームはありません。
準備完了	ゾーンの仮想プラットフォームが確立されています。カーネルにより zsched プロセスが作成され、ネットワークインタフェースが設定されてゾーンで使用可能になり、ファイルシステムがマウントされ、デバイスの構成が完了しています。システムにより、一意のゾーン ID が割り当てられます。この状態では、ゾーンに関連付けられたプロセスは起動されていません。

稼働	ゾーンのアプリケーション環境に関連付けられたユーザープロセスが稼働状態です。アプリケーション環境に関連付けられた最初のユーザープロセス (`init`) が作成されるとすぐに、ゾーンの状態は「稼働」になります。
停止処理中および停止	これらは、ゾーンの停止処理の間に見られる遷移状態です。ただし、なんらかの理由でゾーンを停止処理できない場合は、ゾーンがどちらかの状態で停止します。

`zoneadm` コマンドを使用してこれらの状態間の遷移を開始する方法については、第 20 章「非大域ゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)」および `zoneadm`(1M) のマニュアルページを参照してください。

表 16–1 ゾーンの状態に影響を与えるコマンド

ゾーンの現在の状態	適用できるコマンド

構成済み	`zonecfg -z` *zonename* `verify`
	`zonecfg -z` *zonename* `commit`
	`zonecfg -z` *zonename* `delete`
	`zoneadm -z` *zonename* `attach`
	`zoneadm -z` *zonename* `verify`
	`zoneadm -z` *zonename* `install`
	`zoneadm -z` *zonename* `clone`
	`zonecfg` を使用して、構成済みまたはインストール済みの状態にあるゾーンの名前を変更することもできます。

不完全	`zoneadm -z` *zonename* `uninstall`

インストール済み	`zoneadm -z` *zonename* `ready` (オプション)
	`zoneadm -z` *zonename* `boot`
	`zoneadm -z` *zonename* `uninstall` は、指定されたゾーンの構成をシステムからアンインストールします。
	`zoneadm -z` *zonename* `move` *path*
	`zoneadm -z` *zonename* `detach`
	`zonecfg -z` *zonename* を使用すると、`attr`、`bootargs`、`capped-memory`、`dataset`、`dedicated-cpu`、`device`、`fs`、`ip-type`、`limitpriv`、`net`、`rctl`、または `scheduling-class` プロパティーを追加または削除することができます。インストール済み状態のゾーンの名前を変更することもできます。`inherit-pkg-dir` リソースは変更できません。

表 16–1 ゾーンの状態に影響を与えるコマンド (続き)

ゾーンの現在の状態	適用できるコマンド
準備完了	`zoneadm -z` *zonename* `boot` `zoneadm halt` とシステムリブートを実行すると、準備完了状態のゾーンがインストール済み状態に戻ります。 `zonecfg -z` *zonename* を使用すると、`attr`、`bootargs`、`capped-memory`、`dataset`、`dedicated-cpu`、`device`、`fs`、`ip-type`、`limitpriv`、`net`、`rctl`、または `scheduling-class` プロパティーを追加または削除することができます。`inherit-pkg-dir` リソースは変更できません。

稼働	`zlogin` *options* `zonename`
	
	`zoneadm -z` *zonename* `reboot`
	
	`zoneadm -z`*zonename* `halt` を実行すると、準備完了状態のゾーンがインストール済み状態に戻ります。
	
	`zoneadm halt` とシステムのリブートを実行すると、稼働状態のゾーンがインストール済み状態に戻ります。
	
	`zonecfg -z` *zonename* を使用すると、`attr`、`bootargs`、`capped-memory`、`dataset`、`dedicated-cpu`、`device`、`fs`、`ip-type`、`limitpriv`、`net`、`rctl`、または `scheduling-class` プロパティーを追加または削除することができます。`zonepath` リソースと `inherit-pkg-dir` リソースは変更できません。

注 – `zonecfg` 経由で変更されたパラメータは、稼働中のゾーンには影響しません。変更を適用するには、ゾーンをリブートする必要があります。

非大域ゾーンの特性

ゾーンを使用すると、必要に応じてほぼどのような単位にも細かく隔離できます。専用の CPU、物理デバイス、物理メモリーの一部分などをゾーンに割り当てる必要はありません。このようなリソースは、1 つのドメインまたはシステムで実行される複数のゾーンに渡って多重化するか、オペレーティングシステムに用意されているリソース管理機能を使ってゾーンごとに割り当てることができます。

ゾーンごとにカスタマイズされたサービスを提供できます。基本的なプロセス隔離を強化するために、同じゾーン内のプロセスのみ互いに認識したりシグナルを送信したりできます。ゾーン間で基本的な通信を行うには、各ゾーンに IP 接続機能を持たせます。あるゾーンで実行中のアプリケーションが、別のゾーンのネットワークトラフィックを監視することはできません。それぞれのパケットストリームが同じ物理インタフェースを通過する場合でも、この隔離は維持されます。

各ゾーンには、ファイルシステム階層の一部分が割り当てられます。各ゾーンは、ファイルシステム階層で割り当てられた部分ツリーに限定されます。したがって、特定のゾーンで実行されている作業負荷は、別のゾーンで実行されているほかの作業負荷のディスク上のデータにアクセスすることはできません。

ネームサービスで使用されるファイルは、ゾーン独自のルートファイルシステムのビュー内に置かれます。したがって、異なるゾーンのネームサービスは互いに隔離され、サービスごとに異なる構成を使用できます。

非大域ゾーンでのリソース管理機能の使用

リソース管理機能を使用する場合は、リソース管理制御の境界とゾーンの境界をそろえる必要があります。このように境界をそろえることで、名前空間のアクセス、セキュリティー隔離、およびリソースの使用状況をすべて制御できる、より完成された仮想マシンのモデルを作成できます。

ゾーンで各種のリソース管理機能を使用するための特殊要件については、このドキュメントでこれらの機能に関連する各章を参照してください。

非大域ゾーンによって提供される機能

非大域ゾーンは、次のような機能を提供します。

セキュリティー
: 大域ゾーン以外のゾーンにプロセスを配置したあとは、そのプロセス自体やそのプロセスの子がゾーンを変更することはできません。

  ネットワークサービスをゾーンで実行できます。ネットワークサービスをゾーンで実行すると、セキュリティー違反が発生した場合の損害を抑えることができます。ゾーン内で実行されているソフトウェアのセキュリティー欠陥を侵入者が悪用できた場合でも、そのゾーン内で可能な一連の操作しか実行できません。ゾーン内で使用できる特権は、システム全体で使用できる特権の一部のみです。

隔離
: 複数のアプリケーションが異なる信頼ドメインで動作する場合や、大域リソースへの排他的アクセスを必要とする場合、または、大域の構成を使用すると問題を示すような場合でも、ゾーンを使用することでこれらのアプリケーションを同じマシン上に配備できます。たとえば、各ゾーンにそれぞれ異なる IP アドレスを割り当てるか、ワイルドカードアドレスを使用することで、同じシステム上の異なる共有 IP ゾーンで実行される複数のアプリケーションを同じネットワークポートにバインドできます。アプリケーションが互いのネットワークトラ

フィック、ファイルシステムデータ、プロセスの活動などを監視したり妨害したりすることもできなくなります。

ネットワーク隔離

ゾーンが大域ゾーンやほかの非大域ゾーンとは異なる VLAN や LAN に接続される場合など、ネットワークの IP 層で隔離されている必要がある場合は、セキュリティーの理由から、ゾーンは排他的 IP を持つことができます。排他的 IP ゾーンを使用すると、異なる VLAN や LAN の異なるサブネット上で通信しなければならないアプリケーションを統合することができます。

ゾーンは、共有 IP ゾーンとして構成することもできます。このようなゾーンは、大域ゾーンと同じ VLAN または LAN に接続し、IP ルーティングの構成を大域ゾーンと共有します。共有 IP ゾーンは、個別の IP アドレスを持ちますが、IP のほかの部分は共有します。

仮想化

ゾーンによって提供される仮想環境では、物理デバイスやシステムのプライマリ IP アドレスとホスト名などの詳細をアプリケーションから隠すことができます。同じアプリケーション環境を、物理的に異なるマシンで維持管理することもできます。仮想化された環境では、各ゾーンを個別に管理できます。非大域ゾーンでゾーン管理者によって行われる操作は、システムのほかの部分に影響を与えません。

隔離単位

ゾーンを使用すると、ほぼどのような単位にも細かく隔離できます。詳細は、224 ページの「非大域ゾーンの特性」を参照してください。

環境

セキュリティーや隔離の目標を達成するために必要な場合を除き、アプリケーションの実行される環境がゾーンによって変更されることはありません。ゾーンを使用するために、新しい API や ABI にアプリケーションを移植する必要はありません。代わりに、ゾーンでは Solaris の標準インタフェースとアプリケーション環境が提供されます。ただし、いくらかの制限があります。これらの制限は主に、特権付き操作を実行しようとするアプリケーションに影響を与えます。

大域ゾーンで実行されるアプリケーションは、追加のゾーンが構成されたかどうかにかかわらず、変更なしで実行できます。

システムのゾーンの設定 (タスクマップ)

次の表に、システム上ではじめてゾーンを設定する際に必要となるタスクの基本概要を示します。

タスク	説明	参照先
ゾーンで実行するアプリケーションを特定します。	システムで実行されるアプリケーションを見直します。 ■ ビジネス目標にとってどのアプリケーションが重要かを判定します。 ■ 実行するアプリケーションのシステム要件を評価します。	必要に応じて、ビジネス目標とシステムのドキュメントを参照してください。
構成するゾーンの数を決定します。	次の内容を評価します。 ■ ゾーンで実行する予定のアプリケーションの性能要件 ■ インストールするゾーンあたり 100M バイトの推奨空き容量があるかどうか	260 ページの「現在のシステム設定の評価」を参照してください。
コンテナを作成するためにゾーンでリソースプールを使用するかどうかを決定します。	システム上でリソース管理機能も使用する場合は、リソース管理の境界とゾーンの境界をそろえます。ゾーンを構成する前にリソースプールを構成します。 Solaris 10 8/07 以降のリリースでは、`zonecfg` のプロパティーを使用して、ゾーン規模のリソース制御とプール機能をゾーンにすばやく追加できます。	266 ページの「ゾーンの構成方法」および第 13 章「リソースプールの作成と管理 (タスク)」を参照してください。

タスク	説明	参照先
事前構成タスクを行います。	ゾーン名とゾーンパスを決定します。ゾーンを共有 IP ゾーンにするか排他的 IP ゾーンにするかを決定し、IP アドレスまたはデータリンク名を取得します。各ゾーンに必要なファイルシステムとデバイスを決定します。ゾーンのスケジューリングクラスを決定します。標準のデフォルト特権セットでは十分でない場合は、ゾーンのプロセスを制限するための特権セットを決定します。`zonecfg` の設定の中には自動的に特権を追加するものがあります。たとえば、`ip-type=exclusive` はネットワークスタックの構成および管理に必要な複数の特権を自動的に追加します。	ゾーンの名前とパス、IP タイプ、IP アドレス、ファイルシステム、デバイス、スケジューリングクラス、および特権については、第 17 章「非大域ゾーンの構成 (概要)」および 260 ページの「現在のシステム設定の評価」を参照してください。非大域ゾーンでのデフォルトの特権および構成可能な特権のリストについては、397 ページの「非大域ゾーン内の特権」を参照してください。IP 機能の使用の可否については、387 ページの「共有 IP 非大域ゾーンにおけるネットワーク」および390 ページの「Oracle Solaris 10 8/07: 排他的 IP 非大域ゾーンでのネットワーク」を参照してください。
構成を作成します。	非大域ゾーンの構成を行います。	265 ページの「ゾーンを構成、検証、および確定する」および `zonecfg`(1M) のマニュアルページを参照してください。
大域管理者として、構成されたゾーンの確認とインストールを行います。	ゾーンにログインする前に、ゾーンの確認とインストールを行う必要があります。	第 19 章「非大域ゾーンのインストール、停止、複製、およびアンインストールについて (概要)」および第 20 章「非大域ゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)」を参照してください。
`zlogin` コマンドに `-C` オプションを使用するか、`sysidcfg` ファイルをゾーンの `/etc` ディレクトリに配置して、大域管理者として各非大域ゾーンにログインします。		第 21 章「非大域ゾーンへのログイン (概要)」および第 22 章「非大域ゾーンへのログイン (タスク)」を参照してください。

タスク	説明	参照先
大域管理者として、非大域ゾーンをブートします。	それぞれのゾーンをブートして稼働状態にします。	第 19 章「非大域ゾーンのインストール、停止、複製、およびアンインストールについて (概要)」および第 20 章「非大域ゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)」を参照してください。
この新しいゾーンを本稼働用に準備します。	ユーザーアカウントの作成、ソフトウェアの追加、およびゾーン構成のカスタマイズを行います。	新たにインストールしたマシンを設定するためのドキュメントを参照してください。ゾーン環境に関連する特殊な考慮事項については、このガイドを参照してください。

◆ ◆ ◆ 第 17 章

17 非大域ゾーンの構成 (概要)

この章では、非大域ゾーンの構成の概要について説明します。

この章の内容は次のとおりです。

ゾーンの構成について学んだあとで、第 18 章「非大域ゾーンの計画と構成 (タスク)」に進み、システムにインストールする非大域ゾーンを構成します。

lx ブランドゾーンの構成については、第 32 章「lx ブランドゾーン構成の計画 (概要)」および第 33 章「lx ブランドゾーンの構成 (タスク)」を参照してください。

この章に追加されている説明

Solaris 10 6/06: ネイティブな非大域ゾーンのデータセットリソースを追加する機能も含め、ZFS ファイルシステムのサポートが追加されています。詳細は、252 ページの「リソースタイプのプロパティー」を参照してください。

Solaris 10 11/06: 構成可能な特権のサポートが追加されています。243 ページの「Solaris 10 11/06 以降: 構成可能な特権」を参照してください。

Solaris 10 8/07: 次の機能のサポートが zonecfg コマンドに追加されています。

- リソース管理機能とゾーンの統合を向上させます。zonecfg コマンドを使用して、一時プール、メモリー制限、ゾーンのデフォルトのスケジューリングクラス、およびリソース制御の別名を構成できるようになりました。リソース管理の設定を手動で実行する必要はなくなりました。次に示す新しいリソース制御が追加されました。
  - zone.max-locked-memory
  - zone.max-msg-ids
  - zone.max-sem-ids
  - zone.max-shm-ids
  - zone.max-shm-memory
  - zone.max-swap
- zonecfg コマンドを大域ゾーンで使用できます。
- ゾーンの IP タイプを指定できます。非大域ゾーンに使用できる 2 つの IP タイプは、共有 IP と排他的 IP です。
- limitpriv プロパティーを使用して必要な特権を追加することにより、ゾーンで DTrace を使用できます。
- bootargs プロパティーを使用して、ゾーンでブート引数を使用できます。

Solaris 10 10/08: zonecfg ユーティリティーの net リソースに、共有 IP 非大域ゾーンの defrouter プロパティーが追加されました。このプロパティーを使用して、ネットワークインタフェースのデフォルトのルーターを設定できます。

Solaris 10 の新機能の全一覧および Solaris リリースについての説明は、『Oracle Solaris 10 8/11 の新機能』を参照してください。

ゾーンのリソースについて

リソース管理機能を持っているゾーンは、コンテナと呼ばれます。コンテナ内で制御できるリソースには、次のようなものがあります。

- リソースプールまたは割り当てられる CPU。マシンリソースの区分に使用されます。
- リソース制御機能。システムリソースに対する制約メカニズムを提供します。
- スケジューリングクラス。使用可能な CPU リソースのゾーン間での割り当てを、相対的な配分によって制御できます。ゾーンの作業負荷の重要性を、そのゾーンに割り当てる CPU リソースの配分で表すことができます。

インストール前の構成処理

システムに非大域ゾーンをインストールして使用する前に、そのゾーンを構成する必要があります。

`zonecfg` コマンドを使用すると、構成を作成したり、指定されたリソースやプロパティーが仮定のシステム上で有効かどうかを判定したりできます。特定の構成について `zonecfg` で実行される検査では、次のことが確認されます。

- ゾーンパスが指定されていること
- 各リソースの必須プロパティーがすべて指定されていること

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

ゾーンのコンポーネント

このセクションでは、構成できる必須および省略可能なゾーンコンポーネントについて説明します。詳細は、247 ページの「ゾーン構成データ」を参照してください。

ゾーンの名前とパス

ゾーンの名前とパスを選択する必要があります。

ゾーンの自動ブート

`autoboot` プロパティーの設定により、大域ゾーンのブート時にこのゾーンが自動的にブートされるかどうかが決まります。ゾーンサービス `svc:/system/zones:default` も有効になっている必要があります。

リソースプールの関連付け

第 13 章「リソースプールの作成と管理 (タスク)」の説明に従ってシステムでリソースプールを構成した場合は、ゾーンを構成するときに `pool` プロパティーを使用して、リソースプールの 1 つにゾーンを関連付けることができます。

Solaris 10 8/07 以降のリリースでは、リソースプールが構成されていない場合でも、非大域ゾーンの実行中にシステムのプロセッサの一部をそのゾーン専用に割り当てるよう、`dedicated-cpu` リソースを使用して指定できます。ゾーンの実行中に使用される一時プールが動的に作成されます。`zonecfg` によって指定すると、移行時にプールの設定が伝達されます。

注 – pool プロパティーによって設定される持続的プールを使用するゾーン構成と、dedicated-cpu リソースによって構成される一時プールには、互換性がありません。これら 2 つのプロパティーは、どちらか 1 つしか設定できません。

Solaris 10 8/07: dedicated-cpu リソース

dedicated-cpu リソースは、非大域ゾーンの実行中にシステムのプロセッサの一部をそのゾーン専用に割り当てることを指定します。ゾーンのブート時に、ゾーンの実行中に使用される一時プールが動的に作成されます。

zonecfg で指定すると、移行時にプールの設定が伝達されます。

dedicated-cpu リソースは、ncpus の制限を設定し、必要に応じて importance も設定します。

ncpus
: CPU の数を指定するか、CPU の数の範囲を 2–4 などと指定します。リソースプールの動的な動作を得るために範囲を指定する場合は、次の手順も実行してください。
  - importance プロパティーを設定します。
  - poold サービスを有効にします。手順については、174 ページの「Solaris 10 11/06 以降: svcadm を使って動的リソースプールサービスを有効にする方法」を参照してください。

importance
: 動的な動作を得るために CPU 範囲を使用する場合は、importance プロパティーも設定してください。importance は「省略可能な」プロパティーであり、プールの相対的な重要性を定義します。このプロパティーが必要となるのは、ncpus に範囲を指定した場合で、poold によって管理される動的リソースプールを使用しているときだけです。poold が実行されていない場合、importance は無視されます。poold が実行されている場合、importance が設定されていないと、importance はデフォルト値の 1 になります。詳細は、158 ページの「pool.importance プロパティーの制約」を参照してください。

注 – capped-cpu リソースと dedicated-cpu リソースには互換性がありません。cpu-shares リソース制御と dedicated-cpu リソースには互換性がありません。

Solaris 10 5/08: capped-cpu リソース

capped-cpu リソースは、1 つのプロジェクトまたは 1 つのゾーンで消費可能な CPU リソース量に対して絶対的できめの細かい制限を設けます。プロセッサセットと組み合わせて使用すると、CPU キャップはセット内の CPU 使用率を制限しま

す。capped-cpu リソースには、小数点第 2 位までの正の小数である 1 つの ncpus プロパティーがあります。このプロパティーは、CPU のユニット数に対応しています。このリソースには範囲を指定できません。このリソースには小数を指定できます。ncpus を指定する場合、1 の値は 1 つの CPU の 100% を意味します。1.25 の値は 125% を意味します。100% がシステム上の 1 つの CPU の上限となります。

注 – capped-cpu リソースと dedicated-cpu リソースには互換性がありません。

ゾーンのスケジューリングクラス

公平配分スケジューラ (FSS) を使用すると、使用可能な CPU リソースのゾーン間での割り当てを、ゾーンの作業負荷の重要性に基づいて制御できます。この作業負荷の重要性は、各ゾーンに割り当てる CPU リソースの「配分」で表します。CPU リソースのゾーン間での割り当てを管理するために FSS を使用していない場合でも、ゾーン内のプロジェクトに配分を設定するために FSS を使用するよう、ゾーンのスケジューリングクラスを設定することができます。

cpu-shares プロパティーを明示的に設定すると、公平配分スケジューラ (FSS) はそのゾーンのスケジューリングクラスとして使用されます。ただし、この場合に望ましい FSS の使用方法は、dispadmin コマンドを使用して、FSS をシステムのデフォルトのスケジューリングクラスに設定する方法です。このようにすると、すべてのゾーンがシステムの CPU リソースの公平配分を受けることができます。ゾーンに対して cpu-shares が設定されていない場合、そのゾーンはシステムのデフォルトのスケジューリングクラスを使用します。ゾーンのスケジューリングクラスは、次の処理によって設定されます。

- Solaris 10 8/07 リリースでは、zonecfg の scheduling-class プロパティーを使ってゾーンのスケジューリングクラスを設定できます。
- リソースプール機能を使ってゾーンのスケジューリングクラスを設定できます。ゾーンがプールに関連付けられている場合、そのプールの pool.scheduler プロパティーに有効なスケジューリングクラスが設定されていれば、ゾーンで実行されるプロセスは、デフォルトでそのスケジューリングクラスで実行されます。148 ページの「リソースプールの紹介」 および 181 ページの「プールをスケジューリングクラスに対応付ける方法」 を参照してください。
- cpu-shares リソース制御が設定されている場合で、別の処理を通して FSS がゾーンのスケジューリングクラスとして設定されていないときは、ゾーンのブート時に zoneadmd によってスケジューリングクラスが FSS に設定されます。
- ほかの処理を通してスケジューリングクラスが設定されていない場合、ゾーンはシステムのデフォルトのスケジューリングクラスを継承します。

priocntl (priocntl(1) のマニュアルページを参照) を使用すると、デフォルトのスケジューリングクラスの変更やリブートを行うことなく、実行中のプロセスを別のスケジューリングクラスに移動できます。

Solaris 10 8/07: 物理メモリーの制御と capped-memory リソース

capped-memory リソースは、physical、swap、および locked メモリーの制限を設定します。各制限はオプションですが、少なくとも 1 つは設定する必要があります。

- 大域ゾーンから rcapd を使用してゾーンのメモリー上限を設定する場合は、このリソースの値を決定します。capped-memory リソースの physical プロパティーは、ゾーンの max-rss 値として rcapd で使用されます。
- capped-memory リソースの swap プロパティーは、zone.max-swap リソース制御を設定するための望ましい方法です。
- capped-memory リソースの locked プロパティーは、zone.max-locked-memory リソース制御を設定するための望ましい方法です。

注 – 通常はアプリケーションが多量のメモリーをロックすることはありませんが、ゾーンのアプリケーションによってメモリーがロックされることがわかっている場合は、ロックされるメモリーを設定するとよいでしょう。ゾーンの信頼が問題になる場合は、ロックされるメモリーの上限を、システムの物理メモリーの 10 パーセントまたはゾーンの物理メモリー上限の 10 パーセントに設定することもできます。

詳細は、第 10 章「リソース上限デーモンによる物理メモリーの制御 (概要)」、第 11 章「リソース上限デーモンの管理 (タスク)」、および 266 ページの「ゾーンの構成方法」を参照してください。ゾーンに一時的なリソース上限を設定する方法については、142 ページの「ゾーンに一時的なリソース上限を指定する方法」を参照してください。

ゾーンネットワークインタフェース

ネットワーク接続を提供するために zonecfg コマンドによって構成されるゾーンネットワークインタフェースは、ゾーンのブート時に自動的に設定されてゾーン内に配置されます。

インターネットプロトコル (IP) 層は、ネットワークのパケットの受信と配信を行います。この層には、IP ルーティング、アドレス解決プロトコル (ARP)、IP セキュリティーアーキテクチャー (IPsec)、および IP フィルタが含まれます。

非大域ゾーンに使用できる IP タイプには、共有 IP と排他的 IP の 2 種類があります。共有 IP ゾーンはネットワークインタフェースを共有し、排他的 IP ゾーンには専用のネットワークインタフェースが必要です。

各タイプの IP 機能については、387 ページの「共有 IP 非大域ゾーンにおけるネットワーク」および390 ページの「Oracle Solaris 10 8/07: 排他的 IP 非大域ゾーンでのネットワーク」を参照してください。

共有 IP 非大域ゾーン

デフォルトのタイプは共有 IP ゾーンです。ゾーンには、1 つ以上の専用の IP アドレスが存在する必要があります。共有 IP ゾーンは、IP 層の構成と状態を大域ゾーンと共有します。次の両方の条件が満たされる場合、ゾーンは共有 IP インスタンスを使用すべきです。

- ゾーンは大域ゾーンと同じデータリンクに接続される、つまり、大域ゾーンと同じ IP サブネット上に配置される。
- 排他的 IP ゾーンによって提供されるその他の機能は必要でない。

共有 IP ゾーンには、`zonecfg` コマンドを使用して 1 つ以上の IP アドレスを割り当てます。大域ゾーンでデータリンク名も構成する必要があります。

これらのアドレスは、論理ネットワークインタフェースに関連付けられます。大域ゾーンから `ifconfig` コマンドを使用すると、稼働中のゾーンの論理インタフェースを追加したり削除したりできます。詳細は、388 ページの「共有 IP ネットワークインタフェース」を参照してください。

Solaris 10 8/07: 排他的 IP 非大域ゾーン

排他的 IP ゾーンでは、IP レベルのすべての機能が使用可能です。

排他的 IP ゾーンは、IP 関連の状態を独自に保持します。

排他的 IP ゾーンでは次のような機能を使用できます。

- DHCPv4 および IPv6 ステートレスアドレスの自動構成
- IP フィルタ。ネットワークアドレス変換 (NAT) 機能も含む
- IP ネットワークマルチパス (IPMP)
- IP ルーティング
- TCP/UDP/SCTP および IP/ARP レベルのノブを設定するための `ndd`

- IP セキュリティー (IPsec) と Internet Key Exchange (IKE)。これは、IPsec セキュリティーアソシエーション用の認証済み鍵材料のプロビジョニングを自動化する

排他的 IP ゾーンには、`zonecfg` コマンドを使用して独自のデータリンクセットを割り当てます。`net` リソースの `physical` プロパティーを使用して、ゾーンに `xge0`、`e1000g1`、`bge32001` などのデータリンク名を割り当てます。`net` リソースの `address` プロパティーは設定されません。

データリンクを割り当てると `snoop` コマンドが使用可能になります。

`dladm` コマンドを `show-linkprop` サブコマンドとともに使用して、実行中の排他的 IP ゾーンに対するデータリンクの割り当てを表示できます。`dladm` コマンドを `set-linkprop` サブコマンドとともに使用すると、実行中のゾーンに対して追加のデータリンクを割り当てることができます。使用例については、427 ページの「Oracle Solaris 10 8/07: 排他的 IP 非大域ゾーンでのデータリンクの管理」を参照してください。

実行中の排他的 IP ゾーンの内部で、`ifconfig` コマンドを使用して IP を構成できます。これには、論理インタフェースの追加や削除も含まれます。ゾーンの IP 構成は、大域ゾーンの場合と同様に `sysidtools` を使用して設定できます。詳細は、`sysidcfg(4)` のマニュアルページを参照してください。

注 – 排他的 IP ゾーンの IP 構成は、大域ゾーンから `zlogin` コマンドを使用することによってのみ表示できます。次に例を示します。

```
global# zlogin zone1 ifconfig -a
```

共有 IP 非大域ゾーンと排他的 IP 非大域ゾーンのセキュリティーの相違

共有 IP ゾーン内のアプリケーションは、スーパーユーザーも含め、`zonecfg` ユーティリティーを介してゾーンに割り当てられた IP アドレス以外をソース IP アドレスとしてパケットを送信することはできません。このタイプのゾーンには、任意のデータリンク (レイヤー 2) パケットを送受信するアクセス権はありません。

一方、排他的 IP ゾーンの場合は、`zonecfg` によって指定されたデータリンク全体がゾーンに対して許可されます。その結果、スーパーユーザーは、排他的 IP ゾーンでも大域ゾーンと同様に、偽のパケットをこれらのデータリンク上で送信できます。

共有 IP 非大域ゾーンと排他的 IP 非大域ゾーンの同時使用

共有 IP ゾーンは常に IP 層を大域ゾーンと共有し、排他的 IP ゾーンは常に独自の IP 層インスタンスを持っています。共有 IP ゾーンと排他的 IP ゾーンの両方を同じマシンで使用することができます。

ゾーンでマウントされるファイルシステム

通常、ゾーンでマウントされるファイルシステムには、次のものが含まれます。

- 仮想プラットフォームの初期化時にマウントされる一連のファイルシステム
- アプリケーション環境自体の内部からマウントされる一連のファイルシステム

これには、たとえば次のようなファイルシステムが含まれます。

- ゾーンの `/etc/vfstab` ファイルで指定されたファイルシステム
- `AutoFS` によるマウントおよび `AutoFS` によって引き起こされるマウント
- ゾーン管理者が明示的に実行するマウント

アプリケーション環境内部から実行されるマウントには、いくつかの制限事項があります。これらの制限事項は、ほかのゾーンに悪影響を与えないようにするために、ゾーン管理者がシステムのほかの部分に対するサービスを拒否できないようにします。

一部のファイルシステムについては、ゾーン内部からマウントする場合にセキュリティー制限があります。ほかのファイルシステムは、ゾーン内でマウントされたときに特有の動作を行います。詳細は、380 ページの「ファイルシステムと非大域ゾーン」を参照してください。

注 – 別個の `/var` ファイルシステムを含む `native` 非大域ゾーン構成は、Oracle Solaris 10 ではサポートされません。この構成のシステムでは、`patchadd` コマンドと、`zoneadm install`、`detach`、`attach`、および接続時更新操作が失敗することがあります。サポートされる構成およびサポートされない構成の詳細については、442 ページの「大域ゾーンによってデータが挿入されているファイルシステムをゾーン管理者がマウントする場合」を参照してください。

ゾーンで構成されるデバイス

`zonecfg` コマンドは、規則照合方式を使って、特定のゾーンにどのデバイスを配置するかを指定します。いずれかのルールに一致するデバイスは、ゾーンの `/dev` ファイルシステムに追加されます。詳細は、266 ページの「ゾーンの構成方法」を参照してください。

ゾーン内のホスト ID

非大域ゾーンでは、大域ゾーンの `hostid` とは異なる `hostid` プロパティーを設定できます。この設定は、P2V (Physical-To-Virtual) 機能を使用して物理マシンをゾーンに統合する場合に行われます。ゾーン内に配置されたアプリケーションが元の `hostid`

に依存する場合があり、アプリケーション構成を更新できないことがあります。詳細は、247 ページの「リソースタイプとプロパティータイプ」を参照してください。

ゾーン規模のリソース制御の設定

大域管理者は、ゾーン規模の特権付きリソース制御をゾーンに対して設定できます。ゾーン規模のリソース制御は、ゾーン内のすべてのプロセスエンティティーによる総リソース消費を制限します。

これらの制限は、大域ゾーンと非大域ゾーンのどちらに対しても、`zonecfg` コマンドを使用して指定します。266 ページの「ゾーンの構成方法」を参照してください。

Solaris 10 8/07 以降のリリースでは、ゾーン規模のリソース制御を設定する場合に望ましい、より簡単な方法は、`rctl` リソースの代わりにプロパティー名を使用する方法です。

Solaris 10 5/08: `zone.cpu-cap` リソース制御は、1 つのゾーンで消費可能な CPU リソース量に対する絶対的な制限を設定します。`project.cpu-cap` 設定と同様、`100` の値は 1 つの CPU の 100% を意味します。`125` の値は 125% になります。CPU キャップの使用時は、100% がシステム上の 1 つの CPU の上限となります。

注 – `capped-cpu` リソースを設定する場合は、単位に小数を使用できます。この値は `zone.capped-cpu` リソース制御と相互に関連していますが、設定値はその 100 分の 1 になります。設定値 `1` はリソース制御の設定値 `100` に等しくなります。

`zone.cpu-shares` リソース制御は、公平配分スケジューラ (FSS) の CPU 配分の制限をゾーンに対して設定します。CPU 配分は、まずゾーンに対して割り当てられたあとで、`project.cpu-shares` エントリの指定に従って、ゾーン内のプロジェクトに分配されます。詳細は、429 ページの「ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用」を参照してください。この制御のグローバルプロパティー名は `cpu-shares` です。

`zone.max-locked-memory` リソース制御は、ゾーンで使用できるロックされた物理メモリーの量を制限します。ゾーン内のプロジェクト間でのロックされたメモリーリソースの割り当ては、`project.max-locked-memory` リソース制御を使用して制御できます。詳細は、表 6–1 を参照してください。

1 つのゾーンの LWP が多くなりすぎると、ほかのゾーンに影響を与えることがあります。`zone.max-lwps` リソース制御は、これを防ぐことで、リソースの隔離を向上させます。ゾーン内のプロジェクト間での LWP リソースの割り当ては、`project.max-lwps` リソース制御を使用して制御できます。詳細は、表 6–1 を参照してください。この制御のグローバルプロパティー名は `max-lwps` です。

zone.max-msg-ids、zone.max-sem-ids、zone.max-shm-ids、および zone.max-shm-memory の各リソース制御は、ゾーン内のすべてのプロセスで使用される System V リソースを制限します。ゾーン内のプロジェクト間での System V リソースの割り当ては、これらのリソース制御の project バージョンを使用して制御できます。これらの制御のグローバルプロパティー名は、max-msg-ids、max-sem-ids、max-shm-ids、および max-shm-memory です。

zone.max-swap リソース制御は、ゾーン内のユーザープロセスのアドレス空間マッピングと tmpfs マウントで消費されるスワップを制限します。prstat -Z の出力は「スワップ」列を表示します。報告されるスワップは、ゾーンのプロセスと tmpfs マウントで消費されるスワップの合計量です。この値により、各ゾーンで予約されているスワップを監視しやすくなり、適切な zone.max-swap 設定を選択することができます。

表 17–1 ゾーン規模のリソース制御

制御名	グローバルプロパティー名	説明	デフォルトの単位	使用される値
zone.cpu-cap		**Solaris 10 5/08:** このゾーンに対する CPU リソース量の絶対的な制限。	数量 (CPU の数)、パーセントで表されます。 注 – capped-cpu リソースとして設定する場合は、単位に小数を使用できます。	
zone.cpu-shares	cpu-shares	このゾーンに対する公平配分スケジューラ (FSS) の CPU 配分。	数量 (配分)	

表 17–1 ゾーン規模のリソース制御 (続き)

制御名	グローバルプロパティー名	説明	デフォルトの単位	使用される値
zone.max-locked-memory		ゾーンで使用できるロックされた物理メモリーの合計量。 priv_proc_lock_memory がゾーンに割り当てられている場合、そのゾーンがすべてのメモリーをロックするのを防ぐため、このリソース制御の設定も検討してください。	サイズ (バイト)	capped-memory の locked プロパティー。
zone.max-lwps	max-lwps	このゾーンで同時に使用できる LWP の最大数。	数量 (LWP 数)	
zone.max-msg-ids	max-msg-ids	このゾーンに許容されるメッセージキュー ID の最大数。	数量 (メッセージキュー ID の数)	
zone.max-sem-ids	max-sem-ids	このゾーンに許容されるセマフォー ID の最大数。	数量 (セマフォー ID の数)	
zone.max-shm-ids	max-shm-ids	このゾーンに許容される共有メモリー ID の最大数。	数量 (共有メモリー ID の数)	
zone.max-shm-memory	max-shm-memory	このゾーンに許容される System V 共有メモリーの合計量。	サイズ (バイト)	
zone.max-swap		このゾーンのユーザープロセスのアドレス空間マッピングと tmpfs マウントで消費できるスワップの合計量。	サイズ (バイト)	capped-memory の swap プロパティー

prctl コマンドを使用すると、実行中のプロセスに対してこれらの制限を指定できます。例については、429 ページの「prctl コマンドを使用して大域ゾーンの FSS 配分を設定する方法」を参照してください。prctl コマンドで指定された制限には持続性がありません。システムがリブートされると、制限は無効になります。

Solaris 10 11/06 以降: 構成可能な特権

ゾーンのブート時に、*safe* 特権のデフォルトセットが構成に含められます。これらの特権は、ゾーン内の特権プロセスがシステムのほかの非大域ゾーン内のプロセスや大域ゾーン内のプロセスに影響を及ぼすことを防ぐため、安全と見なされます。`zonecfg` コマンドを使用して、次の操作を実行できます。

- デフォルトの特権セットに追加します。ただし、この種の変更を行うと、あるゾーン内のプロセスがグローバルリソースを制御できるようになって、ほかのゾーン内のプロセスに影響する場合があります。
- デフォルトの特権セットから削除します。ただし、この種の変更を行うと、実行に必要な特権がないため一部のプロセスが正しく動作しなくなる場合があります。

注 – わずかですが、この時点でゾーンのデフォルト特権セットから削除できない特権があります。同じように、特権セットに追加できない特権もあります。

詳細は、397 ページの「非大域ゾーン内の特権」、266 ページの「ゾーンの構成方法」、および `privileges(5)` のマニュアルページを参照してください。

ゾーンのコメントの追加

`attr` リソースの型を使ってゾーンのコメントを追加できます。詳細は、266 ページの「ゾーンの構成方法」を参照してください。

zonecfg コマンドの使用

`zonecfg` コマンド (`zonecfg(1M)` のマニュアルページを参照) は、非大域ゾーンを構成するために使用します。Solaris 10 8/07 リリースでは、大域ゾーンのリソース管理設定を持続的に指定する場合にも、このコマンドを使用できます。

`zonecfg` コマンドは、対話型モード、コマンド行モード、またはコマンドファイルモードで使用できます。このコマンドを使用して、次の操作を実行できます。

- ゾーン構成を作成または削除 (破棄) します
- 特定の構成にリソースを追加します
- 構成に追加したリソースのプロパティーを設定します
- 特定の構成からリソースを削除します
- 構成の照会または確認を行います
- 構成を確定します
- 前の構成に戻します
- ゾーンの名前を変更します

- zonecfg のセッションを終了します

zonecfg のプロンプトは次のような形式です。

zonecfg:*zonename*>

ファイルシステムなど、特定のリソースタイプの構成を行うときは、そのリソースタイプもプロンプトに表示されます。

zonecfg:*zonename*:fs>

この章で説明する zonecfg のさまざまなコンポーネントの使用手順など、詳細については、第 18 章「非大域ゾーンの計画と構成 (タスク)」を参照してください。

zonecfg のモード

このユーザーインタフェースでは「有効範囲」という概念が使用されます。有効範囲は、「大域」または「リソース固有」のどちらかです。デフォルトの有効範囲は大域です。

大域有効範囲で add サブコマンドまたは select サブコマンドを使用すると、特定のリソースが選択されます。すると、有効範囲がそのリソースタイプに変わります。

- add サブコマンドの場合、end、cancel のいずれかのサブコマンドを使用すると、リソースの指定が完了します。
- select サブコマンドの場合、end、cancel のいずれかのサブコマンドを使用すると、リソースの変更が完了します。

すると、有効範囲が大域に戻ります。

add、remove、set などのように、有効範囲によって異なる意味を持つサブコマンドもあります。

zonecfg の対話型モード

対話型モードでは、次のサブコマンドがサポートされます。サブコマンドで使用する意味とオプションの詳細については、zonecfg(1M) のマニュアルページでオプション情報を参照してください。破壊的な操作や作業内容の消失を伴うようなサブコマンドの場合、処理を実行する前にユーザーの確認が求められます。-F (強制) オプションを使用すると、この確認手順を省略できます。

help　一般ヘルプまたは特定のリソースに関するヘルプを表示します。

```
zonecfg:my-zone:inherit-pkg-dir> help
```

create　指定された新しいゾーンに使用するメモリー内構成の構成を開始します。次のような目的に使用されます。

- デフォルト設定を新しい構成に適用します。この方法がデフォルトです。
- `-t` *template* オプションを使用して、指定したテンプレートと同一の構成を作成します。ゾーン名がテンプレート名から新しいゾーン名に変更されます。
- `-F` オプションを使用して、既存の構成を上書きします。
- `-b` オプションを使用して、なにも設定されていない空の構成を作成します。

`export` 標準出力または指定された出力ファイルに、コマンドファイルに使用できる形式で構成を出力します。

`add` 大域有効範囲では、指定されたリソースタイプを構成に追加します。

リソース固有の有効範囲では、指定された名前と値を持つプロパティーを追加します。

詳細は、266 ページの「ゾーンの構成方法」および `zonecfg`(1M) のマニュアルページを参照してください。

`set` 指定されたプロパティー名を、指定されたプロパティー値に設定します。`zonepath` などの大域的なプロパティーと、リソース固有のプロパティーがあることに注意してください。このコマンドは、大域有効範囲とリソース固有の有効範囲の両方で使用できます。

`select` 大域有効範囲でのみ使用できます。指定されたタイプのリソースのうち、指定されたプロパティー名とプロパティー値の対の条件に一致するものを、変更対象として選択します。有効範囲がそのリソースタイプに変わります。リソースが一意に識別されるように、プロパティーの名前と値の対を十分な数だけ指定する必要があります。

`clear` **Solaris 10 8/07:** オプションの設定の値をクリアーします。必須の設定はクリアーできません。ただし、必須の設定のいくつかは、新しい値を割り当てることによって変更できます。

`remove` 大域有効範囲では、指定されたリソースタイプを削除します。リソースタイプが一意に識別されるように、プロパティーの名前と値の対を十分な数だけ指定する必要があります。プロパティーの名前と値の対をまったく指定しないと、すべてのインスタンスが削除されます。該当するものが複数ある場合は、`-F` オプションを使用していない限り、確認を求めるメッセージが表示されます。

リソース固有の有効範囲では、指定された名前と値を持つプロパティーを現在のリソースから削除します。

`end` リソース固有の有効範囲でのみ使用できます。リソースの指定を終了します。

次に、zonecfg コマンドは、現在のリソースが正しく指定されているかどうかを確認します。

- リソースが正しく指定されている場合は、そのリソースがメモリー内に保持される構成に追加され、有効範囲が大域に戻ります。
- 指定が不完全な場合は、必要な作業を示すエラーメッセージが表示されます。

`cancel` リソース固有の有効範囲でのみ使用できます。リソースの指定を終了し、有効範囲を大域に戻します。リソースの指定が不完全な場合、そのリソースは保持されません。

`delete` 指定された構成を破棄します。メモリーと安定した記憶領域の両方から構成を削除します。`delete` に `-F` (強制) オプションを使用する必要があります。

注意 – この操作は即時に実行されます。確定手順は行われず、削除されたゾーンを元に戻すことはできません。

`info` 現在の構成または大域のリソースプロパティー `zonepath`、`autoboot`、および `pool` に関する情報を表示します。リソースタイプが指定されている場合は、そのタイプのリソースについてのみ情報を表示します。リソース固有の有効範囲では、このサブコマンドは、追加または変更しようとしているリソースにのみ適用されます。

`verify` 現在の構成が正しいかどうかを確認します。各リソースに必須プロパティーがすべて指定されていることを確認します。

`commit` 現在の構成をメモリーから安定した記憶領域に確定します。メモリー内の構成を確定するまでは、`revert` サブコマンドで変更内容を削除できます。`zoneadm` で構成を使用するには、その構成を確定する必要があります。`zonecfg` セッションを完了するときに、この操作の実行が自動的に試みられます。正しい構成のみ確定できるので、確定操作では自動的に確認も行われます。

`revert` 構成を最後に確定されたときの状態に戻します。

`exit` `zonecfg` のセッションを終了します。`exit` に `-F` (強制) オプションを使用できます。

必要な場合は、`commit` 操作が自動的に試行されます。EOF 文字を使ってセッションを終了することもできることに注意してください。

zonecfg のコマンドファイルモード

コマンドファイルモードでは、ファイルから入力されます。このファイルを生成するには、244 ページの「zonecfg の対話型モード」で説明されている export サブコマンドを使用します。構成を標準出力に出力するか、-f オプションで指定した出力ファイルに出力することができます。

ゾーン構成データ

ゾーン構成データは 2 種類のエンティティーから成ります。リソースとプロパティーです。各リソースは、タイプのほかにも 1 つ以上のプロパティーを持つことがあります。プロパティーは名前と値から成ります。どのようなプロパティーセットを持つかは、リソースタイプによって異なります。

リソースタイプとプロパティータイプ

リソースタイプとプロパティータイプは次のとおりです。

ゾーン名

ゾーン名は、構成ユーティリティーでゾーンを識別するために使用されます。ゾーン名には次のような規則が適用されます。

- 各ゾーンの名前は一意でなければならない。
- ゾーン名では大文字と小文字が区別される。
- ゾーン名は英数字で始まる必要がある。

  名前には、英数字、下線 (_)、ハイフン (-)、およびピリオド (.) を使用できます。
- 名前の長さは 64 文字以内でなければならない。
- global という名前と SUNW で始まるすべての名前は、予約されているので使用できない。

zonepath

zonepath プロパティーは、ゾーンのルートを含むパスです。各ゾーンでは、大域ゾーンのルートディレクトリファイルシステムの zonepath の下に root ディレクトリがあります。ゾーンのインストール時に、適切な所有者とモードによる zonepath ディレクトリ階層が作成されます。zonepath ディレクトリは、所有者が root、モードが 700 で所有される必要があります。

非大域ゾーンのルートパスは 1 つ下のレベルになります。ゾーンのルートディレクトリの所有権とアクセス権

は、大域ゾーンのルートディレクトリ (/) と同じになります。ゾーンのディレクトリの所有者は root で、モードは 755 であることが必要です。これらのディレクトリは正しいアクセス権を使って自動作成され、ゾーン管理者がこれらのディレクトリを検証する必要はありません。この階層構造により、大域ゾーンのユーザーでも権限を持っていない場合は、非大域ゾーンのファイルシステムと行き来できなくなります。

パス	説明
/home/export/my-zone	zonecfg zonepath
/home/export/my-zone/root	ゾーンのルート
/home/export/my-zone/dev	ゾーン用に作成されたデバイス

この問題の詳細については、386 ページの「ファイルシステムの行き来」を参照してください。

注 – 各リリースでの ZFS の制限については、441 ページの「Oracle Solaris 10 6/06、Oracle Solaris 10 11/06、Oracle Solaris 10 8/07、および Oracle Solaris 10 5/08: 非大域ゾーンのルートファイルシステムを ZFS 上に配置しないでください」を参照してください。

---

autoboot

このプロパティーを true に設定すると、大域ゾーンのブート時にこのゾーンが自動的にブートされます。ゾーンサービス svc:/system/zones:default が無効になっている場合、このプロパティーの設定にかかわらず、ゾーンは自動的にはブートしません。svcadm(1M) のマニュアルページに記載されているように、svcadm コマンドを使用してゾーンサービスを有効にできます。

```
global# svcadm enable zones
```

bootargs

Solaris 10 8/07: このプロパティーは、ゾーンのブート引数を設定するために使用します。reboot、zoneadm boot、または zoneadm reboot コマンドで無効にされた場合を除き、このブート引数が適用されます。286 ページの「Solaris 10 8/07: ゾーンのブート引数」を参照してください。

pool

このプロパティーは、システム上のリソースプールをゾーンに関連付けるために使用します。1 つのプール内

のリソースを複数のゾーンが共有してもかまいません。234 ページの「Solaris 10 8/07: dedicated-cpu リソース」も参照してください。

`limitpriv`

Solaris 10 11/06 以降**:** このプロパティーは、デフォルト以外の特権マスクを指定する場合に使用します。397 ページの「非大域ゾーン内の特権」を参照してください。

特権を追加するには、特権名だけを指定するか、特権名の前に `priv_` 付けて指定します。特権を除外するには、名前の前にダッシュ (-) または感嘆符 (!) を付けます。複数の特権は、コンマで区切り、引用符 (“) で囲みます。

`priv_str_to_set(3C)`で説明されているように、特殊な特権セット `none`、`all`、および `basic` は、それぞれの通常の定義に展開されます。ゾーン構成は大域ゾーンで行われるため、特殊な特権セット `zone` は使用できません。特定の特権を追加または削除してデフォルトの特権セットを変更するのが一般的な使用方法であるため、特殊なセットである `default` はデフォルトの特権セットにマップされます。`limitpriv` プロパティーの先頭に `default` がある場合、デフォルトセットに展開されます。

次のエントリは、`dtrace_proc` 特権と `dtrace_user` 特権だけを必要とする DTrace プログラムをゾーンで使用できるようにします。

```
global# zonecfg -z userzone
zonecfg:userzone> set limitpriv="default,dtrace_proc,dtrace_user"
```

ゾーンの特権セットに不許可の特権が含まれる場合、必須の特権が欠落している場合、または未知の特権が含まれる場合、ゾーンの検証、準備、またはブートの試行は失敗し、エラーメッセージが表示されます。

`scheduling-class`

Solaris 10 8/07: このプロパティーは、ゾーンのスケジューリングクラスを設定します。詳細とヒントについては、235 ページの「ゾーンのスケジューリングクラス」を参照してください。

`ip-type`

Solaris 10 8/07: このプロパティーは、ゾーンが排他的 IP ゾーンである場合のみ設定する必要があります。237 ページの「Solaris 10 8/07: 排他的 IP 非大域ゾーン」および266 ページの「ゾーンの構成方法」を参照してください。

`dedicated-cpu`	**Solaris 10 8/07:** このリソースは、ゾーンの実行中にシステムのプロセッサの一部をそのゾーン専用に割り当てます。`dedicated-cpu` リソースは、`ncpus` の制限を設定し、必要に応じて `importance` も設定します。詳細は、234 ページの「Solaris 10 8/07: `dedicated-cpu` リソース」を参照してください。
`capped-cpu` リソース	**Solaris 10 5/08:** このリソースは、ゾーンの実行中にゾーンで消費可能な CPU リソース量に対する制限を設定します。このリソースは、`ncpus` に制限を設けます。
`capped-memory` リソース	**Solaris 10 8/07:** このリソースは、ゾーンのメモリー上限を設定するためのプロパティーをグループ化します。`capped-memory` リソースは、`physical`、`swap`、および `locked` メモリーの制限を設定します。これらのプロパティーの少なくとも 1 つは指定する必要があります。
`dataset`	**Solaris 10 6/06:** ZFS ファイルシステムのデータセットリソースを追加すると、ストレージ管理を非大域ゾーンに委譲できます。ゾーン管理者は、そのデータセット内のファイルシステムの作成と破棄、クローンの作成と破棄、およびデータセットのプロパティーの変更を行うことができます。ゾーン管理者は、ゾーンに追加されていないデータセットを操作したり、ゾーンに割り当てられているデータセットに設定されている最上位レベルの割り当て制限を超過したりすることはできません。 次の方法で、ZFS データセットをゾーンに追加できます。 ■ lofs マウントされたファイルシステムとして (大域ゾーンとの領域共有のみが目的の場合) ■ 委任されたデータセットとして 『Oracle Solaris ZFS 管理ガイド』の第 10 章「Oracle Solaris ZFS の高度なトピック」および380 ページの「ファイルシステムと非大域ゾーン」を参照してください。 データセットの問題については、第 30 章「Oracle Solaris ゾーンで発生するさまざまな問題のトラブルシューティング」も参照してください。
`fs`	各ゾーンでは、インストール済み状態から準備完了状態に移行するときにマウントする各種のファイルシステムを指定できます。ファイルシステムリソースは、ファイルシステムのマウントポイントのパスを指定します。ゾーンでファイルシステムを使用する方法の詳細に

	ついては、380 ページの「ファイルシステムと非大域ゾーン」を参照してください。
inherit-pkg-dir	完全ルートゾーンでは、このリソースを構成するべきではありません。 疎ルートゾーンで inherit-pkg-dir リソースを使用すると、非大域ゾーンが大域ゾーンと共有するソフトウェアパッケージの保存先ディレクトリを指定できます。 inherit-pkg-dir ディレクトリに転送されるソフトウェアパッケージの内容は、非大域ゾーンでは読み取り専用モードで継承されます。ゾーンのパッケージデータベースが更新され、パッケージが反映されます。zoneadm を使用してゾーンをインストールした後で、これらのリソースを変更または削除することはできません。 注 – 構成には、デフォルトの inherit-pkg-dir リソースが 4 つ含まれています。これらのディレクトリリソースは、大域ゾーンからどのディレクトリが関連パッケージを継承するべきかを指定します。リソースの実装は、読み取り専用のループバックファイルシステムマウントによって行われます。 ■ /lib ■ /platform ■ /sbin ■ /usr
net	ネットワークインタフェースリソースは、インタフェースの名前です。各ゾーンでは、インストール済み状態から準備完了状態に移行するときに設定するべきネットワークインタフェースを指定できます。
device	デバイスリソースは、デバイス照合の指定子です。各ゾーンでは、インストール済み状態から準備完了状態に移行するときに構成するべきデバイスを指定できます。
rctl	rctl リソースは、ゾーン規模のリソース制御に使用されます。リソース制御は、ゾーンがインストール済み状態から準備完了状態に移行するときに有効になります。
hostid	大域ゾーンの hostid とは異なる hostid を設定できます。
attr	この汎用属性は、ユーザーコメントとして使用したり、ほかのサブシステムで使用したりできます。attr の

name プロパティーは、英数字で始まる必要があります。name プロパティーには、英数字、ハイフン (-)、およびピリオド (.) を使用できます。zone. で始まる属性名はシステム用に予約されています。

リソースタイプのプロパティー

リソースには、構成可能なプロパティーもあります。リソースタイプとそれに関連付けられるプロパティーは次のとおりです。

dedicated-cpu　ncpus、importance

Solaris 10 8/07: CPU の数を指定し、必要に応じてプールの相対的な重要性も指定します。次の例では、ゾーン my-zone で使用する CPU の範囲を指定します。importance も設定します。

```
zonecfg:my-zone> add dedicated-cpu
zonecfg:my-zone:dedicated-cpu> set ncpus=1-3
zonecfg:my-zone:dedicated-cpu> set importance=2
zonecfg:my-zone:dedicated-cpu> end
```

capped-cpu　ncpus

CPU の数を指定します。次の例では、ゾーン my-zone の CPU 数のキャップを 3.5 に指定します。

```
zonecfg:my-zone> add capped-cpu
zonecfg:my-zone:capped-cpu> set ncpus=3.5
zonecfg:my-zone:capped-cpu> end
```

capped-memory　physical、swap、locked

ゾーン my-zone のメモリー制限を指定します。各制限はオプションですが、少なくとも 1 つは設定する必要があります。

```
zonecfg:my-zone> add capped-memory
zonecfg:my-zone:capped-memory> set physical=50m
zonecfg:my-zone:capped-memory> set swap=100m
zonecfg:my-zone:capped-memory> set locked=30m
zonecfg:my-zone:capped-memory> end
```

fs　dir、special、raw、type、options

fs リソースのパラメータは、ファイルシステムをマウントする方法と場所を決定する値を指定します。fs のパラメータは次のように定義されています。

dir　ファイルシステムのマウントポイントを指定します

special　大域ゾーンからマウントするブロック型特殊デバイスの名前またはディレクトリを指定します

raw	ファイルシステムをマウントする前に fsck を実行する、raw デバイスを指定します
type	ファイルシステムのタイプを指定します
options	mount コマンドで使用されるオプションに似たマウントオプションを指定します

次の例では、大域ゾーンの /dev/dsk/c0t0d0s2 を、構成中のゾーンに /mnt としてマウントするように指定します。raw プロパティーでデバイスを指定する (オプション) と、ファイルシステムのマウントを実行する前に、そのデバイスに対して fsck コマンドが実行されます。使用するファイルシステムの種類は UFS です。nodevices オプションと logging オプションも追加します。

```
zonecfg:my-zone> add fs
zonecfg:my-zone:fs> set dir=/mnt
zonecfg:my-zone:fs> set special=/dev/dsk/c0t0d0s2
zonecfg:my-zone:fs> set raw=/dev/rdsk/c0t0d0s2
zonecfg:my-zone:fs> set type=ufs
zonecfg:my-zone:fs> add options [nodevices,logging]
zonecfg:my-zone:fs> end
```

詳細は、380 ページの「-o nosuid オプション」、383 ページの「セキュリティーの制限およびファイルシステムの動作」、および fsck(1m) と mount(1M) のマニュアルページを参照してください。また、セクション 1M のマニュアルページには、特定のファイルシステムに固有のマウントオプションに関するものがあります。このようなマニュアルページの名前は、mount_*filesystem* という形式です。

注 – fs リソースプロパティーを使用して ZFS ファイルシステムを追加する方法については、『Oracle Solaris ZFS 管理ガイド』の「ZFS ファイルシステムを非大域ゾーンに追加する」を参照してください。

---

dataset　　name

次の例では、データセット *sales* を非大域ゾーンでマウントして可視にし、大域ゾーンでは不可視にするように指定します。

```
zonecfg:my-zone> add dataset
zonecfg:my-zone> set name=tank/sales
zonecfg:my-zone> end
```

inherit-pkg-dir　　dir

次の例では、大域ゾーンから /opt/sfw をループバックマウントするように指定します。

```
zonecfg:my-zone> add inherit-pkg-dir
zonecfg:my-zone:inherit-pkg-dir> set dir=/opt/sfw
zonecfg:my-zone:inherit-pkg-dir> end
```

net　address、physical、defrouter

注 – 共有 IP ゾーンの場合は、IP アドレスとデバイスの両方を指定します。必要に応じて、デフォルトのルーターを設定できます。

- 大域ゾーンで構成されていないサブネット上に非大域ゾーンがある場合、defrouter プロパティーを使用してデフォルト経路を設定できます。
- defrouter プロパティーを設定したゾーンは、大域ゾーンで構成されていないサブネット上にある必要があります。

共有 IP ゾーンがそれぞれ異なるサブネット上にある場合は、大域ゾーンでデータリンクを構成しないでください。

排他的 IP ゾーンの場合は、物理インタフェースだけを指定します。physical プロパティーは VNIC でもかまいません。

次に示す共有 IP ゾーンの例では、IP アドレス 192.168.0.1 をゾーンに追加します。物理インタフェースとして hme0 カードを使用します。どの物理インタフェースを使用するかを決定するには、システムで ifconfig -a と入力します。出力の各行は、ループバックドライバの行を除き、システムにインストールされているカードの名前で始まります。説明に LOOPBACK が含まれている場合、その行はカードに関するものではありません。

```
zonecfg:my-zone> add net
zonecfg:my-zone:net> set physical=hme0
zonecfg:my-zone:net> set address=192.168.0.1
zonecfg:my-zone:net> end
```

次に示す排他的 IP ゾーンの例では、物理インタフェースとして bge32001 リンクを使用します。使用可能なデータリンクを調べるには、dladm show-link を使用してください。排他的 IP ゾーンで使用するデータリンクは GLDv3 でなければならず、GLDv3 以外のデータリンクは dladm show-link の出力に type: legacy と表示されます。ip-type=exclusive も指定する必要があります。

```
zonecfg:my-zone> set ip-type=exclusive
zonecfg:my-zone> add net
zonecfg:my-zone:net> set physical=bge32001
zonecfg:my-zone:net> end
```

device　match

次の例では、/dev/pts デバイスをゾーンに追加します。

```
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/pts*
zonecfg:my-zone:device> end
```

rctl　name、value

Solaris 10 8/07: このリリースで新しく追加されたリソース制御は、zone.max-locked-memory、zone.max-msg-ids、zone.max-sem-ids、zone.max-shm-ids、zone.max-shm-memory、および zone.max-swap です。

次のゾーン規模のリソース制御を使用できます:

- zone.cpu-shares (推奨: cpu-shares)
- zone.max-locked-memory
- zone.max-lwps (推奨: max-lwps)
- zone.max-msg-ids (推奨: max-msg-ids)
- zone.max-sem-ids (推奨: max-sem-ids)
- zone.max-shm-ids (推奨: max-shm-ids)
- zone.max-shm-memory (推奨: max-shm-memory)
- zone.max-swap

ゾーン規模のリソース制御を設定する場合に望ましい、より簡単な方法は、rctl リソースの代わりにプロパティー名を使用する方法です。詳細は、266 ページの「ゾーンの構成方法」を参照してください。add rctl を使ってゾーン内のゾーン規模のリソース制御エントリを構成する場合、その形式は project データベース内のリソース制御エントリの形式とは異なります。ゾーン構成では、rctl リソースタイプは、名前と値の対 3 つから成ります。これらの名前は、priv、limit、および action です。これらの名前には、単純な値がそれぞれ設定されます。

```
zonecfg:my-zone> add rctl
zonecfg:my-zone:rctl> set name=zone.cpu-shares
zonecfg:my-zone:rctl> add value (priv=privileged,limit=10,action=none)zonecfg:my-zone:rctl> end

zonecfg:my-zone> add rctl
zonecfg:my-zone:rctl> set name=zone.max-lwps
zonecfg:my-zone:rctl> add value (priv=privileged,limit=100,action=deny)
zonecfg:my-zone:rctl> end
```

リソース制御と属性の概要については、第 6 章「リソース制御 (概要)」および 395 ページの「非大域ゾーンで使用されるリソース制御」を参照してください。

`attr` `name`、`type`、`value`

次の例では、ゾーンに関するコメントを追加します。

```
zonecfg:my-zone> add attr
zonecfg:my-zone:attr> set name=comment
zonecfg:my-zone:attr> set type=string
zonecfg:my-zone:attr> set value="Production zone"
zonecfg:my-zone:attr> end
```

`export` サブコマンドを使用すると、ゾーン構成を標準出力に出力できます。構成は、コマンドファイルに使用できる形式で保存されます。

Tecla コマンド行編集ライブラリ

付属の Tecla コマンド行編集ライブラリは、`zonecfg` コマンドで使用できます。このライブラリにより、コマンド行の履歴メカニズムと編集サポートが提供されます。

Tecla コマンド行編集ライブラリについては、次のマニュアルページを参照してください。

- `enhance`(1)
- `libtecla`(3LIB)
- `ef_expand_file`(3TECLA)
- `gl_get_line`(3TECLA)
- `gl_io_mode`(3TECLA)
- `pca_lookup_file`(3TECLA)
- `tecla`(5)

◆ ◆ ◆ 第 18 章

18 非大域ゾーンの計画と構成 (タスク)

この章では、システムにゾーンを構成する前に実行する必要のある操作について説明します。また、ゾーンの構成方法、ゾーン構成の変更方法、およびシステムからゾーン構成を削除する方法についても説明します。

ゾーン構成処理の概要については、第 17 章「非大域ゾーンの構成 (概要)」を参照してください。

非大域ゾーンの計画と構成 (タスクマップ)

ゾーンを使用できるようにシステムを設定する前に、まず、情報を収集してゾーンの構成方法を決定する必要があります。次のタスクマップに、ゾーンの計画および構成方法の概要を示します。

タスク	説明	参照先
ゾーンの全体的な計画を立てます。	■ システムで稼働しているアプリケーションを評価し、ゾーン内で実行するアプリケーションを決定します。 ■ ゾーン内で固有のファイルを保持するディスク領域の可用性を評価します。 ■ リソース管理機能も使用している場合は、リソース管理境界に合わせてゾーンを配列する方法を決定します。	使用状況の履歴を参照してください。260 ページの「ディスク容量の要件」および 150 ページの「ゾーンで使用されるリソースプール」も参照してください。

タスク	説明	参照先
ゾーンの名前を決定します。	命名規則に基づいてゾーンの名前を決定します。	247 ページの「ゾーン構成データ」および 262 ページの「ゾーンのホスト名」を参照してください。
ゾーンパスを決定します。	各ゾーンには、ルートディレクトリのパスが設定されます。これは、大域ゾーンのルートディレクトリに対する相対パスです。	247 ページの「ゾーン構成データ」を参照してください。
リソースプールを構成しない場合は、CPU 制限の必要性を評価します。	アプリケーションの要件を確認します。	234 ページの「Solaris 10 8/07: `dedicated-cpu` リソース」を参照してください。
大域ゾーンから `rcapd` を使用してゾーンのメモリー上限を設定する場合は、メモリー割り当ての必要性を評価します。	アプリケーションの要件を確認します。	第 10 章「リソース上限デーモンによる物理メモリーの制御 (概要)」、第 11 章「リソース上限デーモンの管理 (タスク)」、および 236 ページの「Solaris 10 8/07: 物理メモリーの制御と `capped-memory` リソース」を参照してください。
FSS をシステムのデフォルトのスケジューラにします。	各ゾーンに CPU 配分を与えて、CPU リソースに対するゾーンの使用権を制御します。FSS では、割り当てられた配分に基づいて、プロジェクト間に CPU リソースが公平に配分されることが保証されています。	第 8 章「公平配分スケジューラ (概要)」および 235 ページの「ゾーンのスケジューリングクラス」を参照してください。

タスク	説明	参照先
ゾーンを共有 IP ゾーンにするか排他的 IP ゾーンにするかを決定します。	デフォルトは共有 IP ゾーンで、その場合はゾーンの IP アドレスを取得または構成します。構成に基づき、ネットワークアクセスを行う非大域ゾーンごとに 1 つ以上の IP アドレスを取得する必要があります。 排他的 IP ゾーンの場合は、ゾーンに割り当てるデータリンクを決定します。ゾーンには、1 つ以上のネットワークインタフェースへの排他的アクセスが必要です。インタフェースは、`bge1` などの個別の LAN や、`bge2000` などの個別の VLAN である可能性があります。データリンクは GLDv3 でなければなりません。GLDv3 でないデータリンクは、`dladm show-link` コマンドの出力に `type: legacy` と表示されます。	262 ページの「ゾーンホスト名の決定およびネットワークアドレスの取得」、266 ページの「ゾーンの構成方法」、および『Solaris のシステム管理 (IP サービス)』を参照してください。 GLDv3 インタフェースの詳細については、『Solaris のシステム管理 (IP サービス)』の「Oracle Solaris インタフェースタイプ」を参照してください。
ゾーン内にマウントするファイルシステムを決定します。	アプリケーションの要件を確認します。	詳細は、239 ページの「ゾーンでマウントされるファイルシステム」を参照してください。
ゾーンで使用可能にするべきネットワークインタフェースを決定します。	アプリケーションの要件を確認します。	詳細は、388 ページの「共有 IP ネットワークインタフェース」を参照してください。
非大域ゾーンのデフォルトの特権セットを変更する必要があるかどうかを決定します。	特権セットを確認します。 デフォルトの特権、追加および削除が可能な特権、および現時点では使用できない特権があります。	397 ページの「非大域ゾーン内の特権」を参照してください。
各ゾーンで構成するべきデバイスを決定します。	アプリケーションの要件を確認します。	使用するアプリケーションのドキュメントを参照してください。
ゾーンを構成します。	`zonecfg` を使用してゾーンの構成を作成します。	265 ページの「ゾーンを構成、検証、および確定する」を参照してください。

タスク	説明	参照先
構成したゾーンを検証および確定します。	指定されたリソースおよびプロパティーが仮想サーバー上で有効かどうかを判定します。	265 ページの「ゾーンを構成、検証、および確定する」を参照してください。

現在のシステム設定の評価

ゾーンは、Solaris 10 リリースが稼働する任意のマシンで実行できます。次に、ゾーンの使用に関連したマシンの主な考慮事項を示します。

- 各ゾーン内部で稼働するアプリケーションの性能要件。
- 各ゾーン内部で固有のファイルを保持するディスク容量がどれだけ利用可能か。

ディスク容量の要件

ゾーンが消費可能なディスク容量に関する制限はありません。ディスク容量の制限を設定することは、大域管理者の役割です。大域管理者は、非大域ゾーンのルートファイルシステムを保持するのに十分なローカルストレージがあることを確認する必要があります。小規模な単一プロセッサシステムでも、同時に稼働する多数のゾーンをサポートできます。

大域ゾーンにインストールされるパッケージの特性は、作成される非大域ゾーンの容量要件に影響を及ぼします。パッケージの数およびディスク容量要件が要因となります。

疎ルートゾーン

Solaris 10 リリースでは、`inherit-pkg-dir` リソースを持つ非大域ゾーンは疎ルートゾーンと呼ばれます。

疎ルートゾーンモデルでは、次の方法でオブジェクトの共有を最適化します。

- 大域ゾーンにインストールされたパッケージのサブセットだけが、非大域ゾーンに直接インストールされます。
- それ以外のファイルへのアクセスは、読み取り専用のループバックファイルシステムを介して行われ、これらは `inherit-pkg-dir` リソースと呼ばれます。

このモデルでは、すべてのパッケージが非大域ゾーンにインストールされたように表示されます。ループバックマウントされた読み取り専用のファイルシステムに内容を提供しないパッケージは、完全にインストールされます。ループバックマウントされた読み取り専用ファイルシステムに提供された内容は、大域ゾーンから継承される (および可視になる) ため、この内容をインストールする必要はありません。

- 一般的なガイドラインとして、標準的な Solaris パッケージすべてを大域ゾーンにインストールする場合、ゾーンごとに約 100M バイトの空きディスク容量が必要になります。
- デフォルトでは、大域ゾーンにインストールされたすべての追加パッケージは、非大域ゾーンにも入力されます。この追加パッケージが、`inherit-pkg-dir` リソーススペースに存在するファイルを提供するかどうかによって、必要なディスク容量が増加することもあります。

ゾーンごとに 40M バイトの 追加 RAM が推奨されていますが、十分なスワップ空間のあるマシンでは、これは必須ではありません。

完全ルートゾーン

完全ルートゾーンモデルは、最大限の構成可能性を提供します。Solaris の必須パッケージおよび選択されたオプションパッケージのすべてが、ゾーンの専用ファイルシステムにインストールされます。このモデルの利点として、大域管理者がゾーンファイルシステムのレイアウトをカスタマイズできるということがあります。これはたとえば、任意のアンバンドルパッケージや他社製パッケージを追加するために行います。

このモデルに必要なディスク容量は、大域ゾーンに現在インストールされているパッケージによって使用されるディスク容量によって決まります。

注 – 次の `inherit-pkg-dir` ディレクトリを含む疎ルートゾーンを作成した場合、このゾーンを完全ルートゾーンとしてインストールするには、まず、これらのディレクトリを非大域ゾーンの構成から削除する必要があります。

- `/lib`
- `/platform`
- `/sbin`
- `/usr`

266 ページの「ゾーンの構成方法」を参照してください。

ゾーンサイズを制限する

ゾーンサイズを制限する際、次のオプションを使用できます。

- `lofi` を使用してマウントされたパーティションにゾーンを配置できます。この操作により、ゾーンの消費する容量が、`lofi` の使用するファイルの容量に制限されます。詳細は、`lofiadm(1m)` および `lofi(7D)` のマニュアルページを参照してください。

- ソフトパーティションを使用して、ディスクスライスまたは論理ボリュームをパーティションに分割できます。これらのパーティションをゾーンのルートとして使用することで、ゾーンごとのディスク消費量を制限できます。ソフトパーティションの上限は、8192 パーティションに制限されています。詳細は、『Solaris ボリュームマネージャの管理』の第 12 章「ソフトパーティション (概要)」を参照してください。
- ディスクの標準パーティションをゾーンのルートに使用できるため、ゾーンごとのディスク消費を制限できます。

ゾーンホスト名の決定およびネットワークアドレスの取得

ゾーンのホスト名を決定する必要があります。その後、ネットワーク接続を確立する場合、ゾーンの IPv4 アドレスを割り当てるか、IPv6 アドレスの構成および割り当てを手動で行う必要があります。

ゾーンのホスト名

ゾーン用に選択したホスト名は、大域ゾーン内の `/etc/nsswitch.conf` ファイルでの指定に応じて、`hosts` データベースまたは `/etc/inet/hosts` データベース内で定義する必要があります。ネットワークデータベースは、ネットワークの構成情報を提供するファイルです。`nsswitch.conf` ファイルには、使用するネームサービスを指定します。

ネームサービス用にローカルファイルを使用する場合は、`/etc/inet/hosts` ファイル内で `hosts` データベースが保持されます。ゾーンネットワークインタフェースのホスト名の解決は、`/etc/inet/hosts` 内のローカル `hosts` データベースで行われます。あるいは、ゾーンの構成時に IP アドレス自体を直接指定することで、ホスト名の解決を不要にできます。

詳細は、『Solaris のシステム管理 (IP サービス)』の「TCP/IP 構成ファイル」および『Solaris のシステム管理 (IP サービス)』の「ネットワークデータベースと nsswitch.conf ファイル」を参照してください。

共有 IP ゾーンのネットワークアドレス

ネットワーク接続を必要とする各共有 IP ゾーンには、1 つ以上の一意の IP アドレスが与えられます。IPv4 と IPv6 の両方のアドレスがサポートされます。

IPv4 のゾーンネットワークアドレス

IPv4 を使用している場合、アドレスを取得してゾーンに割り当てます。

IP アドレスとともに接頭辞の長さも指定できます。この接頭辞の書式はアドレス/接頭辞の長さ です。たとえば、192.168.1.1/24 の場合、使用するアドレスは 192.168.1.1 で、使用するネットマスクは 255.255.255.0、または最初の 24 ビットがビット 1 であるマスクです。

IPv6 のゾーンネットワークアドレス

IPv6 を使用している場合、アドレスを手動で構成する必要があります。通常、次の 2 種類のアドレスを最小限構成する必要があります。

リンクローカルアドレス
: リンクローカルアドレスの書式は、fe80::*<64 ビットインタフェース ID>*/10 です。/10 は、接頭辞の長さが 10 ビットであることを示します。

サブネット上で構成された大域接頭辞から作成されたアドレス
: 大域ユニキャストアドレスは、管理者がサブネットごとに構成した 64 ビット接頭辞および 64 ビットのインタフェース ID に基づきます。接頭辞は、IPv6 を使用するよう構成されている同一サブネット上の任意のシステムで、ifconfig コマンドに -a6 オプションを指定して実行しても取得できます。

  通常、64 ビットのインタフェース ID は、システムの MAC アドレスから取得されます。次の方法で、ゾーン用の一意の代替アドレスを大域ゾーンの IPv4 アドレスから取得できます。

  <16 ビットのゼロ>:<IPv4 アドレスの上位 16 ビット>:<IPv4 アドレスの下位 16 ビット>:<ゾーンで一意の番号>

  たとえば、大域ゾーンの IPv4 アドレスが 192.168.200.10 である場合、ゾーン固有の番号 1 を使用する非大域ゾーン用の適正なリンクローカルアドレスは、fe80::c0a8:c80a:1/10 になります。そのサブネットで使用中の大域接頭辞が 2001:0db8:aabb:ccdd/64 である場合、同じ非大域ゾーン用の一意の大域ユニキャストアドレスは 2001:0db8:aabb:ccdd::c0a8:c80a:1/64 です。IPv6 アドレスを構成する際、接頭辞の長さを指定する必要があることに注意してください。

リンクローカルおよび大域ユニキャストアドレスの詳細については、inet6(7P) のマニュアルページを参照してください。

排他的 IP ゾーンのネットワークアドレス

排他的 IP ゾーンの内部で、大域ゾーンと同様の方法でアドレスを構成します。DHCP および IPv6 ステートレスアドレスの自動構成を使用してアドレスを構成することもできます。

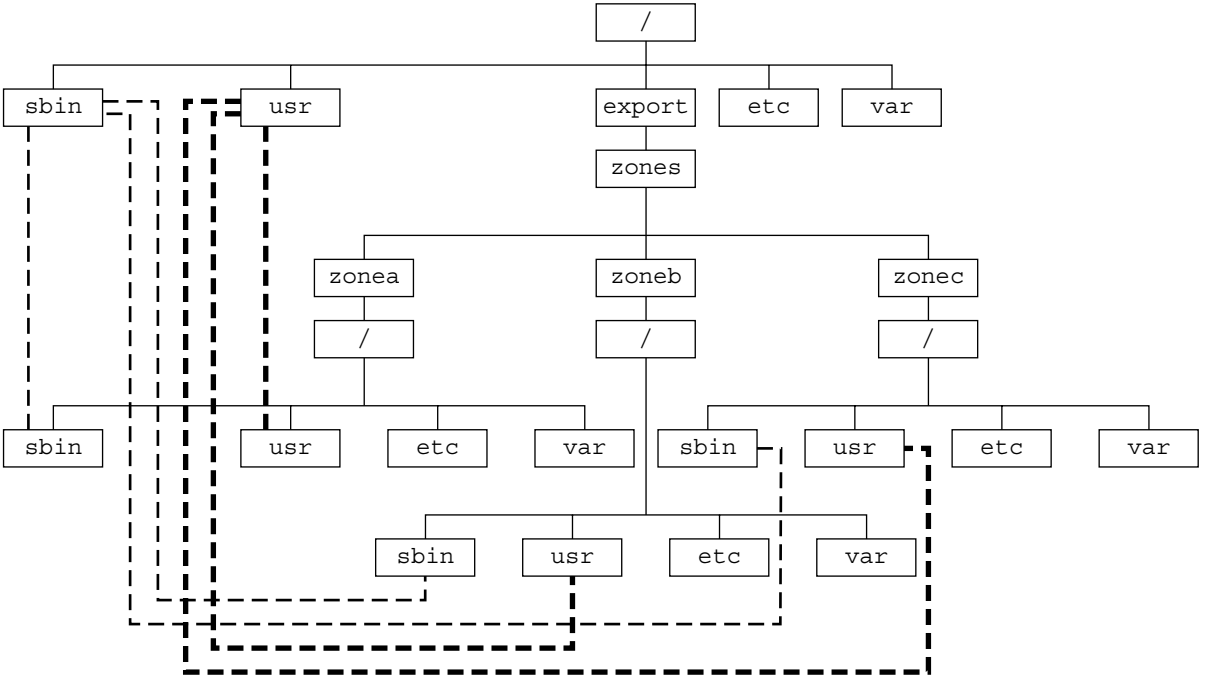
詳細は、sysidcfg(4) のマニュアルページを参照してください。

ファイルシステムの構成

仮想プラットフォームを設定する際、実行するマウントを多数指定できます。ループバック仮想ファイルシステム (LOFS) を使用してファイルシステムをゾーンにループバックマウントする場合、`nodevices` オプションを指定して仮想ファイルシステムをマウントする必要があります。`nodevices` オプションの詳細は、380 ページの「ファイルシステムと非大域ゾーン」を参照してください。

LOFS を使用すると、代替パス名を使用してファイルにアクセスできるように、新しい仮想ファイルシステムを作成できます。非大域ゾーンでは、ループバックマウントにより、ファイルシステム階層がゾーンのルート下に複製されているように見えます。ゾーン内では、ゾーンのルートから始まるパス名を使ってすべてのファイルにアクセスできるようになります。LOFS マウントでは、ファイルシステムの名前空間が維持されます。

図 18–1 ループバックマウントされたファイルシステム



詳細は、`lofs`(7S) のマニュアルページを参照してください。

非大域ゾーン構成の作成、改訂、および削除 (タスクマップ)

タスク	説明	参照先
非大域ゾーンを構成します。	zonecfg コマンドを使用してゾーンの作成、構成の検証および確定を行います。 スクリプトを使用して、システム上の複数のゾーンを構成およびブートすることもできます。zonecfg コマンドを使用して非大域ゾーンの構成を表示できます。	265 ページの「ゾーンを構成、検証、および確定する」、271 ページの「複数のゾーンを構成するスクリプト」
ゾーン構成を変更します。	ゾーン構成内のリソースタイプを変更するか、専用のデバイスをゾーンに追加する場合に、この手順を実行します。	274 ページの「zonecfg コマンドを使用してゾーン構成を変更する」
ゾーン構成を元に戻すか、ゾーン構成を削除します。	zonecfg コマンドを使用して、ゾーン構成に対して行なったリソース設定を取り消すか、ゾーン構成を削除します。	278 ページの「zonecfg コマンドを使用してゾーン構成を元に戻す、または削除する」
ゾーン構成を削除します。	zonecfg コマンドと delete サブコマンドを使用して、システムからゾーン構成を削除します。	279 ページの「ゾーン構成を削除する方法」

ゾーンを構成、検証、および確定する

次の処理を実行するには、zonecfg コマンド (zonecfg(1M) のマニュアルページを参照) を使用します。

- ゾーン構成を作成します
- 必要な情報がすべて存在することを確認します
- 非大域ゾーン構成を確定します

zonecfg コマンドは、大域ゾーンのリソース管理設定を持続的に指定する場合にも使用できます。

zonecfg ユーティリティーを使用してゾーンを構成する際、revert サブコマンドを使用して、リソースの設定を元に戻すことができます。278 ページの「ゾーン構成を元に戻す方法」を参照してください。

システムに複数のゾーンを構成するスクリプトについては、271 ページの「複数のゾーンを構成するスクリプト」を参照してください。

非大域ゾーンの構成を表示する方法については、273 ページの「非大域ゾーンの構成を表示する方法」を参照してください。

▼ ゾーンの構成方法

ネイティブな非大域ゾーンの作成に必須の要素は、`zonename` および `zonepath` プロパティーだけです。そのほかのリソースおよびプロパティーはオプションです。省略可能なリソースには、`dedicated-cpu` リソースと `capped-cpu` リソースのどちらを使用するかを決めるなど、選択肢の中から選ぶ必要があるものもあります。使用可能な `zonecfg` のプロパティーとリソースについては、247 ページの「ゾーン構成データ」を参照してください。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 選択したゾーン名を使用して、ゾーン構成を設定します。

   この手順例では、`my-zone` という名前を使用します。

   ```
   global# zonecfg -z my-zone
   ```

   このゾーンの初回構成時には、次のシステムメッセージが表示されます。

   ```
   my-zone: No such zone configured
   Use 'create' to begin configuring a new zone.
   ```

3. 新しいゾーン構成を作成します。

   この手順では、デフォルト設定を使用します。

   ```
   zonecfg:my-zone> create
   ```

4. ゾーンのパス (この手順では **/export/home/my-zone**) を設定します。

   ```
   zonecfg:my-zone> set zonepath=/export/home/my-zone
   ```

   Solaris 10 10/08 より前のリリースでは、`zonepath` を ZFS 上に設定しないようにしてください。

5 **autoboot** 値を設定します。

`true` に設定すると、大域ゾーンのブート時にこのゾーンが自動的にブートします。ゾーンを自動的にブートするには、ゾーンサービス `svc:/system/zones:default` も有効になっている必要があります。デフォルト値は `false` です。

```
zonecfg:my-zone> set autoboot=true
```

6 ゾーンの持続的なブート引数を設定します。

```
zonecfg:my-zone> set bootargs="-m verbose"
```

7 **1** つの **CPU** をこのゾーン専用に割り当てます。

```
zonecfg:my-zone> add dedicated-cpu
```

a. **CPU** の数を設定します。

```
zonecfg:my-zone:dedicated-cpu> set ncpus=1-2
```

b. **(**オプション**)** 重要性を設定します。

```
zonecfg:my-zone:dedicated-cpu> set importance=10
```

デフォルト値は **1** です。

c. 指定を終了します。

```
zonecfg:my-zone:dedicated-cpu> end
```

8 権限のデフォルトセットを修正します。

```
zonecfg:my-zone> set limitpriv="default,sys_time"
```

この行は、システムクロックを設定する機能をデフォルトの特権セットに追加します。

9 スケジューリングクラスを **FSS** に設定します。

```
zonecfg:my-zone> set scheduling-class=FSS
```

10 メモリー上限を追加します。

```
zonecfg:my-zone> add capped-memory
```

a. メモリー上限を設定します。

```
zonecfg:my-zone:capped-memory> set physical=50m
```

b. スワップメモリーの上限を設定します。

```
zonecfg:my-zone:capped-memory> set swap=100m
```

c. ロックされたメモリーの上限を設定します。

```
zonecfg:my-zone:capped-memory> set locked=30m
```

d. メモリー上限の指定を終了します。

```
zonecfg:my-zone:capped-memory> end
```

11 ファイルシステムを追加します。

```
zonecfg:my-zone> add fs
```

a. ファイルシステムのマウントポイント (この手順では **/usr/local**) を設定します。

```
zonecfg:my-zone:fs> set dir=/usr/local
```

b. 大域ゾーン内の **/opt/zones/my-zone/local** を、構成中のゾーン内で **/usr/local** としてマウントすることを指定します。

```
zonecfg:my-zone:fs> set special=/opt/zones/my-zone/local
```

非大域ゾーン内では、/usr/local ファイルシステムは読み取りおよび書き込みが可能です。

c. ファイルシステムのタイプ (この手順では **lofs**) を指定します。

```
zonecfg:my-zone:fs> set type=lofs
```

このタイプは、カーネルとそのファイルシステムとの相互動作の方法を示します。

d. ファイルシステムの指定を終了します。

```
zonecfg:my-zone:fs> end
```

この手順を複数回実行することで、複数のファイルシステムを追加できます。

12 (オプション) **hostid** を設定します。

```
zonecfg:my-zone> set hostid=80f0c086
```

13 ストレージプール *tank* の *sales* という **ZFS** データセットを追加します。

```
zonecfg:my-zone> add dataset
```

a. **ZFS** データセット *sales* のパスを指定します。

```
zonecfg:my-zone> set name=tank/sales
```

b. データセットの指定を終了します。

```
zonecfg:my-zone> end
```

14 (疎ルートゾーンのみ)大域ゾーンからループバックマウントされた共有ファイルシステムを追加します。

共有ファイルシステムを持たない完全ルートゾーンを作成する場合は、この手順を実行しないでください。260 ページの「ディスク容量の要件」に記述されている完全ルートゾーンの説明を参照してください。

```
zonecfg:my-zone> add inherit-pkg-dir
```

a. 大域ゾーン内の **/opt/sfw** を、構成中のゾーン内で読み取り専用モードでマウントすることを指定します。

```
zonecfg:my-zone:inherit-pkg-dir> set dir=/opt/sfw
```

注 – ゾーンのパッケージデータベースが更新され、パッケージが反映されます。zoneadm を使用してゾーンをインストールした後で、これらのリソースを変更または削除することはできません。

---

b. **inherit-pkg-dir** の指定を終了します。

```
zonecfg:my-zone:inherit-pkg-dir> end
```

この手順を複数回実行することで、複数の共有ファイルシステムを追加できます。

注 – 完全ルートゾーンを作成する場合で、inherit-pkg-dir を使用してデフォルトの共有ファイルシステムリソースがすでに追加されているときは、ゾーンのインストール前に、次のように zonecfg を使用して、これらのデフォルトの inherit-pkg-dir リソースを削除する必要があります。

- zonecfg:my-zone> **remove inherit-pkg-dir dir=/lib**
- zonecfg:my-zone> **remove inherit-pkg-dir dir=/platform**
- zonecfg:my-zone> **remove inherit-pkg-dir dir=/sbin**
- zonecfg:my-zone> **remove inherit-pkg-dir dir=/usr**

---

15 (オプション)排他的 IP ゾーンを作成する場合は、**ip-type** を設定します。

```
zonecfg:my-zone> set ip-type=exclusive
```

注 – add net 手順では、物理デバイスタイプだけを指定します。

---

16 ネットワークインタフェースを追加します。

```
zonecfg:my-zone> add net
```

a. (共有 IP のみ)ネットワークインタフェースの IP アドレス (この手順では **192.168.0.1**) を指定します。

```
zonecfg:my-zone:net> set address=192.168.0.1
```

b. ネットワークインタフェースの物理デバイスタイプ (この手順では **hme** デバイス) を指定します。

```
zonecfg:my-zone:net> set physical=hme0
```

c. **Solaris 10 10/08:** (オプション、共有 IP のみ) ネットワークインタフェースのデフォルトのルーター (この手順では **10.0.0.1**) を設定します。

```
zonecfg:my-zone:net> set defrouter=10.0.0.1
```

d. 指定を終了します。

```
zonecfg:my-zone:net> end
```

この手順を複数回実行することで、複数のネットワークインタフェースを追加できます。

17 デバイスを追加します。

```
zonecfg:my-zone> add device
```

a. デバイスの一致 (この手順では **/dev/sound/***) を設定します。

```
zonecfg:my-zone:device> set match=/dev/sound/*
```

b. デバイスの指定を終了します。

```
zonecfg:my-zone:device> end
```

この手順を複数回実行することで、複数のデバイスを追加できます。

18 プロパティー名を使用して、ゾーン規模のリソース制御を追加します。

```
zonecfg:my-zone> set max-sem-ids=10485200
```

この手順を複数回実行することで、複数のリソース制御を追加できます。

19 リソースタイプ **attr** を使用してコメントを追加します。

```
zonecfg:my-zone> add attr
```

a. 名前を **comment** に設定します。

```
zonecfg:my-zone:attr> set name=comment
```

b. タイプを **string** に設定します。

```
zonecfg:my-zone:attr> set type=string
```

c. 値をゾーンの内容を示すコメントに設定します。

```
zonecfg:my-zone:attr> set value="This is my work zone."
```

d. リソースタイプ **attr** の指定を終了します。

```
zonecfg:my-zone:attr> end
```

20 ゾーンの構成を検証します。

```
zonecfg:my-zone> verify
```

21 ゾーンの構成を確定します。

```
zonecfg:my-zone> commit
```

22 **zonecfg** コマンドを終了します。

```
zonecfg:my-zone> exit
```

プロンプトで `commit` コマンドを明示的に入力しなくても、`exit` を入力するか EOF が発生すると、`commit` の実行が自動的に試みられます。

参考 コマンド行での複数のサブコマンドの使用

ヒント – `zonecfg` コマンドは、複数のサブコマンドもサポートします。次に示すように、同じシェル呼び出しで引用符で囲み、セミコロンで区切ります。

```
global# zonecfg -z my-zone "create ; set zonepath=/export/home/my-zone"
```

次に進む手順

確定済みのゾーン構成をインストールする方法については、292 ページの「ゾーンのインストールとブート」を参照してください。

複数のゾーンを構成するスクリプト

このスクリプトを使用して、システムで複数のゾーンを構成およびブートできます。スクリプトには、次のパラメータを指定します。

- 作成するゾーンの数
- 接頭辞 *zonename*
- 基本ディレクトリとして使用するディレクトリ

このスクリプトを実行するには、大域ゾーン内の大域管理者になる必要があります。大域管理者は、大域ゾーン内でスーパーユーザー権限を保持するか、Primary Administrator 役割になります。

```
#!/bin/ksh
#
# Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident    "%Z%%M%    %I%    %E% SMI"
```

```
if [[ -z "$1" || -z "$2" || -z "$3" ]]; then
        echo "usage: $0 <#-of-zones> <zonename-prefix> <basedir>"
        exit 2
fi

if [[ ! -d $3 ]]; then
        echo "$3 is not a directory"
        exit 1
fi

nprocs=‘psrinfo | wc -l‘
nzones=$1
prefix=$2
dir=$3

ip_addrs_per_if=‘ndd /dev/ip ip_addrs_per_if‘
if [ $ip_addrs_per_if -lt $nzones ]; then
        echo "ndd parameter ip_addrs_per_if is too low ($ip_addrs_per_if)"
        echo "set it higher with ’ndd -set /dev/ip ip_addrs_per_if <num>"
        exit 1
fi

i=1
while [ $i -le $nzones ]; do
    zoneadm -z $prefix$i list > /dev/null 2>&1
    if [ $? != 0 ]; then
        echo configuring $prefix$i
        F=$dir/$prefix$i.config
        rm -f $F
        echo "create" > $F
        echo "set zonepath=$dir/$prefix$i" >> $F
        zonecfg -z $prefix$i -f $dir/$prefix$i.config 2>&1 | \
            sed ’s/^/    /g’
    else
        echo "skipping $prefix$i, already configured"
    fi
    i=‘expr $i + 1‘
done

i=1
while [ $i -le $nzones ]; do
    j=1
    while [ $j -le $nprocs ]; do
        if [ $i -le $nzones ]; then
            if [ ‘zoneadm -z $prefix$i list -p | \
                cut -d’:’ -f 3‘ != "configured" ]; then
                echo "skipping $prefix$i, already installed"
            else
                echo installing $prefix$i
                mkdir -pm 0700 $dir/$prefix$i
                chmod 700 $dir/$prefix$i
                zoneadm -z $prefix$i install > /dev/null 2>&1 &
                sleep 1    # spread things out just a tad
            fi
        fi
        i=‘expr $i + 1‘
        j=‘expr $j + 1‘
    done
    wait
```

```
done

i=1
while [ $i -le $nzones ]; do
    echo setting up sysid for $prefix$i
    cfg=$dir/$prefix$i/root/etc/sysidcfg
    rm -f $cfg
    echo "network_interface=NONE {hostname=$prefix$i}" > $cfg
    echo "system_locale=C" >> $cfg
    echo "terminal=xterms" >> $cfg
    echo "security_policy=NONE" >> $cfg
    echo "name_service=NONE" >> $cfg
    echo "timezone=US/Pacific" >> $cfg
    echo "root_password=Qexr7Y/wzkSbc" >> $cfg  # 'l1a'
    i=‘expr $i + 1‘
done

i=1
para=‘expr $nprocs \* 2‘
while [ $i -le $nzones ]; do
    date
    j=1
    while [ $j -le $para ]; do
        if [ $i -le $nzones ]; then
            echo booting $prefix$i
            zoneadm -z $prefix$i boot &
        fi
        j=‘expr $j + 1‘
        i=‘expr $i + 1‘
    done
    wait
done
```

▼ 非大域ゾーンの構成を表示する方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 ゾーンの構成を表示します。

```
global# zonecfg -z zonename info
```

zonecfg コマンドを使用してゾーン構成を変更する

zonecfg コマンドを使用して、次の操作を実行することもできます。

- ゾーン構成内のリソースタイプを変更します
- ゾーン構成内のプロパティーの値をクリアーします
- 複製したデバイスをゾーンに追加します

▼ ゾーン構成内のリソースタイプを変更する方法

リソースタイプを選択して、そのリソースの仕様を変更できます。

zoneadm を使ってゾーンをインストールしたあとで inherit-pkg-dir ディレクトリ内のソフトウェアパッケージの内容を変更したり削除したりすることはできない点に注意してください。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。
役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 変更するゾーン (この手順では **my-zone**) を選択します。

```
global# zonecfg -z my-zone
```

3 変更するリソースタイプ (リソース制御など) を選択します。

```
zonecfg:my-zone> select rctl name=zone.cpu-shares
```

4 現在の値を削除します。

```
zonecfg:my-zone:rctl> remove value (priv=privileged,limit=20,action=none)
```

5 新しい値を追加します。

```
zonecfg:my-zone:rctl> add value (priv=privileged,limit=10,action=none)
```

6 改定された **rctl** の指定を終了します。

```
zonecfg:my-zone:rctl> end
```

7 ゾーンの構成を確定します。

```
zonecfg:my-zone> commit
```

8 **zonecfg** コマンドを終了します。

```
zonecfg:my-zone> exit
```

プロンプトで commit コマンドを明示的に入力しなくても、exit を入力するか EOF が発生すると、commit の実行が自動的に試みられます。

zonecfg で行なった確定済みの変更は、ゾーンの次回ブート時に適用されます。

▼ Solaris 10 8/07: ゾーン構成内のプロパティータイプをクリアーする方法

スタンドアロンのプロパティーをリセットするには、この手順を使用します。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 変更するゾーン (この手順では **my-zone**) を選択します。

   ```
   global# zonecfg -z my-zone
   ```

3. 変更対象のプロパティー (この手順では既存のプールの関連付け) をクリアーします。

   ```
   zonecfg:my-zone> clear pool
   ```

4. ゾーンの構成を確定します。

   ```
   zonecfg:my-zone> commit
   ```

5. **zonecfg** コマンドを終了します。

   ```
   zonecfg:my-zone> exit
   ```

   プロンプトで commit コマンドを明示的に入力しなくても、exit を入力するか EOF が発生すると、commit の実行が自動的に試みられます。

   zonecfg で行なった確定済みの変更は、ゾーンの次回ブート時に適用されます。

▼ Solaris 10 3/05 から Solaris 10 11/06: ゾーン構成内のプロパティータイプを変更する方法

構成する関連プロパティーを持たないスタンドアロンのプロパティーをリセットするには、この手順を使用します。たとえば、既存のプールの関連付けを削除するには、pool リソースを null にリセットします。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 変更するゾーン (この手順では **my-zone**) を選択します。

   ```
   global# zonecfg -z my-zone
   ```

3. 変更対象のプロパティー (この手順では既存のプールの関連付け) をリセットします。

   ```
   zonecfg:my-zone> set pool=""
   ```

4. ゾーンの構成を確定します。

   ```
   zonecfg:my-zone> commit
   ```

5. **zonecfg** コマンドを終了します。

   ```
   zonecfg:my-zone> exit
   ```

   プロンプトで `commit` コマンドを明示的に入力しなくても、`exit` を入力するか EOF が発生すると、`commit` の実行が自動的に試みられます。

   `zonecfg` で行なった確定済みの変更は、ゾーンの次回ブート時に適用されます。

▼ Solaris 10 8/07: ゾーンの名前を変更する方法

この手順を使用すると、構成済み状態またはインストール済み状態にあるゾーンの名前を変更できます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 名前を変更するゾーン (この手順では **my-zone**) を選択します。

   ```
   global# zonecfg -z my-zone
   ```

3. ゾーンの名前を **newzone** などに変更します。

   ```
   zonecfg:my-zone> set zonename=newzone
   ```

4. 変更を確定します。

   ```
   zonecfg:newzone> commit
   ```

5 **zonecfg** コマンドを終了します。

```
zonecfg:newzone> exit
```

zonecfg で行なった確定済みの変更は、ゾーンの次回ブート時に適用されます。

▼ 専用のデバイスをゾーンに追加する方法

次に、非大域ゾーン構成内に走査デバイスを配置する手順を示します。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 デバイスを追加します。

```
zonecfg:my-zone> add device
```

3 デバイスの一致 (この手順では **/dev/scsi/scanner/c3t4***) を設定します。

```
zonecfg:my-zone:device> set match=/dev/scsi/scanner/c3t4*
```

4 デバイスの指定を終了します。

```
zonecfg:my-zone:device> end
```

5 **zonecfg** コマンドを終了します。

```
zonecfg:my-zone> exit
```

▼ 大域ゾーンの zone.cpu-shares を設定する方法

大域ゾーンの配分を持続的に設定する場合に、ここで説明する手順を使用します。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **zonecfg** コマンドを使用します。

```
# zonecfg -z global
```

3 大域ゾーンの配分を 5 に設定します。

```
zonecfg:global> set cpu-shares=5
```

4 zonecfg を終了します。

```
zonecfg:global> exit
```

zonecfg コマンドを使用してゾーン構成を元に戻す、または削除する

zonecfg(1M) のマニュアルページの記述に従って、zonecfg コマンドを使用し、ゾーンの構成を元に戻すか、またはゾーン構成を削除します。

▼ ゾーン構成を元に戻す方法

zonecfg ユーティリティーによるゾーンの構成中にゾーン構成に対して行なったリソース設定を取り消すには、revert サブコマンドを使用します。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは Primary Administrator 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 tmp-zone という名前のゾーンを構成中に、info と入力して構成を表示します。

```
zonecfg:tmp-zone> info
```

構成の net リソースセグメントが、次のように表示されます。

```
.
.
.
fs:
        dir: /tmp
        special: swap
        type: tmpfs
net:
        address: 192.168.0.1
        physical: eri0
device
        match: /dev/pts/*
.
.
.
```

3 ネットアドレスを削除します。

```
zonecfg:tmp-zone> remove net address=192.168.0.1
```

4 **net** エントリが削除されたことを確認します。

```
zonecfg:tmp-zone> info

.
.
.
fs:
        dir: /tmp
        special: swap
        type: tmpfs
device
        match: /dev/pts/*
.
.
.
```

5 **revert** と入力します。

```
zonecfg:tmp-zone> revert
```

6 次の質問に **yes** で応答します。

```
Are you sure you want to revert (y/[n])? y
```

7 ネットアドレスが再び存在することを確認します。

```
zonecfg:tmp-zone> info

.
.
.
fs:
        dir: /tmp
        special: swap
        type: tmpfs
net:
        address: 192.168.0.1
        physical: eri0
device
        match: /dev/pts/*
.
.
.
```

▼ ゾーン構成を削除する方法

zonecfg と delete サブコマンドを使用して、システムからゾーン構成を削除します。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 次の **2** つの方法のいずれかを使用して、ゾーン **a-zone** のゾーン構成を削除します。

- `-F` オプションを使用して操作を強制実行します。

```
global# zonecfg -z a-zone delete -F
```

- システムプロンプトに yes で応答し、対話的にゾーンを削除します。

```
global# zonecfg -z a-zone delete
Are you sure you want to delete zone a-zone (y/[n])? y
```

◆ ◆ ◆ 第 19 章

19 非大域ゾーンのインストール、停止、複製、およびアンインストールについて (概要)

この章では、Solaris システムへのゾーンのインストールについて説明します。また、仮想プラットフォームおよびアプリケーション環境を管理する 2 つのプロセス zoneadmd および zsched についても説明します。ゾーンの停止、リブート、クローニング、およびアンインストールに関する情報も提供します。

この章の内容は次のとおりです。

非大域ゾーンの複製、非大域ゾーンのインストールとブート、および非大域ゾーンの停止やアンインストールの手順については、第 20 章「非大域ゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)」を参照してください。

lx ブランドゾーンのインストールについては、第 34 章「lx ブランドゾーンのインストール、ブート、停止、複製、およびアンインストールについて (概要)」および第 35 章「lx ブランドゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)」を参照してください。

この章に追加されている説明

Solaris 10 11/06: 非大域ゾーンを複製する機能が使用可能になりました。302 ページの「Solaris 10 11/06: 同一システム上での非大域ゾーンの複製」を参照してください。

Solaris 10 8/07: ブート引数に関する情報も追加されています。286 ページの「Solaris 10 8/07: ゾーンのブート引数」を参照してください。

Solaris 10 5/09: ZFS クローンが実装されています。複製元の `zonepath` と複製先の `zonepath` が両方とも ZFS 上にあり、同じプールに含まれる場合、`zoneadm clone`コマンドは自動的に ZFS を使用してゾーンを複製します。どちらの `zonepath` も ZFS でない場合、または一方が ZFS で他方が ZFS でない場合、コードは既存のコピー手法を使用します。

ゾーンのインストールと管理の概要

`zoneadm` コマンド (`zoneadm(1M)` のマニュアルページを参照) は、非大域ゾーンをインストールおよび管理するための主要なツールです。`zoneadm` コマンドを使用する操作は、大域ゾーンから実行する必要があります。`zoneadm` コマンドを使用すると、次のタスクを実行できます。

- ゾーンを検証します
- ゾーンをインストールします
- ゾーンをブートします。これは、通常の Solaris システムのブートに似ています。
- 稼働中のゾーンに関する情報を表示します
- ゾーンを停止します
- ゾーンをリブートします
- ゾーンをアンインストールします
- 同じシステム上で、ゾーンを別の場所へ再配置します
- 同一システムの既存ゾーンの構成に基づいて、新しいゾーンをプロビジョニングします
- ゾーンを移行します。`zonecfg` コマンドとともに使用します

ゾーンのインストールおよび検証手順については、第 20 章「非大域ゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)」および `zoneadm(1M)` のマニュアルページを参照してください。`zoneadm list` コマンドでサポートされるオプションについては、`zoneadm`(1M) のマニュアルページも参照してください。ゾーンの構成手順については、第 18 章「非大域ゾーンの計画と構成 (タスク)」および `zonecfg(1M)` のマニュアルページを参照してください。ゾーンの状態については、222 ページの「非大域ゾーンの状態モデル」に記載されています。

ゾーンの Solaris 監査レコードの生成を計画している場合は、非大域ゾーンをインストールする前に402 ページの「ゾーン内での Oracle Solaris 監査の使用」を読んでください。

ゾーンの構築

このセクションの内容は、既存のゾーンの複製にではなく、初期のゾーン構築に適用されます。

非大域ゾーンを構成したあとで、システムの構成にゾーンを安全にインストールできることを確認してください。その後、ゾーンをインストールできます。ゾーンのルートファイルシステムに必要とされるファイルは、システムによりゾーンのルートパス内にインストールされます。

非大域ゾーンは、オープンネットワーク構成 (`generic_open.xml`) を使ってインストールされます。ネットワーク構成の種類については、『Solaris のシステム管理 (基本編)』の第 19 章「サービスの管理 (手順)」を参照してください。ゾーン管理者は、`netservices` コマンドを使って、ゾーンを制限されたネットワーク構成 (`generic_limited_net.xml`) に切り替えることができます。SMF コマンドを使って、特定のサービスを有効または無効にできます。

ゾーンのインストールが成功したら、初回のログインおよびブートを実行できます。

パッケージを Solaris インストール内に最初にインストールした時に使用した方法が、非大域ゾーンの生成にも使用されます。

大域ゾーンには、非大域ゾーンの生成に必要なデータがすべて存在する必要があります。ゾーンの生成には、ディレクトリの作成、ファイルのコピー、および構成情報の指定が含まれます。

パッケージから大域ゾーン内に作成されたデータまたは情報だけが、大域ゾーンからゾーンを生成するのに使用されます。詳細は、`pkgparam(1)` および `pkginfo(4)` のマニュアルページを参照してください。

ゾーンのインストール時に、次の場所からデータが参照またはコピーされます。

- インストールされていないパッケージ
- パッチ
- CD および DVD 内のデータ
- ネットワークインストールイメージ
- ゾーンの任意のプロトタイプまたはほかのインスタンス

また、次のタイプの情報が大域ゾーンに存在する場合、これらの情報はインストール中のゾーンにはコピーされません。

- `/etc/passwd` ファイル内の新規または変更されたユーザー
- `/etc/group` ファイル内の新規または変更されたグループ
- DHCP アドレスの割り当て、UUCP、sendmail などのネットワークサービスの構成
- ネームサービスなどのネットワークサービスの構成

- 新規または変更された crontab、プリンタ、およびメールファイル
- システムログ、メッセージ、およびアカウンティングファイル

Solaris 監査を使用する場合、大域ゾーンからコピーされた監査ファイルの変更が必要な場合があります。詳細は、402 ページの「ゾーン内での Oracle Solaris 監査の使用」を参照してください。

非大域ゾーン内では、次の機能を構成することはできません。

- Solaris Live Upgrade のブート環境
- Solaris ボリュームマネージャーのメタデバイス
- 共有 IP ゾーンでの DHCP アドレスの割り当て
- SSL プロキシサーバー

ゾーンの状態がインストール済みから準備完了に移行する際、構成ファイルで指定されたリソースセットが追加されます。システムにより、一意のゾーン ID が割り当てられます。ファイルシステムがマウントされ、ネットワークインタフェースが設定され、デバイスが構成されます。準備完了状態に移行すると、仮想プラットフォームでユーザープロセスを開始する準備が整います。準備完了状態では、仮想プラットフォームを管理するため、zsched および zoneadmd プロセスが開始されます。

- sched に類似したシステムスケジューリングプロセスである zsched が、ゾーンに関連付けられたカーネルリソースの追跡に使用されます。
- zoneadmd は、ゾーン管理デーモンです。

準備完了状態のゾーンには、稼働中のユーザープロセスは存在しません。準備完了状態のゾーンと稼働中のゾーンの主な違いは、稼働中のゾーンでは 1 つ以上のプロセスが稼働している点です。詳細は、init(1M) のマニュアルページを参照してください。

zoneadmd デーモン

ゾーン管理デーモン zoneadmd は、ゾーンの仮想プラットフォーム管理用の主要なプロセスです。このデーモンは、ゾーンのブートおよび停止処理の管理も担当します。システム上のアクティブな (準備完了、稼働中、または停止処理中の) ゾーンごとに、1 つの zoneadmd プロセスが存在します。

zoneadmd デーモンは、ゾーン構成での指定に従ってゾーンを設定します。このプロセスには、次の処理が含まれます。

- ゾーン ID を割り当てて、zsched システムプロセスを開始する。
- ゾーン規模のリソース制御を設定する。
- ゾーン構成の指定に従ってゾーンのデバイスを準備する。詳細は、devfsadmd(1M) のマニュアルページを参照してください。

- 仮想ネットワークインタフェースを設定する。
- ループバックおよび従来のファイルシステムをマウントする。
- ゾーンコンソールデバイスをインスタンス化および初期化する。

zoneadmd デーモンが実行中でない場合、zoneadm によりこのデーモンが自動的に起動されます。このため、何らかの理由でこのデーモンが動作していない場合、zoneadm を呼び出してそのゾーンを管理すると zoneadmd を再起動します。

zoneadmd デーモンのマニュアルページは、zoneadmd(1M) です。

zsched ゾーンスケジューラ

アクティブなゾーンとは、準備完了状態、稼働状態、または停止処理状態のゾーンを指します。すべてのアクティブなゾーンには、カーネルプロセス zsched が関連付けられています。ゾーンのために処理を実行するカーネルスレッドは、zsched により所有されています。zsched プロセスにより、ゾーンサブシステムがカーネルスレッドをゾーンごとに追跡することが可能になります。

ゾーンアプリケーション環境

ゾーンアプリケーション環境の作成には、zoneadm コマンドが使用されます。

非大域ゾーンを最初にブートする前に、ゾーンの内部構成を作成する必要があります。内部構成では、使用するネームサービス、デフォルトのロケールおよびタイムゾーン、ゾーンのルートパスワード、およびほかのアプリケーション環境特性を指定します。アプリケーション環境は、ゾーンコンソールに表示される一連のプロンプトへの応答で確立されます。詳細は、308 ページの「ゾーンの内部構成」を参照してください。ゾーンのデフォルトロケールおよびタイムゾーンは、大域設定には関係なく構成できます。

ゾーンの停止、リブート、およびアンインストールについて

このセクションでは、ゾーンの停止、リブート、およびアンインストール手順の概要について説明します。要求があった際にゾーンの停止に失敗した場合のトラブルシューティングに関するヒントも提供します。

ゾーンを停止する

ゾーンのアプリケーション環境および仮想プラットフォームの両方を削除する場合に、zoneadm halt コマンドを使用します。これにより、ゾーンはインストール済みの

状態に戻されます。すべてのプロセスが終了し、デバイスが構成解除され、ネットワークインタフェースが破棄され、ファイルシステムのマウントが解除され、カーネルデータ構造が破棄されます。

`halt` コマンドにより、ゾーン内部の停止処理スクリプトが実行されることはありません。ゾーンの停止処理を行う方法については、319 ページの「`zlogin` を使用してゾーンを停止処理する方法」を参照してください。

停止操作に失敗する場合は、443 ページの「ゾーンが停止しない」を参照してください。

ゾーンをリブートする

`zoneadm reboot` コマンドを使用してゾーンをリブートします。ゾーンは停止し、その後再ブートします。ゾーンのリブート時に、ゾーン ID が変更されます。

Solaris 10 8/07: ゾーンのブート引数

ゾーンでは、次のブート引数を `zoneadm boot` および `reboot` コマンドに使用できます。

- `-i` *altinit*
- `-m` *smf_options*
- `-s`

次の定義が適用されます。

`-i` *altinit*
: 最初のプロセスとなる代替実行可能ファイルを選択します。*altinit* は実行可能ファイルへの有効なパスでなければなりません。デフォルトの最初のプロセスについては、`init(1M)` のマニュアルページを参照してください。

`-m` *smf_options*
: SMF のブート動作を制御します。復元オプションとメッセージオプションという、2 種類のオプションがあります。メッセージオプションは、ブート中に表示されるメッセージの種類と数を決定します。サービスオプションは、システムのブートに使用されるサービスを決定します。

  復元オプションは次のとおりです。

  debug
  : 標準のサービス別出力と、ログに記録されるすべての `svc.startd` メッセージを出力します。

  `milestone=`*milestone*
  : 指定されたマイルストーンで定義されているサブグラフにブートします。有効なマイルストーンは、`none`、 `single-user`、

multi-user、 multi-user-server、 および all です。

メッセージオプションは次のとおりです。

quiet　標準のサービス別出力と、管理者の介入を必要とするエラーメッセージを出力します。

verbose　標準のサービス別出力と、詳細情報を提供するメッセージを出力します。

-s　マイルストーン svc:/milestone/single-user:default に対してのみブートします。このマイルストーンは、init のレベル s と同等です。

使用例については、296 ページの「ゾーンのブート方法」および 298 ページの「ゾーンをシングルユーザーモードでブートする方法」を参照してください。

Solaris サービス管理機能 (SMF) と init については、『Solaris のシステム管理 (基本編)』の第 18 章「サービスの管理 (概要)」、および svc.startd(1M) と init(1M) のマニュアルページを参照してください。

ゾーンの autoboot

ゾーンの構成内で autoboot リソースプロパティーを true に設定すると、大域ゾーンのブート時にそのゾーンが自動的にブートします。デフォルトの設定は false です。

ゾーンを自動的にブートするには、ゾーンサービス svc:/system/zones:default も有効になっている必要があります。

ゾーンのアンインストール

ゾーンのルートファイルシステム内のすべてのファイルをアンインストールする場合に、zoneadm uninstall コマンドを使用します。-F (force) オプションを合わせて指定しない限り、処理を続行する前に、コマンドプロンプトにより実行の確認が求められます。実行した操作を元に戻すことはできないため、uninstall コマンドは慎重に使用してください。

Solaris 10 11/06 以降: 非大域ゾーンの複製について

クローンを使用すると、システムの既存の構成済みおよびインストール済みゾーンをコピーして、新しいゾーンを同一のシステム上に迅速にプロビジョニングできます。少なくとも、複数のゾーンで同一であってはならないコンポーネントに対しては、プロパティーとリソースをリセットする必要があります。したがって、zonepath は常に変更する必要があります。さらに、共有 IP ゾーンの場合は、各 net リソースの IP アドレスが異なっている必要があります。排他的 IP ゾーンの場合は、各 net リソースの physical プロパティーが異なっている必要があります。

- ゾーンのクローニングは、ゾーンのインストールほど時間がかかりません。
- 新規ゾーンには、パッケージの追加やファイルの変更など、ソースゾーンをカスタマイズする過程で加えられた変更がすべて含まれます。

Solaris 10 5/09: 複製元の zonepath と複製先の zonepath が両方とも ZFS 上にあり、同じプールに含まれる場合、zoneadm clone コマンドは自動的に ZFS を使用してゾーンを複製します。ZFS クローンを使用する場合、データが変更されるまでデータは実際にはコピーされません。したがって、最初のクローンにかかる時間はごくわずかです。zoneadm コマンドは、ソース zonepath の ZFS スナップショットを取得して、ターゲット zonepath を設定します。スナップショットには SUNWzone*X* という形式の名前が付けられます。この *X* は、複数のスナップショットを区別するために使用される一意の ID です。ZFS クローンの名前には、宛先ゾーンの zonepath が使用されます。スナップショットがあとで使用されるときにその妥当性をシステムで検証できるように、ソフトウェアインベントリが実行されます。ソースゾーンを何度も複製できるように zoneadm コマンドでは、既存のスナップショットが使用されるように指定できます。既存のスナップショットがターゲットで使用できるかどうかは、システムによって自動的に検証されます。

『Oracle Solaris ZFS 管理ガイド』の「ZFS スナップショットを作成および破棄する」で説明されているような手作業によるスナップショットは使用できません。このようなスナップショットには、妥当性検査を実行するためのデータが欠如しています。

ソースゾーンを何度も複製する場合で、複製ごとに新しいスナップショットを作成したくないことがあります。clone サブコマンドの -s パラメータを使用すると、以前の複製で作成した既存のスナップショットを使用するように指定できます。304 ページの「Solaris 10 5/09: 既存のスナップショットからゾーンを複製する方法」を参照してください。

スナップショットの内容は過去のある時点でのゾーンを表すため、スナップショットの作成以降に、パッチの適用やアップグレードなど何らかの方法でシステムが更新されている可能性があります。ゾーンがアップグレードされていると、スナップショットを現時点のシステムのゾーンとして使用できなくなる可能性があります。

注 – ZFS クローンを使用してソースを複製できる場合でも、ZFS クローンは行わず ZFS の `zonepath` をコピーするように指定することができます。

詳細は、302 ページの「Solaris 10 11/06: 同一システム上での非大域ゾーンの複製」を参照してください。

◆ ◆ ◆ 第 20 章

非大域ゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)

この章では、非大域ゾーンのインストールおよびブート方法について説明します。クローニングを使って同一のシステムにゾーンをインストールする方法についても説明します。また、ゾーンの停止、リブート、アンインストールなど、インストールに関連するほかのタスクについても説明します。さらに、システムからゾーンを完全に削除する手順についても説明します。

ゾーンのインストールおよび関連する操作に関する一般的な情報については、第 19 章「非大域ゾーンのインストール、停止、複製、およびアンインストールについて (概要)」を参照してください。

lx ブランドゾーンのインストールと複製については、第 34 章「lx ブランドゾーンのインストール、ブート、停止、複製、およびアンインストールについて (概要)」および第 35 章「lx ブランドゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)」を参照してください。

ゾーンのインストール (タスクマップ)

タスク	説明	参照先
(任意) ゾーンをインストールする前に、構成済みのゾーンを検証します。	ゾーンがインストール要件を満たしていることを確認します。この手順を省略した場合、ゾーンのインストール時に検証が自動的に実行されます。	292 ページの「(オプション) インストール前に構成済みのゾーンを検証する方法」
構成済みのゾーンをインストールします。	構成済みの状態にあるゾーンをインストールします。	293 ページの「構成済みのゾーンをインストールする方法」

タスク	説明	参照先
Solaris 8/07: ゾーンの汎用一意識別子 (UUID) を取得します。	ゾーンのインストール時に割り当てられるこの個別の識別子は、ゾーンを識別するための代替手段になります。	294 ページの「Solaris 10 8/07: インストールされた非大域ゾーンの UUID を取得する方法」
(任意) インストール済みのゾーンを準備完了状態に移行します。	ゾーンをすぐにブートして使用する場合、この手順は省略できます。	296 ページの「(オプション) インストール済みのゾーンを準備完了状態に移行する方法」
ゾーンをブートします。	ゾーンをブートすると、ゾーンが稼働状態になります。ゾーンは、準備完了状態またはインストール済み状態からブートできます。ゾーンをはじめてブートしたあとのログインで内部ゾーン構成を実行する必要があります。	296 ページの「ゾーンのブート方法」、308 ページの「ゾーンの内部構成」、312 ページの「初期内部ゾーン構成を実行する」
ゾーンをシングルユーザーモードでブートします。	マイルストーン `svc:/milestone/single-user:default` に対してのみブートします。このマイルストーンは、`init` のレベル `s` と等同です。`init(1M)` および `svc.startd(1M)` のマニュアルページを参照してください。	298 ページの「ゾーンをシングルユーザーモードでブートする方法」

ゾーンのインストールとブート

`zoneadm(1M)` のマニュアルページの記述に従って `zoneadm` コマンドを使用し、非大域ゾーンのインストールタスクを実行します。ゾーンのインストールを実行するには、大域管理者になる必要があります。この章に示す例では、265 ページの「ゾーンを構成、検証、および確定する」で使用したゾーン名およびゾーンパスを使用します。

▼ (オプション) インストール前に構成済みのゾーンを検証する方法

ゾーンをインストールする前に検証できます。この手順を省略した場合、ゾーンのインストール時に検証が自動的に実行されます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **-z** オプションをゾーン名および **verify** サブコマンドとともに使用して、**my-zone** という名前の構成済みゾーンを検証します。

```
global# zoneadm -z my-zone verify
```

ゾーンパスの検証に関する次のメッセージが表示されます。

```
Warning: /export/home/my-zone does not exist, so it cannot be verified.
When 'zoneadm install' is run, 'install' will try to create
/export/home1/my-zone, and 'verify' will be tried again,
but the 'verify' may fail if:
the parent directory of /export/home/my-zone is group- or other-writable
or
/export/home1/my-zone overlaps with any other installed zones.
```

ただし、エラーメッセージが表示され、ゾーンの検証に失敗した場合は、メッセージに従って修正を行い、コマンドを再度実行してください。

エラーメッセージが表示されない場合は、ゾーンをインストールできます。

▼ 構成済みのゾーンをインストールする方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **zoneadm** コマンドに **-z install** オプションを使用して、構成済みのゾーン **my-zone** をインストールします。

```
global# zoneadm -z my-zone install
```

ゾーンのルートファイルシステムに必要なファイルおよびディレクトリがゾーンのルートパスにインストールされる際、さまざまなメッセージが表示されます。

3 (オプション) エラーメッセージが表示され、ゾーンのインストールに失敗した場合は、次のように入力してゾーンの状態を取得します。

```
global# zoneadm -z my-zone list -v
```

- 状態が構成済みであると表示された場合は、メッセージに示された修正を行い、`zoneadm install` コマンドを再度実行します。
- 状態が不完全であると表示された場合は、最初に次のコマンドを実行します。

  ```
  global# zoneadm -z my-zone uninstall
  ```

  次にメッセージに示された修正を行い、`zoneadm install` コマンドを再度実行します。

4 インストールが完了したら、**list** サブコマンドに **-i** オプションおよび **-v** オプションを指定してインストール済みのゾーンを一覧表示し、ステータスを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
 -  my-zone  installed    /export/home/my-zone           native     shared
```

注意事項 ゾーンのインストールが中断または失敗した場合は、ゾーンの状態は不完全なままになります。`uninstall -F` を使用して、ゾーンを構成済みの状態にリセットします。

次の手順 デフォルトでは、このゾーンは、『Solaris のシステム管理 (基本編)』の第 19 章「サービスの管理 (手順)」に説明されているオープンなネットワーク構成でインストールされています。ゾーンへのログイン時に、オープンなネットワーク構成に切り替えることも、個別のサービスを有効または無効に設定することもできます。詳細は、319 ページの「非大域ゾーンの別のネットワークサービス構成への切り替え」を参照してください。

▼ Solaris 10 8/07: インストールされた非大域ゾーンの UUID を取得する方法

ゾーンのインストール時に、汎用一意識別子 (UUID) がゾーンに割り当てられます。UUID は、`zoneadm` に `list` サブコマンドと `-p` オプションを使うことで取得できます。UUID は、5 番目に表示されるフィールドです。

- インストールされたゾーンの **UUID** を表示します。

```
global# zoneadm list -p
```

次のような情報が表示されます。

```
0:global:running:/:
6:my-zone:running:/export/home/my-zone:61901255-35cf-40d6-d501-f37dc84eb504
```

例 **20–1** コマンド内で UUID を使用する方法

```
global# zoneadm -z my-zone -u 61901255-35cf-40d6-d501-f37dc84eb504 list -v
```

`-u` *uuid-match* と `-z` *zonename* の両方が存在する場合、最初に UUID に基づいてマッチングが行われます。指定した UUID のゾーンが見つかった場合はそのゾーンが使用され、`-z` パラメータは無視されます。指定した UUID のゾーンが見つからなかった場合、システムはゾーン名で検索を実行します。

参考 UUID について

ゾーンをアンインストールすることも、同名のゾーンを内容を変えて再インストールすることもできます。ゾーンの内容を変更せずにゾーンの名前を変更することも可能です。こうした理由から、UUID はゾーン名よりも信頼性の高いハンドルです。

参照 詳細は、zoneadm(1M) および libuuid(3LIB) を参照してください。

▼ Solaris 10 8/07: インストールした非大域ゾーンに不完全のマークを付ける方法

システムに加えられた管理上の変更のためにゾーンが使用不可になるか、矛盾が生じた場合、インストールしたゾーンの状態を不完全に変更できます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 ゾーン **testzone** に不完全のマークを付けます。

```
global# zoneadm -z testzone mark incomplete
```

3 **list** サブコマンドに **-i** オプションと **-v** オプションを使って、ステータスを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
 -  my-zone  installed    /export/home/my-zone           native     shared
 -  testzone incomplete   /export/home/testzone          native     shared
```

参考 ゾーンへの不完全のマーク付け

-R *root* オプションを、 zoneadm の mark サブコマンドや list サブコマンドとともに使用して、代替ブート環境を指定できます。詳細は、zoneadm(1M) のマニュアルページを参照してください。

注 – ゾーンへの不完全のマーク付けは、取り消すことができません。不完全のマークが付けられたゾーンに実行可能なのは、ゾーンをアンインストールして、構成済みの状態に戻す操作だけです。301 ページの「ゾーンをアンインストールする方法」を参照してください。

▼ (オプション) インストール済みのゾーンを準備完了状態に移行する方法

準備完了状態に移行すると、仮想プラットフォームでユーザープロセスを開始する準備が整います。準備完了状態のゾーンには、内部で実行中のユーザープロセスは存在しません。

ゾーンをすぐにブートして使用する場合、この手順は省略できます。ゾーンのブート時に、準備完了状態への移行が自動的に行われます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **zoneadm** コマンドを **-z** オプション、ゾーン名 (**my-zone**)、および **ready** サブコマンドとともに使用することで、そのゾーンを準備完了状態に移行します。

   ```
   global# zoneadm -z my-zone ready
   ```

3. プロンプトで、**zoneadm list** コマンドに **-v** オプションを指定して、ステータスを確認します。

   ```
   global# zoneadm list -v
   ```

   次のような情報が表示されます。

   ```
   ID  NAME     STATUS       PATH                           BRAND      IP
    0  global   running      /                              native     shared
    1  my-zone  ready        /export/home/my-zone           native     shared
   ```

   システムにより一意のゾーン ID 1 が割り当てられていることに注目してください。

▼ ゾーンのブート方法

ゾーンをブートすると、ゾーンが稼働状態になります。ゾーンは、準備完了状態またはインストール済み状態からブートできます。ブートしたインストール済み状態

のゾーンは、準備完了状態から稼働状態に透過的に移行します。稼働状態のゾーンに対してはゾーンへのログインが可能です。

ヒント – ゾーンにはじめてログインするときに、内部ゾーン構成を実行します。これについては、308 ページの「ゾーンの内部構成」で説明しています。

314 ページの「/etc/sysidcfg ファイルを使用して初期ゾーン構成を行う方法」の記述に従い、/etc/sysidcfg ファイルを使用して初期ゾーン構成を実行する場合は、sysidcfg ファイルを作成してゾーンの /etc ディレクトリに配置してから、ゾーンをブートします。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **zoneadm** コマンドを **-z** オプション、ゾーン名 (**my-zone**)、および **boot** サブコマンドとともに使用することで、ゾーンをブートします。

```
global# zoneadm -z my-zone boot
```

3 ブートが完了したら、**list** サブコマンドに **-v** オプションを指定してステータスを確認します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
 1  my-zone  running      /export/home/my-zone           native     shared
```

例 20–2 ゾーンのブート引数を指定する

-m verbose オプションを使用してゾーンをブートします。

```
global# zoneadm -z my-zone boot -- -m verbose
```

-m verbose ブートオプションを使用してゾーンをリブートします。

```
global# zoneadm -z my-zone reboot -- -m verbose
```

ゾーン管理者が -m verbose オプションを使用してゾーン *my-zone* をリブートします。

```
my-zone# reboot -- -m verbose
```

注意事項　ゾーン構成で指定された IP アドレス用のネットマスクをシステムが検出できなかったことを示すメッセージが表示された場合は、444 ページの「ゾーンブート時に netmasks の警告が表示される」を参照してください。このメッセージは単なる警告であり、コマンドは成功しています。

▼ ゾーンをシングルユーザーモードでブートする方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。
2. ゾーンをシングルユーザーモードでブートします。

```
global# zoneadm -z my-zone boot -s
```

次に進む手順

ゾーンにログインして初期内部構成を実行する方法については、第 21 章「非大域ゾーンへのログイン (概要)」および第 22 章「非大域ゾーンへのログイン (タスク)」を参照してください。

非大域ゾーンの停止、リブート、アンインストール、複製、および削除 (タスクマップ)

タスク	説明	参照先
ゾーンを停止します。	停止手順を実行して、ゾーンのアプリケーション環境と仮想プラットフォームの両方を削除します。この手順により、ゾーンが準備完了状態からインストール済み状態に戻されます。ゾーンの完全な停止処理を行う方法については、319 ページの「zlogin を使用してゾーンを停止処理する方法」を参照してください。	299 ページの「ゾーンの停止方法」

タスク	説明	参照先
ゾーンをリブートします。	リブートの手順を実行すると、ゾーンが停止してから再びブートします。	300 ページの「ゾーンをリブートする方法」
ゾーンをアンインストールします。	ゾーンのルートファイルシステム内のすべてのファイルを削除します。「この手順は、十分注意して実行する必要があります。」実行した操作を元に戻すことはできません。	301 ページの「ゾーンをアンインストールする方法」
同一システムの既存ゾーンの構成に基づいて、新しい非大域ゾーンをプロビジョニングします。	ゾーンのクローニングは、ゾーンのインストールより高速な代替手段です。ただし、インストールの前にはやはり新規ゾーンを構成してください。	302 ページの「Solaris 10 11/06: 同一システム上での非大域ゾーンの複製」
システムから非大域ゾーンを削除します。	この手順を実行すると、システムからゾーンが完全に削除されます。	305 ページの「システムから非大域ゾーンを削除する」

ゾーンの停止、リブート、およびアンインストール

▼ ゾーンの停止方法

ゾーンのアプリケーション環境と仮想プラットフォームの両方を削除する場合に、この停止手順を実行します。ゾーンの完全な停止処理を行う方法については、319 ページの「zlogin を使用してゾーンを停止処理する方法」を参照してください。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 システムで稼働中のゾーンの一覧を表示します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
 1  my-zone  running      /export/home/my-zone           native     shared
```

3 **zoneadm** コマンドを **-z** オプション、ゾーン名 (**my-zone** など)、および **halt** サブコマンドとともに使用することで、指定されたゾーンを停止します。

```
global# zoneadm -z my-zone halt
```

4 システム内のゾーンの一覧を再度表示して、**my-zone** が停止していることを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
 -  my-zone  installed    /export/home/my-zone           native     shared
```

5 ゾーンを再び起動する場合は、次のコマンドをブートします。

```
global# zoneadm -z my-zone boot
```

注意事項 停止操作が失敗する場合は、443 ページの「ゾーンが停止しない」 でトラブルシューティングのヒントを参照してください。

▼ ゾーンをリブートする方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 システムで稼働中のゾーンの一覧を表示します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
 1  my-zone  running      /export/home/my-zone           native     shared
```

3 **zoneadm** コマンドを **-z reboot** オプションとともに使用することで、ゾーン **my-zone** をリブートします。

```
global# zoneadm -z my-zone reboot
```

4 システム内のゾーンの一覧を再度表示して、**my-zone** がリブートしたことを確認します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
 2  my-zone  running      /export/home/my-zone           native     shared
```

ヒント – my-zone のゾーン ID が変更されていることに注目してください。通常、リブートするとゾーン ID は変更されます。

▼ ゾーンをアンインストールする方法

注意 – この手順は、注意深く実行してください。ゾーンのルートファイルシステム内のファイルすべてを削除した後で、操作を元に戻すことはできません。

ゾーンは稼働状態であっていけません。uninstall 操作は、稼働中のゾーンに対しては無効です。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. システム内のゾーンの一覧を表示します。

   ```
   global# zoneadm list -v
   ```

   次のような情報が表示されます。

   ```
   ID  NAME     STATUS       PATH                           BRAND      IP
    0  global   running      /                              native     shared
    -  my-zone  installed    /export/home/my-zone           native     shared
   ```

3. **zoneadm** コマンドを **-z uninstall** オプションとともに使用することで、ゾーン **my-zone** を削除します。

   -F オプションを使用すると、処理を強制的に実行できます。このオプションが指定されていない場合、システムにより確認を求めるメッセージが表示されます。

   ```
   global# zoneadm -z my-zone uninstall -F
   ```

4. システム内のゾーンの一覧を再度表示して、**my-zone** が一覧に含まれていないことを確認します。

   ```
   global# zoneadm list -v
   ```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
```

注意事項 ゾーンのアンインストールが中断した場合、ゾーンの状態は不完全なままになります。`zoneadm uninstall` コマンドを使用して、ゾーンを構成済みの状態にリセットしてください。

実行した操作を元に戻すことはできないため、`uninstall` コマンドは慎重に使用してください。

Solaris 10 11/06: 同一システム上での非大域ゾーンの複製

複製操作は、複製元の `zonepath` から複製先の `zonepath` にデータをコピーすることにより、システム上に新しいゾーンをプロビジョニングするのに使用されます。

Solaris 10 5/09 以降では、複製元の `zonepath` と複製先の `zonepath` が両方とも ZFS 上にあり、同じプールに含まれる場合、`zoneadm clone` コマンドは自動的に ZFS を使用してゾーンを複製します。ただし、ZFS の `zonepath` のコピーは行い、ZFS の複製は行わないように指定することもできます。

▼ ゾーンをクローンする方法

新規ゾーンをインストールする前に、そのゾーンを構成する必要があります。`zoneadm create` サブコマンドに渡されるパラメータは、クローンするゾーンの名前です。このソースゾーンを停止する必要があります。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 クローンされるソースゾーン (この手順では **my-zone**) を停止します。

```
global# zoneadm -z my-zone halt
```

3 ソースゾーン **my-zone** の構成をファイル (たとえば、**master**) にエクスポートすることにより、新規ゾーンの構成を開始します。

```
global# zonecfg -z my-zone export -f /export/zones/master
```

注 – 既存の構成を変更する代わりに、266 ページの「ゾーンの構成方法」で説明されている手順を使って、新規ゾーン構成を作成することもできます。この方法を使用する場合は、ゾーンを作成したあとで手順 6 に進みます。

4 **master** ファイルを編集します。複数のゾーンで同一であってはならないコンポーネントに対して、異なるプロパティーとリソースを設定します。たとえば、新しい **zonepath** を設定する必要があります。共有 **IP** ゾーンの場合は、各 **net** リソースの **IP** アドレスを変更する必要があります。排他的 **IP** ゾーンの場合は、各 **net** リソースの **physical** プロパティーを変更する必要があります。

5 *master* ファイル内のコマンドを使って、新規ゾーン **zone1** を作成します。

```
global# zonecfg -z zone1 -f /export/zones/master
```

6 **my-zone** をクローニングして、新規ゾーン **zone1** をインストールします。

```
global# zoneadm -z zone1 clone my-zone
```

システムには次のように表示されます。

```
Cloning zonepath /export/home/my-zone...
```

Solaris 10 5/09 以降では、複製元の `zonepath` が ZFS プール上にある場合 (例: `zeepool`)、システムには次のように表示されます。

```
Cloning snapshot zeepool/zones/my-zone@SUNWzone1
Instead of copying, a ZFS clone has been created for this zone.
```

7 システム内のゾーンの一覧を表示します。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
 -  my-zone  installed    /export/home/my-zone           native     shared
 -  zone1    installed    /export/home/zone1             native     shared
```

参考

Solaris 10 5/09: ZFS ファイルシステム上にある複製元の zonepath が複製される場合

独自の ZFS ファイルシステム上にある複製元 `zonepath` を `zoneadm` コマンドで複製すると、次の処理が実行されます。

- `zoneadm` コマンドは、ソフトウェア目録を作成します。
- `zoneadm` コマンドは、ZFS スナップショットを作成し、`SUNWzone`*X* という形式の名前 (例: `SUNWzone`*1*) を付けます。
- `zoneadm` コマンドは、ZFS クローンを使用してスナップショットを複製します。

▼ Solaris 10 5/09: 既存のスナップショットからゾーンを複製する方法

最初にゾーンを複製したときに作成された既存のスナップショットから、元のゾーンを何度も複製することができます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 ゾーン **zone2** を構成します。

3 既存のスナップショットを使用して **new-zone2** を作成することを指定します。

```
global# zoneadm -z zone2 clone -s zeepool/zones/my-zone@SUNWzone1 my-zone
```

システムには次のように表示されます。

```
Cloning snapshot zeepool/zones/my-zone@SUNWzone1
```

zoneadm コマンドは、スナップショット SUNWzone*1* のソフトウェアを検証し、スナップショットを複製します。

4 システム内のゾーンの一覧を表示します。

```
ID  NAME     STATUS       PATH                          BRAND      IP
 0  global   running      /                             native     shared
 -  my-zone  installed    /zeepool/zones/my-zone        native     shared
 -  zone1    installed    /zeepool/zones/zone1          native     shared
 -  zone2    installed    /zeepool/zones/zone2          native     shared
```

▼ Solaris 10 5/09: ZFS クローンの代わりにコピーを使用する方法

ZFS ファイルシステム上のゾーンの自動複製を防止し、代わりに zonepath をコピーするように指定する場合は、ここで説明する手順を使用します。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **ZFS** クローンは行わず、**ZFS** 上の **zonepath** をコピーするように指定します。

```
global# zoneadm -z zone1 clone -m copy my-zone
```

システムから非大域ゾーンを削除する

このセクションでは、システムからゾーンを完全に削除する手順を説明します。

▼ 非大域ゾーンを削除する方法

1 ゾーン **my-zone** を停止処理します。

```
global# zlogin my-zone shutdown -y -g0 -i0
```

2 **my-zone** のルートファイルシステムを削除します。

```
global# zoneadm -z my-zone uninstall -F
```

3 **my-zone** の構成を削除します。

```
global# zonecfg -z my-zone delete -F
```

4 システム内のゾーンの一覧を表示し、**my-zone** が一覧に含まれていないことを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
```

◆ ◆ ◆ 第 21 章

21 非大域ゾーンへのログイン (概要)

この章では、大域ゾーンからゾーンへのログインについて説明します。

この章の内容は次のとおりです。

手順および使用法については、第 22 章「非大域ゾーンへのログイン (タスク)」を参照してください。

zlogin コマンド

ゾーンのインストール後に、ゾーンにログインしてアプリケーション環境を完成させる必要があります。ゾーンにログインして、管理タスクを行うこともできます。-C オプションを使用してゾーンコンソールに接続しない限り、zlogin を使用してゾーンにログインすると、新しいタスクが開始されます。1 つのタスクを 2 つのゾーンで実行することはできません。

zlogin コマンドを使用して、大域ゾーンから稼働状態または準備完了状態にある任意のゾーンにログインします。

注 – 稼働状態にないゾーンへのログインに使用できるのは、zlogin コマンドと -C オプションの組み合わせだけです。

---

317 ページの「非対話型モードを使用してゾーンにアクセスする方法」に記述されているとおり、ゾーン内での実行を指示するコマンドを指定することで、zlogin コマ

ンドを非対話型モードで使用できます。ただし、このコマンドおよびこのコマンドの処理対象となるファイルは、いずれも NFS 上に存在していてはなりません。開かれているファイルのいずれか、またはそのアドレス空間のいずれかの部分が NFS 上に存在する場合、コマンドは失敗します。アドレス空間には、コマンドの実行可能ファイル自体またはリンクされたライブラリが含まれます。

大域ゾーンを操作する大域管理者が使用できるのは、`zlogin` コマンドだけです。詳細は、`zlogin(1)` のマニュアルページを参照してください。

ゾーンの内部構成

インストール後、ゾーンは未構成の状態です。ゾーンには、ネームサービス用の内部構成は存在せず、 ロケールおよびタイムゾーンは設定されておらず、ほかのさまざまな構成タスクも実行されていません。このため、ゾーンコンソールログインの初回使用時に、`sysidtool` プログラムが実行されます。詳細は、`sysidtool(1M)` のマニュアルページを参照してください。

次の 2 つの方法で、必要な構成を実行できます。

- ゾーンコンソールログイン。この場合、システムにより一連の質問が表示されます。次の情報を提供する準備をしておく必要があります。
  - 言語
  - 使用する端末の種類
  - ホスト名
  - セキュリティーポリシー (Kerberos または標準 UNIX)
  - ネームサービスの種類 (`None` も使用可)
  - ネームサービスドメイン
  - ネームサーバー
  - デフォルトのタイムゾーン
  - root パスワード

  手順については、312 ページの「初期内部ゾーン構成を実行する」を参照してください。
- `/etc/sysidcfg` ファイル。ゾーンを最初にブートする前に、このファイルを作成してゾーン内に配置することができます。詳細は、`sysidcfg(4)` のマニュアルページを参照してください。

非大域ゾーンへのログイン方法

このセクションでは、ゾーンへのログインに使用可能な方法について説明します。

ゾーンコンソールログイン

各ゾーンは、仮想コンソール `/dev/console` を保持します。コンソール上で操作を実行することを、コンソールモードと呼びます。ゾーンコンソールは、システム上のシリアルコンソールに非常に似ています。コンソールへの接続は、ゾーンをリブートしても持続します。コンソールモードと `telnet` などのログインセッションとの違いを理解するには、310 ページの「リモートログイン」を参照してください。

ゾーンコンソールへの接続には、`zlogin` コマンドと `-C` オプション、および *zonename* を使用します。ゾーンを稼働状態にする必要はありません。

ゾーン内部のプロセスが、コンソールを開いてメッセージを書き込むことができます。`zlogin -C` プロセスが終了すると、別のプロセスがコンソールにアクセスできるようになります。

ユーザーログインの方法

ユーザー名を使ってゾーンにログインする場合は、`zlogin` コマンドと `-l` オプション、ユーザー名、および *zonename* を使用します。たとえば、大域ゾーンの管理者は、`zlogin` に `-l` オプションを指定することで、通常のユーザーとして非大域ゾーンにログインできます。

```
global# zlogin -l user zonename
```

ユーザー `root` でログインするには、オプションを指定せずに `zlogin` を使用します。

フェイルセーフモード

ログインで問題が発生し、`zlogin` コマンドまたは `zlogin` コマンドと `-C` オプションを使用してゾーンにアクセスできない場合、代替手段が存在します。`zlogin` コマンドと `-S` (safe) オプションを使用することで、ゾーンに入ることができます。このモードは、ほかのログイン方法が成功しなかったときに、損傷を受けたゾーンを復元する場合にのみ使用してください。この最小環境では、ゾーンログインが失敗した理由を診断できる場合があります。

リモートログイン

ゾーンにリモートでログインする機能は、選択したネットワークサービスに依存します。デフォルトでは通常、rlogin、ssh、および telnet を使用してのログインが機能します。これらのコマンドの詳細は、rlogin(1)、ssh(1)、および telnet(1) のマニュアルページを参照してください。

対話型モードと非対話型モード

zlogin コマンドを使ってゾーンにアクセスし、ゾーン内でコマンドを実行する方法がさらに 2 つ存在します。その方法が、対話型モードおよび非対話型モードです。

対話型モード

対話型モードでは、ゾーン内部で使用する新しい仮想端末が割り当てられます。コンソールデバイスへの排他的なアクセスが許可されるコンソールモードとは異なり、対話型モードでは、いつでも任意の数の zlogin セッションを開くことができます。対話型モードが有効になるのは、発行するコマンドが含まれていない場合です。エディタなどの端末デバイスを必要とするプログラムは、このモードで正常に動作します。

非対話型モード

非対話型モードは、ゾーンを管理するシェルスクリプトを実行する場合に使用します。非対話型モードでは、新しい仮想端末は割り当てられません。ゾーン内部で実行されるコマンドを指定すると、非対話型モードが有効になります。

◆ ◆ ◆ 第 22 章

22 非大域ゾーンへのログイン (タスク)

この章では、インストール済みのゾーンの構成を完了して大域ゾーンからゾーンにログインし、ゾーンを停止処理する手順について説明します。また、zonename コマンドを使用して現在のゾーン名を出力する方法についても説明します。

ゾーンへのログイン処理の概要については、第 21 章「非大域ゾーンへのログイン (概要)」を参照してください。

初期ゾーンブートおよびゾーンログインの手順 (タスクマップ)

タスク	説明	参照先
内部構成を実行します。	ゾーンコンソールにログインするか、/etc/sysidcfg ファイルを使用して初期ゾーン構成を実行します。	312 ページの「初期内部ゾーン構成を実行する」
ゾーンにログインします。	ゾーンへのログインには、コンソールを使用する、対話型モードを使って仮想端末を割り当てる、またはゾーン内で実行するコマンドを指定する方法があります。実行するコマンドを指定する場合、仮想端末は割り当てられません。ゾーンへの接続が拒否された場合は、フェイルセーフモードを使用してログインすることもできます。	316 ページの「ゾーンへのログイン」
非大域ゾーンから抜けます。	非大域ゾーンへの接続を切り離します。	318 ページの「非大域ゾーンから抜ける方法」

タスク	説明	参照先
ゾーンを停止処理します。	shutdown ユーティリティーまたはスクリプトを使用して、ゾーンを停止処理します。	319 ページの「zlogin を使用してゾーンを停止処理する方法」
ゾーン名を出力します。	現在のゾーンの名前を出力します。	320 ページの「現在のゾーンの名前を出力する」

初期内部ゾーン構成を実行する

次のいずれかの方法で、ゾーンを構成する必要があります。

- 308 ページの「ゾーンの内部構成」に記載された手順に従い、ゾーンにログインして構成を行います。
- 314 ページの「/etc/sysidcfg ファイルを使用して初期ゾーン構成を行う方法」に記載された手順に従い、/etc/sysidcfg ファイルを使用してゾーンを構成します。

ヒント – 内部構成を実行したあとは、非大域ゾーンの構成のコピーを作成することをお勧めします。このバックアップを使用して、あとでゾーンを復元することができます。スーパーユーザーまたは Primary Administrator として、ゾーン my-zone の構成をファイルに出力してください。次の例では、 my-zone.config というファイルを使用しています。

```
global# zonecfg -z my-zone export > my-zone.config
```

詳細は、434 ページの「非大域ゾーンを個別に復元する方法」を参照してください。

▼ ゾーンコンソールにログインして初期ゾーン構成を行う方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **zlogin** コマンドと **-C** オプション、およびゾーン名 **(**この手順では **my-zone)** を使用します。

```
global# zlogin -C my-zone
```

3 別の端末ウィンドウからゾーンをブートします。

```
global# zoneadm -z my-zone boot
```

次のような内容が、zlogin ウィンドウに表示されます。

```
[NOTICE: Zone booting up]
```

4 コンソールへの初回ログイン時に、一連の質問に応答するよう求められます。画面には、次のようなメッセージが表示されます。

```
SunOS Release 5.10 Version Generic 64-bit
Copyright 1983-2006 Sun Microsystems, Inc.  All rights reserved.
Use is subject to license terms.

Hostname: my-zone
Loading smf(5) service descriptions:
Select a Language

     1. English
     2. es
     2. fr
Please make a choice (0 - 1), or press h or ? for help:

Select a Locale

      1. English (C - 7-bit ASCII)
      2. Canada (English) (UTF-8)
      4. U.S.A. (UTF-8)
      5. U.S.A. (en_US.ISO8859-1)
      6. U.S.A. (en_US.ISO8859-15)
      7. Go Back to Previous Screen
Please make a choice (0 - 9), or press h or ? for help:

What type of terminal are you using?
      1) ANSI Standard CRT
      2) DEC VT52
      3) DEC VT100
      4) Heathkit 19
      5) Lear Siegler ADM31
      6) PC Console
      7) Sun Command Tool
      8) Sun Workstation
      9) Televideo 910
      10) Televideo 925
      11) Wyse Model 50
      12) X Terminal Emulator (xterms)
      13) CDE Terminal Emulator (dtterm)
      14) Other
Type the number of your choice and press Return:
13
.
.
.
```

応答する必要のある質問の全一覧については、308 ページの「ゾーンの内部構成」を参照してください。

5 (オプション) 手順 3 で説明した 2 つのウィンドウを使用していない場合、構成情報の指定を求める初期メッセージが表示されない可能性があります。ゾーンへのログイン時に、構成情報の指定を求めるメッセージの代わりに、次のシステムメッセージが表示される場合があります。

```
[connected to zone zonename console]
```

この場合は、Return キーを押すと構成情報の指定を求めるメッセージが表示されます。

不正な情報を指定したために構成をやり直す場合、うまく再指定できない場合があります。原因は、`sysidtools` により以前の指定が保存されていることが考えられます。

この問題が発生した場合は、大域ゾーンから次の回避手順を実行して構成処理を再実行してください。

```
global# zlogin -S zonename /usr/sbin/sys-unconfig
```

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

▼ `/etc/sysidcfg` ファイルを使用して初期ゾーン構成を行う方法

Solaris 10 8/07: キーワード `nfs4_domain` が追加されました。例のファイルにはこのキーワードが示されています。次の手順 4 は、これより前のリリースを実行している場合の追加手順です。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 大域ゾーンから、非大域ゾーンの `/etc` ディレクトリに移動します。

```
global# cd /export/home/my-zone/root/etc
```

3 **`sysidcfg`** ファイルを作成して、このディレクトリに配置します。

ファイルには、次のような情報が記載されています。

- 共有 **IP** ゾーンの場合:

```
system_locale=C
terminal=dtterm
network_interface=primary {
```

```
            hostname=my-zone
}
security_policy=NONE
name_service=NIS {
            domain_name=special.example.com
            name_server=bird(192.168.112.3)
}
nfs4_domain=domain.com
timezone=US/Central
root_password=m4qtoWN
```

- 静的 **IP** 構成を使用する排他的 **IP** ゾーンの場合:

```
system_locale=C
terminal=dtterm
network_interface=primary {
         hostname=my-zone
         default_route=10.10.10.1
         ip_address=10.10.10.13
         netmask=255.255.255.0
}
nfs4_domain=domain.com
timezone=US/Central
root_password=m4qtoWN
```

- **DHCP** と **IPv6** オプションを使用する排他的 **IP** ゾーンの場合:

```
system_locale=C
terminal=dtterm
network_interface=primary {
            dhcp protocol_ipv6=yes
}
security_policy=NONE
name_service=DNS {
         domain_name=example.net
         name_server=192.168.224.11,192.168.224.33
}
nfs4_domain=domain.com
timezone=US/Central
root_password=m4qtoWN
```

4 **Solaris 10 8/07** より前のリリースを実行している場合、キーワード **nfs4_domain** が **sysidcfg** ファイルに含まれていません。デフォルトでは、別個のモジュールが **nfsmapid** コマンドの使用する **NFSv4** ドメインパラメータを要求します。自動初期ゾーン構成を完了するには、**default/nfs** ファイルを編集して **NFSMAPID_DOMAIN** パラメータをコメント解除し、ドメインを適切な **NFSv4** ドメインに設定します。

```
global# vi default/nfs
       .
       .
       .
       NFSMAPID_DOMAIN=domain
```

このディレクトリに `.NFS4inst_state.domain` ファイルを作成して、NFSv4 ドメインが設定済みであることを示します。

```
global# touch .NFS4inst_state.domain
```

NFSv4 ドメインパラメータについては、nfsmapid(1M) のマニュアルページを参照してください。

5 ゾーンをブートします。

参照 詳細は、sysidcfg(4) のマニュアルページを参照してください。

ゾーンへのログイン

zlogin コマンドを使用して、大域ゾーンから稼働状態または準備完了状態にある任意のゾーンへログインします。詳細は、zlogin(1) のマニュアルページを参照してください。

次の手順で説明されているように、ゾーンへのログインはさまざまな方法で実行できます。310 ページの「リモートログイン」で説明されているように、リモートでログインすることも可能です。

▼ ゾーンコンソールへのログイン方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。
役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **zlogin** コマンドを **-C** オプションと **my-zone** などのゾーン名とともに使用します。

```
global# zlogin -C my-zone
```

注 – zoneadm boot コマンドの実行後、すぐに zlogin セッションを開始すると、ゾーンからのブートメッセージが表示されます。

```
SunOS Release 5.10 Version Generic 64-bit
Copyright 1983-2005 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
starting rpc services: rpcbind done.
syslog service starting.
The system is ready.
```

3 ゾーンコンソールが表示されたら、**root** でログインし、**Return** キーを押します。プロンプトが表示されたら **root** のパスワードを入力します。

```
my-zone console login: root
Password:
```

▼ 対話型モードを使用してゾーンにアクセスする方法

対話型モードでは、ゾーン内部で使用する新しい仮想端末が割り当てられます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 大域ゾーンからゾーン (例: **my-zone**) にログインします。

```
global# zlogin my-zone
```

次のような情報が表示されます。

```
[Connected to zone 'my-zone' pts/2]
Last login: Wed Jul  3 16:25:00 on console
Sun Microsystems Inc. SunOS 5.10 Generic June 2004
```

3 **exit** と入力して、接続を閉じます。

次のようなメッセージが表示されます。

```
[Connection to zone 'my-zone' pts/2 closed]
```

▼ 非対話型モードを使用してゾーンにアクセスする方法

ゾーン内部で実行されるコマンドを指定すると、非対話型モードが有効になります。非対話型モードでは、新しい仮想端末は割り当てられません。

コマンドおよびコマンドの処理対象のファイルは、いずれも NFS 上に存在してはならないことに注意してください。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 大域ゾーンから **my-zone** ゾーンにログインして、コマンド名を入力します。

ここではコマンド zonename を使用します。

```
global# zlogin my-zone zonename
```

次の出力が表示されます。

```
my-zone
```

▼ 非大域ゾーンから抜ける方法

- 非大域ゾーンへの接続を切り離すには、次のいずれかの方法を使用します。
  - ゾーンの非仮想コンソールを終了するには、次の操作を行います。

    ```
    zonename# exit
    ```

  - ゾーンの仮想コンソールへの接続を切り離すには、次のようにチルダ (~) 文字とピリオドを使用します。

    ```
    zonename# ~.
    ```

    画面には、次のようなメッセージが表示されます。

    ```
    [Connection to zone 'lx-zone' pts/6 closed]
    ```

参照　`zlogin` コマンドのオプションの詳細については、`zlogin(1)` のマニュアルページを参照してください。

▼ フェイルセーフモードを使用してゾーンに入る方法

ゾーンへの接続が拒否された場合、`zlogin` コマンドと `-S` オプションを使用して、ゾーン内の最小環境に入ることができます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 大域ゾーンから、**zlogin** コマンドと **-S** オプションを使用してゾーン (例: **my-zone**) にアクセスします。

   ```
   global# zlogin -S my-zone
   ```

▼ zlogin を使用してゾーンを停止処理する方法

注 – 大域ゾーンで init 0 を実行して Solaris システムの完全な停止処理を実行すると、システム内のそれぞれの非大域ゾーンでも init 0 が実行されます。init 0 は、ローカルユーザーとリモートユーザーに対してシステムが停止する前にログオフするよう警告しないので、注意してください。

ゾーンを正しく停止処理するには、次の手順を実行します。停止処理スクリプトを実行せずにゾーンを停止する方法については、299 ページの「ゾーンの停止方法」を参照してください。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 停止処理を行うゾーン (例: **my-zone**) にログインし、ユーティリティーの名前として **shutdown** を、状態として **init 0** を指定します。

   ```
   global# zlogin my-zone shutdown -y -g0 -i 0
   ```

   サイトによっては、特定の環境に合わせた独自の停止処理スクリプトが存在する場合もあります。

参考 非対話型モードでの shutdown の使用

現時点では、非対話型モードで shutdown コマンドを使って、ゾーンをシングルユーザー状態にすることはできません。詳細は、CR 6214427 を参照してください。

317 ページの「対話型モードを使用してゾーンにアクセスする方法」の説明に従って、対話型ログインを使用できます。

非大域ゾーンの別のネットワークサービス構成への切り替え

このゾーンは、『Solaris のシステム管理 (基本編)』の第 19 章「サービスの管理 (手順)」に説明されているオープンなネットワーク構成でインストールされています。このゾーンを制限されたネットワーク構成に切り替えることも、ゾーン内の個別のサービスを有効または無効にすることもできます。

▼ ゾーンを制限されたネットワークサービス構成に切り替える方法

1 大域ゾーンからゾーン (例: **my-zone**) にログインします。

```
global# zlogin my-zone
```

2 **netservices** コマンドを実行して、ゾーンを制限されたネットワーク構成に切り替えます。

```
my-zone# /usr/sbin/netservices limited
```

次のような情報が表示されます。プロンプトに *y* と入力して、`dtlogin` を再起動します。

```
restarting syslogd
restarting sendmail
dtlogin needs to be restarted. Restart now? [Y] y
restarting dtlogin
```

▼ ゾーン内で特定のサービスを有効にする方法

1 大域ゾーンからゾーン (例: **my-zone**) にログインします。

```
global# zlogin my-zone
```

2 **svcadm** コマンドを実行して、リソース上限デーモンを使用した物理メモリー制御を有効にします。

```
my-zone# svcadm enable svc:/system/rcap:default
```

3 サービスの一覧を表示して、**rcapd** が有効になっていることを確認します。

```
my-zone# svcs -a
.
.
.
online    14:04:21 svc:/system/rcap:default
.
.
.
```

現在のゾーンの名前を出力する

`zonename`(1) のマニュアルページに説明されているように、`zonename` コマンドにより現在のゾーンの名前が出力されます。次に、大域ゾーン内で `zonename` を使用した場合の出力例を示します。

```
# zonename
global
```

◆ ◆ ◆ 第 23 章

23 非大域ゾーンの移動と移行 (タスク)

この章は、Solaris 10 11/06 リリースで新たに追加されました。以降のリリースでは、新しい機能が追加されています。

この章では、次の操作を行う方法について説明します。

- 既存の非大域ゾーンを同じマシンの新しい場所に移動する
- 実際の移行を実行する前に、非大域ゾーンの移行で何が発生するかを検証する
- 既存の非大域ゾーン新しいマシンに移行する
- `zoneadm detach` および `zoneadm attach` コマンドを使用して、低いパッチレベルを持つゾーンを高いパッチレベルで大域ゾーンのレベルに更新する

Solaris 10 10/08 リリース以降では、ゾーンに依存するパッケージおよび関連パッチが新しいホストに存在する場合、それらのバージョンが移行元と同じまたは移行元より新しいときは、`zoneadm attach` と `-u` オプションを使用すると、パッケージの最小セットが更新され、その非大域ゾーンを新しいホストで使用できるようになります。新しいホストに移行元ホストよりも高いバージョンと低いバージョンのパッチが混在している場合、接続操作中の更新はできません。

`zoneadm attach` コマンドと `-u` オプションを使用すると、`sun4u` から `sun4v` への移行など、マシンクラス間の移行も可能です。

Solaris 10 9/10 リリース以降では、`zoneadm attach` と `-U` オプションを使用すると、ゾーンのすべてのパッケージが、このホストに新しくインストールされた非大域ゾーンで表示される内容と一致するように更新されます。ゾーン内にインストールされ、大域ゾーンにはインストールされていないパッケージがある場合、それらのパッケージは無視され、そのままの状態になります。このオプションを使用すると、`sun4u` から `sun4v` への移行など、マシンクラス間の自動移行も可能です。

通常のパッチ適用の代わりに、大域ゾーンにパッチを適用する間は各ゾーンを切り離しておき、適用後に `-U` オプションを使用して大域ゾーンを再接続することで、大域ゾーンのパッチレベルを一致させることができます。

lx ブランドゾーンの移動と移行については、第 37 章「lx ブランドゾーンの移動と移行 (タスク)」を参照してください。

Solaris 10 11/06: 非大域ゾーンの移動

zonepath を変更してゾーンを同じシステムの新しい場所に移動する場合に、ここで説明する手順を使用します。ゾーンは、停止する必要があります。新規 zonepath がローカルファイルシステムに存在する必要があります。247 ページの「リソースタイプとプロパティータイプ」に説明されている、 zonepath の通常の基準が適用されます。

▼ ゾーンを移動する方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 移動するゾーン **(**この手順では **db-zone)** を停止します。

   ```
   global# zoneadm -z db-zone halt
   ```

3. **zoneadm** コマンドを **move** サブコマンドとともに使用して、ゾーンを新規の **zonepath** である **/export/zones/db-zone** に移動します。

   ```
   global# zoneadm -z db-zone move /export/zones/db-zone
   ```

4. パスを確認します。

   ```
   ID  NAME     STATUS       PATH                          BRAND      IP
    0  global   running      /                             native     shared
    -  my-zone  installed    /export/home/my-zone          native     shared
    -  db-zone  installed    /export/zones/db-zone         native     shared
   ```

Solaris 10 11/06: 別のマシンへの非大域ゾーンの移行

Solaris 10 5/08 リリースでは、実際にゾーンを別のマシンに移行する前にゾーンの移行を試験的に実行することができます。詳細は、328 ページの「Solaris 10 5/08: 移行を行う前のゾーンの移行の検証について」を参照してください。

ゾーンの移行について

Solaris 10 11/06 リリース以降、このセクションに新しい情報が追加されています。

zonecfg および zoneadm コマンドを使用して、既存の非大域ゾーンをあるシステムから別のシステムに移行できます。ゾーンは停止され、現在のホストから切り離されます。zonepath はターゲットホストに移動され、そこで接続されます。

次の制限が、ゾーンの移行に適用されます。

- ターゲットシステム上の大域ゾーンで、元のソースホストと同じかそれ以降の Oracle Solaris リリースが稼働している必要があります。
- ゾーンが正常に動作することを保証するため、移行元ホストにインストールされているものと同じバージョンの次の必須オペレーティングシステムパッケージおよびパッチが、移行先システムにもインストールされている必要があります。
  - ファイルを inherit-pkg-dir リソース内に格納するパッケージ
  - SUNW_PKG_ALLZONES=true であるパッケージ

  他社製品のパッケージやパッチなど、ほかのパッケージおよびパッチは異なっていてもかまいません。
- **Solaris 10 10/08:** ゾーンに依存するパッケージおよび関連パッチが新しいホストに存在する場合、それらのバージョンが移行元より新しいときは、zoneadm attach と -u オプションを使用すると、ゾーン内のこのようなパッケージが新しいホストに一致するように更新されます。接続時更新ソフトウェアは、移行対象のゾーンを調べ、新しいホストに一致するように更新する必要のあるパッケージを決定します。このようなパッケージだけが更新されます。それ以外のパッケージおよびその関連パッチは、ゾーンごとに異なる可能性があります。このオプションを使用すると、sun4u から sun4v への移行など、マシンクラス間の自動移行も可能です。

  Solaris 10 9/10: パッケージおよび関連パッチが新しいホストに存在する場合、それらのバージョンが移行元より新しいときは、zoneadm attach と -U オプションを使用すると、ゾーン内のこのようなパッケージが、このホストに新しくインストールされた非大域ゾーンで表示される内容と一致するように更新されます。ゾーン内にインストールされ、大域ゾーンにはインストールされていないパッケージがある場合、それらのパッケージは無視され、そのままの状態になります。このオプションを使用すると、sun4u から sun4v への移行など、マシンクラス間の自動移行も可能です。

  Solaris 10 5/09: -b オプションを使用すると、更新の前にゾーンから除去するパッチを指定できます。
- -u オプションを使用すると、sun4u と sun4v のマシンクラス間の移行が可能ですが、このオプションを使用しない場合は、移行元と移行先システムのマシンアーキテクチャーは同じでなければいけません。
- **Solaris 10 5/09:** -b オプションを使用すると、接続時にゾーンから除去するパッチを指定できます。指定できるパッチは、公式パッチと IDR (Interim Diagnostics/Relief) パッチのどちらかです。複数の -b オプションを指定できます。何らかの理由でいずれかのパッチを除去できない場合、attach は失敗し、どのパッチも除去されません。

このオプションは、SVr4 パッケージを使用しているゾーンブランドだけに適用されます。

Solaris リリースとマシンアーキテクチャーを確認するには、次のように入力します。

```
#uname -m
```

zoneadm detach プロセスにより、別のシステムでゾーンを接続するのに必要な情報が作成されます。 zoneadm attach プロセスは、ターゲットマシンがゾーンのホストとして機能するための適正な構成を保持していることを確認します。

新規ホストで zonepath を使用可能にする方法は複数存在するため、あるシステムから別のシステムへの zonepath の実際の移動は、大域管理者が手動で行います。

新規システムへの接続時に、ゾーンはインストール済みの状態になります。

▼ 非大域ゾーンを移行する方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 移行するゾーン (この手順では **my-zone**) を停止します。

   ```
   host1# zoneadm -z my-zone halt
   ```

3. ゾーンを切り離します。

   ```
   host1# zoneadm -z my-zone detach
   ```

   切り離されたゾーンは、現在、構成済みの状態にあります。

4. **my-zone** の **zonepath** を新規ホストに移動します

   詳細は、326 ページの「zonepath を新規ホストに移動する方法」を参照してください。

5. 新規ホスト上でゾーンを構成します

   ```
   host2# zonecfg -z my-zone
   ```

   次のシステムメッセージが表示されます

   ```
   my-zone: No such zone configured
   Use 'create' to begin configuring a new zone.
   ```

6 新規ホスト上にゾーン **my-zone** を作成するには、**zonecfg** コマンドに **-a** オプションおよび新規ホストの **zonepath** を指定します。

```
zonecfg:my-zone> create -a /export/zones/my-zone
```

7 (オプション) 構成を表示します。

```
zonecfg:my-zone> info
zonename: my-zone
zonepath: /export/zones/my-zone
autoboot: false
pool:
inherit-pkg-dir:
        dir: /lib
inherit-pkg-dir:
        dir: /platform
inherit-pkg-dir:
        dir: /sbin
inherit-pkg-dir:
        dir: /usr
net:
        address: 192.168.0.90
        physical: bge0
```

8 構成に必要な調整を加えます。

たとえば、新規ホストではネットワーク物理デバイスが異なる場合があります。また、構成に含まれるデバイスの名前が新規ホストでは異なることもあります。

```
zonecfg:my-zone> select net physical=bge0
zonecfg:my-zone:net> set physical=e1000g0
zonecfg:my-zone:net> end
```

9 構成を確定して終了します。

```
zonecfg:my-zone> commit
zonecfg:my-zone> exit
```

10 次のいずれかの方法で、ゾーンを新しいホストに接続します。

- 妥当性検査を使用して、ゾーンを接続します。

  ```
  host2# zoneadm -z my-zone attach
  ```

  次の条件のいずれかまたは両方に当てはまる場合、実行が必要な操作がシステム管理者に通知されます。

  - 必須パッケージおよびパッチが新規マシンに存在しない。
  - ソフトウェアレベルがマシン間で異なる。

- **Solaris 10 10/08:** 妥当性検査を使用してゾーンを接続します。また、より新しいバージョンのゾーン依存パッケージがホストで実行されている場合や、ホストのマシンクラスが異なっている場合は、ホストに一致するように接続時にゾーンを更新します。

  ```
  host2# zoneadm -z my-zone attach -u
  ```

ヒント – **Solaris 10 10/08:** 移行元システムで古いバージョンの Solaris システムが実行されている場合は、ゾーンを切り離す際に正しいパッケージリストが生成されないことがあります。移行先に正しいパッケージリストが確実に生成されるようにするには、`SUNWdetached.xml` ファイルを `zonepath` から削除します。このファイルを削除すると、移行先システムによって新しいパッケージリストが生成されます。

これは Solaris 10 5/09 以降のリリースでは必要ありません。

- **Solaris 10 9/10:** 妥当性検査を使用してゾーンを接続し、ゾーンのすべてのパッケージを、このホストに新しくインストールされた非大域ゾーンで表示される内容と一致するように更新します。ゾーン内にインストールされ、大域ゾーンにはインストールされていないパッケージがある場合、それらのパッケージは無視され、そのままの状態になります。

  ```
  host2# zoneadm -z my-zone attach -U
  ```

- **Solaris 10 5/09** 以降: **-b** オプションを使用して、指定したパッチを接続時にゾーンから除去します。指定できるパッチは、公式パッチまたは **IDR** パッチです。

  ```
  host2# zoneadm -z my-zone attach -u -b IDR246802-01 -b 123456-08
  ```

  `-b` オプションは、`-u` または `-U` オプションとは独立して使用できます。

- 妥当性検査を実行せずに、接続操作を強制的に実行します。

  ```
  host2# zoneadm -z my-zone attach -F
  ```

注意 – `-F` オプションを使用すると、妥当性検査を実行せずに `attach` が強制的に実行されます。これは、クラスタ環境やバックアップ/復元操作など、特定の場合に役立ちますが、システムがゾーンのホストとして動作するよう正しく構成されている必要があります。構成が不正な場合、あとで未定義の動作が実行される可能性があります。

▼ zonepath を新規ホストに移動する方法

`zonepath` のアーカイブの作成には、いくつもの方法があります。たとえば、`cpio` または `pax` コマンドを使用できます。詳細は、`cpio(1)` および `pax(1)` のマニュアルページを参照してください。

アーカイブを新規ホストに転送する方法も、複数存在します。`zonepath` を転送元ホストから転送先ホストに転送するメカニズムは、ローカルの構成によって異なります。SAN などのいくつかの場合には、`zonepath` データを実際には移動できないこともあります。SAN の場合は、`zonepath` が新規ホストに表示されるように、再構成が実行されるだけです。それ以外の場合は、`zonepath` をテープに書き込み、それを新規サイトに送付することもあります。

これらの理由のために、この手順は自動化されていません。システム管理者は、zonepath を新規ホストに移動する最適な手法を選択する必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **zonepath** を新規ホストに移動します。この手順で説明した方法を使用することも、別の方法を選んで使用することもできます。

例 23–1 tar コマンドを使用した zonepath のアーカイブおよび移動

1. host1 上で zonepath の tar ファイルを作成し、 sftp コマンドを使って host2 に転送します。

```
host1# cd /export/zones
host1# tar cf my-zone.tar my-zone
host1# sftp host2
Connecting to host2...
Password:
sftp> cd /export/zones
sftp> put my-zone.tar
Uploading my-zone.tar to /export/zones/my-zone.tar
sftp> quit
```

2. host2 上で tar ファイルを展開します。

```
host2# cd /export/zones
host2# tar xf my-zone.tar
```

詳細は、sftp(1) および tar(1) を参照してください。

注意事項 次の情報のトラブルシューティングについては、444 ページの「zoneadm 接続操作の問題解決」を参照してください。

- パッチおよびパッケージが同期しない。
- オペレーティングシステムのリリースが一致しない。

次の手順 SAN を再構成せず、データをコピーした場合、ゾーンが構成済みの状態になっても zonepath のデータはソースホスト上に表示されたままになってます。データを新しいホストに移行し終わったあと手動で zonepath をソースホストから削除するか、ソースホストにゾーンを再接続し、zoneadm uninstall コマンドを使って zonepath を削除することができます。

Solaris 10 5/08: 移行を行う前のゾーンの移行の検証について

「no execute」(実行しない) オプションである -n を使用することで、ゾーンを新しいマシンに移行する前に試行を行うことができます。

-n オプションを指定して zoneadm detach サブコマンドを使用すると、実際にゾーンを切り離さずに実行中のゾーンでマニフェストを生成できます。移行元のシステムのゾーンの状態は変わりません。ゾーンのマニフェストは標準出力に送信されます。大域管理者は、この出力をファイルに送ったり、移行先ホストですぐに検証されるようにリモートコマンドにパイプしたりできます。-n オプションを指定して zoneadm attach サブコマンドを使用すると、このマニフェストを読み取り、実際に接続を行わずに、移行先のマシンがゾーンのホストとして機能するための適正な構成を保持しているかどうかを確認できます。

試行接続を行う前に、新規ホストで移行先システムのゾーンを構成する必要はありません。

▼ Solaris 10 5/08: 移行を行う前にゾーンの移行を検証する方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 次のいずれかを実行します。

   - **my-zone** の移行元ホストでマニフェストを生成し、移行先ホストをすぐに検証するリモートコマンドにその出力をパイプします。

     ```
     global# zoneadm -z my-zone detach -n | ssh remotehost zoneadm attach -n -
     ```

     行の最後にあるハイフン (–) は、パスとして標準入力を指定します。

     検証内容は、移行元ホストの画面である stdout に出力されます。

   - **my-zone** の移行元ホストでマニフェストを生成し、その出力をファイルに送ります。

     ```
     global# zoneadm -z my-zone detach -n > filename
     ```

326 ページの「zonepath を新規ホストに移動する方法」の説明に従って、そのマニフェストを新しいホストシステムにコピーし、検証を行います。

```
global# zoneadm attach -n path_to_manifest
```

パスを – にすると標準入力を指定できます。

使用できないマシンからゾーンを移行する

ネイティブ Solaris ゾーンをホストするマシンが使用できなくなる場合があります。しかし、ゾーンが格納されている SAN などのストレージがまだ使用できる場合は、ゾーンを新しいホストに正常に移行できる可能性があります。ゾーンの zonepath を新しいホストに移動できます。SAN などのいくつかの場合には、zonepath データを実際には移動できないこともあります。SAN の場合は、zonepath が新規ホストに表示されるように、再構成が実行されるだけです。ゾーンが適切に切り離されなかったため、初めに zonecfg コマンドを使って新しいホストにゾーンを作成する必要があります。一度これを行なったあと、新しいホストでゾーンを接続してください。新しいホストから、ゾーンが適切に切り離されなかったというメッセージが出ますが、それでもシステムは接続を試行します。

このタスクの手順は、324 ページの「非大域ゾーンを移行する方法」の手順 4 - 8 で説明されています。326 ページの「zonepath を新規ホストに移動する方法」も参照してください。

パッチ適用のソリューションとして、接続時更新を使用する

接続時の更新のプロセスは、ゾーンを別のシステムに移行するために開発されたものですが、ゾーンにパッチを適用する場合にも使用できます。この方法を利用すると、大域ゾーンをより迅速に使用可能な状態にすることができます。その後、システム管理者は、あまり重要でないゾーンを更新してブートする前に、どのゾーンを先に更新して稼働させるかを調整できます。

次のプロセスでは、そのゾーンがシステムに新しくインストールされたゾーンと同じように見えるように、すべてのパッチを更新します。

1. 大域ゾーンにパッチバンドルを適用する前に、すべての非大域ゾーンを切り離します。
2. 大域ゾーンにパッチバンドルを適用します。
3. パッチバンドルが適用され、システムがリブートしたら、zoneadm attach コマンドを -u オプションを付けて使用し、非大域ゾーンを大域ゾーンと同じパッチレベルにします。

ゾーン内にインストールされ、大域ゾーンにはインストールされていない
パッケージがある場合、それらのパッケージは無視され、影響を受けません。

`patchadd` ユーティリティーを使用した高速のパッチ適用ソリューションについては、361 ページの「Oracle Solaris 10 10/09: パッチ適用時間を短縮するためのゾーンの並列パッチ」を参照してください。

◆ ◆ ◆ 第 24 章

Oracle Solaris 10 9/10: ゾーンへの物理的な Oracle Solaris システムの移行 (タスク)

既存の Oracle Solaris 10 システムを Oracle Solaris 10 ターゲットシステム上のネイティブゾーンに移行するには、P2V (Physical to Virtual) 機能を使用します。Oracle Solaris 10 システムを、Oracle Solaris 11 リリース上で使用可能な `solaris10` ブランド非大域ゾーンに移行するには、『Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management』を参照してください。

zonep2vchk ユーティリティーを使用したシステムへのアクセス

`zonep2vchk` ユーティリティーを使用して、別の Oracle Solaris 10 ホスト上のゾーンへの移行のための Oracle Solaris 10 ホストの評価と、`zonecfg` テンプレートの作成を行います。このユーティリティーは、移行が始まる前にソースシステム上で実行されます。ユーティリティーには次の機能があります。

- 使用中のネットワーク、ストレージ、およびオペレーティングシステム機能などの、Oracle Solaris 構成の解析
- アプリケーションバイナリの解析
- 実行中のアプリケーションの解析
- ターゲットシステムで使用する `zonecfg` テンプレート `zonecfg` コマンドファイルの生成。ゾーンはソースシステムの構成と一致します。

`zonep2vchk` ユーティリティーの説明は、`zonep2vchk(1M)` のマニュアルページで説明されています。

Oracle Solaris 10 1/13: zonep2vchk ユーティリティーの取得

`zonep2vchk` ユーティリティーは Oracle Solaris 10 1/13 システム上で使用できます。

Oracle Solaris 10 の以前のバージョンでこのユーティリティーを使用する場合は、OTN (http://www.oracle.com/technetwork/server-storage/solaris10/downloads) からアンバンドルのパッケージをダウンロードできます。

注 – アンバンドルのパッケージを追加しても、その後システムをアップグレードするか、またはシステムにパッチを適用すれば、Oracle Solaris 10 1/13 によって提供されたバージョンと干渉しません。アンバンドル版は、/opt/SUNWzonep2vchk にインストールされます。Oracle Solaris 10 1/13 のアップグレードまたはパッチにより、/usr/sbin にバンドル版が追加されます。以前に取得したアンバンドルのパッケージはアンインストールできます。

移行に関するその他の考慮事項

元の Oracle Solaris 10 システムで実行されていたサービスに応じて、大域管理者は、ゾーンがインストールされたあとに新規ホスト上のゾーンを手動でカスタマイズすることが必要になる場合があります。たとえば、ゾーンに割り当てられた特権の変更が必要になる場合があります。この操作は自動的に実行されることはありません。また、すべてのシステムサービスがゾーン内で動作するとは限らないため、すべての物理システムがゾーンへの移行に適しているわけではありません。

P2V でインストールする元のソースシステムイメージが、移行先ホストのオペレーティングシステムリリースよりもあとのリリースの場合、インストールは失敗します。

Oracle Solaris システムをゾーンに直接移行するために使用するイメージの作成

インストール済みの Solaris システムのイメージを Flash アーカイブツールを使用して作成し、ゾーンに直接移行することができます。

イメージを作成する前に、ゾーンで実行するすべてのソフトウェアを含む完全なシステムを構成できます。その後、このイメージは、ゾーンのインストール時にインストールプログラムによって使用されます。

注意 – ZFS ルートを持つ Oracle Solaris 10 システムの Oracle Solaris フラッシュアーカイブ (flar) を作成する場合、デフォルトでは、flar は実際には ZFS send ストリームであり、これを使用してルートプールを再作成できます。このイメージを使用して Oracle Solaris 10 リリース上にゾーンをインストールすることはできません。システムに ZFS ルートがある場合は、明示的な cpio または pax アーカイブによる flar を作成する必要があります。

flarcreate コマンドを -L *archiver* オプションとともに使用し、ファイルのアーカイブ方法として cpio または pax を指定します。次の手順の 4 を参照してください。

▼ flarcreate を使用してイメージを作成する方法

flarcreate コマンド (flarcreate(1M) のマニュアルページを参照) を使用してシステムイメージを作成します。この手順例では、ターゲットの Oracle Solaris システムにフラッシュアーカイブを配置するために NFS を使用しますが、ほかの方法でファイルを移動することもできます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
2. アーカイブする移行元システムにログインします。
3. ディレクトリをルートディレクトリに変更します。

   ```
   # cd /
   ```

4. 移行元システムで **flarcreate** を使用して **s10-system** という名前のフラッシュアーカイブイメージファイルを作成し、そのアーカイブを移行先システムに配置します。

   ```
   source-system # flarcreate -S -n s10-system -L cpio /net/target/export/s10-system.flar
   Determining which filesystems will be included in the archive...
   Creating the archive...
   cpio: File size of "etc/mnttab" has
   increased by 435
   2068650 blocks
   1 error(s)
   Archive creation complete.
   ```

   移行先のマシンは、/export ファイルシステムに対する root 書き込みアクセス権を必要とします。ホストシステム上のファイルシステムのサイズによっては、アーカイブが数 G バイトのサイズになることがあるため、移行先のファイルシステムには十分な空き容量が必要です。

ヒント – 場合によっては、flarcreate の実行時に cpio コマンドからのエラーが表示されることがあります。もっとも多いのは、「File size of etc/mnttab has increased by 435」のようなメッセージです。これらのメッセージがログファイルまたはシステム状態を反映するファイルに関連するものであれば、無視してもかまいません。必ずすべてのエラーメッセージを確認してください。

ほかのアーカイブ作成方法

別の方法を使用してアーカイブを作成することもできます。インストーラは次のアーカイブフォーマットを受け入れることができます。

- cpio アーカイブ
- gzip で圧縮された cpio アーカイブ
- bzip2 で圧縮された cpio アーカイブ
- -x xustar (XUSTAR) 形式で作成された pax アーカイブ
- ufsdump レベル 0 (完全) バックアップ

インストーラは、ファイルのアクセス権、所有権、およびリンクを保存および復元するアーカイブユーティリティーを使用して作成されたファイルのディレクトリのみを受け入れることができます。

詳細は、cpio(1)、pax(1)、bzip2 (1)、gzip(1)、および ufsdump(1M) のマニュアルページを参照してください。

注 – フラッシュアーカイブ以外の方法を使用して P2V 用アーカイブを作成する場合は、アーカイブを作成する前に、ソースシステム上でプロセッサ依存の libc.so.1 の、lofs でマウントされたハードウェア機能 (hwcap) ライブラリをマウント解除する必要があります。それらをマウント解除しなければ、そのアーカイブを使用してインストールしたゾーンがターゲットシステムでブートしない可能性があります。アーカイブの作成後に、lofs と mount -O オプションを使用して、/lib/libc.so.1 上に適切なハードウェア機能ライブラリを再マウントできます。

```
source-system# unmount /lib/libc.so.1
source-system# mount -O -F lofs /lib/libc.so.1
```

ホスト ID のエミュレーション

物理的な Oracle Solaris システムから新しいシステム上のゾーンにアプリケーションを移行すると、hostid は新しいマシンの hostid に変更されます。

場合によっては、アプリケーションが元の hostid に依存しており、アプリケーション構成を更新できないことがあります。このような場合は、元のシステムの hostid を使用するようにゾーンを構成することができます。そのためには、266 ページの「ゾーンの構成方法」の説明に従って、zonecfg プロパティーを設定して hostid を指定します。値としては、元のシステムで hostid コマンドを実行した場合の出力を使用してください。インストール済みゾーンで hostid を表示する場合も、hostid コマンドを使用します。

ホスト ID の詳細は、hostid(1) のマニュアルページを参照してください。

ゾーンの構成

zonep2vchk ユーティリティーによって作成されたテンプレートの zonecfg コマンドファイルを使用して、ターゲットシステム上に新規ゾーン構成を作成します。266 ページの「ゾーンの構成方法」の手順も参照してください。

ヒント – CD または DVD を使用してアプリケーションを新しいゾーンにインストールする場合は、ブランドゾーンを最初に構成するときに add fs を使用して、大域ゾーンの CD または DVD メディアに読み取り専用のアクセスを行う権限を追加します。アクセス権を追加したら、CD または DVD を使用して製品をブランドゾーンにインストールできます。詳細は、421 ページの「非大域ゾーンで CD または DVD メディアにアクセスする権限を追加する方法」を参照してください。

ゾーンのインストール

zoneadm コマンド (パート II「ゾーン」および zoneadm(1M) のマニュアルページを参照) は、非大域ゾーンをインストールおよび管理するための主要なツールです。zoneadm を使用する操作は、移行先システムの大域ゾーンから実行する必要があります。

インストールプロセスは、アーカイブからファイルを展開するほかに、ゾーンがホスト上で最適に実行されることを保証するために、検査や必要な後処理などの機能を実行します。

ゾーンで実行されるすべてのソフトウェアを備えた、完全に構成済みの Oracle Solaris システムのイメージを使用できます。

既存のシステムから Oracle Solaris システムアーカイブを作成した場合は、ゾーンをインストールするときに -p (sysidcfg を維持する) オプションを使用すると、イメージの作成に使用されたシステムと同じ ID がゾーンに設定されます。

-u (sys-unconfig) オプションを使用して移行先にゾーンをインストールすると、ホスト名やネームサービスの構成されていないゾーンが作成されます。

注意 – -p オプションか -u オプションのどちらかを指定する必要があります。どちらも指定しない場合、エラーが発生します。

インストーラオプション

オプション	説明
-a *archive*	システムイメージのコピー元となるアーカイブの場所。完全なフラッシュアーカイブと cpio、gzip で圧縮された cpio、bzip で圧縮された cpio、およびレベル 0 ufsdump がサポートされています。SUNWsfman パッケージに用意されている gzip のマニュアルページを参照してください。
-d *path*	システムイメージのコピー元となるディレクトリの場所。
-d –	-d オプションとダッシュ (-) パラメータを使用して、zonepath で既存のディレクトリレイアウトが使用されるように指定します。このため、インストールの前に管理者が手動で zonepath ディレクトリを設定する場合に、-d – オプションを使用してそのディレクトリがすでに存在するかどうかを示すことができます。
-p	システム ID を維持します。
-s	サイレントインストールします。
-u	ゾーンに対して sys-unconfig を実行します。
-v	詳細情報を出力します。
-b*patchid*	システムイメージにインストールされているパッチのパッチ ID を指定するために、1 つ以上の -b オプションを使用できます。これらのパッチは、インストールプロセス中にバックアウトされます。

-a オプションと -d オプションは相互に排他的です。-p、-s、-u、および -v の各オプションは、-a または -d が指定されている場合にのみ使用できます。

▼ ゾーンのインストール方法

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

2 **zoneadm** コマンドに **install -a** オプションとアーカイブのパスを指定して、構成済みゾーン **s-zone** をインストールします。

```
global# zoneadm -z s-zone install -u -a /net/machine_name/s-system.flar
```

インストールの完了につれてさまざまなメッセージが表示されます。これにはしばらく時間がかかることがあります。

インストールが完了したら、`list` サブコマンドに `-i` オプションおよび `-v` オプションを指定してインストール済みのゾーンを一覧表示し、ステータスを確認します。

注意事項 インストールが失敗した場合は、ログファイルを確認してください。成功した場合、ログファイルはゾーン内の `/var/log` にあります。失敗した場合、ログファイルは大域ゾーン内の `/var/tmp` にあります。

ゾーンのインストールが中断または失敗した場合は、ゾーンの状態は不完全なままになります。`uninstall -F` を使用して、ゾーンを構成済みの状態にリセットします。

ゾーンのブート

▼ ゾーンのブート方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

`-u` オプションを使用した場合は、さらに、`zlogin` でゾーンコンソールにログインしてシステム構成を実行する必要があります。この操作については312 ページの「初期内部ゾーン構成を実行する」を参照してください。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

2 **zoneadm** コマンドを **-z** オプション、ゾーン名 **(s-zone)**、および **boot** サブコマンドとともに使用することで、ゾーンをブートします。

```
global# zoneadm -z s-zone boot
```

3 ブートが完了したら、**list** サブコマンドに **-v** オプションを指定してステータスを確認します。

```
global# zoneadm list -v
```

◆ ◆ ◆ 第 25 章

ゾーンがインストールされている Oracle Solaris システムでのパッケージとパッチについて (概要)

Oracle Solaris 10 1/06: この章は、全面的に改訂されています。

この章では、ゾーンがインストールされるときに、Oracle Solaris オペレーティングシステムを維持する方法について説明します。大域ゾーンおよびインストール済みのすべての非大域ゾーン内で、パッケージやパッチをオペレーティングシステムに追加する際の情報を提供します。パッケージおよびパッチの削除に関する情報も含まれます。この章の内容は、既存の Oracle Solaris インストールおよびパッチに関するドキュメントの内容を補うものです。詳細は、Oracle Solaris 10 Release and Installation Collection - Japanese および『Solaris のシステム管理 (基本編)』を参照してください。

この章で扱う内容は、次のとおりです。

ゾーンがインストールされているときのパッケージとパッチの操作について追加された説明

パッチのソフトウェアダウンロードサイトは、My Oracle Support (`https://support.oracle.com`) です。「パッチと更新 (Patches & Updates)」タブをクリックします。そのサイトでは、ダウンロード手順を確認し、イメージをダウンロードすることができます。パッチに関する追加情報については、サポートプロバイダにお問い合わせください。

Oracle Solaris 10 1/06: この章は、非大域ゾーンがインストールされているシステム上でのパッケージコマンドとパッチコマンドの最新の動作を説明するために、Oracle Solaris 10 以降に改訂されています。

Oracle Solaris 10 6/06: `SUNW_PKG_ALLZONES`、`SUNW_PKG_HOLLOW`、および `SUNW_PKG_THISZONE` パッケージパラメータに関する情報が改訂されています。341 ページの「パッケージツールとパッチツールの概要」および 350 ページの「パッケージパラメータの情報」を参照してください。

Oracle Solaris 10 8/07 以降のリリース:

- `pkgadd` コマンドに `-G` オプションを付けてパッケージをインストールした場合、`patchadd` コマンドを使用してそのパッケージにパッチを追加するときに、`patchadd` に `-G` オプションを付ける必要はなくなりました。
- さまざまな状態の非大域ゾーンが存在しているシステムで、`pkgadd`、`pkgrm`、`patchadd`、および `patchrm` の各コマンドを使用するとどうなるかを示す表が追加されました。345 ページの「ゾーン状態がパッチとパッケージの操作に与える影響」を参照してください。
- patchadd `-G` と `pkginfo` 変数との相互作用に関する説明が追加されました。363 ページの「ゾーンが含まれているシステムでの patchadd `-G` と `pkginfo` 変数の相互作用」を参照してください。
- 遅延起動パッチに関する情報が追加されました。360 ページの「Oracle Solaris 10 8/07: 遅延起動パッチ」を参照してください。
- `pkgrm` コマンドの `-G` オプションに関する情報が削除されました。

Oracle Solaris 10 10/09: ゾーンの並列パッチは、標準の Oracle Solaris 10 パッチユーティリティーの拡張機能です。Oracle Solaris 10 10/09 より前のリリースでは、このパッチは、119254-66 以降のリビジョン (SPARC) および 119255-66 以降のリビジョン (x86) のパッチユーティリティーのパッチで提供されます。361 ページの「Oracle Solaris 10 10/09: パッチ適用時間を短縮するためのゾーンの並列パッチ」および372 ページの「Oracle Solaris 10 10/09: 非大域ゾーンに並列でパッチを適用する方法」を参照してください。329 ページの「パッチ適用のソリューションとして、接続時更新を使用する」も参照してください。ゾーンを持つシステムでパッチをすばやく更新するには、この方法をお勧めします。

Oracle Solaris 10 の新機能の全一覧および Oracle Solaris リリースについての説明は、http://www.oracle.com/webfolder/technetwork/hcl/index.html の『*Oracle Solaris OS: Hardware Compatibility List*』を参照してください。

パッケージツールとパッチツールの概要

Oracle Solaris パッケージツールはゾーン環境の管理に使用されます。大域管理者は、システムを Oracle Solaris の新しいバージョンに更新でき、大域ゾーンと非大域ゾーンの両方を更新します。

大域ゾーンで Oracle Solaris Live Upgrade、Oracle Solaris 標準の対話型インストールプログラム、またはカスタム JumpStart インストールプログラムを使用して、非大域ゾーンが含まれているシステムをアップグレードできます。`zonepath` が ZFS 上に設定されているゾーンには、次の制限が適用されます。

- `zonepath` が ZFS 上に設定されているシステムで Oracle Solaris Live Upgrade がサポートされるのは、Oracle Solaris 10 10/08 リリース以降です。
- システムのアップグレードに使用できるのは Oracle Solaris Live Upgrade だけです。

詳細は、『Oracle Solaris ZFS 管理ガイド』の「ゾーンが含まれているシステムを Live Upgrade を使用して移行またはアップグレードする (Solaris 10 10/08)」を参照してください。

ゾーン管理者は、このドキュメントに記載されている制限の範囲内でパッケージツールを使用して、非大域ゾーンにインストールされたすべてのソフトウェアを管理できます。

ゾーンがインストールされている場合は、次の一般的な指針が適用されます。

- 大域管理者は、システムのすべてのゾーン内のソフトウェアを管理できます。
- 非大域ゾーンのルートファイルシステムは、Oracle Solaris パッケージツールおよびパッチツールを使用することで、大域ゾーンから管理できます。Oracle Solaris パッケージツールおよびパッチツールは、共通パッケージ (バンドル) 製品、スタンドアロン (別パッケージ) 製品、および他社製製品を管理するために、非大域ゾーン内でサポートされます。
- パッケージツールおよびパッチツールは、ゾーン対応の環境で動作します。このツールを使用すると、大域ゾーンにインストールされたパッケージやパッチを非大域ゾーンにもインストールできます。
- `SUNW_PKG_ALLZONES` パッケージパラメータにより、パッケージの「ゾーン範囲」が定義されます。この範囲により、個別のパッケージをインストール可能なゾーンの種類が決まります。このパラメータの詳細は、354 ページの「`SUNW_PKG_ALLZONES` パッケージパラメータ」を参照してください。

- あるパッケージをすべてのゾーンにインストールする必要があり、そのパッケージがすべてのゾーンで同一でなければならない場合、SUNW_PKG_HOLLOW パッケージパラメータはそのパッケージの「可視性」を定義します。このパラメータの詳細については、356 ページの「SUNW_PKG_HOLLOW パッケージパラメータ」を参照してください。
- SUNW_PKG_THISZONE パッケージパラメータは、あるパッケージを現在のゾーンだけにインストールする必要があるかどうかを定義します。このパラメータについては、358 ページの「SUNW_PKG_THISZONE パッケージパラメータ」を参照してください。
- ゾーンのパッケージパラメータの値が定義されていないパッケージでは、デフォルトの設定は false です。
- 非大域ゾーン内から可視であるパッケージ情報は、Oracle Solaris パッケージツールおよびパッチツールを使用してそのゾーンにインストールされたファイルと一致します。このパッケージ情報は、inherit-pkg-dir ディレクトリと同期しています。
- 大域ゾーンに追加されたパッチ、パッケージなどの変更は、すべてのゾーンに適用できます。この機能により、大域ゾーンと各非大域ゾーン間の一貫性が維持されます。
- パッケージコマンドを使用して、パッケージの追加、削除、および調査を実行できます。パッチコマンドを使用して、パッチを追加および削除できます。

注 – 特定のパッケージおよびパッチ操作が行われている間、ゾーンはこの種のほかの操作から一時的にロックされます。システムはまた、要求された操作について管理者に確認してから処理を続行することがあります。

パッケージとゾーンについて

非大域ゾーンのインストール時に、大域ゾーンにインストールされた Oracle Solaris パッケージの一部だけが完全にレプリケートされます。たとえば、非大域ゾーン内では、Oracle Solaris カーネルを含む多数のパッケージは必要ありません。すべての非大域ゾーンは、大域ゾーンと同一の Oracle Solaris カーネルを暗黙的に共有します。ただし、パッケージのデータが不要であるか、非大域ゾーンでは使用されない場合でも、パッケージが大域ゾーンにインストールされるという情報が非大域ゾーン内で必要になる場合があります。この情報を使用すると、非大域ゾーンをソースとするパッケージの依存関係を大域ゾーンで適切に解決できます。

パッケージが保持するパラメータにより、非大域ゾーンのインストールされたシステムで内容を配布および可視にする方法が制御されます。SUNW_PKG_ALLZONES、SUNW_PKG_HOLLOW、および SUNW_PKG_THISZONE パッケージパラメータは、ゾーンがインストールされているシステムでのパッケージの特性を定義します。ゾーン環境内でパッケージを追加または削除する際、システム管理者

は、必要に応じてこれらのパッケージパラメータ設定を検査してパッケージの適用範囲を確認できます。pkgparam コマンドを使用して、これらのパラメータの値を表示できます。パラメータの詳細については、350 ページの「パッケージパラメータの情報」を参照してください。使用方法については、374 ページの「ゾーンがインストールされているシステムでのパッケージパラメータ設定の検査」を参照してください。

パッケージの特性およびパラメータの詳細は、pkginfo(4) のマニュアルページを参照してください。パッケージのパラメータ値の表示については、pkgparam(1) のマニュアルページを参照してください。

パッケージ用に生成されるパッチ

パッケージ用にパッチを生成するときは、パラメータを元のパッケージと同じ値に設定する必要があります。

対話型パッケージ

対話型であることが必要とされるパッケージ、つまり要求スクリプトを含むパッケージは、現在のゾーンだけに追加されます。パッケージはほかのゾーンには伝達されません。大域ゾーンに対話型パッケージを追加すると、そのパッケージは、pkgadd コマンドに -G オプションを指定して追加する場合と同様に処理されます。このオプションの詳細については、346 ページの「ゾーン内でのパッケージの追加について」を参照してください。

ゾーンの同期を維持する

非大域ゾーンにインストールされたソフトウェアと、大域ゾーンにインストールされたソフトウェアとの同期を最大限維持するのが最善です。これにより、複数のゾーンがインストールされたシステムで、管理の問題が発生するのを最小限に抑えることができます。

この目標を達成するために、大域ゾーン内でパッケージを追加または削除する際、次の規則がパッケージツールにより適用されます。

大域ゾーン内で実行可能なパッケージ操作

現在、パッケージが大域ゾーンにも、どの非大域ゾーンにもインストールされていない場合、パッケージは次の場所にインストールできます。

- SUNW_PKG_ALLZONES=false の場合、大域ゾーンのみ

- SUNW_PKG_THISZONE=true の場合、現在の (大域) ゾーンのみ
- 大域ゾーンおよびすべての非大域ゾーン

現在、パッケージが大域ゾーンだけにインストールされている場合:

- すべての非大域ゾーンにパッケージをインストールできます。
- パッケージを大域ゾーンから削除できます。

現在、パッケージが大域ゾーンおよび非大域ゾーンの一部だけにインストールされている場合:

- SUNW_PKG_ALLZONES が false に設定されている必要があります。
- すべての非大域ゾーンにパッケージをインストールできます。すべての非大域ゾーン内の既存のインスタンスは、インストール中のバージョンに更新されます。
- パッケージを大域ゾーンから削除できます。
- パッケージを大域ゾーンおよびすべての非大域ゾーンから削除できます。

現在、パッケージが大域ゾーンおよびすべての非大域ゾーンにインストールされている場合、そのパッケージを大域ゾーンおよびすべての非大域ゾーンから削除できます。

これらの規則により、次のことが保証されます。

- 大域ゾーンにインストールされたパッケージは、大域ゾーンだけにインストールされているか、大域ゾーンとすべての非大域ゾーンにインストールされています。
- 大域ゾーンとすべての非大域ゾーンの両方にインストールされているパッケージは、すべてのゾーンで同一です。

非大域ゾーン内で実行可能なパッケージ操作

すべての非大域ゾーン内で実行可能なパッケージ操作を次に示します。

- 現在、パッケージがその非大域ゾーンにインストールされていない場合、SUNW_PKG_ALLZONES=false の場合にだけ、パッケージをインストールできます。
- SUNW_PKG_THISZONE=true の場合、パッケージを現在の (非大域) ゾーンにインストールできます。
- 現在、パッケージがその非大域ゾーンにインストールされている場合:
  - SUNW_PKG_ALLZONES=false である場合にだけ、パッケージの既存のインスタンス上にパッケージをインストールできます。
  - SUNW_PKG_ALLZONES=false である場合にだけ、パッケージをその非大域ゾーンから削除できます。

ゾーン状態がパッチとパッケージの操作に与える影響

さまざまな状態の非大域ゾーンが存在しているシステムで、pkgadd、pkgrm、patchadd、および patchrm の各コマンドを使用するとどうなるかを、次の表に示します。

Oracle Solaris 10 5/08 リリースでは、この表のインストール済み状態の説明にいくつかの修正が加えられました。

ゾーンの状態	パッケージとパッチの操作に与える影響
構成済み	パッチツールとパッケージツールを実行できます。ソフトウェアはまだインストールされていません。
インストール済み	パッチツールとパッケージツールを実行できます。パッチとパッケージの操作中、システムはゾーンをインストール済み状態から「マウント済み」と呼ばれる新しい内部状態に移行します。パッチの適用が完了したあとで、ゾーンはインストール済み状態に戻されます。 zoneadm -z *zonename* install が完了した直後にも、ゾーンはインストール済み状態に移行されます。今までにブートしたことのないインストール済み状態のゾーンは、パッチを適用することも、パッケージコマンドを実行することもできません。少なくとも 1 度はゾーンをブートして稼働状態にする必要があります。ゾーンを少なくとも一度ブートして、その後 zoneadm halt によってインストール済み状態に戻したあとは、パッチおよびパッケージのコマンドを実行できます。
準備完了	パッチツールとパッケージツールを実行できます。
稼働	パッチツールとパッケージツールを実行できます。
不完全	zoneadm によってインストール中または削除中のゾーンです。パッチツールとパッケージツールは使用できません。ツールでは、ツールを使用できる適切な状態にゾーンを移行させることはできません。

ゾーン内でのパッケージの追加について

ゾーンがインストールされている Oracle Solaris システムにパッケージが追加するには、pkgadd(1M) のマニュアルページに記載されている pkgadd システムユーティリティーが使用されます。

大域ゾーン内での pkgadd の使用

大域ゾーンで -G オプションを指定して pkgadd ユーティリティーを使用すると、大域ゾーンだけにパッケージを追加できます。パッケージはほかのゾーンには伝達されません。SUNW_PKG_THISZONE=true の場合、-G オプションは不要です。SUNW_PKG_THISZONE=false の場合、-G オプションがオーバーライドします。

大域ゾーン内で pkgadd ユーティリティーを実行する場合、次の操作が適用されます。

- pkgadd ユーティリティーを使用して、パッケージを次の場所に追加できます。
  - 大域ゾーンのみ (パッケージが SUNW_PKG_ALLZONES=true である場合を除く)
  - 大域ゾーンとすべての非大域ゾーン
  - すべての非大域ゾーンのみ (パッケージが大域ゾーンにインストール済みの場合)
  - 現在のゾーンのみ (SUNW_PKG_THISZONE=true の場合)
- pkgadd ユーティリティーを使用して、パッケージを次の場所に追加することはできません。
  - 非大域ゾーンの一部
  - すべての非大域ゾーン (パッケージが大域ゾーンにインストール済みの場合を除く)
- pkgadd ユーティリティーを -G オプションを指定せずに実行し、かつ SUNW_PKG_THISZONE=false であった場合、指定されたパッケージはデフォルトですべてのゾーンに追加されます。パッケージに「大域ゾーンだけにインストール」されているというマークが付けられることはありません。
- pkgadd ユーティリティーを -G オプションを指定せずに実行し、かつ SUNW_PKG_THISZONE=true であった場合、指定されたパッケージはデフォルトで現在のゾーン (大域ゾーン) に追加されます。パッケージには「大域ゾーンだけにインストール」されているというマークが付けられます。
- -G オプションを指定する場合、pkgadd ユーティリティーは指定されたパッケージを大域ゾーンだけに追加します。パッケージには「大域ゾーンだけにインストール」されているというマークが付けられます。いずれかの非大域ゾーンがインストールされる場合、パッケージはインストールされません。

パッケージを大域ゾーンとすべての非大域ゾーンに追加する

パッケージを大域ゾーンおよびすべての非大域ゾーンに追加するには、大域ゾーンで pkgadd ユーティリティーを実行します。大域管理者として、-G オプションを指定せずに pkgadd を実行します。

パッケージを大域ゾーンおよびすべての非大域ゾーンに追加する際、パッケージの影響を受ける領域について意識する必要はありません。

pkgadd ユーティリティーにより、次の手順が実行されます。

- 大域ゾーンおよび非大域ゾーン上でパッケージの依存関係が検査されます。必須のパッケージがどのゾーンにもインストールされていない場合、依存性の検査は失敗します。システムから大域管理者に対し、作業を続行するかどうかの問い合わせが行われます。
- パッケージが大域ゾーンに追加されます。
- 大域ゾーンのパッケージデータベースが更新されます。
- パッケージが各非大域ゾーンに追加され、大域ゾーン内のデータベースが更新されます。
- 各非大域ゾーンのパッケージデータベースが更新されます。

パッケージを大域ゾーンだけに追加する

パッケージを大域ゾーンだけに追加するには、大域ゾーンの大域管理者になり、-G オプションだけを指定して pkgadd ユーティリティーを実行します。

次の条件を両方満たす場合に、パッケージを大域ゾーンに追加できます。

- パッケージの内容が、いずれかの非大域ゾーンと共有されている大域ゾーンのどの領域にも影響を及ぼさない。
- パッケージに SUNW_PKG_ALLZONES=false が設定されている。

pkgadd ユーティリティーにより、次の手順が実行されます。

- パッケージの内容がいずれかの非大域ゾーンと共有されている大域ゾーンのいずれかの領域に影響を及ぼす場合、またはパッケージに SUNW_PKG_ALLZONES=true が設定されている場合、pkgadd は失敗します。パッケージを大域ゾーンおよびすべての非大域ゾーンに追加する必要があることを示すエラーメッセージが表示されます。
- 大域ゾーンだけでパッケージの依存関係が検査されます。必須のパッケージがインストールされていない場合、依存性の検査は失敗します。システムから大域管理者に対し、作業を続行するかどうかの問い合わせが行われます。
- パッケージが大域ゾーンに追加されます。
- 大域ゾーンのパッケージデータベースが更新されます。

- 大域ゾーンのパッケージ情報に、このパッケージが大域ゾーンだけにインストールされることを示す注釈が付けられます。将来、非大域ゾーンがインストールされる場合、このパッケージはインストールされません。

大域ゾーンにインストールされているパッケージをすべての非大域ゾーンに追加する

大域ゾーンにすでにインストールされているパッケージをすべての非大域ゾーンに追加するには、大域ゾーンからそのパッケージを削除し、すべてのゾーンに再インストールする必要があります。

次の手順で、大域ゾーンにすでにインストールされているパッケージをすべての非大域ゾーンに追加します。

1. 大域ゾーンで、`pkgrm` を使ってパッケージを削除します。
2. `-G` オプションを指定せずにパッケージを追加します。

非大域ゾーン内での **pkgadd** の使用

指定された非大域ゾーン内でパッケージを追加するには、ゾーン管理者として、オプションを指定せずに `pkgadd` ユーティリティーを実行します。次のすべての条件が適用されます。

- `pkgadd` ユーティリティーは、それが使用される非大域ゾーン内だけでパッケージを追加できます。
- 大域ゾーンから共有されるゾーンのどの領域に対しても、パッケージは操作を行うことはできません。
- パッケージに `SUNW_PKG_ALLZONES=false` が設定されていなければなりません。

`pkgadd` ユーティリティーにより、次の手順が実行されます。

- パッケージの追加前に、非大域ゾーンのパッケージデータベース上でパッケージの依存関係が検査されます。必須のパッケージがインストールされていない場合、依存性の検査は失敗します。システムから非大域ゾーン管理者に対し、作業を続行するかどうかの問い合わせが行われます。次のいずれかが当てはまる場合、検査は失敗します。
  - パッケージのいずれかのコンポーネントが、大域ゾーンから共有されているゾーンのいずれかの領域に影響を及ぼす。
  - パッケージに `SUNW_PKG_ALLZONES=true` が設定されている。
- パッケージがゾーンに追加されます。
- ゾーンのパッケージデータベースが更新されます。

ゾーン内でのパッケージの削除について

pkgrm(1M) のマニュアルページに記載されている pkgrm ユーティリティーは、ゾーンがインストールされている Oracle Solaris システムでのパッケージの削除をサポートします。

大域ゾーン内での pkgrm の使用

大域ゾーンで pkgrm ユーティリティーを使用する場合、次の操作が適用されます。

- pkgrm は、大域ゾーンとすべての非大域ゾーンを対象として、またはパッケージが大域ゾーンだけにインストールされている場合は、大域ゾーンだけを対象としてパッケージを削除できます。
- パッケージが非大域ゾーンにもインストールされている場合、pkgrm は大域ゾーンだけからパッケージを削除できません。削除を実行すると、非大域ゾーンの一部からパッケージが削除されます。

次の条件が満たされる場合、ゾーン管理者は、管理する非大域ゾーンを対象としてパッケージの削除だけを実行できます。

- 大域ゾーンから共有される非大域ゾーン内のどの領域に対しても、パッケージは影響を及ぼさない。
- パッケージに SUNW_PKG_ALLZONES=false が設定されている。

パッケージを大域ゾーンとすべての非大域ゾーンから削除する

パッケージを大域ゾーンおよび非大域ゾーンから削除するには、大域管理者として大域ゾーン内で pkgrm ユーティリティーを実行します。

パッケージを大域ゾーンおよび非大域ゾーンから削除する際、パッケージの影響を受ける領域について意識する必要はありません。

pkgrm ユーティリティーにより、次の手順が実行されます。

- 大域ゾーンおよび非大域ゾーン上でパッケージの依存関係が検査されます。依存関係の検査が失敗すると、pkgrm は失敗します。システムから大域管理者に対し、作業を続行するかどうかの問い合わせが行われます。
- パッケージが各非大域ゾーンから削除されます。
- 各非大域ゾーンのパッケージデータベースが更新されます。
- パッケージが大域ゾーンから削除されます。
- 大域ゾーンのパッケージデータベースが更新されます。

非大域ゾーン内での pkgrm の使用

ゾーン管理者として、非大域ゾーン内で pkgrm ユーティリティーを使用してパッケージを削除します。次の制限が適用されます。

- pkgrm では、非大域ゾーンからのパッケージの削除だけを実行できます。
- 大域ゾーンから共有されるゾーンのどの領域に対しても、パッケージは操作を行うことはできません。
- パッケージに SUNW_PKG_ALLZONES=false が設定されていなければなりません。

pkgrm ユーティリティーにより、次の手順が実行されます。

- 非大域ゾーンのパッケージデータベース上で、依存関係が検査されます。依存関係の検査が失敗すると、pkgrm が失敗し、ゾーン管理者に通知が行われます。次のいずれかが当てはまる場合、検査は失敗します。
  - パッケージのいずれかのコンポーネントが、大域ゾーンから共有されているゾーンのいずれかの領域に影響を及ぼす。
  - パッケージに SUNW_PKG_ALLZONES=true が設定されている。
- パッケージがゾーンから削除されます。
- ゾーンのパッケージデータベースが更新されます。

パッケージパラメータの情報

ゾーンのパッケージパラメータの設定

SUNW_PKG_ALLZONES、SUNW_PKG_HOLLOW、および SUNW_PKG_THISZONE パッケージパラメータは、ゾーンがインストールされているシステムでのパッケージの特性を定義します。非大域ゾーンがインストールされたシステムでパッケージを管理できるようにするには、これらのパラメータを設定する必要があります。

次の表に、パッケージパラメータの設定に使用できる 4 つの有効な組み合わせを示します。次の表に示されていない設定の組み合わせは無効であり、そのような設定を選択するとパッケージのインストールは失敗します。

3 つのパッケージパラメータをすべて設定したことを確認してください。3 つのパッケージパラメータをすべて空のままにしてもかまいません。ゾーンのパッケージパラメータが見つからない場合、パッケージツールではその設定は false として解釈されますが、パラメータの設定は省略しないように強くお勧めします。3 つのパッケージパラメータをすべて設定することにより、パッケージをインストールまたは削除するときのパッケージツールの動作を正確に指定します。

表 25–1 有効なパッケージパラメータ設定

SUNW_PKG_ALLZONES の設定	SUNW_PKG_HOLLOW の設定	SUNW_PKG_THISZONE の設定	パッケージの説明
false	false	false	これは、ゾーンのパッケージパラメータのすべてに値を指定しないパッケージに対するデフォルト設定です。 この設定を持つパッケージは、大域ゾーンまたは非大域ゾーンにインストールできます。 ■ 大域ゾーン内で pkgadd コマンドを実行すると、パッケージは大域ゾーンおよびすべての非大域ゾーンにインストールされます。 ■ 非大域ゾーン内で pkgadd コマンドを実行すると、パッケージはその非大域ゾーンだけにインストールされます。 どちらの場合も、パッケージがインストールされたすべてのゾーンで、パッケージの内容全体が可視になります。
false	false	true	この設定を持つパッケージは、大域ゾーンまたは非大域ゾーンにインストールできます。インストール後に新しい非大域ゾーンを作成した場合、パッケージはこれらの新しい非大域ゾーンには伝達されません。 ■ 大域ゾーン内で pkgadd コマンドを実行すると、パッケージは大域ゾーンだけにインストールされます。 ■ 非大域ゾーン内で pkgadd コマンドを実行すると、パッケージはその非大域ゾーンだけにインストールされます。 どちらの場合も、パッケージがインストールされたゾーンで、パッケージの内容全体が可視になります。

表 25–1 有効なパッケージパラメータ設定 (続き)

SUNW_PKG_ALLZONES の設定	SUNW_PKG_HOLLOW の設定	SUNW_PKG_THISZONE の設定	パッケージの説明
true	false	false	この設定を持つパッケージは、大域ゾーンだけにインストールできます。pkgadd コマンドを実行すると、パッケージは大域ゾーンおよびすべての非大域ゾーンにインストールされます。すべてのゾーンで、パッケージの内容全体が可視になります。 注 – パッケージを非大域ゾーンにインストールしようとすると失敗します。

表 25–1 有効なパッケージパラメータ設定 (続き)

SUNW_PKG_ALLZONES の設定	SUNW_PKG_HOLLOW の設定	SUNW_PKG_THISZONE の設定	パッケージの説明
true	true	false	この設定を持つパッケージは、大域管理者が大域ゾーンだけにインストールできます。pkgadd コマンドを実行すると、パッケージの内容が大域ゾーンに完全にインストールされます。パッケージパラメータの値がこのように設定されている場合、パッケージの内容自体はどの非大域ゾーンにも提供されません。パッケージをインストール済みとして表示するために必要なパッケージインストール情報だけが、すべての非大域ゾーンにインストールされます。これにより、このパッケージに依存するほかのパッケージをインストールできるようになります。 パッケージの依存関係を検査できるように、パッケージはすべてのゾーンでインストール済みとして表示されます。 ■ 大域ゾーンでは、パッケージの内容全体が可視になります。 ■ 完全ルート非大域ゾーンでは、パッケージの内容全体が不可視になります。 ■ 非大域ゾーンが大域ゾーンからファイルシステムを継承する場合、このファイルシステムにインストールされているパッケージは非大域ゾーンで可視になります。パッケージで提供されるほかのすべてのファイルは、非大域ゾーン内では不可視になります。たとえば、疎ルート非大域ゾーンは、特定のディレクトリを大域ゾーンと共有します。これらのディレクトリは読み取り専用です。疎ルート非大域ゾーンは、/platform ファイルシステムをほかのゾーンと共有します。もう 1 つの例は、ブートするハードウェアだけに関連するファイルがパッケージで提供されている場合です。 注 – パッケージを非大域ゾーンにインストールしようとすると失敗します。

SUNW_PKG_ALLZONES パッケージパラメータ

オプションの SUNW_PKG_ALLZONES パッケージパラメータには、パッケージのゾーン範囲が記述されます。このパラメータにより、次のことが定義されます。

- パッケージをすべてのゾーンにインストールする必要があるかどうか
- パッケージがすべてのゾーンで同一である必要があるかどうか

SUNW_PKG_ALLZONES パッケージパラメータに指定可能な値は 2 つあります。その値は、true および false です。デフォルト値は false です。このパラメータが設定されていない場合か、または true および false 以外の値に設定されている場合には、値 false が使用されます。

すべてのゾーンでパッケージのバージョンとパッチの改訂レベルが同一で「なければならない」パッケージでは、SUNW_PKG_ALLZONES パラメータを true に設定することをお勧めします。たとえば Oracle Solaris 10 など、特定の Oracle Solaris カーネルに依存した機能を提供するパッケージでは、このパラメータを true に設定することをお勧めします。パッケージに適用するパッチの SUNW_PKG_ALLZONES パラメータは、インストール済みパッケージに設定されている値と同じ値に設定する必要があります。このパラメータを true に設定しているパッケージのパッチ改訂レベルは、すべてのゾーンで一致していなければなりません。

他社製パッケージや Sun コンパイラなど、特定の Oracle Solaris カーネルに依存しない機能を提供するパッケージでは、このパラメータを false に設定することをお勧めします。このパラメータを false に設定しているパッケージのパッチでも、このパラメータを false に設定する必要があります。このパラメータを false に設定しているパッケージの場合は、各ゾーンのパッケージバージョンまたはパッチ改訂レベルが異なっていてもかまいません。たとえば、2 つの非大域ゾーンにインストールされている Web サーバーのバージョンは、一致していなくてもかまいません。

SUNW_PKG_ALLZONES パッケージパラメータの値については、次の表で説明します。

表 25–2 SUNW_PKG_ALLZONES パッケージパラメータの値

値	説明
false	このパッケージは、大域ゾーンから大域ゾーンだけ、または大域ゾーンから大域ゾーンおよびすべての非大域ゾーンにインストールできます。また、任意の非大域ゾーンから同じ非大域ゾーンにもインストールできます。 ■ 大域管理者は、パッケージを大域ゾーンだけにインストールできます。 ■ 大域管理者は、パッケージを大域ゾーンおよびすべての非大域ゾーンにインストールできます。 ■ ゾーン管理者は、パッケージを非大域ゾーンにインストールできます。 パッケージを大域ゾーンから削除しても、ほかのゾーンからそのパッケージは削除されません。パッケージは非大域ゾーンごとに削除できます。 ■ パッケージを大域ゾーンにインストールする必要はありません。 ■ パッケージをいずれかの非大域ゾーンにインストールする必要はありません。 ■ パッケージはすべてのゾーンで同一である必要はありません。ゾーンごとにパッケージのバージョンが異なっていてもかまいません。 ■ パッケージは、すべてのゾーンで暗黙的に共有されていないソフトウェアを提供します。これは、パッケージがオペレーティングシステムに固有ではないことを意味します。アプリケーションレベルのソフトウェアの大半がこのカテゴリに属します。たとえば、StarSuite 製品や Web サーバーがこれに該当します。

表 25–2 SUNW_PKG_ALLZONES パッケージパラメータの値 (続き)

値	説明
true	パッケージを大域ゾーンにインストールする場合、すべての非大域ゾーンにもインストールする必要があります。パッケージを大域ゾーンから削除する場合、すべての非大域ゾーンからも削除する必要があります。 ■ パッケージをインストールする場合、大域ゾーンにインストールする必要があります。その後、パッケージはすべての非大域ゾーンに自動的にインストールされます。 ■ パッケージのバージョンは、すべてのゾーンで一致している必要があります。 ■ パッケージは、すべてのゾーンで暗黙的に共有されているソフトウェアを提供します。パッケージは、すべてのゾーンで暗黙的に共有されるソフトウェアのバージョンに依存します。パッケージは、すべての非大域ゾーンで可視である必要があります。これには、カーネルモジュールも含まれます。 パッケージ全体をすべての非大域ゾーンにインストールすることを要求することで、これらのパッケージを使用して、大域ゾーンにインストールされるパッケージの依存関係を非大域ゾーンから解決できます。 ■ 大域管理者だけがパッケージをインストールできます。ゾーン管理者が非大域ゾーンにパッケージをインストールすることはできません。

SUNW_PKG_HOLLOW パッケージパラメータ

SUNW_PKG_HOLLOW パッケージパラメータは、パッケージがすべてのゾーンにインストールされ、かつすべてのゾーンで同一であることが求められる場合、そのパッケージをすべての非大域ゾーン内で可視にするべきかどうかを定義します。

SUNW_PKG_HOLLOW パッケージパラメータは、2 つの値 true、false のいずれかを取ります。

- SUNW_PKG_HOLLOW が設定されていないか、または true および false 以外の値に設定されている場合、値 false が使用されます。
- SUNW_PKG_ALLZONES が false に設定されている場合、SUNW_PKG_HOLLOW パラメータは無視されます。
- SUNW_PKG_ALLZONES が false に設定されている場合、SUNW_PKG_HOLLOW を true に設定することはできません。

SUNW_PKG_HOLLOW パッケージパラメータ値については、次の表で説明します。

表 25–3　SUNW_PKG_HOLLOW パッケージパラメータ値

値	説明
false	これは「hollow」パッケージではありません。 ■ 大域ゾーンにインストールする場合、すべての非大域ゾーンでパッケージ内容およびインストール情報が要求されます。 ■ このパッケージの提供するソフトウェアは、すべての非大域ゾーンで可視でなければなりません。たとえば、truss コマンドを提供するパッケージがこれに該当します。 ■ 現在の SUNW_PKG_ALLZONES パッケージパラメータ設定に適用される制限を除き、追加の制限は定義されません。

表 25–3 SUNW_PKG_HOLLOW パッケージパラメータ値 (続き)

値	説明
true	これは「hollow」パッケージです。 ■ パッケージの内容はどの非大域ゾーンにも提供されません。ただし、すべての非大域ゾーンでパッケージのインストール情報が求められます。 ■ このパッケージの提供するソフトウェアは、すべての非大域ゾーンで可視であってはいけません。たとえば、大域ゾーンでのみ機能するカーネルドライバやシステム構成ファイルがこれに該当します。この設定により、実際にパッケージデータをインストールしなくても、非大域ゾーンが、大域ゾーンにのみインストールされるパッケージの依存関係を解決することが可能になります。 ■ インストール中のパッケージに依存するほかのパッケージにより、このパッケージが依存関係の検査目的ですべてのゾーンにインストールされたと認識されます。 ■ このパッケージ設定には、SUNW_PKG_ALLZONES を true に設定するために定義されたすべての制限が含まれます。 ■ 大域ゾーン内で、パッケージはインストール済みとして認識され、パッケージのすべてのコンポーネントがインストールされます。パッケージのインストール時に、ディレクトリの作成、ファイルのインストール、およびクラス操作とほかのスクリプトが、必要に応じて実行されます。 ■ 非大域ゾーン内で、パッケージはインストール済みとして認識されますが、パッケージのコンポーネントは 1 つもインストールされません。パッケージのインストール時に、ディレクトリの作成、ファイルのインストール、およびクラス操作やほかのインストールスクリプトは実行されません。 ■ パッケージが大域ゾーンから削除される場合、パッケージは完全にインストールされていたと認識されます。パッケージの削除時に、該当するディレクトリおよびファイルが削除され、クラス操作またはほかのインストールスクリプトが実行されます。

SUNW_PKG_THISZONE パッケージパラメータ

SUNW_PKG_THISZONE パッケージパラメータは、パッケージを現在のゾーン (大域ゾーン、非大域ゾーンのいずれか) だけにインストールする必要があるかどうかを定義します。SUNW_PKG_THISZONE パッケージパラメータに設定可能な値は 2 つあります。その値は、true および false です。デフォルト値は false です。

SUNW_PKG_THISZONE パッケージパラメータの値については、次の表で説明します。

表 25–4 SUNW_PKG_THISZONE パッケージパラメータの値

値	説明
false	■ 非大域ゾーン内で pkgadd を実行する場合、パッケージは現在のゾーンだけにインストールされます。 ■ 大域ゾーン内で pkgadd を実行する場合、パッケージは大域ゾーンにインストールされるだけでなく、現在インストールされているすべての非大域ゾーンにもインストールされます。さらに、将来新しくインストールされるすべての非大域ゾーンにも、そのパッケージが伝達されます。
true	■ パッケージは現在のゾーンだけにインストールされます。 ■ 大域ゾーン内でパッケージをインストールする場合、そのパッケージは現存の非大域ゾーンや将来作成される非大域ゾーンには一切追加されません。これは、pkgadd に -G オプションを指定した場合の動作と同じです。

パッケージ情報の照会

pkginfo(1) のマニュアルページに記載されている pkginfo ユーティリティーは、ゾーンがインストールされている Oracle Solaris システムでのソフトウェアパッケージデータベースの照会をサポートします。データベースの詳細は、364 ページの「製品データベース」を参照してください。

大域ゾーンで、pkginfo ユーティリティーを使用すると、大域ゾーン内だけでソフトウェアパッケージデータベースを照会できます。非大域ゾーンで pkginfo ユーティリティーを使用すると、非大域ゾーン内だけでソフトウェアパッケージデータベースを照会できます。

ゾーン内でのパッチの追加について

通常、パッチは次のコンポーネントで構成されます。

- パッチ情報:
  - 識別情報 (パッチのバージョンおよびパッチ ID)
  - 適用対象 (オペレーティングシステムの種類、バージョン、およびアーキテクチャー)
  - 依存関係 (必須や廃止など)
  - プロパティー (適用後にリブートが必要、など)

- パッチを適用する 1 つ以上のパッケージ。各パッケージには次のものが含まれます。
  - パッチを適用可能なパッケージのバージョン
  - パッチ情報 (ID、廃止、必須など)
  - パッチの適用されるパッケージの 1 つ以上のコンポーネント

`patchadd` コマンドを使用してパッチを適用する場合、稼働中のシステムにパッチを適用可能かどうかの判別にパッチ情報が使用されます。適用不可と判別された場合、パッチは適用されません。パッチの依存関係も、システムのすべてのゾーンに対して検査されます。必須の依存関係のいずれかが満たされない場合、パッチは適用されません。これには、以降のバージョンのパッチがインストール済みである場合も含まれます。

パッチに含まれる各パッケージも検査されます。パッケージがどのゾーンにもインストールされない場合、そのパッケージは省略され、パッチは適用されません。

すべての依存関係が満たされる場合、いずれかのゾーンにインストールされるパッチ内のパッケージすべてが、システムへのパッチ適用に使用されます。パッケージおよびパッチデータベースの更新も行われます。

注 – **Oracle Solaris 10 3/05** から **Oracle Solaris 10 11/06:** パッケージが `pkgadd -G` を使ってインストールされている場合、またはパッケージの `pkginfo` 設定が `SUNW_PKG_THISZONE=true` になっている場合は、`patchadd -G` でのみパッケージにパッチを適用できます。この制限は Oracle Solaris 8/07 リリースで削除されています。

Oracle Solaris 10 8/07: 遅延起動パッチ

パッチ 119254-41 および 119255-41 以降、`patchadd` と `patchrm` のパッチインストールユーティリティーが変更され、機能を提供する特定のパッチの処理方法が変わりました。この変更は、これらのパッチをどの Oracle Solaris 10 リリースにインストールする場合にも影響を与えます。これらの遅延起動パッチによって、Oracle Solaris 10 3/05 リリース以降の各 Oracle Solaris 10 リリースに関連するカーネルパッチなどの機能パッチで提供される、大規模な変更の処理能力が向上します。

遅延起動パッチは、実行中のシステムの安定性を保証するためにループバックファイルシステム (lofs) を使用します。パッチが実行中のシステムに適用されると、lofs はパッチプロセス中の安定性を保持します。これらの大規模なカーネルパッチは必ずリブートを必要としますが、この必須リブートで lofs による変更がアクティブ化されるようになりました。パッチの README には、どのパッチでリブートが必要になるかが説明されています。

非大域ゾーンを実行しているか、lofs を無効にしている場合は、遅延起動パッチをインストールまたは削除する際に次の点を考慮してください。

- このパッチ操作を行うには、すべての非大域ゾーンが停止される必要があります。パッチを適用する前に非大域ゾーンを停止してください。
- 遅延起動パッチには、ループバックファイルシステム (lofs) が必要です。Sun Cluster 3.1 または Sun Cluster 3.2 を実行しているシステムでは、lofs が有効になっていると HA-NFS 機能が制限されるため、lofs がオフになっている可能性があります。このため、遅延起動パッチをインストールする前に、`/etc/system` ファイルの次の行を削除するかコメントにして、ループバックファイルシステムを再び有効にする必要があります。

  ```
  exclude:lofs
  ```

  その次に、システムをリブートしてパッチをインストールします。パッチのインストール操作を完了した後、`/etc/system` ファイルから同じ行を復元するか、コメントを解除します。そのあと通常の操作を再開するにはリブートする必要があります。

注 – Oracle Solaris Live Upgrade を使用してパッチの管理を行うと、実行中のシステムへのパッチに関連する問題を防ぐことができます。Oracle Solaris Live Upgrade は、パッチの適用に伴う停止時間を短縮し、問題発生時のフォールバック機能を提供することでリスクを低減できます。システムがまだ本稼働していても、アクティブでないブート環境 (BE) にパッチを適用したり、新しい BE で問題が発見された場合、元の BE に戻ってブートすることができます。『Oracle Solaris 10 9/10 インストールガイド (Solaris Live Upgrade とアップグレードの計画)』の「パッケージまたはパッチによるシステムのアップグレード」を参照してください。

Oracle Solaris 10 10/09: パッチ適用時間を短縮するためのゾーンの並列パッチ

ゾーンの並列パッチは、標準のパッチユーティリティーの拡張機能で、Oracle Solaris 10 システム上の非大域ゾーンにパッチを適用するためにサポートされている方法です。この機能は、非大域ゾーンにも並行してパッチを適用することで、ゾーンパッチングのパフォーマンスを向上します。

Oracle Solaris 10 10/09 より前のリリースでは、この機能は、119254-66 以降のリビジョン (SPARC) および 119255-66 以降のリビジョン (x86) のパッチユーティリティーのパッチで提供されます。

並列でパッチを適用する非大域ゾーンの最大数は `patchadd`, `/etc/patch/pdo.conf` の新しい構成ファイルに設定できます。このパッチのリビジョン 66 以降は、すべての Oracle Solaris 10 システムおよび Sun xVM Ops Center など高いレベルのパッチ自動化ツールで機能します。

ただし、最初にパッチを適用する必要があるのは大域ゾーンです。大域ゾーンへのパッチの適用が終了した時点で、`num_proc=` に設定されている数の非大域ゾーンに対

して一緒にパッチの適用が行われます。最大数は、オンライン CPU の数の 1.5 倍であり、システム上の実際の非大域ゾーンの数まで設定できます。

例:

- オンライン CPU の数は 4 つです。
- 設定は `num_proc=6` です。

システム上にこの数を超える非大域ゾーンがある場合は、最初の 6 つに並列でパッチを適用し、その後、最初のグループへのパッチの適用プロセスが終了すると、残りの非大域ゾーンにパッチが適用されます。

Oracle Solaris Live Upgrade とパッチを管理するこの新しいパッチを使用することで、問題が発生した場合のフォールバック機能が提供されます。システムがまだ本稼働していても、アクティブでないブート環境 (BE) にパッチを適用したり、新しい BE で問題が発見された場合、元の BE に戻ってブートすることができます。

372 ページの「Oracle Solaris 10 10/09: 非大域ゾーンに並列でパッチを適用する方法」も参照してください。

注 – ホストに新しくインストールされた非大域ゾーンで表示される内容と一致するように、ゾーンのすべてのパッケージをすばやく更新するには、大域ゾーンにパッチを適用する間は各ゾーンを切り離しておき、適用後に `-U` オプションを使用して再接続して、大域ゾーンのレベルと一致させます。詳細は、329 ページの「パッチ適用のソリューションとして、接続時更新を使用する」を参照してください。

ゾーンがインストールされている Oracle Solaris システムでのパッチの適用

大域ゾーンで適用されるすべてのパッチが、すべてのゾーンで適用されます。非大域ゾーンをインストールする際も、大域ゾーンと同じパッチレベルでパッチが適用されます。大域ゾーンにパッチを適用する際、すべての非大域ゾーンで同様にパッチが適用されます。この処理により、すべてのゾーンにわたって同じパッチレベルが維持されます。

`patchadd(1M)` のマニュアルページに記載されているように、`patchadd` システムユーティリティーを使用して、ゾーンがインストールされているシステムにパッチを追加できます。

大域ゾーンでの patchadd の使用

パッチを大域ゾーンおよびすべての非大域ゾーンに追加するには、大域ゾーン内で大域管理者として `patchadd` を実行します。

大域ゾーンで patchadd を使用する場合、次の条件が両方適用されます。

- patchadd ユーティリティーは、パッチを大域ゾーンおよびすべての非大域ゾーンだけに追加できます。これはデフォルトの動作です。
- patchadd ユーティリティーは、パッチを大域ゾーンだけ、または非大域ゾーンの一部に追加することはできません。

パッチを大域ゾーンおよびすべての非大域ゾーンに追加する際、大域ゾーンから共有される領域にパッチが影響を及ぼすかどうかを意識する必要はありません。

patchadd ユーティリティーにより、次の手順が実行されます。

- パッチが大域ゾーンに追加されます。
- 大域ゾーンのパッチデータベースが更新されます。
- パッチが各非大域ゾーンに追加されます。
- 各非大域ゾーンのパッチデータベースが更新されます。

非大域ゾーン内での patchadd の使用

ゾーン管理者が非大域ゾーン内で patchadd を使用する場合、そのゾーンにパッチを追加する用途だけにコマンドを使用できます。次の場合に、パッチを非大域ゾーンに追加できます。

- パッチが大域ゾーンから共有されるゾーンのどの領域にも影響を及ぼさない。
- パッチ内部のパッケージすべてが SUNW_PKG_ALLZONES=false に設定されている。

patchadd ユーティリティーにより、次の手順が実行されます。

- パッチがゾーンに追加されます。
- ゾーンのパッチデータベースが更新されます。

ゾーンが含まれているシステムでの patchadd -G と pkginfo 変数の相互作用

次に、大域ゾーンおよび非大域ゾーンにパッチを追加する際の -G オプションと SUNW_PKG_ALLZONES 変数の相互作用について説明します。

大域ゾーン、-G 指定あり
: SUNW_PKG_ALLZONES=TRUE に設定されているパッケージがある場合、この使用法の結果はエラーとなり、処理は行われません。

  SUNW_PKG_ALLZONES=TRUE に設定されているパッケージがない場合は、大域ゾーンのパッケージだけにパッチが適用されます。

大域ゾーン、-G 指定なし
: SUNW_PKG_ALLZONES=TRUE に設定されているパッケージがある場合は、すべてのゾーンにあるそれらのパッケージにパッチが適用されます。

SUNW_PKG_ALLZONES=TRUE に設定されていないパッケージがある場合は、該当するすべてのゾーンにあるそれらのパッケージにパッチが適用されます。「大域ゾーンのみ」のパッケージは、大域ゾーンだけにインストールされます。

非大域ゾーン、-G 指定ありまたは指定なし
SUNW_PKG_ALLZONES=TRUE に設定されているパッケージがある場合、この使用法の結果はエラーとなり、処理は行われません。

SUNW_PKG_ALLZONES=TRUE に設定されているパッケージがない場合は、非大域ゾーンのパッケージだけにパッチが適用されます。

ゾーンがインストールされている Oracle Solaris システムでのパッチの削除

patchrm システムユーティリティー (patchrm(1M) のマニュアルページを参照) を使用して、ゾーンがインストールされているシステムでパッチを削除します。

大域ゾーン内での patchrm の使用

大域ゾーン管理者として大域ゾーン内で patchrm ユーティリティーを使用して、パッチを削除できます。patchrm ユーティリティーは、大域ゾーンのみ、または非大域ゾーンの一部からパッチを削除することはできません。

非大域ゾーン内での patchrm の使用

ゾーン管理者として非大域ゾーン内で patchrm ユーティリティーを使用すると、その非大域ゾーンだけを対象としてパッチの削除を実行できます。共有されている領域がパッチの影響を受けることはありません。

製品データベース

各ゾーンの個別パッケージ、パッチ、および製品レジストリデータベースには、そのゾーンで使用可能なインストール済みのソフトウェアすべてが記載されています。インストールする追加ソフトウェアまたはパッチの依存性検査はすべて、パッケージまたはパッチが大域ゾーンおよび 1 つ以上の非大域ゾーンでインストールまたは削除中でない限り、ほかのゾーンのデータベースに一切アクセスすることなく実行されます。パッケージまたはパッチが大域ゾーンおよび 1 つ以上の非大域ゾーンでインストールまたは削除中の場合は、適切な非大域ゾーンデータベースにアクセスする必要があります。

データベースの詳細については、pkgadm(1M) のマニュアルページを参照してください。

◆ ◆ ◆ 第 26 章

ゾーンがインストールされている Oracle Solaris システムでのパッケージとパッチの追加および削除 (タスク)

Oracle Solaris 10 1/06: このリリースでは、この章は全面的に改訂されています。この章では、非大域ゾーンがインストールされているシステムで最新のパッケージとパッチを追加および削除する手順について説明します。

Oracle Solaris 10 6/06: 367 ページの「パッケージを大域ゾーンだけに追加する方法」の手順に注が追加されました。

Oracle Solaris 10 8/07: 371 ページの「大域ゾーンだけにパッチを適用する方法」のタスクから注が削除されました。

Oracle Solaris 10 の新機能の一覧および Oracle Solaris リリースについての説明は、『Oracle Solaris 10 8/11 の新機能』を参照してください。

この章では、ゾーンがインストールされているシステムでパッケージとパッチを追加および削除する方法について説明します。パッケージパラメータの設定やパッケージ情報の取得など、パッケージとパッチの管理に関連したほかのタスクについても説明します。ゾーンがインストールされているシステムでのパッチおよびパッケージの概念に関する概要は、第 25 章「ゾーンがインストールされている Oracle Solaris システムでのパッケージとパッチについて (概要)」を参照してください。

ゾーンがインストールされている Oracle Solaris システムでのパッケージとパッチの追加および削除 (タスクマップ)

タスク	説明	参照先
パッケージを追加します。	ゾーンがインストールされているシステムでパッケージを追加します。	366 ページの「ゾーンがインストールされている Oracle Solaris システムでのパッケージの追加」
パッケージ情報を検査します。	ゾーンがインストールされているシステムでパッケージ情報を検査します。	369 ページの「ゾーンがインストールされている Oracle Solaris システムでのパッケージ情報の検査」
パッケージを削除します。	ゾーンがインストールされているシステムでパッケージを削除します。	370 ページの「ゾーンがインストールされている Solaris システムからのパッケージの削除」
パッチを適用します。	ゾーンがインストールされているシステムでパッチを適用します。	371 ページの「ゾーンがインストールされている Oracle Solaris システムへのパッチの適用」
パッチを削除します。	ゾーンがインストールされているシステムでパッチを削除します。	373 ページの「ゾーンがインストールされているシステムでのパッチの削除」
(オプション) パッケージパラメータの設定を検査します。	パッケージを追加または削除する際、パッケージパラメータの設定が実行する操作をサポートしていることを確認します。	374 ページの「ゾーンがインストールされているシステムでのパッケージパラメータ設定の検査」

ゾーンがインストールされている Oracle Solaris システムでのパッケージの追加

pkgadd(1M) のマニュアルページで説明しているように、pkgadd システムユーティリティーを使用すると、次のタスクを実行できます。

- パッケージを大域ゾーンだけに追加します
- パッケージを大域ゾーンとすべての非大域ゾーンの両方に追加します
- 大域ゾーンにインストール済みのパッケージを非大域ゾーンに追加します
- パッケージを指定された非大域ゾーンだけに追加します

パッケージを追加する場合、SUNW_PKG_ALLZONES および SUNW_PKG_HOLLOW パッケージパラメータ設定は、true または false のいずれか適正な値でなければなりませ

ん。ほかの値が設定されている場合、意図した結果を得ることができません。パッケージパラメータ設定の効果に関する詳細は、342 ページの「パッケージとゾーンについて」を参照してください。これらのパッケージパラメータ設定の検査方法については、374 ページの「ゾーンがインストールされているシステムでのパッケージパラメータ設定の検査」を参照してください。

▼ パッケージを大域ゾーンだけに追加する方法

パッケージを大域ゾーンだけに追加する場合、SUNW_PKG_ALLZONES パッケージパラメータが false に設定されている必要があります。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 大域ゾーン内で、**pkgadd -d** コマンドに続けてパッケージの場所、**-G** オプション、およびパッケージ名を指定して実行します。

   - CD-ROM からパッケージをインストールする場合は、次のように入力します。

     ```
     global# pkgadd -d /cdrom/cdrom0/directory -G package_name
     ```

   - そのパッケージが既にコピーされているディレクトリからパッケージをインストールする場合は、次のように入力します。

     ```
     global# pkgadd -d disk1/image -G package_name
     ```

     ここで、*disk1* にはパッケージのコピー先の場所を指定します。

注 – pkgadd ユーティリティーを -G オプションを指定せずに実行し、かつ SUNW_PKG_THISZONE=true であった場合、指定されたパッケージはデフォルトで現在のゾーン (大域ゾーン) に追加されます。

▼ パッケージを大域ゾーンとすべての非大域ゾーンに追加する方法

この手順では、pkgadd にオプション -G を指定しないでください。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。
2. 大域ゾーン内で、**pkgadd -d** コマンドに続けてパッケージの場所とパッケージ名を指定して実行します。
   - CD-ROM からパッケージをインストールする場合は、次のように入力します。

     ```
     global# pkgadd -d /cdrom/cdrom0/directory package_name
     ```
   - そのパッケージが既にコピーされているディレクトリからパッケージをインストールする場合は、次のように入力します。

     ```
     global# pkgadd -d disk1/image package_name
     ```

     ここで、*disk1* にはパッケージのコピー先の場所を指定します。

▼ 大域ゾーンにインストールしたパッケージをすべての非大域ゾーンに追加する方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。
2. 大域ゾーンで、**pkgrm** を使ってパッケージを削除します。
3. **-G** オプションを指定せずにパッケージを追加します。

▼ 指定された非大域ゾーンだけにパッケージを追加する方法

指定された非大域ゾーンだけにパッケージを追加する場合、`SUNW_PKG_ALLZONES` パッケージパラメータが `false` に設定されている必要があります。この手順では、`pkgadd` に `-G` オプションを指定しないでください。

この手順を実行するには、非大域ゾーン内のゾーン管理者になる必要があります。

1. ゾーン管理者として非大域ゾーンにログインします。

2 非大域ゾーン (この手順では **my-zone**) 内で、**pkgadd -d** コマンドに続けてパッケージの場所、およびパッケージ名を指定して実行します。

- CD-ROM からパッケージをインストールする場合は、次のように入力します。

  ```
  my-zone# pkgadd -d /cdrom/cdrom0/directory package_name
  ```

- そのパッケージが既にコピーされているディレクトリからパッケージをインストールする場合は、次のように入力します。

  ```
  my-zone# pkgadd -d disk1/image package_name
  ```

  ここで、*disk1* にはパッケージのコピー先の場所を指定します。

ゾーンがインストールされている Oracle Solaris システムでのパッケージ情報の検査

pkginfo コマンドを実行することで、大域ゾーンや非大域ゾーンのソフトウェアパッケージデータベースを照会できます。このコマンドの詳細は、pkginfo(1) のマニュアルページを参照してください。

▼ 大域ゾーンだけでパッケージ情報を検査する方法

- 大域ゾーンだけのソフトウェアパッケージデータベースを検査する場合、**pkginfo** に続けてパッケージ名を指定します。

  ```
  global% pkginfo package_name
  ```

例 **26–1** 大域ゾーンで pkginfo コマンドを使用する

```
global% pkginfo SUNWcsr SUNWcsu
system      SUNWcsr Core Oracle Solaris, (Root)
system      SUNWcsu Core Oracle Solaris, (Usr)
```

▼ 指定された非大域ゾーンだけのパッケージ情報を検査する方法

- 指定された非大域ゾーンのソフトウェアパッケージデータベースを検査する場合は、非大域ゾーンにログインし、**pkginfo** に続けてパッケージ名を指定します。

  ```
  my-zone% pkginfo package_name
  ```

例 **26–2** 非大域ゾーンで pkginfo コマンドを使用する

```
my-zone% pkginfo SUNWcsr SUNWcsu
system      SUNWcsr Core Oracle Solaris, (Root)
system      SUNWcsu Core Oracle Solaris, (Usr)
```

ゾーンがインストールされている Solaris システムからのパッケージの削除

pkgrm(1M) のマニュアルページで説明しているように、pkgrm システムユーティリティーを使用すると、次のタスクを実行できます。

- 大域ゾーンおよびすべての非大域ゾーンからパッケージを削除します
- 指定された非大域ゾーンだけからパッケージを削除します

パッケージを削除する場合、SUNW_PKG_ALLZONES および SUNW_PKG_HOLLOW パッケージパラメータ設定は、true または false のいずれか適正な値でなければなりません。ほかの値が設定されている場合、意図した結果を得ることができません。パッケージパラメータ設定の効果に関する詳細は、342 ページの「パッケージとゾーンについて」を参照してください。これらのパッケージパラメータ設定の検査方法については、374 ページの「ゾーンがインストールされているシステムでのパッケージパラメータ設定の検査」を参照してください。

▼ 大域ゾーンとすべての非大域ゾーンからパッケージを削除する方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 大域ゾーン内で、**pkgrm** コマンドに続けてパッケージ名を指定します。

   ```
   global# pkgrm package_name
   ```

▼ 指定された非大域ゾーンだけからパッケージを削除する方法

指定された非大域ゾーンだけからパッケージを削除する場合、SUNW_PKG_ALLZONES パッケージパラメータを false に設定する必要があります。

この手順を実行するには、非大域ゾーン内のゾーン管理者になる必要があります。

1 ゾーン管理者として非大域ゾーンにログインします。

2 非大域ゾーン (この手順では **my-zone**) 内で、**pkgrm** コマンドに続けてパッケージ名を指定して実行します。

```
my-zone# pkgrm package_name
```

ゾーンがインストールされている Oracle Solaris システムへのパッチの適用

patchadd(1M) のマニュアルページで説明しているように、patchadd システムユーティリティーを使用すると、次のタスクを実行できます。

- 大域ゾーンだけにパッチを適用します
- 大域ゾーンおよびすべての非大域ゾーンにパッチを適用します
- 指定された非大域ゾーンだけにパッチを適用します

▼ 大域ゾーンだけにパッチを適用する方法

注 – **Oracle Solaris 10 3/05** から **Oracle Solaris 10 11/06:** pkgadd コマンドに -G オプションを指定して追加されたパッケージにパッチを適用する場合は、patchadd コマンドに -G オプションを指定してパッチを適用する必要があります。この制限は Oracle Solaris 8/07 リリースで削除されています。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **patchadd** コマンドに続けて **-G** オプションとパッチ **ID** を指定して実行します。

```
global# patchadd -G patch_id
```

▼ 大域ゾーンとすべての非大域ゾーンにパッチを適用する方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

361 ページの「Oracle Solaris 10 10/09: パッチ適用時間を短縮するためのゾーンの並列パッチ」および372 ページの「Oracle Solaris 10 10/09: 非大域ゾーンに並列でパッチを適用する方法」も参照してください。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。
役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **patchadd** コマンドに続けてパッチ **ID** を指定して実行します。

```
global# patchadd patch_id
```

▼ 指定された非大域ゾーンだけにパッチを適用する方法

指定された非大域ゾーンだけにパッチを適用するには、パッチセット内のすべてのパッケージの SUNW_PKG_ALLZONES パッケージパラメータが false に設定されている必要があります。

この手順を実行するには、非大域ゾーン内のゾーン管理者になる必要があります。

1 ゾーン管理者として非大域ゾーンにログインします。

2 非大域ゾーン (この手順では **my-zone**) 内で、**patchadd** コマンドに続けてパッチ **ID** を指定して実行します。

```
my-zone# patchadd patch_id
```

▼ Oracle Solaris 10 10/09: 非大域ゾーンに並列でパッチを適用する方法

patchadd 構成ファイル /etc/patch/pdo.conf に、並列でパッチを適用する非大域ゾーンの数を設定します。大域ゾーンへのパッチの適用が終了した時点で、num_proc= に設定されている数の非大域ゾーンに対して一緒にパッチの適用が行われます。

Oracle Solaris 10 10/09 より前のリリースを実行している場合は、パッチ 119254-66 以降のリビジョン (SPARC) または 119255-66 以降のリビジョン (x86) をダウンロードします。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **(オプション、Oracle Solaris 10 10/09 より前のリリースのみ必要) パッチ 119254-66 (SPARC) または 119255-66 (x86)** をダウンロードします。

3. **`/etc/patch/pdo.conf`** ファイルに、オンライン **CPU** が **4** つあるシステム上で、並列でパッチを一緒に適用する **6** つの非大域ゾーンを設定します。

   ```
   num_proc=6
   ```

   システム上に 6 つを超える非大域ゾーンがある場合は、まず最初の 6 つに並列でパッチが適用され、最初の 6 つの非大域ゾーンへのパッチの適用プロセスが終了したら、残りの非大域ゾーンにパッチが適用されます。

ゾーンがインストールされているシステムでのパッチの削除

`patchrm(1M)` のマニュアルページで説明しているように、`patchrm` システムユーティリティーを使用すると、次のタスクを実行できます。

- 大域ゾーンとすべての非大域ゾーンからパッチを削除します
- 指定された非大域ゾーンだけからパッチを削除します

▼ 大域ゾーンとすべての非大域ゾーンからパッチを削除する方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **`patchrm`** コマンドに続けてパッチ **ID** を指定して実行します。

   ```
   global# patchrm patch_id
   ```

▼ 指定された非大域ゾーンだけからパッチを削除する方法

指定された非大域ゾーンだけからパッチを削除するには、パッチセット内のすべてのパッケージの SUNW_PKG_ALLZONES パッケージパラメータが false に設定されている必要があります。

この手順を実行するには、非大域ゾーン内のゾーン管理者になる必要があります。

1. ゾーン管理者として非大域ゾーンにログインします。
2. 非大域ゾーン (この手順では **my-zone**) 内で、**patchrm** コマンドに続けてパッチ **ID** を指定して実行します。

```
my-zone# patchrm patch_id
```

ゾーンがインストールされているシステムでのパッケージパラメータ設定の検査

ソフトウェアパッケージを追加または削除する前に、pkgparam コマンドを使用してパッケージパラメータ設定を検査できます。この手順はオプションです。パッケージが意図したとおりに追加または削除されない場合のトラブルシューティングにも、この検査は使用できます。パッケージのパラメータ値の表示については、pkgparam(1) のマニュアルページを参照してください。

▼ (オプション) システムにインストールされたパッケージの設定を検査する方法

- 大域ゾーンまたは非大域ゾーンにインストール済みのパッケージのパッケージパラメータ設定を検査する場合、**pkgparam** に続けてパッケージ名およびパラメータ名を指定します。

```
my-zone% pkgparam package_name SUNW_PKG_ALLZONES
true
my-zone% pkgparam package_name SUNW_PKG_HOLLOW
false
```

▼ (オプション) CD-ROM に収録されたソフトウェア内のパッケージの設定を検査する方法

- **CD-ROM** に収録されているソフトウェア内のアンインストール済みパッケージのパッケージパラメータを検査する場合、**pkgparam -d** に続けて **CD-ROM** のパス、パッケージ名、およびパラメータ名を指定します。

```
my-zone% pkgparam -d /cdrom/cdrom0/directory package_name SUNW_PKG_ALLZONES
true
my-zone% pkgparam -d /cdrom/cdrom0/directory package_name SUNW_PKG_HOLLOW
false
```

◆ ◆ ◆ 第 27 章

Oracle Solaris ゾーンの管理 (概要)

この章では、次の一般的なゾーン管理について説明します。

lx ブランドゾーンについては、パート III「lx ブランドゾーン」を参照してください。

この章に追加されている説明

Oracle Solaris 10 1/06: 382 ページの「ゾーン内でのファイルシステムのマウント解除」というセクションが新しく追加されています。

Oracle Solaris 10 1/06: ゾーンのバックアップ手順と復元手順に関するセクションが新しく追加されています。404 ページの「ゾーンがインストールされている Oracle Solaris システムのバックアップについて」を参照してください。

Oracle Solaris 10 6/06: 381 ページの「ゾーン内でのファイルシステムのマウント」の表に ZFS エントリが追加されています。

Oracle Solaris 10 8/07: このリリースでは、次の情報が新しく追加または更新されています。

- このリリースでは、非大域ゾーンに 2 つの IP タイプを使用できるようになりました。IP タイプごとに使用可能な機能に関する説明が追加されています。387 ページの「共有 IP 非大域ゾーンにおけるネットワーク」および390 ページの「Oracle Solaris 10 8/07: 排他的 IP 非大域ゾーンでのネットワーク」を参照してください。
- Oracle Solaris IP フィルタは共有 IP ゾーンで使用できるようになりました。詳細は、389 ページの「共有 IP ゾーンでの Oracle Solaris IP フィルタ」を参照してください。
- ゾーンの特権設定に関する説明が改訂されています。表 27–1 を参照してください。
- 408 ページの「ゾーンがインストールされている Oracle Solaris システムで使用するコマンド」の内容が更新されています。

Oracle Solaris 10 の新機能の一覧および Oracle Solaris リリースについての説明は、『Oracle Solaris 10 8/11 の新機能』を参照してください。

大域ゾーンの可視性とアクセス

大域ゾーンは、システムのデフォルトゾーンとしても、システム規模の管理制御用ゾーンとしても機能します。この二重の役割と関係のある管理上の問題が存在します。ゾーン内部のアプリケーションはほかのゾーン内のプロセスおよびほかのシステムオブジェクトにアクセスするため、管理操作の効果が予期したものよりも大きい場合があります。たとえば、サービスの停止スクリプトでは、プロセスを終了させるためのシグナルの送信にしばしば `pkill` が使用されます。大域ゾーンからこの種のスクリプトを実行すると、システム内の該当するすべてのプロセスが、ゾーンに関係なくシグナルを送信します。

多くの場合、システム規模の範囲が必要になります。たとえば、システム規模のリソース使用状況を監視する場合、システム全体のプロセス統計情報を表示する必要があります。大域ゾーンのアクティビティーだけのビューには、システムリソース

の一部または全体を共有可能なシステム内のほかのゾーンからの関連情報が欠落しています。この種のビューは、CPU などのシステムリソースがリソース管理機能を使用して厳密に区分されていない場合、特に重要です。

このため、大域ゾーン内のプロセスから、非大域ゾーン内のプロセスおよびほかのオブジェクトを監視できます。これにより、この種のプロセスがシステム規模の監視機能を備えることが可能になります。ほかのゾーン内のプロセスを制御したりシグナルを送信したりする機能は、PRIV_PROC_ZONE 特権により制限されます。この特権は、特権のないプロセスに設定された制限をオーバーライドできるため、PRIV_PROC_OWNER に類似しています。この場合の制限は、大域ゾーン内の特権のないプロセスはほかのゾーン内のプロセスにシグナルを送信したり制御したりすることはできない、というものです。これは、プロセスのユーザー ID が一致するか、動作しているプロセスが PRIV_PROC_OWNER 特権を保持している場合でも適用されます。PRIV_PROC_ZONE 特権を、そうでなければ特権の付与されたプロセスから削除して、大域ゾーンへの操作に制限できます。

zoneidlist を使用した照合プロセスの詳細は、pgrep(1) および pkill(1) のマニュアルページを参照してください。

ゾーン内でのプロセス ID の可視性

同一ゾーン内のプロセスだけが、kill や priocntl コマンドなどの、プロセス ID を指定するシステムコールインタフェースを介して表示されます。詳細は、kill(1) および priocntl(1) のマニュアルページを参照してください。

ゾーン内のシステム監視機能

ps コマンドに、次の変更が加えられました。

- 出力形式の指定には -o オプションを使用します。このオプションを使用すると、プロセスのゾーン ID またはプロセスを実行中のゾーンの名前を出力できます。
- 指定されたゾーン内のプロセスだけをリスト表示するには、-z *zonelist* オプションを使用します。ゾーンの指定には、ゾーン名またはゾーン ID を使用できます。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。
- プロセスに関連するゾーンの名前を出力するには -Z オプションを使用します。列見出し ZONE の下に名前が出力されます。

詳細は、ps(1) のマニュアルページを参照してください。

-z *zonename* オプションが、次の Oracle Solaris ユーティリティーに追加されました。このオプションを使用して情報をフィルタ処理し、指定したゾーンだけを含めることができます。

- ipcs (ipcs(1) のマニュアルページを参照)
- pgrep (pgrep(1) のマニュアルページを参照)
- ptree (proc(1) のマニュアルページを参照)
- prstat (prstat(1M) のマニュアルページを参照)

コマンドに加えられた変更の全一覧については、表 27–5 を参照してください。

非大域ゾーンのノード名

uname n により返される -/etc/nodename 内のノード名は、ゾーン管理者が設定できます。ノード名は一意でなければなりません。

ファイルシステムと非大域ゾーン

このセクションでは、ゾーンがインストールされている Oracle Solaris システムでファイルシステムを使用する場合の問題について説明します。各ゾーンは、ゾーンの root と呼ばれるディレクトリをルートとする、ファイルシステム階層の独自セクションを保持します。ゾーン内のプロセスは、ゾーンルート以下の階層部分内のファイルだけにアクセスできます。ゾーン内で chroot ユーティリティーを使用できますが、プロセスをゾーン内のルートパスに制限する場合だけです。chroot の詳細については、chroot(1M) のマニュアルページを参照してください。

-o nosuid オプション

mount ユーティリティーで -o nosuid オプションを指定する場合、次の機能を利用できます。

- nosetuid オプションを使用してマウントされたファイルシステム上の setuid バイナリに基づくプロセスは、setuid バイナリの権限では動作しません。プロセスは、バイナリを実行するユーザーの権限で動作します。

  たとえば、ユーザーが root の所有する setuid バイナリを実行する場合、プロセスはそのユーザーの権限で動作します。
- ファイルシステム内のデバイス特殊エントリを開くことはできません。この動作は、nodevices オプションを指定する場合と同じです。

mount(1M) のマニュアルページに記載されているように、このファイルシステム固有オプションは mount ユーティリティーでマウントできるすべての Oracle Solaris ファイルシステムで使用できます。これらのファイルシステムの一覧については、このガイドの 381 ページの「ゾーン内でのファイルシステムのマウント」を参照してください。マウント機能についても説明します。-o nosuid オプションの詳細は、『Solaris のシステム管理 (ネットワークサービス)』の「ネットワークファイルシステムへのアクセス (リファレンス)」を参照してください。

ゾーン内でのファイルシステムのマウント

ファイルシステムをゾーン内部でマウントする場合、nodevices オプションが適用されます。たとえば、ゾーンに、UFS ファイルシステムに対応するブロックデバイス (/dev/dsk/c0t0d0s7) および raw デバイス (/dev/rdsk/c0t0d0s7) へのアクセスが許可される場合、ゾーン内部からマウントを行うと、ファイルシステムのマウントで自動的に nodevices オプションが適用されます。この規則は、zonecfg 構成を使用して指定されたマウントには適用されません。

次の表で、非大域ゾーン内でファイルシステムをマウントする場合のオプションを説明します。これらの補助的なマウント方法の実行手順については、265 ページの「ゾーンを構成、検証、および確定する」および 418 ページの「稼働中の非大域ゾーン内でファイルシステムをマウントする」を参照してください。

/usr/lib/*fstype*/mount 内にマウントバイナリが存在する場合、表に含まれない任意のファイルシステムタイプを構成内で指定できます。

ファイルシステム	非大域ゾーン内のマウントオプション
AutoFS	マウントに zonecfg を使用することはできません。大域ゾーンから非大域ゾーン内に手動でマウントすることもできません。ゾーン内部からマウントすることは可能です。
CacheFS	非大域ゾーン内では使用できません。
FDFS	zonecfg を使用してマウントできます。大域ゾーンから非大域ゾーン内に手動でマウントできます。ゾーン内部からマウントできます。
HSFS	zonecfg を使用してマウントできます。大域ゾーンから非大域ゾーン内に手動でマウントできます。ゾーン内部からマウントできます。
LOFS	zonecfg を使用してマウントできます。大域ゾーンから非大域ゾーン内に手動でマウントできます。ゾーン内部からマウントできます。
MNTFS	マウントに zonecfg を使用することはできません。大域ゾーンから非大域ゾーン内に手動でマウントすることもできません。ゾーン内部からマウントすることは可能です。
NFS	マウントに zonecfg を使用できません。ゾーン内で現在サポートされているバージョンである V2、V3、および V4 を、ゾーン内からマウントできます。

ファイルシステム	非大域ゾーン内のマウントオプション
PCFS	`zonecfg` を使用してマウントできます。大域ゾーンから非大域ゾーン内に手動でマウントできます。ゾーン内部からマウントできます。
PROCFS	マウントに `zonecfg` を使用することはできません。大域ゾーンから非大域ゾーン内に手動でマウントすることもできません。ゾーン内部からマウントすることは可能です。
TMPFS	`zonecfg` を使用してマウントできます。大域ゾーンから非大域ゾーン内に手動でマウントできます。ゾーン内部からマウントできます。
UDFS	`zonecfg` を使用してマウントできます。大域ゾーンから非大域ゾーン内に手動でマウントできます。ゾーン内部からマウントできます。
UFS	`zonecfg` を使用してマウントできます。大域ゾーンから非大域ゾーン内に手動でマウントできます。ゾーン内部からマウントできます。
XMEMFS	`zonecfg` を使用してマウントできます。大域ゾーンから非大域ゾーン内に手動でマウントできます。ゾーン内部からマウントできます。 このファイルシステムのサポートは、Oracle Solaris システムの将来のリリースから削除される予定です。
ZFS	`zonecfg dataset` および `fs` リソースタイプを使用してマウントできます。

詳細は、266 ページの「ゾーンの構成方法」、418 ページの「稼働中の非大域ゾーン内でファイルシステムをマウントする」、および `mount(1M)` のマニュアルページを参照してください。

ゾーン内でのファイルシステムのマウント解除

ファイルシステムをマウント解除できるかどうかは、どの管理者がそのファイルシステムを最初にマウントしたかによって決まります。`zonecfg` コマンドを使用してゾーンを構成するときにファイルシステムが指定されている場合は、大域ゾーンがそのマウントの所有者となるので、非大域ゾーンの管理者はそのファイルシステムのマウントを解除することはできません。非大域ゾーンの `/etc/vfstab` ファイルにマウントを指定するなどの方法でファイルシステムが非大域ゾーンからマウントされている場合には、その非大域ゾーンの管理者はそのファイルシステムのマウントを解除できます。

セキュリティーの制限およびファイルシステムの動作

ゾーン内部から特定のファイルシステムをマウントする場合、適用されるセキュリティーの制限が存在します。ほかのファイルシステムは、ゾーン内でマウントされたときに特有の動作を行います。変更されたファイルシステムの一覧を、次に示します。

AutoFS
: autofs は、自動的に適切なファイルシステムをマウントするためのクライアント側のサービスです。クライアントが現在マウントされていないファイルシステムにアクセスしようとすると、AutoFS ファイルシステムはその要求に介入し、`automountd` を呼び出して要求されたディレクトリをマウントします。ゾーン内で確立された AutoFS マウントは、そのゾーンだけで有効です。大域ゾーンを含むほかのゾーンからそのマウントにアクセスすることはできません。ゾーンが停止またはリブートすると、マウントは削除されます。AutoFS の詳細は、『Solaris のシステム管理 (ネットワークサービス)』の「autofs のしくみ」を参照してください。

  各ゾーンは、`automountd` の独自コピーを実行します。自動マップおよびタイムアウトは、ゾーン管理者により制御されます。大域ゾーンから非大域ゾーンの AutoFS マウントポイントを横断的に使用して、別のゾーン内でマウントを始動させることはできません。

  別のマウントが始動する際、カーネル内で特定の AutoFS マウントが作成されます。この種のマウントは、一括してマウントまたはマウント解除する必要があるため、通常の `umount` インタフェースを使用して削除することはできません。この機能は、ゾーンの停止処理用であることに注意してください。

MNTFS
: MNTFS は、ローカルシステムのマウント済みファイルシステムのテーブルに読み取り専用アクセスを提供する仮想ファイルシステムです。非大域ゾーン内から `mnttab` を使用して表示可能なファイルシステムのセットは、ゾーン内でマウントされたファイルシステムセットおよびルート (/) のエントリで構成されます。`/dev/rdsk/c0t0d0s0` などの、ゾーン内からアクセス不可能な特殊なデバイスを保持するマウントポイントは、マウントポイントと同じ特殊なデバイスセットを保持します。システム内のすべてのマウントが、大域ゾーンの `/etc/mnttab` テーブルから表示可能になります。MNTFS の詳細は、『Solaris のシステム管理 (デバイスとファイルシステム)』の「Oracle Solaris ファイルシステムのマウントおよびマウント解除」を参照してください。

NFS
: ゾーン内部で確立された NFS マウントは、そのゾーンでのみ有効です。大域ゾーンを含むほかのゾーンからそのマウントにアクセスすることはできません。ゾーンが停止またはリブートすると、マウントは削除されます。

mount_nfs(1M) のマニュアルページに記載されているように、NFS サーバーが独自のファイルシステムのマウントを試みることはできません。このため、大域ゾーンによりエクスポートされたファイルシステムを、ゾーンが NFS マウントしてはいけません。ゾーンを NFS サーバーにすることはできません。ゾーン内部からの NFS マウントは、nodevices オプションを使用してマウントされたかのように動作します。

nfsstat コマンドの出力は、コマンドが実行されたゾーンにのみ関係があります。たとえば、コマンドが大域ゾーン内で実行される場合、大域ゾーンに関する情報だけが出力されます。nfsstat コマンドの詳細は、nfsstat(1M) のマニュアルページを参照してください。

開かれているファイルのいずれか、またはそのアドレス空間のいずれかの部分が NFS 上に存在する場合、zlogin コマンドは失敗します。詳細は、307 ページの「zlogin コマンド」を参照してください。

PROCFS
: PROCFS とも呼ばれる /proc ファイルシステムは、プロセスの可視性とアクセス制限、およびプロセスのゾーン関連性に関する情報を提供します。/proc では、同じゾーン内のプロセスだけを表示できます。

  大域ゾーン内のプロセスから、非大域ゾーン内のプロセスおよびほかのオブジェクトを監視できます。これにより、この種のプロセスがシステム規模の監視機能を備えることが可能になります。

  ゾーン内部からは、procfs マウントは nodevices オプションを使用してマウントされたかのように動作します。procfs の詳細は、proc(4) のマニュアルページを参照してください。

LOFS
: LOFS でマウント可能なファイルシステムの範囲は、ゾーンで表示可能なファイルシステム部分に限定されています。このため、ゾーン内での LOFS マウントには制限はありません。

UFS、UDFS、PCFS、およびその他のストレージに基づいたファイルシステム
: zonecfg コマンドを使用して、UFS などの fsck バイナリを保持するストレージに基づいたファイルシステムを構成する場合、ゾーン管理者は raw パラメータを指定する必要があります。このパラメータは、/dev/rdsk/c0t0d0s7 などの raw (文字) デバイスを示します。zoneadmd は、このデバイス上で fsck コマンドを非対話型の検査専用モード (fsck -m) で自動実行してから、ファイルシステムをマウントします。fsck が失敗した場合、zoneadmd を使用してゾーンを準備完了状態にすることはできません。raw により指定されるパスを、相対パスにするこ とはできません。

  /usr/lib/*fstype*/fsck 内で fsck バイナリを提供しないファイルシステムで、デバイスを fsck に指定するのは誤りです。また、そのファイルシステムに fsck バイナリが存在する場合に、デバイスを fsck に指定しないことも誤りです。

詳細は、284 ページの「zoneadmd デーモン」および fsck(1m) のマニュアルページを参照してください。

ZFS

zonecfg コマンドに add dataset リソースを指定して実行することにより、非大域ゾーンに ZFS データセットを追加できます。データセットは非大域ゾーンでマウントされ可視になり、大域ゾーンでは不可視になります。ゾーン管理者は、そのデータセット内のファイルシステムの作成と破棄、クローンの作成と破棄、およびデータセットのプロパティーの変更を行うことができます。

zfs の zoned 属性は、データセットが非大域ゾーンに追加されたかどうかを示します。

```
# zfs get zoned tank/sales
NAME          PROPERTY    VALUE       SOURCE
tank/sales    zoned       on          local
```

大域ゾーンのデータセットを共有する場合は、zonecfg コマンドとともに add fs サブコマンドを使用して、LOFS マウントした ZFS ファイルシステムを追加できます。大域管理者は、データセットのプロパティーの設定および制御を担当します。

ZFS の詳細については、『Oracle Solaris ZFS 管理ガイド』の第 10 章「Oracle Solaris ZFS の高度なトピック」を参照してください。

NFS クライアントとして機能する非大域ゾーン

ゾーンは、NFS クライアントとしても機能できます。バージョン 2、バージョン 3、およびバージョン 4 プロトコルがサポートされます。これらの NFS バージョンの詳細は、『Solaris のシステム管理 (ネットワークサービス)』の「NFS サービスの機能」を参照してください。

デフォルトのバージョンは、NFS バージョン 4 です。次のいずれかの方法を使用して、クライアント上でほかの NFS バージョンを有効にできます。

- /etc/default/nfs を編集して NFS_CLIENT_VERSMAX=*number* を設定することで、指定したバージョンをゾーンのデフォルトとして使用できます。『Solaris のシステム管理 (ネットワークサービス)』の「NFS サービスの設定」を参照してください。タスクマップの「/etc/default/nfs ファイルを変更して、クライアント上で異なるバージョンの NFS を選択する方法」に記載された手順を実行します。
- バージョンマウントを手動で作成できます。この方法を使用すると、/etc/default/nfs の内容がオーバーライドされます。『Solaris のシステム管理 (ネットワークサービス)』の「NFS サービスの設定」を参照してください。タスクマップの「コマンド行を使用して、クライアント上で異なるバージョンの NFS を選択する方法」に記載された手順を実行します。

ゾーン内での mknod の使用禁止

mknod(1M) のマニュアルページに記載されているように、mknod コマンドを使用して、非大域ゾーン内で特殊ファイルを作成することはできません。

ファイルシステムの行き来

ゾーンのファイルシステム名前空間は、大域ゾーンからアクセス可能な名前空間の一部です。大域ゾーン内の特権のないプロセスが、非大域ゾーンのファイルシステム階層を行き来することはできません。これには、次のような理由があります。

- root だけが、ゾーンルートの親ディレクトリを所有、読み込み可能、書き込み可能、および実行可能に指定する
- /proc によりエクスポートされたディレクトリへのアクセスが制限される

別のゾーン用にマウントされた AutoFS ノードへのアクセスを試みても、失敗します。大域管理者は、その子孫がほかのゾーンに含まれる自動マップを保持してはいけません。

大域ゾーンから非大域ゾーンにアクセスする際の制限

非大域ゾーンのインストール後に、システムのバックアップユーティリティー以外のコマンドを使用して、大域ゾーンからそのゾーンに直接アクセスしてはいけません。また、非大域ゾーンを未知の環境に公開した後は、そのゾーンが安全であると考えることはできません。たとえば、公開されたネットワーク上に配置されたゾーンについて考えてみましょう。この場合、ゾーンのセキュリティーが低下し、ファイルシステムの内容が変更される可能性があります。セキュリティーが低下する可能性がある場合、大域管理者はゾーンを信頼できないものとして処理する必要があります。

次の両方の条件が当てはまる場合、-R オプションまたは -b オプション (またはこれと同等なオプション) を使って代替ルートを指定可能なコマンドを使用してはいけません。

- コマンドが大域ゾーン内で実行される。
- 代替ルートが非大域ゾーン内のいずれかのパスを参照している。現在動作中のシステムの大域ゾーンからの相対パスの場合や、代替ルート内の大域ゾーンからの相対パスの場合の場合を含む。

たとえば、pkgadd ユーティリティーに -R *root_path* オプションを指定して、非大域ゾーンのルートパスを保持する大域ゾーンから実行する場合が、この条件に該当します。

次に、-R と代替ルートパスを使用するコマンド、プログラム、およびユーティリティーを一覧表示します。

- auditreduce
- bart
- flar
- flarcreate
- installf
- localeadm
- makeuuid
- metaroot
- patchadd
- patchrm
- pkgadd
- pkgadm
- pkgask
- pkgchk
- pkgrm
- prodreg
- removef
- routeadm
- showrev
- syseventadm

次に、-b と代替ルートパスを使用するコマンドおよびプログラムを一覧表示します。

- add_drv
- pprosetup
- rem_drv
- roleadd
- sysidconfig
- update_drv
- useradd

共有 IP 非大域ゾーンにおけるネットワーク

ゾーンがインストールされている Oracle Solaris システムでは、ゾーンはネットワーク経由で互いに通信できます。ゾーンはすべて別個の結合または接続を保持します。また、すべてのゾーンは独自のサーバーデーモンを実行できます。これらのデーモンは、同一のポート番号で競合することなく待機できます。IP スタックは、着信接続の IP アドレスを考慮に入れることで競合を解決します。IP アドレスにより、ゾーンが識別されます。

共有 IP ゾーンの区分化

ゾーンをサポートするシステム内の IP スタックは、ゾーン間のネットワークトラフィックの分離を実装します。IP トラフィックを受け取るアプリケーションは、同じゾーンに送信されたトラフィックの受信だけを実行できます。

システム上の各論理インタフェースは、特定のゾーンに所属します。デフォルトは、大域ゾーンです。`zonecfg` ユーティリティー経由でゾーンに割り当てられた論理ネットワークインタフェースは、ネットワーク経由での通信に使用されます。各ストリームおよび接続は、それを開いたプロセスのゾーンに所属します。

上位層ストリームと論理インタフェース間の結合は、制限されます。ストリームが確立できるのは、同一ゾーン内の論理インタフェースへの結合だけです。同様に、論理インタフェースからのパケットを渡すことができるのは、論理インタフェースと同じゾーン内の上位層ストリームに対してだけです。

各ゾーンは、独自のバインドセットを保持します。アドレスが使用中であるため、各ゾーンは、同一のポート番号で待機する同じアプリケーションを、バインドが失敗することなく稼働可能です。各ゾーンは、次に示すサービスの独自のバージョンを実行できます。

- 完全な構成ファイルを保持するインターネットサービスデーモン (`inetd(1M)` のマニュアルページを参照)
- `sendmail` (`sendmail(1M)` のマニュアルページを参照)
- `apache` (`apache(1M)` のマニュアルページを参照)

大域ゾーン以外のゾーンは、ネットワークへのアクセスが制限されています。標準の TCP および UDP ソケットインタフェースが利用可能ですが、`SOCK_RAW` ソケットインタフェースは ICMP (Internet Control Message Protocol) に制限されています。ICMP は、ネットワークのエラー状況を検出および報告したり、`ping` コマンドを使用するのに必要です。

共有 IP ネットワークインタフェース

ネットワーク接続を必要とする非大域ゾーンには、それぞれ 1 つ以上の専用 IP アドレスがあります。これらのアドレスは、`ifconfig` コマンドを使ってゾーン内に配置可能な論理ネットワークインタフェースに関連付けられています。`zonecfg` により構成されるゾーンネットワークインタフェースは、ブート時に自動的に設定されてゾーン内に配置されます。`ifconfig` コマンドを使用すると、ゾーンの稼働中に論理インタフェースを追加または削除できます。インタフェース構成およびネットワーク経路を変更できるのは、大域管理者だけです。

非大域ゾーン内部では、そのゾーンのインタフェースだけを `ifconfig` で表示できます。

詳細は、ifconfig(1m) および if_tcp(7P) のマニュアルページを参照してください。

同一マシン上の共有 IP ゾーン間の IP トラフィック

次の表に示す宛先およびゾーンの「一致する経路」が存在する場合にのみ、同一マシン上の 2 つのゾーン間でパケットの配信が許可されます。

一致情報は、次のように実装されます。

- パケットの送信元アドレスが、一致する経路で指定された出力インタフェース上で選択されます。
- デフォルトでは、同一のサブネット上にアドレスを保持する 2 つのゾーン間のトラフィックが許可されます。この場合の一致する経路は、サブネットのインタフェース経路です。
- 指定されたゾーンのデフォルト経路が存在する場合 (ゲートウェイはゾーンのいずれかのサブネット上にある)、このゾーンからほかのすべてのゾーンへのトラフィックが許可されます。この場合の一致する経路は、デフォルト経路です。
- RTF_REJECT フラグの付いた一致する経路が存在する場合、パケットにより ICMP 到達不能メッセージがトリガーされます。RTF_BLACKHOLE フラグの付いた一致する経路が存在する場合、パケットは破棄されます。大域管理者は、次の表に示す route コマンドオプションをこれらのフラグとともに使用して、経路を作成できます。

修飾子	フラグ	説明
-reject	RTF_REJECT	一致した場合に、ICMP 到達不能メッセージを発行します。
-blackhole	RTF_BLACKHOLE	更新時に、メッセージを表示せずにパケットを破棄します。

詳細は、route(1M) のマニュアルページを参照してください。

共有 IP ゾーンでの Oracle Solaris IP フィルタ

Oracle Solaris IP フィルタは、ステートフルパケットフィルタリングとネットワークアドレス変換 (NAT) を行います。ステートフルパケットフィルタは、アクティブな接続の状態を監視し、取得した情報を使用して、ファイアウォールの通過を許可するネットワークパケットを決定することができます。Oracle Solaris IP フィルタには、ス

テートレスパケットフィルタリングと、アドレスプールの作成および管理を行う機能もあります。詳細は、『Solaris のシステム管理 (IP サービス)』の第 25 章「Oracle Solaris の ＩＰ フィルタ (概要)」を参照してください。

『Solaris のシステム管理 (IP サービス)』の第 26 章「IP フィルタ (手順)」に記載されているように、ループバックフィルタリングをオンに設定することで、非大域ゾーンで Oracle Solaris IP フィルタを有効にできます。

Oracle Solaris IP フィルタは、オープンソースの IP Filter ソフトウェアを基にしています。

共有 IP ゾーン内の IP ネットワークマルチパス

IP ネットワークマルチパス (IPMP) は、同一の IP リンク上に複数のインタフェースを保持するシステムで、物理インタフェースの障害検出および透過的なネットワークアクセスフェイルオーバーを提供します。IPMP も、複数のインタフェースを保持するシステムについて、パケットの負荷分散を提供します。

すべてのネットワーク構成は、大域ゾーン内で行われます。大域ゾーン内で IPMP を構成した後で、この機能を非大域ゾーンに拡張できます。ゾーンの構成時に、ゾーンのアドレスを IPMP グループ内に配置することでこの機能は拡張されます。その後、大域ゾーン内のいずれかのインタフェースで障害が発生すると、非大域ゾーンアドレスが別のインタフェースカードに移されます。1 つの共有 IP ゾーンが複数の IP アドレスを持つこと、共有 IP ゾーンが複数の IPMP グループに属すること、および特定の IPMP グループを複数の共有 IP ゾーンで使用することが可能です。

指定された非大域ゾーン内で、ゾーンに関連するインタフェースだけを `ifconfig` コマンドを使用して表示できます。

425 ページの「IP ネットワークマルチパス機能を共有 IP 非大域ゾーンに拡張する方法」を参照してください。ゾーンの構成手順については、266 ページの「ゾーンの構成方法」を参照してください。IPMP の機能、コンポーネント、および使用方法の詳細は、『Solaris のシステム管理 (IP サービス)』の第 30 章「IPMP の紹介 (概要)」を参照してください。

Oracle Solaris 10 8/07: 排他的 IP 非大域ゾーンでのネットワーク

排他的 IP ゾーンは、IP 関連の状態とチューニング変数を独自に保持します。ゾーンの構成時に、独自のデータリンクセットがゾーンに割り当てられます。

排他的 IP 非大域ゾーンで使用できる機能については、237 ページの「Solaris 10 8/07: 排他的 IP 非大域ゾーン」を参照してください。IP ndd 変数のチューニングについては、『Oracle Solaris カーネルのチューンアップ・リファレンスマニュアル』を参照してください。

排他的 IP ゾーンの区分化

排他的 IP ゾーンはそれぞれ個別の TCP/IP スタックを持っているため、下位のデータリンク層まで分離されます。排他的 IP ゾーンには、1 つ以上のデータリンク名が大域管理者によって割り当てられます。データリンク名は、NIC または NIC 上の VLAN の場合があります。ゾーン管理者は、大域ゾーンの場合と同じ柔軟性とオプションで、これらのデータリンクの IP を構成できます。

排他的 IP データリンクインタフェース

1 つのデータリンク名は、1 つのゾーンだけに割り当てる必要があります。

`dladm show-link` コマンドを使用して、実行中のゾーンに割り当てられているデータリンクを表示できます。

詳細は、`dladm(1M)` のマニュアルページを参照してください。

同一マシン上の排他的 IP ゾーン間の IP トラフィック

排他的 IP ゾーン間での IP パケットの内部ループバックはありません。すべてのパケットが下位のデータリンクまで送信されます。これは通常、パケットがネットワークインタフェース上に送信されることを意味します。その後、Ethernet スイッチや IP ルーターなどのデバイスがパケットを宛先へ転送します。この送信先は、送信元と同じマシン上の別のゾーンである可能性もあります。

排他的 IP ゾーンにおける Oracle Solaris IP フィルタ

排他的 IP ゾーンでは、大域ゾーンのものと同じ IP フィルタ機能を使用できます。排他的 IP ゾーンでの IP フィルタの構成方法も大域ゾーンと同じです。

排他的 IP ゾーン内の IP ネットワークマルチパス

IP ネットワークマルチパス (IPMP) は、同一の IP リンク上に複数のインタフェースを保持するシステムで、物理インタフェースの障害検出および透過的なネットワークアクセスフェイルオーバーを提供します。IPMP は、耐障害性に加えて、複数のインタフェースを保持するシステムについて、パケットの負荷分散も提供します。

データリンク構成は、大域ゾーン内で行われます。最初に、`zonecfg` を使用して、複数のデータリンクインタフェースをゾーンに割り当てます。複数のデータリンクインタフェースを同じ IP サブネットに接続する必要があります。その後、排他的 IP ゾーン内からゾーン管理者が IPMP を構成することができます。複数の IPMP グループを特定の排他的 IP ゾーンに割り当てることができますが、それらの IPMP グループを他のゾーンと共有することはできません。

非大域ゾーンでのデバイスの使用

あるゾーンのプロセスが別のゾーンで実行中のプロセスに干渉することがないよう、ゾーン内部で利用可能なデバイスセットには制限が課されています。たとえば、ゾーン内のプロセスが、カーネルメモリーおよびルートディスクの内容を変更することはできません。このため、デフォルトでは、ゾーン内で安全に利用可能であると見なされる特定の仮想デバイスだけを使用できます。`zonecfg` ユーティリティーを使用すると、利用可能なデバイスを特定のゾーンに追加できます。

`/dev` および `/devices` 名前空間

Oracle Solaris システムでは、`/devices` を管理するために、devfs(7FS) のマニュアルページに記載されている `devfs` ファイルシステムが使用されます。この名前空間内の各要素は、ハードウェアデバイス、仮想デバイス、またはネクサスデバイスへの物理パスを表します。名前空間には、デバイスツリーが反映されます。したがって、ファイルシステムは、ディレクトリおよびデバイス特殊ファイルの階層により生成されます。

現在はルート (`/`) ファイルシステムの一部になっている `/dev` ファイル階層は、`/devices` 内に存在する物理パスへのシンボリックリンク (論理パス) で構成されています。アプリケーションは、`/dev` 内に存在するデバイスへの論理パスを参照します。`/dev` ファイルシステムは、読み取り専用のマウントを使用して、ゾーン内にループバックマウントされます。

`/dev` ファイル階層を管理するシステムは、次に示すコンポーネントで構成されます。

- `devfsadm` (devfsadm(1M) のマニュアルページを参照)
- `syseventd` (syseventd(1M) のマニュアルページを参照)

- libdevinfo デバイス情報ライブラリ (libdevinfo(3LIB) のマニュアルページを参照)
- devinfo ドライバ (devinfo(7D) のマニュアルページを参照)
- RCM (Reconfiguration Coordination Manager) (『Solaris のシステム管理 (デバイスとファイルシステム)』の「Reconfiguration Coordination Manager (RCM) スクリプトの概要」を参照)

注意 – /devices パス名に依存するサブシステムは、/dev パス名が確立されるまで、非大域ゾーンでは実行できません。

排他使用のデバイス

デバイスを特定のゾーンに割り当てることが必要な場合があります。特権のないユーザーがブロックデバイスにアクセスできるようにすると、これらのデバイスの使用が許可されて、システムパニックやバスリセットなどの不具合が生じる場合があります。この種の割り当てを行う前に、次の点を考慮してください。

- SCSI テープデバイスを特定のゾーンに割り当てる前に、sgen(7D) のマニュアルページを参照してください。
- 物理デバイスを複数のゾーンに配置する場合、ゾーン間に隠れたチャネルが作成される場合があります。大域ゾーンアプリケーションでこの種のデバイスを使用すると、非大域ゾーンによるデータ整合性の損失や、データの破壊が発生する危険があります。

デバイスドライバの管理

modinfo(1M) のマニュアルページに記載されているように、非大域ゾーン内で modinfo コマンドを使用して、読み込まれたカーネルモジュールのリストを検査できます。

カーネル、デバイス、およびプラットフォームの管理に関係した大半の操作は、非大域ゾーンの内部では機能しません。これは、プラットフォームハードウェア構成を変更すると、ゾーンのセキュリティーモデルに違反するためです。これらの操作には、次のことが含まれます。

- ドライバの追加および削除
- カーネルモジュールの明示的な読み込みおよび読み込み解除
- 動的再構成 (DR) 操作の開始
- 物理プラットフォームの状態に影響を与える機能の使用

非大域ゾーンで動作しないか、変更されるユーティリティー

非大域ゾーンで動作しないユーティリティー

次のユーティリティーは、通常は使用できないデバイスに依存しているため、ゾーン内では動作しません。

- `cdrecord` (`/usr/share/man/man1` ディレクトリのマニュアルページを参照してください。)
- `cdrw` (`cdrw(1)` のマニュアルページを参照)
- `rmformat` (`rmformat(1)` のマニュアルページを参照)
- `add_drv` (`add_drv(1M)` のマニュアルページを参照)
- `disks` (`disks(1M)` のマニュアルページを参照)
- `prtconf` (`prtconf(1M)` のマニュアルページを参照)
- `prtdiag` (`prtdiag(1M)` のマニュアルページを参照)
- `rem_drv` (`rem_drv(1M)` のマニュアルページを参照)

SPARC: 非大域ゾーンでの使用に合わせて変更されたユーティリティー

`eeprom` ユーティリティーをゾーン内で使用して、設定を表示できます。このユーティリティーを使用して、設定を変更することはできません。詳細は、`eeprom(1M)` および `openprom(7D)` のマニュアルページを参照してください。

非大域ゾーンでのアプリケーションの実行

通常は、すべてのアプリケーションを非大域ゾーンで実行できます。ただし、次のタイプのアプリケーションは、この環境に適さない場合があります。

- システム全体に影響を与える特権付きの操作を実行するアプリケーション。大域システムクロックを設定したり、物理メモリーをロックダウンする操作が、これに該当します。
- `/dev/kmem` などの、非大域ゾーン内に存在しない特定のデバイスに依存するいくつかのアプリケーション。
- 実行時、あるいはインストール、パッチの適用、またはアップグレードを行う際に、`/usr` に書き込み可能であることを前提とするアプリケーション。これは、デフォルトでは、非大域ゾーンで `/usr` が読み取り専用であるためです。時には、この種のアプリケーションに関連する問題を、アプリケーション自体を変更せずに回避できる場合があります。

- 共有 IP ゾーンでは、アプリケーションは /dev/ip 内のデバイスに依存します。

非大域ゾーンで使用されるリソース制御

ゾーン内でのリソース管理機能の使用に関する追加情報については、このガイドの第 1 部の、この機能について説明した章を参照してください。

リソース管理の章に記載されたリソース制御および属性はすべて、大域ゾーンおよび非大域ゾーンの /etc/project ファイル、NIS マップ、または LDAP ディレクトリサービスで設定できます。指定されたゾーンの設定は、そのゾーンにのみ影響を及ぼします。異なるゾーン内で自動実行中のプロジェクトは、ゾーンごとに別個の制御セットを保持できます。たとえば、大域ゾーン内のプロジェクト A を project.cpu-shares=10 に設定し、非大域ゾーン内の Project A を project.cpu-shares=5 に設定できます。それぞれが該当するゾーン内でのみ機能する、rcapd のインスタンスをシステム上で複数実行できます。

ゾーン内部のプロジェクト、タスク、およびプロセスを制御するため、ゾーン内で使用するリソースの制御および属性は、プールおよびゾーン規模のリソース制御に関する追加要件に従います。

非大域ゾーンには、「1 ゾーン、1 プール」という規則が適用されます。1 つのプール内のリソースを複数の非大域ゾーンが共有してもかまいません。ただし、十分な特権を付与されたプロセスを使って、大域ゾーン内のプロセスを任意のプールにバインドすることが可能です。リソースコントローラ poold は、大域ゾーン内だけで動作します。大域ゾーン内には、リソースコントローラが動作するプールが複数存在します。poolstat ユーティリティーを非大域ゾーンで実行すると、そのゾーンに関連付けられているプールの情報だけが表示されます。非大域ゾーンで引数なしで pooladm コマンドを実行すると、そのゾーンに関連付けられているプールの情報だけが表示されます。

ゾーン規模のリソース制御が project ファイルで設定されている場合、そのリソース制御は有効にはなりません。ゾーン規模のリソース制御は、zonecfg ユーティリティーを使って設定されます。

ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラ

このセクションでは、ゾーンで公平配分スケジューラ (FSS) を使用する方法について説明します。

非大域ゾーン内の FSS 配分分割

ゾーンの FSS CPU 配分は、階層的です。大域ゾーンおよび非大域ゾーンの配分は、ゾーン規模のリソース制御 zone.cpu-shares を使って大域管理者が設定します。次に、そのゾーン内のプロジェクトごとにリソース制御 project.cpu-shares を定義して、ゾーン規模の制御で設定された配分をさらに分割できます。

zonecfg コマンドを使用してゾーンに配分を割り当てる方法については、277 ページの「大域ゾーンの zone.cpu-shares を設定する方法」を参照してください。project.cpu-shares の詳細は、87 ページの「使用可能なリソース制御」を参照してください。配分を一時的に設定する方法を示す手順例については、429 ページの「ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用」も参照してください。

ゾーン間の配分均衡

zone.cpu-shares を使用して、FSS 配分を大域ゾーンと非大域ゾーンに割り当てることができます。FSS がシステムのデフォルトのスケジューラになっている場合で、配分が割り当てられていないときは、大域ゾーンも含め各ゾーンにはデフォルトで 1 つの配分が付与されます。システムに 1 つの非大域ゾーンが存在し、zone.cpu-shares を使ってこのゾーンに 2 つの配分を付与する場合、これにより非大域ゾーンが大域ゾーンとの関連で受ける CPU の比率が定義されます。2 つのゾーン間の CPU 比率は 2:1 です。

ゾーンがインストールされている Oracle Solaris システムでの拡張アカウンティング

拡張アカウンティングサブシステムを大域ゾーンで実行した場合、非大域ゾーンを含むシステム全体の情報が収集および報告されます。大域管理者は、ゾーンごとのリソース消費を決定することもできます。

拡張アカウンティングサブシステムを使用すると、プロセスおよびタスクに基づいたアカウンティングに対応した異なるアカウンティング設定およびファイルを、ゾーン単位で指定することが可能になります。exacct レコードに、プロセスの場合はゾーン名 EXD_PROC_ZONENAME のタグを付け、タスクの場合はゾーン名 EXD_TASK_ZONENAME のタグを付けることができます。アカウンティングレコードは、大域ゾーンのアカウンティングファイルおよびゾーン単位のアカウンティングファイルに書き込まれます。EXD_TASK_HOSTNAME、EXD_PROC_HOSTNAME、および EXD_HOSTNAME レコードには、大域ゾーンのノード名の代わりに、プロセスまたはタスクが実行されたゾーンの uname -n 値が含まれます。

IPQoS フローアカウンティングの詳細は、『Solaris のシステム管理 (IP サービス)』の第 36 章「フローアカウンティングの使用と統計情報の収集 (手順)」を参照してください。

非大域ゾーン内の特権

プロセスは、特権の一部に制限されています。特権を制限することで、ほかのゾーンに影響を及ぼす可能性がある操作がゾーンで実行されないようにします。特権セットにより、特権が付与されたユーザーがゾーン内で実行可能な機能が制限されます。ゾーン内で利用可能な特権のリストを表示するには、`ppriv` ユーティリティーを使用します。

次の表に、Oracle Solaris の特権すべて、およびゾーン内での各特権のステータスを示します。省略可能な特権は、デフォルト特権セットの一部ではありませんが、`limitpriv` プロパティーを使って指定できます。必須の特権が、生成される特権セットに含まれている必要があります。禁止された特権を、生成される特権セットに含めることはできません。

Oracle Solaris 10 11/06 リリースから、`limitpriv` プロパティーが使用可能になりました。

表 27–1 ゾーン内の特権のステータス

特権	ステータス	注意事項
`cpc_cpu`	任意	特定の `cpc`(3CPC) カウンタへのアクセス
`dtrace_proc`	任意	`fasttrap` および `pid` プロバイダ。`plockstat`(1M)
`dtrace_user`	任意	`profile` および `syscall` プロバイダ
`graphics_access`	任意	`ioctl`(2) による `agpgart_io`(7I) へのアクセス
`graphics_map`	任意	`mmap`(2) による `agpgart_io`(7I) へのアクセス
`net_rawaccess`	共有 IP ゾーンではオプション。 排他的 IP ゾーンではデフォルト。	raw `PF_INET/PF_INET6` パケットアクセス
`proc_clock_highres`	任意	高解像度タイマーの使用
`proc_priocntl`	任意	スケジューリングの制御。`priocntl`(1)
`sys_ipc_config`	任意	IPC メッセージキューのバッファサイズの引き上げ

表 27–1 ゾーン内の特権のステータス (続き)

特権	ステータス	注意事項
sys_time	任意	システム時間の操作。xntp(1M)
dtrace_kernel	禁止	現在、未サポート
proc_zone	禁止	現在、未サポート
sys_config	禁止	現在、未サポート
sys_devices	禁止	現在、未サポート
sys_linkdir	禁止	現在、未サポート
sys_net_config	禁止	現在、未サポート
sys_res_config	禁止	現在、未サポート
sys_suser_compat	禁止	現在、未サポート
proc_exec	必須、デフォルト	init(1M) の起動に使用
proc_fork	必須、デフォルト	init(1M) の起動に使用
sys_mount	必須、デフォルト	必須ファイルシステムのマウントに必要
sys_ip_config	必須、排他的 IP ゾーンではデフォルト 共有 IP ゾーンでは禁止	ゾーンのブートおよび排他的 IP ゾーンの IP ネットワークの初期化に必要
contract_event	デフォルト	契約ファイルシステムで使用
contract_observer	デフォルト	UID とは無関係な契約観察
file_chown	デフォルト	ファイル所有権の変更
file_chown_self	デフォルト	所有するファイルの所有者/グループの変更
file_dac_execute	デフォルト	モード/ACL に依存しない実行アクセス
file_dac_read	デフォルト	モード/ACL に依存しない読み取りアクセス
file_dac_search	デフォルト	モード/ACL に依存しない検索アクセス
file_dac_write	デフォルト	モード/ACL に依存しない書き込みアクセス
file_link_any	デフォルト	所有者に依存しないリンクアクセス
file_owner	デフォルト	所有者に依存しないその他のアクセス
file_setid	デフォルト	setid、setgid、 setuid ファイルのアクセス権の変更

表 27–1 ゾーン内の特権のステータス (続き)

特権	ステータス	注意事項
ipc_dac_read	デフォルト	モードに依存しない IPC 読み取りアクセス
ipc_dac_owner	デフォルト	モードに依存しない IPC 書き込みアクセス
ipc_owner	デフォルト	モードに依存しないその他の IPC アクセス
net_icmpaccess	デフォルト	ICMP パケットアクセス: ping(1M)
net_privaddr	デフォルト	特権ポートへのバインド
proc_audit	デフォルト	監査レコードの生成
proc_chroot	デフォルト	root ディレクトリの変更
proc_info	デフォルト	プロセスの検査
proc_lock_memory	デフォルト	メモリーのロック。shmctl(2) および mlock(3C) この特権がシステム管理者によって非大域ゾーンに割り当てられている場合、ゾーンがすべてのメモリーをロックするのを防ぐために zone.max-locked-memory リソース制御の設定も検討してください。
proc_owner	デフォルト	所有者に依存しないプロセス制御
proc_session	デフォルト	セッションに依存しないプロセス制御
proc_setid	デフォルト	ユーザー/グループ ID の任意設定
proc_taskid	デフォルト	呼び出し元へのタスク ID の割り当て
sys_acct	デフォルト	アカウンティングの管理
sys_admin	デフォルト	単純なシステム管理タスク
sys_audit	デフォルト	監査の管理
sys_nfs	デフォルト	NFS クライアントのサポート
sys_resource	デフォルト	リソース制限の操作

次の表に、Oracle Solaris Trusted Extensions の特権すべて、および各特権のゾーン内のステータスを示します。省略可能な特権は、デフォルト特権セットの一部ではありませんが、limitpriv プロパティーを使って指定できます。

注 – これらの特権が解釈されるのは、システムが Oracle Solaris Trusted Extensions を使って構成されている場合だけです。

表 27–2 ゾーン内での Oracle Solaris Trusted Extensions の特権のステータス

Oracle Solaris Trusted Extensions の特権	ステータス	注意事項
file_downgrade_sl	任意	ファイルまたはディレクトリの機密ラベルを、既存の機密ラベルを優先する機密ラベルに設定します。
file_upgrade_sl	任意	ファイルまたはディレクトリの機密ラベルを、既存の機密ラベルよりも優先される機密ラベルに設定します。
sys_trans_label	任意	機密ラベルの制御下にないラベルの変換
win_colormap	任意	カラーマップ制限のオーバーライド
win_config	任意	X サーバーにより常時保持されるリソースの構成または破棄
win_dac_read	任意	クライアントのユーザー ID が所有していないウィンドウリソースからの読み取り
win_dac_write	任意	クライアントのユーザー ID が所有していないウィンドウリソースへの書き込みまたは作成
win_devices	任意	入力デバイスでの操作の実行
win_dga	任意	ダイレクトグラフィックスアクセス X プロトコル拡張機能の使用。フレームバッファー特権が必要
win_downgrade_sl	任意	ウィンドウリソースの機密ラベルを、既存ラベルの制御下にある新規ラベルに変更
win_fontpath	任意	フォントパスの追加
win_mac_read	任意	クライアントのラベルを制御するラベルを使用した、ウィンドウリソースからの読み取り
win_mac_write	任意	クライアントのラベルと同等ではないラベルを使用した、ウィンドウリソースへの書き込み
win_selection	任意	確認者の介入なしでの要求データの移動

表 27–2　ゾーン内での Oracle Solaris Trusted Extensions の特権のステータス　(続き)

Oracle Solaris Trusted Extensions の特権	ステータス	注意事項
win_upgrade_sl	任意	ウィンドウリソースの機密ラベルを、既存ラベルの制御下にない新規ラベルに変更
net_bindmlp	デフォルト	マルチレベルポート (MLP) へのバインドの許可
net_mac_aware	デフォルト	NFS を使用した読み取りの許可

非大域ゾーン構成内の特権を変更する方法については、265 ページの「ゾーンを構成、検証、および確定する」を参照してください。

特権セットを検査する方法については、414 ページの「ppriv ユーティリティーの使用」を参照してください。特権の詳細は、ppriv(1) のマニュアルページおよび『Solaris のシステム管理ガイド (セキュリティサービス)』を参照してください。

ゾーン内での IP セキュリティーアーキテクチャーの使用

IP データグラム保護を提供する IPsec (Internet Protocol Security Architecture) については、『Solaris のシステム管理 (IP サービス)』の第 19 章「IP セキュリティーアーキテクチャー (概要)」を参照してください。IKE (Internet Key Exchange) プロトコルを使用して、認証および暗号化に必要な鍵材料が自動的に管理されます。

詳細は、ipsecconf(1M) および ipseckey(1M) のマニュアルページを参照してください。

共有 IP ゾーン内の IP セキュリティーアーキテクチャー

IPsec は、大域ゾーン内で使用できます。ただし、非大域ゾーン内では、IPsec で IKE を使用することはできません。そのため、大域ゾーンで Internet Key Exchange (IKE) プロトコルを使用して、非大域ゾーンの IPsec キーおよび IPSec ポリシーを管理する必要があります。構成中の非大域ゾーンに対応するソースアドレスを使用します。

Oracle Solaris 10 8/07: 排他的 IP ゾーンでの IP セキュリティーアーキテクチャー

IPsec は、排他的 IP ゾーン内で使用できます。

ゾーン内での Oracle Solaris 監査の使用

Oracle Solaris 監査については、『Solaris のシステム管理 (セキュリティサービス)』の第 28 章「Oracle Solaris 監査 (概要)」に記載されています。監査を使用する際のゾーンの考慮事項については、次のセクションを参照してください。

- 『Solaris のシステム管理 (セキュリティサービス)』の第 29 章「Oracle Solaris 監査の計画」
- 『Solaris のシステム管理 (セキュリティサービス)』の「監査と Oracle Solaris ゾーン」

監査レコードには、システムへのログインやファイルへの書き込みなどのイベントが記載されます。レコードは、監査データのセットであるトークンで構成されます。`zonename` トークンを使用して Oracle Solaris 監査を構成することで、監査イベントをゾーンごとに識別できます。`zonename` トークンを使用すると、次の情報を生成できます。

- レコードを生成したゾーンの名前がマーク付けされた監査レコード
- 大域管理者がゾーン管理者に提供可能な特定のゾーンの監査ログ

大域ゾーン内での監査の構成

Oracle Solaris 監査トレールは、大域ゾーンで構成されます。監査ポリシーが大域ゾーン内で設定され、すべてのゾーン内のプロセスに適用されます。イベントが発生したゾーンの名前を使って、監査レコードをマーク付けできます。監査レコード内にゾーン名を含めるには、非大域ゾーンをインストールする前に `/etc/security/audit_startup` ファイルを編集する必要があります。ゾーン名選択では大文字と小文字が区別されます。

すべてのゾーン監査レコードが含まれるよう、大域ゾーン内で監査を構成するには、`/etc/security/audit_startup` ファイルに次の行を追加します。

```
/usr/sbin/auditconfig -setpolicy +zonename
```

大域ゾーンの大域管理者として、`auditconfig` ユーティリティーを実行します。

```
global# auditconfig -setpolicy +zonename
```

追加情報については、`audit_startup(1M)` と `auditconfig(1M)` のマニュアルページ、および『Solaris のシステム管理ガイド (セキュリティサービス)』の「監査ファイルの構成 (タスクマップ)」を参照してください。

非大域ゾーンのユーザー監査特性を構成する

非大域ゾーンのインストール時に、大域ゾーン内の audit_control ファイルおよび audit_user ファイルがゾーンの /etc/security ディレクトリにコピーされます。ゾーンの監査ニーズに合わせて、これらのファイルの修正が必要な場合があります。

たとえば、一部のユーザーをほかのユーザーとは異なる方法で監査するように、各ゾーンを構成できます。ユーザーごとに異なる事前選択基準を適用するには、audit_control ファイルおよび audit_user ファイルの両方を編集する必要があります。非大域ゾーン内の audit_user ファイルも必要に応じて改訂し、ゾーンのユーザーベースを反映する必要があります。監査するユーザーに応じて各ゾーンの構成を変えることができるため、audit_user ファイルを空にすることもできます。

詳細は、audit_control(4) および audit_user(4) のマニュアルページを参照してください。

特定の非大域ゾーンの監査レコードを提供する

402 ページの「大域ゾーン内での監査の構成」で説明したように、zonename トークンを含めることで、Oracle Solaris 監査レコードをゾーン別に分類できます。auditreduce コマンドを使用してレコードをゾーンごとに収集して、特定のゾーンのログを作成できます。

詳細は、audit_startup(1M) および auditreduce(1M) のマニュアルページを参照してください。

ゾーン内のコアファイル

coreadm コマンドを使用して、異常終了するプロセスにより生成されるコアファイルの名前と場所を指定できます。%z 変数を指定することで、プロセスが実行されたゾーンの *zonename* を含むコアファイルパスを生成できます。パス名は、ゾーンのルートディレクトリに対する相対パスです。

詳細は、coreadm(1M) および core(4) のマニュアルページを参照してください。

非大域ゾーン内での DTrace の実行

dtrace_proc 特権と dtrace_user 特権だけを必要とする DTrace プログラムは、非大域ゾーンで実行できます。非大域ゾーンで使用できる特権のセットにこれらの特権を追加するには、zonecfg limitpriv プロパティーを使用します。手順については、416 ページの「DTrace を使用する方法」を参照してください。

dtrace_proc によってサポートされるプロバイダは、fasttrap と pid です。dtrace_user によってサポートされるプロバイダは、profile と syscall です。DTrace のプロバイダおよびアクションの有効範囲は、ゾーンに制限されます。

詳細は、397 ページの「非大域ゾーン内の特権」も参照してください。

ゾーンがインストールされている Oracle Solaris システムのバックアップについて

非大域ゾーンを個別にバックアップしたり、大域ゾーンからシステム全体をバックアップしたりできます。

ループバックファイルシステムのディレクトリのバックアップ

非大域ゾーンと大域ゾーンでファイルを共有するときには、多くの場合、ループバックファイルシステムの読み取り専用マウント (通常は、/usr、 /lib、/sbin、および /platform) が使用されます。このため、lofs ディレクトリをバックアップするときには、大域ゾーンのバックアップ方式を使用する必要があります。

注意 – 非大域ゾーンで、大域ゾーンと共有されている lofs ファイルシステムをバックアップしないでください。非大域ゾーンの管理者が非大域ゾーンから lofs ファイルシステムを復元しようとすると、重大な問題が発生することがあります。

大域ゾーンからのシステムのバックアップ

次のような場合は、大域ゾーンからバックアップを実行することをお勧めします。

- 非大域ゾーンの構成をアプリケーションデータと一緒にバックアップする場合。
- 障害から回復することが最も重要である場合。使用しているゾーンのルートファイルシステムおよびそれらの構成データ、使用している大域ゾーン内のデータなど、システム上のすべてまたはほぼすべての情報を復元する必要がある場合。このような場合は、大域ゾーンでバックアップを実行してください。

- ufsdump コマンドを使ってデータをバックアップする場合。物理ディスクデバイスを非大域ゾーンにインポートすると、ゾーンのセキュリティープロファイルが変更されます。このため、ufsdump を使用するときは、必ず大域ゾーンから実行してください。
- 市販のネットワークバックアップソフトウェアを使用する場合。

注 – ネットワークバックアップソフトウェアを使用するときには、可能であれば、継承された lofs ファイルシステムはすべてスキップするように構成することをお勧めします。バックアップは、ゾーンとそのアプリケーションがバックアップ対象のデータを休止させた状態のときに、行うことをお勧めします。

システム上の非大域ゾーンを個別にバックアップ

次のような場合は、非大域ゾーン内でバックアップを実行することをお勧めします。

- 非大域ゾーンの管理者が、重大度の低い障害から回復する機能、またはゾーンに固有のアプリケーションデータまたはユーザーデータを復元する機能を必要とする場合。
- tar や cpio など、ファイル単位でバックアップを行うプログラムを使用する場合。tar(1) および cpio(1) のマニュアルページを参照してください。
- ゾーン内で動作する特定のアプリケーションまたはサービスのバックアップソフトウェアを使用する場合。ディレクトリパスやインストール済みソフトウェアなどのアプリケーション環境が大域ゾーンと非大域ゾーンとの間で異なっている場合には、バックアップソフトウェアを大域ゾーンから実行するのが困難な場合があります。

  アプリケーションが非大域ゾーンごとのバックアップスケジュールに基づいてスナップショットを実行し、そのバックアップデータを大域ゾーンからエクスポートした書き込み可能なディレクトリに格納できる場合には、大域ゾーンの管理者は大域ゾーンからバックアップする処理の中でそれら個々のバックアップデータを個別に選択できます。

非大域ゾーン内でバックアップするデータの決定

非大域ゾーン内のデータは、すべてバックアップできます。ゾーンの構成が頻繁に変更されない場合には、アプリケーションデータだけをバックアップすることもできます。

アプリケーションデータのみのバックアップ

アプリケーションデータがファイルシステムの特定の場所に格納されている場合には、このデータだけを定期的にバックアップすることもできます。ゾーンのルートファイルシステムはそれほど頻繁には変更されないため、頻繁にバックアップする必要がない場合もあります。

アプリケーションファイルがどこに格納されているかを確認する必要があります。アプリケーションファイルは次のような場所に格納されている可能性があります。

- ユーザーのホームディレクトリ
- `/etc` (構成データファイルの場合)
- `/var`

アプリケーション管理者がデータの格納場所を認識している場合には、ゾーンごとに書き込み可能ディレクトリを割り当てるように、システムを作成することもできます。バックアップがゾーンごとに格納されるので、大域ゾーンの管理者はその場所をシステム上のバックアップ対象の 1 つとして選択することができます。

一般的なデータベースバックアップ操作

データベースアプリケーションデータがデータベース固有のディレクトリに存在していない場合には、次の規則が適用されます。

- 最初にデータベースが安定した状態にあることを確認します。

  データベースが休止している必要があります。内部バッファーのデータがディスクにフラッシュされるためです。大域ゾーンからバックアップを開始する前に、非大域ゾーン内のデータベースが安定した状態になっていることを確認してください。
- ゾーンごとに、ファイルシステム機能を使用してデータのスナップショットを作成してから、そのスナップショットを大域ゾーンから直接バックアップします。

  このようにすると、バックアップのための時間が短縮されるだけでなく、すべてのゾーンのクライアントやモジュールをバックアップする必要がなくなります。

テープバックアップ

非大域ゾーンだけが使用するファイルシステムについては、そのゾーンにとって都合のよい時間帯に、アプリケーションがわずかに休止している時間を利用して、スナップショットを作成することもできます。それらのスナップショットは、アプリケーションがサービスに戻ったあとに大域ゾーンからバックアップしてテープに格納できます。

この方法により、次の利点が得られます。

- 必要なテープデバイスが少なくてすみます。
- 非大域ゾーンの間で調整する必要がなくなります。
- デバイスを直接ゾーンに割り当てる必要がないため、セキュリティー機能が向上します。
- 大域ゾーンでシステム管理を続けることができるので、通常はこの方法をお勧めします。

非大域ゾーンの復元について

大域ゾーンの管理者は、大域ゾーンから実行したバックアップを復元するときには、関係するゾーンを再インストールしてから、そのゾーンのファイルを復元できます。この処理を行うときには、ゾーンが次の状態である必要があります。

- 復元するゾーンの構成が、バックアップしたときと同じ構成であること。
- バックアップしてからゾーンを復元するまでの間に、大域ゾーンをアップグレードしたり、大域ゾーンにパッチを適用したりしていないこと。

これらの前提を満たしていない場合は、一部のファイルが復元によって上書きされ、手作業でマージしなければならないことがあります。

たとえば、バックアップしてから非大域ゾーンを復元するまでに大域ゾーンにパッチを適用している場合には、手作業によるファイルのマージが必要になることがあります。このような状況でゾーンのバックアップファイルを復元するときには、注意が必要です。大域ゾーンにパッチを適用したあとに、新しくゾーンをインストールして再構築した場合には、バックアップファイルがそのゾーンと互換性を持たなくなることがあります。このような場合には、ファイルを個別に調べて、それらを新しくインストールしたゾーンのコピーと比較する必要があります。ほとんどの場合、ファイルを直接コピーすることができます。しかし、場合によっては、バックアップファイルに行なっていた変更を、ゾーン内で新しくインストールしたコピーまたはパッチを適用したコピーにマージする必要があります。

注 – 大域ゾーンのすべてのファイルシステムが失われた場合には、大域ゾーンのすべてのファイルを復元すると、非大域ゾーンも復元されます。ただし、非大域ゾーンの各ルートファイルシステムがバックアップに含まれている必要があります。

ゾーンがインストールされている Oracle Solaris システムで使用するコマンド

表 27–3 に示すコマンドにより、ゾーン機能に対するプライマリ管理インタフェースが提供されます。

表 27–3 ゾーンの管理に使用されるコマンド

コマンドのリファレンス	説明
zlogin(1)	非大域ゾーンにログインします
zonename(1)	現在のゾーンの名前を出力します
zoneadm(1M)	システムのゾーンを管理します
zonecfg(1M)	ゾーン構成の設定に使用されます
getzoneid(3C)	ゾーン ID と名前のマッピングに使用されます
zones(5)	ゾーン機能の説明を提供します
zcons(7D)	ゾーンコンソールのデバイスドライバ

zoneadmd デーモンは、ゾーンの仮想プラットフォームを管理する基本プロセスです。zoneadmd デーモンのマニュアルページは、zoneadmd(1M) です。このデーモンは、プログラミングインタフェースの構成要素ではありません。

次の表に示すコマンドは、リソース上限デーモンとともに使用されます。

表 27–4 rcapd とともに使用されるコマンド

コマンドのリファレンス	説明
rcapstat(1)	上限が定義されたプロジェクトのリソース使用効率を監視します。
rcapadm(1M)	リソース上限デーモンを構成します。構成済みのリソース上限デーモンの現在のステータスを表示します。リソース上限制御を有効または無効にします。一時的なメモリー上限を設定する場合にも使用されます。
rcapd(1M)	リソース上限デーモン。

次の表で示すコマンドは、ゾーンがインストールされている Oracle Solaris システムで使用できるように変更されています。これらのコマンドには、ゾーンに固有のオプションが用意されています。指定するオプションによって異なる情報が表示されます。コマンドは、マニュアルページのセクション別に記載されています。

表 27–5　ゾーンがインストールされている Oracle Solaris システムで使用するために変更されたコマンド

コマンドのリファレンス	説明
ipcrm(1)	-z *zone* オプションが追加されました。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。
ipcs(1)	-z *zone* オプションが追加されました。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。
pgrep(1)	-z *zoneidlist* オプションが追加されました。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。
ppriv(1)	-l オプションとともに使用する式 zone が追加されました。これを使用すると、現在のゾーン内で使用可能なすべての特権が一覧表示されます。また、zone に -v オプションを指定して、冗長出力を取得できます。
priocntl(1)	*idlist* と -i *idtype* でゾーン ID を使用することで、プロセスを指定できます。priocntl -i *zoneid* コマンドを使用すると、実行中のプロセスを非大域ゾーン内の別のスケジューリングクラスに移動できます。
proc(1)	ptree だけに -z *zone* オプションが追加されました。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。
ps(1)	-o オプションが使用する、認識される format 名リストに、*zonename* と *zoneid* が追加されました。 指定したゾーン内のプロセスだけを一覧表示するため、-z *zonelist* が追加されました。ゾーンの指定には、ゾーン名またはゾーン ID を使用できます。このオプションは、大域ゾーン内でコマンドを実行する場合にのみ有用です。 プロセスに関連するゾーンの名前を出力するため、-Z が追加されました。名前は、追加された列ヘッダー ZONE の下に出力されます。
renice(1)	有効な引数をリスト表示するため、-i オプションとともに使用する *zoneid* が追加されました。
sar(1)	プール機能が有効な非大域ゾーン内で実行する際、-b、-c、-g、-m、-p、-u、-w、および -y オプションを指定すると、ゾーンのバインド先プールのプロセッサセット内のプロセッサの値だけが表示されます。
auditconfig(1M)	zonename トークンが追加されました。
auditreduce(1M)	-z *zone-name* オプションが追加されました。ゾーンの監査ログを取得する機能が追加されました。
coreadm(1M)	プロセスが実行されたゾーンを識別するための変数 %z が追加されました。

表 27–5 ゾーンがインストールされている Oracle Solaris システムで使用するために変更されたコマンド (続き)

コマンドのリファレンス	説明
df(1M)	すべての可視ゾーン内のマウントを表示する -z オプションが追加されました。
ifconfig(1m)	大域ゾーンで使用する zone オプション (デフォルト)、および非大域ゾーンで使用する -zone *zonename* が追加されました。
iostat(1M)	プール機能が有効な非大域ゾーン内で実行すると、ゾーンのバインド先プールのプロセッサセット内のプロセッサ情報だけが提供されます。
kstat(1M)	大域ゾーンで実行すると、すべてのゾーンの kstat が表示されます。非大域ゾーンで実行すると、一致する *zoneid* を持つ kstat だけが表示されます。
mpstat(1M)	プール機能が有効な非大域ゾーン内で実行すると、ゾーンのバインド先プールのプロセッサセット内のプロセッサの行だけが表示されます。
ndd(1M)	大域ゾーン内で使用された場合、すべてのゾーンの情報を表示します。排他的 IP ゾーンの TCP/IP モジュールに対して ndd を実行すると、そのゾーンの情報だけが表示されます。
netstat(1M)	現在のゾーンのみの情報を表示します。
nfsstat(1M)	現在のゾーンのみの統計情報を表示します。
poolbind(1M)	*zoneid* リストが追加されました。ゾーンとリソースプールの併用については、150 ページの「ゾーンで使用されるリソースプール」も参照してください。
prstat(1M)	-z *zoneidlist* オプションが追加されました。また、-Z オプションも追加されました。 プール機能が有効な非大域ゾーンで実行した場合、ゾーンのバインド先プールのプロセッサセット内のプロセッサだけを対象にして、プロセスが使用した最新の CPU 時間の比率が表示されます。 -a、-t、-T、-J、および -Z の各オプションを指定すると、出力にはサイズ列の代わりにスワップ列が表示されます。報告されるスワップは、ゾーンのプロセスと tmpfs マウントで消費されるスワップの合計量です。この値により、各ゾーンで予約されているスワップを監視しやすくなり、適切な zone.max-swap 設定を選択することができます。
psrinfo(1M)	非大域ゾーン内で実行した場合、ゾーンで表示可能なプロセッサの情報だけが表示されます。
traceroute(1M)	使用方法が変更されました。非大域ゾーン内から指定した場合、-F オプションをしても効果はありません。理由は、「断片化しない」というビットが常に設定されているためです。

表 27–5 ゾーンがインストールされている Oracle Solaris システムで使用するために変更されたコマンド (続き)

コマンドのリファレンス	説明
vmstat(1M)	プール機能が有効な非大域ゾーン内で実行すると、ゾーンのバインド先プールのプロセッサセット内のプロセッサ統計情報だけが報告されます。-p オプション指定時の出力と、page、faults、および cpu 報告フィールドに適用されます。
auditon(2)	各監査レコードとともにゾーン ID トークンを生成する、AUDIT_ZONENAME が追加されました。
priocntl(2)	P_ZONEID *id* 引数が追加されました。
processor_info(2)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
p_online(2)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
pset_bind(2)	idtype として *P_ZONEID* が追加されました。P_MYID 仕様の選択肢にゾーンが追加されました。EINVAL エラー説明内の有効な *idtype* リストに P_ZONEIDが追加されました。
pset_info(2)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
pset_list(2)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
pset_setattr(2)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
sysinfo(2)	PRIV_SYS_CONFIG が PRIV_SYS_ADMIN に変更されました。
umount(2)	*file* が参照しているファイルが絶対パスでない場合、ENOENT が返されます。
getloadavg(3C)	呼び出し側が非大域ゾーン内にあり、プール機能が有効な場合、PS_MYID の psetid を使用して呼び出した場合と動作は同じになります。
getpriority(3C)	ゾーン ID が、指定可能なターゲットプロセスに追加されました。ゾーン ID が EINVAL のエラー説明に追加されました。
priv_str_to_set(3C)	呼び出し側のゾーン内部で使用可能なすべての特権セットで、「zone」文字列が追加されました。

表 27–5 ゾーンがインストールされているOracle Solarisシステムで使用するために変更されたコマンド (続き)

コマンドのリファレンス	説明
pset_getloadavg(3C)	呼び出し側が非大域ゾーン内にあり、プール機能が有効であるが、プロセッサがゾーンのバインド先プールのプロセッサセット内にない場合、エラーが返されます。
sysconf(3C)	呼び出し側が非大域ゾーン内にあり、プール機能が有効な場合、sysconf(_SC_NPROCESSORS_CONF) およびsysconf(_SC_NPROCESSORS_ONLN) は、ゾーンのバインド先プールのプロセッサセット内のプロセッサ数を返します。
ucred_get(3C)	ucred_getzoneid() 関数が追加されました。この関数は、プロセスのゾーン ID を返します。ただし、ゾーン ID を取得できなかった場合は -1 を返します。
core(4)	n_type: NT_ZONENAME が追加されました。このエントリには、プロセスが実行されていたゾーンの名前を示す文字列が含まれます。
pkginfo(4)	ゾーンのサポート内でオプションパラメータおよび環境変数が提供されるようになりました。
proc(4)	ゾーン内で実行中のプロセスに関する情報を取得する機能が追加されました。
audit_syslog(5)	in<*zone name*> フィールドが追加されました。このフィールドは、zonename 監査ポリシーが設定されている場合に使用されます。
privileges(5)	プロセスによる追跡またはほかのゾーン内のプロセスへのシグナル送信を可能にする PRIV_PROC_ZONE が追加されました。zones(5) を参照してください。
if_tcp(7P)	ゾーンの ioctl() 呼び出しが追加されました。
cmn_err(9F)	ゾーンパラメータが追加されました。
ddi_cred(9F)	cr の指し示すユーザー資格からゾーン ID を返す、crgetzoneid() が追加されました。

◆ ◆ ◆ 第 28 章

Oracle Solaris ゾーンの管理 (タスク)

この章では、一般的な管理タスクおよびその使用例を示します。

この章に追加されている説明

このセクションでは、製品の新しい機能を一覧表示し、このガイドに追加された内容を示します。

Oracle Solaris 10 の新機能の一覧および Oracle Solaris リリースについての説明は、『Oracle Solaris 10 8/11 の新機能』を参照してください。

Oracle Solaris 10 1/06 についてこの章に追加されている説明

メディアにアクセスするための新しい手順が追加されています。421 ページの「非大域ゾーンで CD または DVD メディアにアクセスする権限を追加する方法」を参照してください。

ゾーン内のファイルをバックアップおよび復元する手順が追加されています。431 ページの「ゾーンがインストールされている Oracle Solaris システムのバックアップ」および434 ページの「非大域ゾーンの復元」を参照してください。

Oracle Solaris 10 6/06 についてこの章に追加されている説明

新しい手順が追加されています。421 ページの「大域ゾーンのファイルシステムを非大域ゾーンにマウントする方法」および423 ページの「非大域ゾーンの /usr の下に書き込み可能ディレクトリを追加する方法」を参照してください。

Oracle Solaris 10 8/07 についてこの章に追加されている説明

新しい手順が追加されています。416 ページの「DTrace を使用する方法」、427 ページの「Oracle Solaris 10 8/07: 排他的 IP 非大域ゾーンでのデータリンクの管理」、417 ページの「非大域ゾーンの SMF サービスのステータスの確認」を参照してください。

ppriv ユーティリティーの使用

ppriv ユーティリティーを使用してゾーンの特権を表示します。

▼ 大域ゾーンでの Oracle Solaris の特権を一覧表示する方法

ppriv ユーティリティーを -l オプションとともに使用して、システムで使用可能な特権を一覧表示します。

- プロンプトで **ppriv -l zone** と入力し、ゾーンで使用可能な特権セットを表示します。

```
global# ppriv -l zone
```

次のような内容が表示されます。

```
contract_event
contract_observer
cpc_cpu
.
.
.
```

▼ 非大域ゾーンの特権セットの表示方法

ppriv ユーティリティーを -l オプションおよび式 zone とともに使用して、ゾーンの特権を表示します。

1 非大域ゾーンにログインします。この例では、*my-zone* という名前のゾーンを使用します。

2 プロンプトで **ppriv -l zone** と入力し、ゾーンで使用可能な特権セットを表示します。

```
my-zone# ppriv -l zone
```

次のような内容が表示されます。

```
contract_event
contract_observer
file_chown

.
.
.
```

▼ 非大域ゾーンの特権セットを冗長出力で表示する方法

ppriv ユーティリティーを -l オプション、式 zone、および -v オプションとともに使用して、ゾーンの特権を一覧表示します。

1 非大域ゾーンにログインします。この例では、*my-zone* という名前のゾーンを使用します。

2 プロンプトで **ppriv -l -v zone** と入力して、ゾーン内で使用可能な特権セットおよび各特権の説明を出力します。

```
my-zone# ppriv -l -v zone
```

次のような内容が表示されます。

```
contract_event
        Allows a process to request critical events without limitation.
        Allows a process to request reliable delivery of all events on
        any event queue.
contract_observer
        Allows a process to observe contract events generated by
        contracts created and owned by users other than the process's
        effective user ID.
        Allows a process to open contract event endpoints belonging to
        contracts created and owned by users other than the process's
        effective user ID.
file_chown
        Allows a process to change a file's owner user ID.
        Allows a process to change a file's group ID to one other than
        the process' effective group ID or one of the process'
        supplemental group IDs.
.
.
.
```

非大域ゾーン内での DTrace の使用

404 ページの「非大域ゾーン内での DTrace の実行」で説明されている DTrace 機能を使用するには、次の手順を実行します。

▼ DTrace を使用する方法

1. **zonecfg limitpriv** プロパティーを使用して、**dtrace_proc** 特権と **dtrace_user** 特権を追加します。

   ```
   global# zonecfg -z my-zone
   zonecfg:my-zone> set limitpriv="default,dtrace_proc,dtrace_user"
   zonecfg:my-zone> exit
   ```

   注 – 必要に応じて、どちらか一方の特権を追加することも、両方の特権を追加することもできます。

2. ゾーンをブートします。

   ```
   global# zoneadm -z my-zone boot
   ```

3. ゾーンにログインします。

   ```
   global# zlogin my-zone
   ```

4. **DTrace** プログラムを実行します。

   ```
   my-zone# dtrace -l
   ```

非大域ゾーンの SMF サービスのステータスの確認

ネイティブな非大域ゾーンの SMF サービスのステータスを確認するには、zlogin コマンドを使用します。

▼ コマンド行から SMF サービスのステータスを確認する方法

1. スーパーユーザーまたは Primary Administrator 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。
2. コマンド行から次のように入力して、無効になっているサービスも含むすべてのサービスを表示します。

```
global# zlogin my-zone svcs -a
```

参照　詳細は、第 22 章「非大域ゾーンへのログイン (タスク)」および svcs(1) のマニュアルページを参照してください。

▼ ゾーン内から SMF サービスのステータスを確認する方法

1. スーパーユーザーまたは Primary Administrator 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。
2. ゾーンにログインします。

```
global# zlogin my-zone
```

3. **svcs** コマンドに **-a** オプションを付けて実行して、無効になっているサービスも含むすべてのサービスを表示します。

```
my-zone# svcs -a
```

参照　詳細は、第 22 章「非大域ゾーンへのログイン (タスク)」および svcs(1) のマニュアルページを参照してください。

稼働中の非大域ゾーン内でファイルシステムをマウントする

稼働中の非大域ゾーン内でファイルシステムをマウントできます。具体的には、次の作業について説明しています。

- 大域ゾーンの大域管理者として、raw デバイスおよびブロックデバイスを非大域ゾーンにインポートできます。デバイスのインポート後に、ゾーン管理者はディスクにアクセスできます。その後、ゾーン管理者はディスク上に新しいファイルシステムを作成して、次のいずれかの操作を実行できます。
  - ファイルシステムを手動でマウントします
  - ファイルシステムがゾーンのブート時にマウントされるように、`/etc/vfstab` 内に配置します
- 大域管理者として、大域ゾーンのファイルシステムを非大域ゾーンにマウントすることもできます。

▼ zonecfg を使用して raw デバイスおよびブロックデバイスをインポートする方法

この手順では、`lofi` ファイルドライバを使用します。このドライバは、ファイルをブロックデバイスとしてエクスポートします。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. ディレクトリを **/usr/tmp** に変更します。

   ```
   global# cd /usr/tmp
   ```

3. 新しい **UFS** ファイルシステムを作成します。

   ```
   global# mkfile 10m fsfile
   ```

4. ファイルをブロックデバイスとして接続します。
   ほかの `lofi` デバイスが作成されていない場合、使用可能な最初のスロット `/dev/lofi/1` が使用されます。

   ```
   global# lofiadm -a ‘pwd‘/fsfile
   ```

   必要な文字デバイスも取得します。

5 デバイスをゾーン **my-zone** にインポートします。

```
global# zonecfg -z my-zone
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/rlofi/1
zonecfg:my-zone:device> end
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/lofi/1
zonecfg:my-zone:device> end
```

6 ゾーンをリブートします。

```
global# zoneadm -z my-zone boot
```

7 ゾーンにログインして、デバイスのインポートが成功したことを確認します。

```
my-zone# ls  -l /dev/*lofi/*
```

次のような内容が表示されます。

```
brw-------   1 root     sys      147,  1 Jan  7 11:26 /dev/lofi/1
crw-------   1 root     sys      147,  1 Jan  7 11:26 /dev/rlofi/1
```

参照 詳細は、lofiadm(1m) および lofi(7D) のマニュアルページを参照してください。

▼ ファイルシステムを手動でマウントする方法

この手順を実行するには、ゾーン管理者になり、Zone Management プロファイルを保持する必要があります。この手順では、newfs コマンドを使用します。このコマンドの詳細は、newfs(1M) のマニュアルページを参照してください。

1 スーパーユーザーになるか、プロファイルリスト内の **Zone Management** 権利プロファイルを保持します。

2 ゾーン **my-zone** 内で、ディスク上に新しいファイルシステムを作成します。

```
my-zone# newfs /dev/lofi/1
```

3 プロンプトに **yes** で応答します。

```
newfs: construct a new file system /dev/rlofi/1: (y/n)? y
```

次のような内容が表示されます。

```
/dev/rlofi/1:   20468 sectors in 34 cylinders of 1 tracks, 602 sectors
        10.0MB in 3 cyl groups (16 c/g, 4.70MB/g, 2240 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 9664, 19296,
```

4 ファイルシステムのエラーを検査します。

```
my-zone# fsck -F ufs /dev/rlofi/1
```

次のような内容が表示されます。

```
** /dev/rlofi/1
** Last Mounted on
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
2 files, 9 used, 9320 free (16 frags, 1163 blocks, 0.2% fragmentation)
```

5 ファイルシステムをマウントします。

```
my-zone# mount -F ufs /dev/lofi/1 /mnt
```

6 マウントを検証します。

```
my-zone# grep /mnt /etc/mnttab
```

次のような内容が表示されます。

```
/dev/lofi/1     /mnt    ufs
rw,suid,intr,largefiles,xattr,onerror=panic,zone=foo,dev=24c0001
1073503869
```

▼ ゾーンのブート時にマウントするファイルシステムを /etc/vfstab に配置する方法

ここでは、ブロックデバイス /dev/lofi/1 をファイルシステムパス /mnt にマウントする手順を説明します。ブロックデバイスには UFS ファイルシステムが含まれます。次のオプションを使用します。

- マウントオプションとして logging を使用します。
- yes は、ゾーンのブート時にファイルシステムを自動的にマウントするよう、システムに指示します。
- /dev/rlofi/1 は文字 (または raw) デバイスです。必要に応じて、fsck コマンドを raw デバイスで実行します。

1 スーパーユーザーになるか、プロファイルリスト内の **Zone Management** 権利プロファイルを保持します。

2 ゾーン **my-zone** 内で、次の行を **/etc/vfstab** に追加します。

```
/dev/lofi/1 /dev/rlofi/1  /mnt   ufs  2  yes logging
```

▼ 大域ゾーンのファイルシステムを非大域ゾーンにマウントする方法

ゾーンが zonepath /export/home/my-zone を保持するものとします。ここで、大域ゾーンのディスク /dev/lofi/1 を非大域ゾーン内の /mnt にマウントします。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. ディスクを非大域ゾーンの **/mnt** にマウントするには、大域ゾーンから次のように入力します。

   ```
   global# mount -F ufs /dev/lofi/1 /export/home/my-zone/root/mnt
   ```

参照　lofi の詳細については、lofiadm(1M) および lofi(7D) のマニュアルページを参照してください。

大域ゾーン内の特定のファイルシステムへのアクセス権を非大域ゾーンに追加する

▼ 非大域ゾーンで CD または DVD メディアにアクセスする権限を追加する方法

この手順を使用して、非大域ゾーンで CD または DVD メディアに読み取り専用のアクセスを行う権限を追加できます。メディアをマウントするときには、大域ゾーンでボリューム管理ファイルシステムが使用されます。アクセス権を追加したら、CD または DVD を使用して製品を非大域ゾーンにインストールできます。この手順では、jes_05q4_dvd という DVD を使用します。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 大域ゾーンでボリューム管理ファイルシステムが稼働しているかどうかを調べます。

```
global# svcs volfs
STATE          STIME    FMRI
online         Sep_29   svc:/system/filesystem/volfs:default
```

3 (オプション) 大域ゾーンでボリューム管理ファイルシステムが稼働していない場合は、起動します。

```
global# svcadm volfs enable
```

4 メディアを挿入します。

5 ドライブにメディアが入っているかどうかを確認します。

```
global# volcheck
```

6 DVD が自動マウントされているかどうかをテストします。

```
global# ls /cdrom
```

次のような情報が表示されます。

```
cdrom   cdrom1   jes_05q4_dvd
```

7 **ro,nodevices** オプション (読み取り専用、デバイスなし) を指定して、非大域ゾーンでファイルシステムをループバックマウントします。

```
global# zonecfg -z my-zone
zonecfg:my-zone> add fs
zonecfg:my-zone:fs> set dir=/cdrom
zonecfg:my-zone:fs> set special=/cdrom
zonecfg:my-zone:fs> set type=lofs
zonecfg:my-zone:fs> add options [ro,nodevices]
zonecfg:my-zone:fs> end
zonecfg:my-zone> commit
zonecfg:my-zone> exit
```

8 非大域ゾーンをリブートします。

```
global# zoneadm -z my-zone reboot
```

9 **zoneadm list** コマンドに **-v** オプションを指定して、ステータスを確認します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
 1  my-zone  running      /export/home/my-zone           native     shared
```

10 非大域ゾーンにログインします

```
global# zlogin my-zone
```

11 **DVD-ROM** がマウントされているかを確認します。

```
my-zone# ls /cdrom
```

次のような内容が表示されます。

```
cdrom   cdrom1   jes_05q4_dvd
```

12 製品のインストールガイドで説明されているとおりに、製品をインストールします。

13 非大域ゾーンから抜けます。

```
my-zone# exit
```

ヒント – `/cdrom` ファイルシステムを非大域ゾーンに残すこともできます。マウントするときには、常に CD-ROM ドライブの現在の内容が反映されます。つまり、ドライブが空の場合は、ディレクトリは空になります。

---

14 (オプション) 非大域ゾーンから **/cdrom** ファイルシステムを削除する場合は、次の手順を使用します。

```
global# zonecfg -z my-zone
zonecfg:my-zone> remove fs dir=/cdrom
zonecfg:my-zone> commit
zonecfg:my-zone> exit
```

▼ 非大域ゾーンの /usr の下に書き込み可能ディレクトリを追加する方法

疎ルートゾーンでは、`/usr` は大域ゾーンから読み取り専用でマウントされます。ここで説明する手順を使って、ゾーンの `/usr` の下に `/usr/local` などの書き込み可能ディレクトリを追加できます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 大域ゾーンに **/usr/local** ディレクトリを作成します。

```
global# mkdir -p /usr/local
```

3 ゾーンの **/usr/local** ディレクトリのバッキングストアとして使用する、大域ゾーン内のディレクトリを指定します。

```
global# mkdir -p /storage/local/my-zone
```

4 ゾーン *my-zone* の構成を編集します。

```
global# zonecfg -z my-zone
```

5 ループバックマウントされたファイルシステムを追加します。

```
zonecfg:my-zone> add fs
zonecfg:my-zone:fs> set dir=/usr/local
        zonecfg:my-zone:fs> set special=/storage/local/my-zone
        zonecfg:my-zone:fs> set type=lofs
        zonecfg:my-zone:fs> end
        zonecfg:my-zone> commit
        zonecfg:my-zone> exit
```

6 ゾーンをブートします。

▼ 大域ゾーン内のホームディレクトリを非大域ゾーンにエクスポートする方法

大域ゾーンから同じシステム上の非大域ゾーンにホームディレクトリまたはほかのファイルシステムをエクスポートする場合に、この手順を使用します。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 ループバックマウントされたファイルシステムを追加します。

```
global# zonecfg -z my-zone
zonecfg:my-zone> add fs
zonecfg:my-zone:fs> set dir=/export/home
zonecfg:my-zone:fs> set special=/export/home
zonecfg:my-zone:fs> set type=lofs
zonecfg:my-zone:fs> set options=nodevices
zonecfg:my-zone:fs> end
zonecfg:my-zone> commit
zonecfg:my-zone> exit
```

3 ゾーンの **/etc/auto_home** ファイルに次の行を追加します。

```
$HOST:/export/home/&
```

ゾーンがインストールされている Oracle Solaris システムでの IP ネットワークマルチパスの使用

▼ Oracle Solaris 10 8/07: 排他的 IP 非大域ゾーンで IP ネットワークマルチパスを使用する方法

排他的 IP ゾーンの IP ネットワークマルチパス (IPMP) は、大域ゾーンと同じ方法で構成します。

IP マルチパスグループ (IPMP グループ) に 1 つ以上の物理インタフェースを構成できます。IPMP を構成すると、IPMP グループのインタフェースに障害が発生していないかどうかをシステムが自動的に監視します。グループのインタフェースに障害が発生した場合や、保守のためにインタフェースが削除された場合、IPMP は自動的に、そのインタフェースの IP アドレスを移行して処理を継続します。フェイルオーバーされたアドレスは、障害が発生したインタフェースの IPMP グループ内の機能中のインタフェースが受け取ります。IPMP のフェイルオーバー機能は、接続を保持し、既存の接続の切断を防止します。さらに、IPMP は、ネットワークトラフィックを自動的に IPMP グループ内のインタフェースのセットに分散することによって、ネットワークパフォーマンス全体を向上させます。この処理は負荷分散と呼ばれます。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 『**Solaris** のシステム管理 **(IP** サービス**)**』の「**IPMP** グループの構成」に記載されているように **IPMP** グループを構成します。

▼ IP ネットワークマルチパス機能を共有 IP 非大域ゾーンに拡張する方法

大域ゾーン内で IPMP を構成し、IPMP 機能を非大域ゾーンに拡張する場合に、ここで説明する手順を使用します。

ゾーンの構成時に、各アドレスつまり論理インタフェースを非大域ゾーンと関連付ける必要があります。手順については、243 ページの「zonecfg コマンドの使用」および 266 ページの「ゾーンの構成方法」を参照してください。

この手順を実行すると、次のことが達成されます。

- カード bge0 および hme0 がグループ内でともに構成されます。
- アドレス 192.168.0.1 が非大域ゾーン *my-zone* と関連付けられます。
- bge0 カードが物理インタフェースとして設定されます。このため、IP アドレスが bge0 および hme0 カードを含むグループ内に収容されます。

稼働中のゾーンで、ifconfig コマンドを使用して関連付けを行うことができます。388 ページの「共有 IP ネットワークインタフェース」および ifconfig(1m) のマニュアルページを参照してください。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 大域ゾーン内で、『**Solaris のシステム管理 (IP サービス)**』の「**IPMP グループの構成**」に記載されているように **IPMP** グループを構成します。

3. **zonecfg** コマンドを使用してゾーンを構成します。**net** リソースを構成する際、アドレス **192.168.0.1** および物理インタフェース **bge0** をゾーン *my-zone* に追加します。

   ```
   zonecfg:my-zone> add net
   zonecfg:my-zone:net> set address=192.168.0.1
   zonecfg:my-zone:net> set physical=bge0
   zonecfg:my-zone:net> set defrouter=10.0.0.1
   zonecfg:my-zone:net> end
   ```

   非大域ゾーン *my-zone* 内に bge0 だけが表示されます。

参考 bge0 が連続して失敗した場合

bge0 が連続して失敗し、bge0 データアドレスが大域ゾーン内の hme0 に引き継がれる場合、*my-zone* アドレスも移行します。

アドレス 192.168.0.1 が hme0 に移動する場合、非大域ゾーン *my-zone* 内で hme0 だけが表示されます。このカードは、アドレス 192.168.0.1 に関連付けられ、bge0 は表示されなくなります。

Oracle Solaris 10 8/07: 排他的 IP 非大域ゾーンでのデータリンクの管理

データリンクを管理するには、大域ゾーンから dladm コマンドを使用します。

▼ dladm show-linkprop の使用方法

dladm コマンドを show-linkprop サブコマンドとともに使用して、実行中の排他的 IP ゾーンに対するデータリンクの割り当てを表示できます。

データリンクを管理するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは Primary Administrator 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 システムのデータリンクの割り当てを表示します。

```
global# dladm show-linkprop
```

例 28–1 dladm コマンドを show-linkprop サブコマンドとともに使用する

1. ゾーン 49bge には bge0 が割り当てられていますが、最初の画面ではこのゾーンはブートしていません。

```
global# dladm show-linkprop
LINK         PROPERTY        VALUE          DEFAULT        POSSIBLE
bge0         zone            --             --             --
ath0         channel         6              --             --
ath0         powermode       ?              off            off,fast,max
ath0         radio           ?              on             on,off
ath0         speed           11             --
1,2,5.5,6,9,11,12,18,24,36,48,54
ath0         zone            --             --             --
```

2. ゾーン 49bge をブートします。

```
global# zoneadm -z 49bge boot
```

3. コマンド dladm show-linkprop をもう一度実行します。すると、bge0 リンクが 49bge に割り当てられていることに注目してください。

```
global# dladm show-linkprop
LINK         PROPERTY        VALUE          DEFAULT        POSSIBLE
bge0         zone            49bge          --             --
ath0         channel         6              --             --
ath0         powermode       ?              off            off,fast,max
ath0         radio           ?              on             on,off
ath0         speed           11             --
```

```
1,2,5.5,6,9,11,12,18,24,36,48,54
ath0         zone            --              --              --
```

▼ dladm set-linkprop の使用方法

dladm コマンドを set-linkprop サブコマンドとともに使用して、実行中の排他的 IP ゾーンに対してデータリンクを一時的に割り当てることができます。持続的な割り当ては zonecfg コマンドを使用して行う必要があります。

データリンクを管理するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **dladm set-linkprop** を **-t** とともに使用して、**excl** という実行中のゾーンに **bge0** を追加します。

```
global# dladm set-linkprop -t -p zone=excl bge0
LINK         PROPERTY        VALUE          DEFAULT        POSSIBLE
bge0         zone            excl           --             --
```

ヒント – -p オプションを使用すると、マシンで構文解析できる安定した形式の表示が生成されます。

▼ dladm reset-linkprop の使用方法

dladm コマンドを reset-linkprop サブコマンドとともに使用して、bge0 リンクの値を未設定にリセットできます。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **dladm reset-linkprop** を **-t** とともに使用して、ゾーンへの **bge0** デバイスの割り当てを解除します。

```
global# dladm reset-linkprop -t -p zone=excl bge0
LINK         PROPERTY        VALUE          DEFAULT        POSSIBLE
bge0         zone            excl           --             --
```

ヒント – -p オプションを使用すると、マシンで構文解析できる安定した形式の表示が生成されます。

注意事項　実行中のゾーンでそのデバイスが使用中の場合、再割り当ては失敗し、エラーメッセージが表示されます。441 ページの「排他的 IP ゾーンがデバイスを使用しているために dladm reset-linkprop が失敗する」を参照してください。

ゾーンがインストールされている Oracle Solaris システムでの公平配分スケジューラの使用

prctl コマンドで指定された制限には持続性がありません。システムがリブートされると、制限は無効になります。ゾーンの配分を持続的に設定する方法については、266 ページの「ゾーンの構成方法」および 277 ページの「大域ゾーンの zone.cpu-shares を設定する方法」を参照してください。

▼ prctl コマンドを使用して大域ゾーンの FSS 配分を設定する方法

大域ゾーンには、デフォルトで 1 つの配分が付与されます。ここで説明する手順を使って、デフォルトの割り当てを変更できます。システムをリブートするたびに prctl コマンドで割り当てた配分を設定し直す必要があります。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは Primary Administrator 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **prctl** ユーティリティーを使用して、2 つの配分を大域ゾーンに割り当てます。

   ```
   # prctl -n zone.cpu-shares -v 2 -r -i zone global
   ```

3. (オプション) 大域ゾーンに割り当てられた配分の数を確認するには、次のように入力します。

   ```
   # prctl -n zone.cpu-shares -i zone global
   ```

参照　prctl ユーティリティーの詳細は、prctl(1) のマニュアルページを参照してください。

▼ ゾーンの zone.cpu-shares 値を動的に変更する方法

この手順は、大域ゾーンだけでなく任意のゾーンに対して使用できます。

1 スーパーユーザーまたは Primary Administrator 役割になります。
役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 `prctl` コマンドを使って `cpu-shares` の新しい値を指定します。

```
# prctl -n zone.cpu-shares -r -v value -i zone zonename
```

idtype は *zonename*、*zoneid* のいずれかです。*value* は新しい値です。

ゾーン管理での権利プロファイルの使用

このセクションでは、非大域ゾーンで権利プロファイルを使用することに関連したタスクについて説明します。

▼ Zone Management プロファイルを割り当てる方法

ユーザーは、Zone Management プロファイルを使用することで、システムの非大域ゾーンをすべて管理できます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは Primary Administrator 役割になります。
役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 Zone Management 権利プロファイルを含む役割を作成し、その役割をユーザーに割り当てます。

- Oracle Solaris Management Console を使用して役割の作成および割り当てを行う方法については、『Solaris のシステム管理 (セキュリティサービス)』の「RBAC の構成 (作業マップ)」を参照してください。「GUI を使用して役割の作成および割り当てを行う方法」のタスクを参照してください。
- コマンド行で役割の作成および割り当てを行う方法については、『Solaris のシステム管理 (セキュリティサービス)』の「RBAC の管理」を参照してください。「コマンド行から役割を作成する方法」のタスクを参照してください。

例 — ゾーンコマンドでのプロファイルシェルの使用

pfexec プログラムを使用して、プロファイル内でゾーンコマンドを実行できます。プログラムは、exec_attr データベース内のユーザープロファイルで指定された属性を使って、コマンドを実行します。プログラムの呼び出しには、プロファイルシェル pfksh、pfcsh、および pfsh が使用されます。

pfexec プログラムを使用して、 my-zone などのゾーンにログインします。

```
machine$ pfexec zlogin my-zone
```

ゾーンがインストールされている Oracle Solaris システムのバックアップ

次の手順を使用して、ゾーン内のファイルをバックアップできます。ゾーンの構成ファイルもバックアップしてください。

▼ ufsdump を使用してバックアップを実行する方法

ufsdump コマンドを使用して、完全バックアップまたは増分バックアップを実行できます。この手順では、ゾーン /export/my-zone を /backup/my-zone.ufsdump にバックアップします。ここで *my-zone* は使用しているシステム上のゾーンの名前に置き換えてください。/backup にマウントされたファイルシステムなど、別のファイルシステムにバックアップを格納することもできます。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. (オプション) 共有ファイルシステムのバックアップを作成しないように、ゾーンを停止して休止状態にします。

   ```
   global# zlogin -S my-zone init 0
   ```

3. ゾーンのステータスを検査します。

   ```
   global# zoneadm list -cv
   ```

   次のような情報が表示されます。

   ```
   ID  NAME     STATUS       PATH                           BRAND      IP
    0  global   running      /                              native     shared
    -  my-zone  installed    /export/home/my-zone           native     shared
   ```

4 バックアップを実行します。

```
global# ufsdump 0f /backup/my-zone.ufsdump /export/my-zone
```

次のような情報が表示されます。

```
DUMP: Date of this level 0 dump: Wed Aug 10 16:13:52 2005
DUMP: Date of last level 0 dump: the epoch
DUMP: Dumping /dev/rdsk/c0t0d0s0 (bird:/) to /backup/my-zone.ufsdump.
DUMP: Mapping (Pass I) [regular files]
DUMP: Mapping (Pass II) [directories]
DUMP: Writing 63 Kilobyte records
DUMP: Estimated 363468 blocks (174.47MB).
DUMP: Dumping (Pass III) [directories]
DUMP: Dumping (Pass IV) [regular files]
DUMP: 369934 blocks (180.63MB) on 1 volume at 432 KB/sec
DUMP: DUMP IS DONE
```

5 ゾーンをブートします。

```
global# zoneadm -z my-zone boot
```

▼ fssnap を使用して UFS スナップショットを作成する方法

この手順では、fssnap コマンドを使用して、バックアップのためにファイルシステムの一時的なイメージを作成します。

この方法を利用すれば、ゾーンファイルだけの純粋で一貫性のあるバックアップファイルを作成することができ、しかも、その作業をゾーンの動作中に実行できます。ただし、スナップショットを作成するときには、ファイルを更新中のアクティブなアプリケーションを中断するか、チェックポイントを設定することをお勧めします。スナップショットが作成されるときにファイルを更新中のアプリケーションは、これらのファイルを、内部的に一貫性のない状態、一部を切り捨てた状態、または使用できない状態にすることがあります。

下記の手順の例では、次のようになっています。

- /export/home の下に my-zone というゾーンがあります。
- /export/home は別のファイルシステムです。

始める前に バックアップ先は /backup/my-zone.ufsdump です。/ の下に backup ディレクトリを作成する必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 スナップショットを作成します。

```
global# fssnap -o bs=/export /export/home
```

次のような情報が表示されます。

```
dev/fssnap/0
```

3 スナップショットをマウントします。

```
global# mount -o ro /dev/fssnap/0 /mnt
```

4 スナップショットから **my-zone** をバックアップします。

```
global# ufsdump 0f /backup/my-zone.ufsdump /mnt/my-zone
```

次のような情報が表示されます。

```
DUMP: Date of this level 0 dump: Thu Oct 06 15:13:07 2005
   DUMP: Date of last level 0 dump: the epoch
   DUMP: Dumping /dev/rfssnap/0 (pc2:/mnt) to /backup/my-zone.ufsdump.
   DUMP: Mapping (Pass I) [regular files]
   DUMP: Mapping (Pass II) [directories]
   DUMP: Writing 32 Kilobyte records
   DUMP: Estimated 176028 blocks (85.95MB).
   DUMP: Dumping (Pass III) [directories]
   DUMP: Dumping (Pass IV) [regular files]
   DUMP: 175614 blocks (85.75MB) on 1 volume at 2731 KB/sec
   DUMP: DUMP IS DONE
```

5 スナップショットをアンマウントします。

```
global# umount /mnt
```

6 スナップショットを削除します。

```
global# fssnap -d /dev/fssnap/0
```

システムをリブートすると、スナップショットもシステムから削除されます。

▼ **find** および **cpio** を使用してバックアップを実行する方法

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 ディレクトリをルートディレクトリに変更します。

```
global# cd /
```

3 ループバックマウントされていない **my-zone** ファイルを **/backup/my-zone.cpio** にバックアップします。

```
global# find export/my-zone -fstype lofs -prune -o -local
 | cpio -oc -O /backup/my-zone.cpio      type as one line
```

4 結果を確認します。

```
global# ls -l backup/my-zone.cpio
```

次のような情報が表示されます。

```
-rwxr-xr-x   1 root     root     99680256 Aug 10 16:13 backup/my-zone.cpio
```

▼ ゾーン構成のコピーを出力する方法

非大域ゾーン構成のバックアップファイルを作成することをお勧めします。必要に応じて、このバックアップを使用してゾーンをあとで再作成することができます。ゾーンにはじめてログインして `sysidtool` からの質問に回答したあとに、ゾーンの構成のコピーを作成します。この手順では、処理をわかりやすく説明するために、`my-zone` という名前のゾーンと `my-zone.config` という名前のバックアップファイルを使用します。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 ゾーン **my-zone** の構成を **my-zone.config** というファイルに出力します。

```
global# zonecfg -z my-zone export > my-zone.config
```

非大域ゾーンの復元

▼ 非大域ゾーンを個別に復元する方法

必要に応じて、非大域ゾーン構成のバックアップファイルを使用して非大域ゾーンを復元できます。この手順では、ゾーンを復元する処理をわかりやすく説明するために、`my-zone` という名前のゾーンと `my-zone.config` という名前のバックアップファイルを使用します。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 ゾーン **my-zone** を再作成するために、**my-zone.config** を **zonecfg** コマンドファイルとして使用することを指定します。

```
global# zonecfg -z my-zone -f my-zone.config
```

3 ゾーンをインストールします。

```
global# zoneadm -z my-zone install
```

4 ゾーンに最初にログインするときに **sysidtool** からの質問が表示されないようにする場合は、たとえば次のようにして *zonepath*/**root/etc/.UNCONFIGURED** ファイルを削除します。

```
global# rm /export/home/my-zone/root/etc/.UNCONFIGURED
```

5 ゾーン固有のファイル (アプリケーションデータなど) を復元する場合は、新しく作成したゾーンのルートファイルシステムに対してバックアップのファイルを手動で復元します。手作業でのマージが必要な場合があります。

◆ ◆ ◆ 第 29 章

29 非大域ゾーンにインストールされている Oracle Solaris 10 システムのアップグレード

この章では、Oracle Solaris コンテナ (ゾーン) を実行中の場合に、Oracle Solaris 10 システムをそれ以降のリリースにアップグレードする方法について説明します。Oracle Solaris のインストールに関するドキュメントへのリンクを記載しています。

Oracle Solaris 10 8/07 についてこの章に追加されている説明

ゾーンがインストールされているシステムで Oracle Solaris Live Upgrade がサポートされるようになりました。`zonepath` を ZFS 上に設定することはできません。

Oracle Solaris 10 10/08 についてこの章に追加されている説明

このリリース以降、`zonepath` が ZFS 上に設定されているシステムで Oracle Solaris Live Upgrade がサポートされるようになります。`zonepath` が ZFS 上に設定されているゾーンの場合、システムのアップグレードに使用できるのは Oracle Solaris Live Upgrade だけです。

Oracle Solaris Live Upgrade 機能を使用して、ゾーンを ZFS ルートファイルシステムに移行できます。共有でないファイルシステム内のゾーンは、UFS ルートファイルシステムが ZFS ルートファイルシステムに移行される際に、自動的に移行されます。共有 UFS ファイルシステム内のゾーンは、以前の Oracle Solaris リリースと同様の方法でアップグレードする必要があります。詳細は、『Oracle Solaris ZFS 管理ガイド』の「ZFS ルートファイルシステムへの移行または ZFS ルートファイルシステムの更新 (Live Upgrade)」を参照してください。

アップグレードする前のシステムのバックアップ

アップグレードを実行する前に、Oracle Solaris システムの大域ゾーンと非大域ゾーンをバックアップしてください。詳細は、404 ページの「ゾーンがインストールされている Oracle Solaris システムのバックアップについて」および431 ページの「ゾーンがインストールされている Oracle Solaris システムのバックアップ」を参照してください。

ゾーンがインストールされているシステムの Oracle Solaris 10 8/07 以降の更新リリースへのアップグレード

ゾーンがインストールされている Oracle Solaris システムをアップグレードするときには、Oracle Solaris Live Upgrade、Oracle Solaris 標準の対話型インストールプログラム、またはカスタム JumpStart インストールプログラムを使用できます。詳細は、『Solaris 10 8/07 Installation Guide: Planning for Installation and Upgrade』の「Upgrading With Non-Global Zones」を参照してください。`zonepath` が ZFS 上にあるとき、437 ページの「Oracle Solaris 10 8/07 についてこの章に追加されている説明」および437 ページの「Oracle Solaris 10 10/08 についてこの章に追加されている説明」も参照してください。

Oracle Solaris ゾーンで Oracle Solaris Live Upgrade を使用するためのガイドライン

ゾーンがインストールされているシステムで Live Upgrade を使用する場合は、考慮すべき事項がいくつかあります。`lucreate` および `lumount` 操作の実行中にゾーン状態が遷移しないようにすることが非常に重要です。

- ある特定のゾーンが実行していない場合に、`lucreate` コマンドを使って代替ブート環境 (ABE) を生成するときは、`lucreate` が完了するまで、そのゾーンをブートすることはできません。
- ある特定のゾーンが実行している場合に、`lucreate` コマンドを使って ABE を生成するときは、`lucreate` が完了するまで、そのゾーンを停止したり、リブートしたりしてはいけません。
- ABE が `lumount` によってマウントされるときは、ゾーンをブートまたはリブートすることはできません。ただし、`lumount` 操作の前に実行していたゾーンは引き続き実行できます。

非大域ゾーンは非大域ゾーン管理者だけでなく大域管理者も制御できるため、`lucreate` または `lumount` の操作中はすべてのゾーンを停止するのが最善の策です。

Live Upgrade 操作の進行中は、非大域ゾーン管理者が介入してはいけません。アップグレードは、アップグレードによって発生する変更に対処する予定の管理者の作業に影響を及ぼします。ゾーン管理者は、すべてのローカルパッケージが一連の操作を通じて確実に安定しているようにし、構成ファイルの調整といったアップグレード後のタスクをすべて行い、通常はシステムの機能停止を避けたスケジュールを立てる必要があります。

ゾーンがインストールされているシステムの Oracle Solaris 10 6/06 または Oracle Solaris 10 11/06 へのアップグレード

システムをアップグレードする前に、446 ページの「lofs タイプで定義された fs リソースを持つゾーンを Oracle Solaris 10 11/06 リリースにアップグレードできない」を読んでください。

ゾーンがインストールされている Oracle Solaris システムをアップグレードするときには、Oracle Solaris 標準の対話型インストールプログラムまたはカスタム JumpStart インストールプログラムを使用できます。Oracle Solaris Live Upgrade は、このリリースではサポートされていません。詳細は、『Solaris 10 11/06 Installation Guide: Solaris Live Upgrade and Upgrade Planning』および『Solaris 10 11/06 Installation Guide: Custom JumpStart and Advanced Installations』を参照してください。

- すべてのタイプのインストールとアップグレードに関する全般的な計画情報および要件については、『Solaris 10 11/06 Installation Guide: Planning for Installation and Upgrade』の第 4 章「System Requirements, Guidelines, and Upgrade (Planning)」に記載されています。インストールするときには、DVD、または DVD から作成したネットワークインストールイメージのいずれかのメディアを使用する必要があります。
- Oracle Solaris 10 リリースのインタフェースについては、『Solaris 10 11/06 Installation Guide: Basic Installations 』に記載されています。
- カスタム JumpStart インストールに固有の考慮事項と制限については、『Solaris 10 11/06 Installation Guide: Custom JumpStart and Advanced Installations』の第 8 章「Custom JumpStart (Reference)」に記載されています。
- ネットワーク経由でインストールまたはアップグレードする方法については、『Solaris 10 11/06 Installation Guide: Network-Based Installations』に記載されています。

◆ ◆ ◆ 第 30 章

30 Oracle Solaris ゾーンで発生するさまざまな問題のトラブルシューティング

この章は、Oracle Solaris 10 6/06 リリースで新たに追加されました。

Oracle Solaris 10 の新機能の一覧および Oracle Solaris リリースについての説明は、『Oracle Solaris 10 8/11 の新機能』を参照してください。

Oracle Solaris 10 6/06、Oracle Solaris 10 11/06、Oracle Solaris 10 8/07、および Oracle Solaris 10 5/08: 非大域ゾーンのルートファイルシステムを ZFS 上に配置しないでください

これらのリリースでは、非大域ゾーンの `zonepath` を ZFS 上に設定すべきではありません。この操作の結果、パッチ適用に関する問題が発生し、システムを新しい Oracle Solaris 10 更新リリースにアップグレードできなくなる可能性があります。

Oracle Solaris 10 10/08 リリース以降では、非大域ゾーンのルートファイルシステムを ZFS 上に設定できます。その場合は、Oracle Solaris Live Upgrade を使用してシステムをアップグレードできます。

排他的 IP ゾーンがデバイスを使用しているために `dladm reset-linkprop` が失敗する

次のエラーメッセージが表示された場合

```
dladm: warning: cannot reset link property 'zone' on 'bge0': operation failed
```

428 ページの「dladm reset-linkprop の使用方法」で説明されているように、dladm reset-linkprop を使用しようとして失敗しました。実行中のゾーン excl は、ゾーン内部で ifconfig bge0 plumb を実行することによって割り当てられたデバイスを使用中です。

値をリセットするには、手続き ifconfig bge0 unplumb をゾーン内部で実行し、dladm コマンドを再度実行します。

大域ゾーンによってデータが挿入されているファイルシステムをゾーン管理者がマウントする場合

非大域ゾーンの初回ブート時にファイルシステム階層内にファイルが存在している場合は、そのファイルシステムデータが大域ゾーンによって管理されていることを示しています。非大域ゾーンがインストールされたときに、大域ゾーン内のいくつかのパッケージファイルがそのゾーン内に複製されています。これらのファイルは zonepath ディレクトリの下に置かれている必要があります。ゾーンに追加されているディスクデバイスや ZFS データセット上にゾーン管理者によってファイルシステムが作成され、そこにこれらのファイルが置かれている場合は、パッケージとパッチの問題が発生する可能性があります。

大域ゾーンによって管理されているファイルシステムデータを、ゾーンにローカルなファイルシステムに保存する場合の問題は、ZFS を例にとって説明することができます。ZFS データセットが非大域ゾーンに委任されている場合、ゾーン管理者は、大域ゾーンによって管理されているファイルシステムデータの保存のためにそのデータセットを使用するべきではありません。構成にパッチやアップグレードを正しく適用できなくなります。

たとえば、委任されている ZFS データセットを /var ファイルシステムとして使用するべきではありません。Oracle Solaris オペレーティングシステムでは、/var にコンポーネントをインストールする主要パッケージが提供されています。これらのパッケージは、アップグレードやパッチの適用時に /var にアクセスする必要がありますが、委任されている ZFS データセットに /var がマウントされているとそれが不可能になります。

大域ゾーンによって制御されている階層の部分の下にファイルシステムをマウントすることはサポートされています。たとえば、大域ゾーンに空の /usr/local ディレクトリが存在している場合、ゾーン管理者はそのディレクトリにほかの内容をマウントすることができます。

非大域ゾーンの /export などのように、パッチやアップグレードの適用時にアクセスする必要のないファイルシステムには、委任されている ZFS データセットを使用することができます。

ゾーンが停止しない

ゾーンに関連付けられたシステム状態を破棄できない場合には、停止処理は中途で失敗します。このため、稼働中とインストール済みの間で、ゾーンが中間状態のままになります。この状態では、アクティブなユーザープロセスやカーネルスレッドは存在せず、何も作成できません。停止操作が失敗した場合は、手動で処理を完了する必要があります。

障害のもっとも一般的な原因は、システムがすべてのファイルシステムをマウント解除できないことです。システム状態が破棄される Oracle Solaris システムの従来の停止処理とは異なり、ゾーンでは、ゾーンのブートまたは操作時に実行されたマウントがゾーンの停止後に残らないことを保証する必要があります。`zoneadm` によりゾーン内で実行中のプロセスが存在しないことが確認されても、大域ゾーン内のプロセスによりゾーン内のファイルが開かれた場合には、マウント解除操作が失敗することがあります。`proc`(1) (`pfiles` に記載) および `fuser`(1M) のマニュアルページに記載されているツールを使用してこれらのプロセスを検索し、適切な処理を実行してください。これらのプロセスを処理したあとで `zoneadm halt` を再度呼び出すと、ゾーンが完全に停止するはずです。

停止できないゾーンがある場合、Oracle Solaris 10 10/09 リリースでは、`zoneadm attach -F` オプションを使用して妥当性検査を行わずに接続を強制的に実行すれば、切り離されなかったゾーンを移行できます。ターゲットシステムは、ゾーンをホストするように正しく構成されている必要があります。構成が不正な場合、未定義の動作が実行される可能性があります。また、ゾーン内のファイルの状態を知る方法はありません。

ゾーン構成内に不正な特権セットが指定されている

ゾーンの特権セットに許可されない特権が含まれる場合、必須の特権が欠落している場合、または不明な特権名が含まれる場合、ゾーンの検証、準備、またはブートの試行は失敗し、次のようなエラーメッセージが表示されます。

```
zonecfg:zone5> set limitpriv="basic"
.
.
.
global# zoneadm -z zone5 boot
     required privilege "sys_mount" is missing from the zone's privilege set
     zoneadm: zone zone5 failed to verify
```

ゾーンブート時に netmasks の警告が表示される

296 ページの「ゾーンのブート方法」の説明に従ってゾーンをブートする際、次のメッセージが表示されることがあります。

```
# zoneadm -z my-zone boot
zoneadm: zone 'my-zone': WARNING: hme0:1: no matching subnet
    found in netmasks(4) for 192.168.0.1; using default of
    255.255.255.0.
```

このメッセージは単なる警告であり、コマンドは成功しています。このメッセージは、ゾーン構成で指定された IP アドレス用のネットマスク をシステムが検出できなかったことを示します。

以降のリブートでこの警告が表示されないようにするには、大域ゾーン内の /etc/nsswitch.conf ファイルに適切な netmasks データベースが記載されていること、およびこれらのデータベースの 1 つ以上にゾーン my-zone で使用されるサブネットおよびネットマスクが含まれることを確認します。

たとえば、大域ゾーン内のネットマスクの解決に /etc/inet/netmasks ファイルおよびローカルの NIS データベースが使用される場合、/etc/nsswitch.conf 内の適切なエントリは次のようになります。

```
netmasks: files nis
```

その後、ゾーン my-zone のサブネットおよび対応するネットマスク情報を /etc/inet/netmasks に追加して、これを使用可能にします。

netmasks コマンドの詳細は、netmasks(4) のマニュアルページを参照してください。

zoneadm 接続操作の問題解決

▼ パッチおよびパッケージが同期しない

ターゲットシステムでは、元のホストにインストールされているものと同じバージョンの、次の必須オペレーティングシステムパッケージおよびパッチが実行されている必要があります。

- ファイルを inherit-pkg-dir リソース内に格納するパッケージ
- SUNW_PKG_ALLZONES=true であるパッケージ

1 パッケージおよびパッチが移行元ホストと新規ホスト間で異なる場合、次のようなメッセージが表示されることがあります。

```
host2# zoneadm -z my-zone attach
    These packages installed on the source system are inconsistent with this system:
            SUNWgnome-libs (2.6.0,REV=101.0.3.2005.12.06.20.27) version mismatch
```

```
                (2.6.0,REV=101.0.3.2005.12.19.21.22)
        SUNWudaplr (11.11,REV=2005.12.13.01.06) version mismatch
                (11.11,REV=2006.01.03.00.45)
        SUNWradpu320 (11.10.0,REV=2005.01.21.16.34) is not installed
        SUNWaudf (11.11,REV=2005.12.13.01.06) version mismatch
                (11.11,REV=2006.01.03.00.45)
        NCRos86r (11.10.0,REV=2005.01.17.23.31) is not installed
These packages installed on this system were not installed on the source system:
        SUNWukspfw (11.11,REV=2006.01.03.00.45) was not installed
        SUNWsmcmd (1.0,REV=2005.12.14.01.53) was not installed
These patches installed on the source system are inconsistent with this system:
        120081 is not installed
        118844 is not installed
        118344 is not installed
These patches installed on this system were not installed on the source system:
        118669 was not installed
        118668 was not installed
        116299 was not installed
```

2 ゾーンの移行を成功させるには、次のいずれかの方法を使用します。

- 正しいパッケージとパッチを使って新規ホストを更新し、両方のシステムで同じ内容になるようにします。詳細は、第 25 章「ゾーンがインストールされている Oracle Solaris システムでのパッケージとパッチについて (概要)」および第 26 章「ゾーンがインストールされている Oracle Solaris システムでのパッケージとパッチの追加および削除 (タスク)」を参照してください。
- ゾーンに依存するパッケージまたは関連パッチが新しいホストに存在する場合、それらのバージョンが移行元より新しいときは、**zoneadm attach** と **-u** または **-U** オプションを使用してゾーン内のこのようなパッケージを更新し、新しいホストに一致させます。322 ページの「ゾーンの移行について」を参照してください。

▼ オペレーティングシステムのリリースが一致しない

ゾーンの移行を成功させるには、移行元ホストで稼働しているものと同じ Oracle Solaris リリースを、同じアーキテクチャーを持つシステムにインストールしてください。

1 移行元システムで稼働している **Oracle Solaris** のリリースを確認してください。

```
host1# uname -a
```

2 新規ホストに同じリリースをインストールします。

`docs.sun.com` にある Oracle Solaris インストールドキュメントを参照してください。

▼ マシンアーキテクチャーが一致しない

ゾーンの移行を成功させるには、zoneadm attach に -u オプションを使用します。

1. 両方のシステムのシステムアーキテクチャーを確認します。

```
host1# uname -a
```

2. アーキテクチャーが異なっている場合は、**zoneadm attach** に **-u** オプションを使用して接続を実行します。

```
host2# zoneadm -z my-zone attach -u
```

詳細は、324 ページの「非大域ゾーンを移行する方法」を参照してください。

lofs タイプで定義された fs リソースを持つゾーンを Oracle Solaris 10 11/06 リリースにアップグレードできない

注 – この問題は Oracle Solaris 10 8/07 リリースで修正されています。

lofs fs リソースで構成されたすべての非大域ゾーンが miniroot 内に存在するディレクトリをマウントしている場合、標準のアップグレードを使用して、以前の Oracle Solaris 10 リリースから Oracle Solaris 10 11/06 リリースにシステムをアップグレードできます。たとえば、lofs でマウントされた /opt ディレクトリは、問題なくアップグレードできます。

ただし、いずれかの非大域ゾーンが非標準の lofs マウントで構成されている場合 (たとえば、/usr/local ディレクトリが lofs でマウントされている場合)、次のエラーメッセージが表示されます。

```
The zones upgrade failed and the system needs to be restored
from backup.  More details can be found in the file
/var/sadm/install_data/upgrade_log on the upgrade root file
system.
```

このエラーメッセージには、システムをバックアップから復元する必要があると表示されていますが、システムは実際には正常であり、次の回避方法を使用すれば問題なくアップグレードできます。

1. インストールされている OS でシステムをリブートします。
2. ゾーンを再構成し、lofs タイプで定義された fs リソースを削除します。
3. これらのリソースを削除したあとで、システムを Oracle Solaris 10 11/06 にアップグレードします。

4. アップグレード後にゾーンを再構成して、削除した `fs` リソースを復元できます。

パート III

lx ブランドゾーン

Oracle Solaris 10 8/07: ブランドゾーンは、このリリースから使用可能になりました。

BrandZ は、ネイティブでないオペレーティング環境を含む非大域ブランドゾーンを作成するためのフレームワークを提供します。ブランドゾーンは、Oracle Solaris オペレーティングシステムでアプリケーションを実行するために使用します。

使用可能な最初のブランドは lx ブランドで、これは Linux アプリケーション用の Oracle Solaris コンテナでした。lx ブランドはアプリケーションに Linux 環境を提供するもので、x86 および x64 マシンで稼働します。

◆ ◆ ◆ 第 31 章

ブランドゾーンと Linux ブランドゾーンについて

Oracle Solaris 10 8/07 リリースから、ブランドゾーンが使用可能になりました。以降の更新リリースで追加された機能は、リリースごとに分類されています。

Oracle Solaris オペレーティングシステムのブランドゾーン機能は、Oracle Solaris ゾーンの単純な拡張です。この章では、ブランドゾーンの概念と、Linux ブランドゾーン機能を実装する `lx` ブランドについて説明します。Linux ブランドゾーンは、Linux アプリケーション用 Solaris コンテナとも呼ばれます。

注 – ラベルが有効になっている Oracle Trusted Solaris システムにブランドゾーンを構成してインストールすることはできますが、このシステム構成でブランドゾーンをブートすることはできません。

注 – Oracle Solaris オペレーティングシステムでは、追加のブランドがサポートされています。

Oracle Solaris 10 8/07 オペレーティングシステムまたはそれ以降の Oracle Solaris 10 リリースを実行する SPARC マシンでは、次の 2 つのブランドがサポートされています。

- 『Solaris のシステム管理: Solaris 8 Containers』に記載されている `solaris8` ブランド、Oracle Solaris 8 コンテナ
- 『Solaris のシステム管理: Solaris 9 Containers』に記載されている `solaris9` ブランド、Oracle Solaris 9 コンテナ

`docs.sun.com` の Sun Cluster 3.2 1/09 Software Collection for Solaris OS に記載されているクラスタ ブランドも、Solaris 10 リリースでサポートされています。

Oracle Solaris システムでのゾーンの使用について

Oracle Solaris システムでのゾーンの使用に関する一般的な情報については、第 16 章「Solaris ゾーンの紹介」を参照してください。

次に示すゾーンおよびリソース管理の概念について、よく理解しておくようにしてください。

- 大域ゾーンと非大域ゾーン。219 ページの「ゾーンのしくみ」に記載されています。
- 大域管理者とゾーン管理者。221 ページの「非大域ゾーンの管理のしくみ」および 222 ページの「非大域ゾーンの作成のしくみ」に記載されています。
- ゾーンの状態モデル。222 ページの「非大域ゾーンの状態モデル」に記載されています。
- ゾーン隔離の特徴。詳細は、224 ページの「非大域ゾーンの特性」を参照してください。
- 特権。397 ページの「非大域ゾーン内の特権」に記載されています。
- ネットワーク。387 ページの「共有 IP 非大域ゾーンにおけるネットワーク」に記載されています。
- Oracle Solaris コンテナの概念、つまり、リソースプールなどのリソース管理機能をゾーンで使用する方法。ゾーンとリソース管理機能の使用法と相互動作については、225 ページの「非大域ゾーンでのリソース管理機能の使用」、240 ページの「ゾーン規模のリソース制御の設定」、第 27 章「Oracle Solaris ゾーンの管理 (概要)」、およびこのマニュアルのパート 1「リソース管理」で各リソース管理機能について説明した章を参照してください。たとえば、リソースプールについては、第 12 章「リソースプール (概要)」および第 13 章「リソースプールの作成と管理 (タスク)」を参照してください。
- 公平配分スケジューラ (FSS)。これは、配分に基づいて CPU 時間を割り当てることができるようにするスケジューリングクラスです。詳細は、第 8 章「公平配分スケジューラ (概要)」および第 9 章「公平配分スケジューラの管理 (タスク)」を参照してください。
- リソース上限デーモン (`rcapd`)。これを大域ゾーンから使用して、ブランドゾーンの常駐セットサイズ (RSS) 使用量を制御できます。`zonecfg capped-memory` リソースのプロパティーは、ゾーンの `max-rss` を設定します。この値は、大域ゾーンで実行中の `rcapd` によって適用されます。詳細は、第 10 章「リソース上限デーモンによる物理メモリーの制御 (概要)」、第 11 章「リソース上限デーモンの管理 (タスク)」、および `rcapd(1M)` のマニュアルページを参照してください。

用語集には、ゾーンとリソース管理機能で使用される用語の定義が記載されています。

システムでブランドゾーンを使用するために必要な追加情報は、ガイドのこのパートに記載されています。

注 – このガイドで、次の章はブランドゾーンには当てはまりません。

- 第 25 章「ゾーンがインストールされている Oracle Solaris システムでのパッケージとパッチについて (概要)」
- 第 26 章「ゾーンがインストールされている Oracle Solaris システムでのパッケージとパッチの追加および削除 (タスク)」

ブランドゾーン技術

ブランドゾーン (BrandZ) フレームワークは、このマニュアルの
パート II「ゾーン」で説明されている Oracle Solaris ゾーンインフラストラクチャーを拡張して、ブランドの作成機能を追加します。「ブランド」という用語は、さまざまなオペレーティング環境を指す場合があります。BrandZ を使用すると、アプリケーションを実行するための、ネイティブでないオペレーティング環境を含む非大域ブランドゾーンを作成できます。ブランドタイプにより、ゾーンのインストール時およびブート時に実行されるスクリプトが決定されます。また、ゾーンのブランドにより、アプリケーションの起動時に正しいアプリケーションタイプが適切に識別されます。すべてのブランド管理は、現在のゾーン構造の拡張を通して実行されます。

ブランドでは、単純な環境を提供することも、複雑な環境を提供することもできます。たとえば、単純な環境では、Oracle Solaris の標準ユーティリティーを同等の GNU ユーティリティーで置き換えることができます。複雑な環境では、Linux アプリケーションの実行をサポートする完全な Linux ユーザー空間を提供できます。

すべてのゾーンに、それぞれ関連するブランドが構成されます。デフォルトは `native` ブランド、つまり Oracle Solaris です。ブランドゾーンは、ネイティブでないバイナリのブランドを 1 つだけサポートします。つまり、1 つのブランドゾーンは 1 つのオペレーティング環境を提供します。

BrandZ はゾーンのツールを次のように拡張します。

- `zonecfg` コマンドを使用して、ゾーンの構成時にゾーンのブランドタイプを設定します。
- `zoneadm` コマンドを使用して、ゾーンのブランドタイプの報告とゾーンの管理を行います。

注 – 構成済み状態にあるゾーンのブランドは変更することができます。ブランドゾーンのインストールが完了したあとは、そのブランドの変更や削除を行うことはできません。

ブランドゾーンで実行中のプロセス

ブランドゾーンでは、ブランドゾーンで実行中のプロセスだけに適用される一連の介入ポイントがカーネル内に用意されます。

- これらのポイントは、`syscall` パス、プロセスローディングパス、スレッド作成パスなどのパス内に見つかります。
- これらの各ポイントで、ブランドは Oracle Solaris の標準的な動作を補完したり置き換えたりすることができます。

ブランドは `librtld_db` のプラグインライブラリを提供することもできます。デバッガ (`mdb(1)` に記載) や DTrace (`dtrace(1M)` に記載) といった Oracle Solaris のツールは、このプラグインライブラリを使用することによって、ブランドゾーン内で実行中のプロセスのシンボル情報にアクセスできます。

ブランドゾーンのデバイスのサポート

各ゾーンでサポートされるデバイスについては、そのブランドに関するマニュアルページやほかのドキュメントに記載されています。デバイスのサポートは、ブランドによって定義されます。ブランドでは、サポートされていないデバイスや認識されないデバイスの追加を禁止するように選択することができます。

ブランドゾーンのファイルシステムのサポート

ブランドゾーンに必要なファイルシステムは、ブランドによって定義されます。

ブランドゾーンの特権

ブランドゾーンで使用できる特権は、ブランドによって定義されます。特権の詳細については、397 ページの「非大域ゾーン内の特権」および 468 ページの「lx ブランドゾーンで構成可能な特権」を参照してください。

lx ブランドについて

lx ブランドは、ブランドゾーンフレームワークを使用して、Linux バイナリアプリケーションを変更することなく、Oracle Solaris システムのカーネルを備えたマシンで実行できるようにします。

マシンは、サポートされている次のプロセッサタイプのいずれかを備えている必要があります。

- Intel
  - Pentium Pro
  - Pentium II
  - Pentium III
  - Celeron
  - Xeon
  - Pentium 4
  - Pentium M
  - Pentium D
  - Pentium Extreme Edition
  - Core
  - Core 2

  AMD

  - Opteron
  - Athlon XP
  - Athlon 64
  - Athlon 64 X2
  - Athlon FX
  - Duron
  - Sempron
  - Turion 64
  - Turion 64 X2

サポートされている Linux ディストリビューション

lx ブランドには、CentOS 3.*x* または Red Hat Enterprise Linux 3.*x* ディストリビューションを非大域ゾーン内にインストールするために必要なツールが含まれています。各ディストリビューションのバージョン 3.5 から 3.8 までがサポートされています。このブランドでは、32 ビットまたは 64 ビットモードの Oracle Solaris システムが稼働している x86 マシンおよび x64 マシンで、32 ビット Linux アプリケーションを実行できます。

lx ブランドは、Linux 2.4.21 カーネルで提供されるシステムコールインタフェースをエミュレートします。このカーネルは、Red Hat の RHEL 3.*x* ディストリビューションでの変更に従っています。このカーネルは、Red Hat によってリリースされた `glibc` バージョン 2.3.2 で消費されるシステムコールインタフェースを提供します。

また、lx ブランドは、Linux の `/dev` と `/proc` のインタフェースを部分的にエミュレートします。

注意 – lx ブランドゾーンにパッケージを追加する場合は、サポートされている構成を維持する必要があります。詳細は、529 ページの「サポートされている構成の保守について」を参照してください。

アプリケーションのサポート

Oracle Solaris システムでは、lx ブランドゾーン内で実行できる Linux アプリケーションの数は制限されていません。十分なメモリーを使用できることが必要です。461 ページの「システム要件と容量要件」も参照してください。

配下のカーネルに関わらず、実行できるのは 32 ビット Linux アプリケーションだけです。

lx ゾーンでは、ユーザーレベルの Linux アプリケーションだけがサポートされます。Linux デバイスドライバ、Linux カーネルモジュール、または Linux ファイルシステムを lx ゾーン内から使用することはできません。

アプリケーションのインストール例については、531 ページの「lx ブランドゾーンにアプリケーションをインストールする方法」を参照してください。

lx ゾーン内部で Oracle Solaris アプリケーションを実行することはできません。ただし、lx ゾーンを使用すると、Oracle Solaris システムを使用して Linux アプリケーションの開発、テスト、および配備を行うことができます。たとえば、Linux アプリケーションを lx ゾーンに配置し、大域ゾーンから Oracle Solaris ツールを実行してそれを解析できます。その後、改善を加えチューニングしたアプリケーションを、ネイティブな Linux システムに配備することができます。

デバッグツール

DTrace や mdb などの Oracle Solaris デバッグツールを、ゾーン内部で実行中の Linux プロセスに適用できます。ただし、ツール自体は大域ゾーンで実行する必要があります。生成されるコアファイルは、すべて Oracle Solaris 形式で作成され、Oracle Solaris ツールでのみデバッグできます。

DTrace lxsyscall 動的トレースプロバイダにより、Linux アプリケーションに対して DTrace が使用可能になります。このプロバイダの動作は、DTrace syscall プロバイダに似ています。lxsyscall プロバイダは、スレッドが Linux システムコールのエントリポイントに入ったり戻ったりするたびに起動するプローブを提供します。

デバッグオプションの詳細については、『Oracle Solaris 動的トレースガイド』および dtrace(1M) と mdb(1) のマニュアルページを参照してください。『Solaris 動的トレースガイド』では、DTrace 機能に使用できる公式なインタフェースが説明されています。syscall プロバイダに関するドキュメントは、lxsyscall プロバイダにも使用できます。

注 – NFS はネームサービスに依存しており、ネームサービスはゾーンに固有であるため、現在のゾーンの外部にマウントされている NFS ファイルシステムにアクセスすることはできません。したがって、NFS に基づく Linux プロセスを大域ゾーンからデバッグすることはできません。

コマンドとその他のインタフェース

次の表に示すコマンドにより、ゾーン機能に対する主要な管理インタフェースが提供されます。

表 31–1 lx ブランドゾーンで使用されるコマンドとその他のインタフェース

コマンドのリファレンス	説明
zlogin(1)	非大域ゾーンにログインします
zoneadm(1M)	システムのゾーンを管理します
zonecfg(1M)	ゾーン構成の設定に使用されます
getzoneid(3C)	ゾーン ID と名前のマッピングに使用されます
brands(5)	ブランドゾーン機能の説明を提供します
lx(5)	Linux ブランドゾーンの説明を提供します
zones(5)	ゾーン機能の説明を提供します
lx_systrace(7D)	DTrace の Linux システムコールトレースプロバイダ
zcons(7D)	ゾーンコンソールのデバイスドライバ

zoneadmd デーモンは、ゾーンの仮想プラットフォームを管理する基本プロセスです。zoneadmd デーモンのマニュアルページは、zoneadmd(1M) です。このデーモンは、プログラミングインタフェースの構成要素ではありません。

注 – 表 27–5 に示すコマンドを大域ゾーンで使用すると、ブランドゾーンも含むすべての非大域ゾーンに関する情報を表示できます。表 27–4 に示すコマンドは、リソース上限デーモンで使用されます。

システムの lx ブランドゾーンの設定 (タスクマップ)

次の表に、システム上ではじめて lx ゾーンを設定する際に必要となるタスクの概要を示します。

タスク	説明	参照先
ゾーンで実行する 32 ビット Linux アプリケーションをそれぞれ特定します。	システム要件を評価します。	必要に応じて、ビジネス目標とシステムのドキュメントを参照してください。
構成するゾーンの数を決定します。	次の内容を評価します。 ■ 実行する予定の Linux アプリケーションの数。 ■ Linux ブランドゾーンに必要なディスク容量 ■ スクリプトを使用する必要があるかどうか。	456 ページの「アプリケーションのサポート」、461 ページの「システム要件と容量要件」、260 ページの「現在のシステム設定の評価」、487 ページの「複数の lx ブランドゾーンを構成するスクリプト」を参照してください。
コンテナを作成するためにゾーンでリソースプールを使用するかどうかを決定します。	リソースプールを使用する場合は、ゾーンを構成する前にリソースプールを構成します。 zonecfg のプロパティーを使用すると、ゾーン規模のリソース制御とプール機能をゾーンにすばやく追加できます。	482 ページの「lx ブランドゾーンの構成方法」、第 13 章「リソースプールの作成と管理 (タスク)」を参照してください。
事前構成タスクを行います。	各ゾーンのゾーン名とゾーンパスを決定します。ネットワーク接続が必要な場合は、IP アドレスを取得します。ゾーンのスケジューリングクラスを決定します。標準のデフォルト特権セットでは十分でない場合は、ゾーンのプロセスを制限するための特権セットを決定します。	ゾーン名、ゾーンパス、IP アドレス、およびスケジューリングクラスについては、463 ページの「lx ブランドゾーン構成のコンポーネント」を参照してください。非大域ゾーンでのデフォルトの特権および構成可能な特権のリストについては、397 ページの「非大域ゾーン内の特権」を参照してください。 リソースプールの関連付けについては、219 ページの「ゾーンのしくみ」および 482 ページの「lx ブランドゾーンの構成方法」を参照してください。

タスク	説明	参照先
構成を作成します。	非大域ゾーンの構成を行います。	265 ページの「ゾーンを構成、検証、および確定する」および zonecfg(1M) のマニュアルページを参照してください。
大域管理者として、構成されたゾーンの確認とインストールを行います。	ゾーンをブートする前に、ゾーンの確認とインストールを行う必要があります。Linux ブランドゾーンをインストールする前に、Linux のディストリビューションを入手する必要があります。	第 34 章「lx ブランドゾーンのインストール、ブート、停止、複製、およびアンインストールについて (概要)」および第 35 章「lx ブランドゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)」を参照してください。
大域管理者として、非大域ゾーンをブートします。	それぞれのゾーンをブートして稼働状態にします。	第 35 章「lx ブランドゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)」を参照してください。
この新しいゾーンを本稼働用に準備します。	Linux システム管理の標準のツールと方法をゾーン内から使用して、ユーザーアカウントの作成、ソフトウェアの追加、およびゾーン構成のカスタマイズを行います。	新たにインストールしたマシンの設定およびアプリケーションのインストールを行うためのドキュメントを参照してください。ゾーンがインストールされているシステムに関連する特殊な考慮事項については、このガイドを参照してください。

◆ ◆ ◆ 第 32 章

lx ブランドゾーン構成の計画 (概要)

この章では、x64 システムまたは x86 システムに lx ブランドゾーンを構成する前に実行する必要のある作業について説明します。この章では、zonecfg コマンドの使用方法についても説明します。

システム要件と容量要件

次に、lx ブランドゾーンの使用に関連したマシンの主な考慮事項を示します。

- マシンは x64 ベースまたは x86 ベースでなければなりません。
- 各 lx ゾーンに固有のファイルを保持するために十分なディスク容量が必要です。lx ゾーンに必要なディスク容量は、インストールされる RPM (Linux パッケージ) のサイズと数によって決まります。
- lx ブランドは完全ルートモデルのみをサポートするので、インストールされる各ゾーンは独自にすべてのファイルのコピーを保持します。

ゾーンが消費可能なディスク容量に関する制限はありません。ディスク容量の制限を設定することは、大域管理者の役割です。大域管理者は、非大域ゾーンのルートファイルシステムを保持するのに十分なローカルストレージがあることを確認する必要があります。十分なストレージがあれば、小規模な単一プロセッサシステムでも、同時に稼働する多数のゾーンをサポートできます。

ブランドゾーンのサイズを制限する

ゾーンサイズを制限する際、次のオプションを使用できます。

- lofi を使用してマウントされたパーティションにゾーンを配置できます。この操作により、ゾーンの消費する容量が、lofi の使用するファイルの容量に制限されます。詳細は、lofiadm(1m) および lofi(7D) のマニュアルページを参照してください。

- ソフトパーティションを使用して、ディスクスライスまたは論理ボリュームをパーティションに分割できます。これらのパーティションをゾーンのルートとして使用することで、ゾーンごとのディスク消費量を制限できます。ソフトパーティションの上限は、8192 パーティションに制限されています。詳細は、『Solaris ボリュームマネージャの管理』の第 12 章「ソフトパーティション (概要)」を参照してください。
- ディスクの標準パーティションをゾーンのルートに使用できるため、ゾーンごとのディスク消費を制限できます。

ブランドゾーンのネットワークアドレス

ネットワーク接続を必要とする各ゾーンには、1 つ以上の一意の IP アドレスが与えられます。IPv4 アドレスがサポートされています。ゾーンに IPv4 アドレスを割り当てる必要があります。詳細は、462 ページの「ブランドゾーンのネットワークアドレス」を参照してください。必要に応じて、ネットワークインタフェースのデフォルトのルーターを設定することもできます。手順については、482 ページの「lx ブランドゾーンの構成方法」を参照してください。

lx ブランドゾーンの構成処理

`zonecfg` コマンドは、次の目的に使用されます。

- ゾーンのブランドを設定します。
- `lx` ゾーンの構成を作成します。
- 指定されたリソースおよびプロパティーが仮定の x64 システムまたは x86 システムで有効で内部的に一貫しているかどうかを調べるために、構成を確認します。
- ブランド固有の確認を実行します。確認により次のことが保証されます。
  - ゾーンでは、継承されたパッケージディレクトリ、ZFS データセット、または追加されたデバイスは保持できません。
  - オーディオを使用するようにゾーンを構成する場合、デバイスを指定するときは、`none`、`default`、または 1 桁の数字でなければなりません。

特定の構成について `zonecfg verify` コマンドで実行される検査では、次のことが確認されます。

- ゾーンパスが指定されていること
- 各リソースの必須プロパティーがすべて指定されていること
- ブランドの要件が満たされていること

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

lx ブランドゾーン構成のコンポーネント

このセクションでは、次のコンポーネントについて説明します。

- zonecfg コマンドを使用して構成できるゾーンのリソースとプロパティー
- デフォルトで構成に含まれているリソース

lx ブランドゾーンのゾーン名とゾーンパス

ゾーンの名前とパスを選択する必要があります。

lx ブランドゾーンでのゾーンの自動ブート

autoboot プロパティーの設定により、大域ゾーンのブート時にこのゾーンが自動的にブートされるかどうかが決まります。

lx ブランドゾーンでのリソースプールの関連付け

第 13 章「リソースプールの作成と管理 (タスク)」の説明に従ってシステムでリソースプールを構成した場合は、ゾーンを構成するときに pool プロパティーを使用して、リソースプールの 1 つにゾーンを関連付けることができます。

リソースプールが構成されていない場合でも、非大域ゾーンの実行中にシステムのプロセッサの一部をそのゾーン専用に割り当てるよう、dedicated-cpu リソースを使用して指定できます。ゾーンの実行中に使用される一時プールが動的に作成されます。

注 – pool プロパティーによって設定される持続的プールを使用するゾーン構成と、dedicated-cpu リソースによって構成される一時プールには、互換性がありません。これら 2 つのプロパティーは、どちらか 1 つしか設定できません。

---

dedicated-cpu リソースを指定する

dedicated-cpu リソースは、非大域ゾーンの実行中にシステムのプロセッサの一部をそのゾーン専用に割り当てることを指定します。ゾーンのブート時に、ゾーンの実行中に使用される一時プールが動的に作成されます。

dedicated-cpu リソースは、ncpus の制限を設定し、必要に応じて importance も設定します。

ncpus　　CPU の数を指定するか、CPU の数の範囲を 2–4 などと指定します。リソースプールの動的な動作を得るために範囲を指定する場合は、次の手順も実行してください。

- importance プロパティーを設定します。
- 173 ページの「プール機能の有効化と無効化」の説明に従って、動的リソースプールサービスを有効にします。

importance　動的な動作を得るために CPU 範囲を使用する場合は、importance プロパティーも設定してください。importance は「省略可能な」プロパティーであり、プールの相対的な重要性を定義します。このプロパティーが必要となるのは、ncpus に範囲を指定した場合で、poold によって管理される動的リソースプールを使用しているときだけです。poold が実行されていない場合、importance は無視されます。poold が実行されている場合、importance が設定されていないと、importance はデフォルト値の 1 になります。詳細は、158 ページの「pool.importance プロパティーの制約」を参照してください。

注 – cpu-shares リソース制御と dedicated-cpu リソースには互換性がありません。

Oracle Solaris10 5/08: capped-cpu リソースを指定する

capped-cpu リソースは、1 つのプロジェクトまたは 1 つのゾーンで消費可能な CPU リソース量に対して絶対的な制限を設けます。capped-cpu リソースには、小数点第 2 位までの正の小数である 1 つの ncpus プロパティーがあります。このプロパティーは、CPU のユニット数に対応しています。このリソースには範囲を指定できません。このリソースには小数を指定できます。ncpus を指定する場合、1 の値は 1 つの CPU の 100% を意味します。100% がシステム上の 1 つの完全な CPU に対応するため、値 1.25 は 125% を意味します。

注 – capped-cpu リソースと dedicated-cpu リソースには互換性がありません。

ゾーンのスケジューリングクラス

公平配分スケジューラ (FSS) を使用すると、使用可能な CPU リソースのゾーン間での割り当てを、ゾーンの重要性に基づいて制御できます。この重要性は、各ゾーンに割り当てる CPU リソースの「配分」で表します。

cpu-shares プロパティーを明示的に設定すると、公平配分スケジューラ (FSS) はそのゾーンのスケジューリングクラスとして使用されます。ただし、この場合に望ましい FSS の使用方法は、dispadmin コマンドを使用して、FSS をシステムのデフォルトのスケジューリングクラスに設定する方法です。このようにすると、すべてのゾーンがシステムの CPU リソースの公平配分を受けることができます。ゾーンに対

して cpu-shares が設定されていない場合、そのゾーンはシステムのデフォルトのスケジューリングクラスを使用します。ゾーンのスケジューリングクラスは、次の処理によって設定されます。

- zonecfg の scheduling-class プロパティーを使ってゾーンのスケジューリングクラスを設定できます。
- リソースプール機能を使ってゾーンのスケジューリングクラスを設定できます。ゾーンがプールに関連付けられている場合、そのプールの pool.scheduler プロパティーに有効なスケジューリングクラスが設定されていれば、ゾーンで実行されるプロセスは、デフォルトでそのスケジューリングクラスで実行されます。148 ページの「リソースプールの紹介」 および 181 ページの「プールをスケジューリングクラスに対応付ける方法」を参照してください。
- cpu-shares リソース制御が設定されている場合で、別の処理を通して FSS がゾーンのスケジューリングクラスとして設定されていないときは、ゾーンのブート時に zoneadmd によってスケジューリングクラスが FSS に設定されます。
- ほかの処理を通してスケジューリングクラスが設定されていない場合、ゾーンはシステムのデフォルトのスケジューリングクラスを継承します。

priocntl (priocntl(1) のマニュアルページを参照) を使用すると、デフォルトのスケジューリングクラスの変更やリブートを行うことなく、実行中のプロセスを別のスケジューリングクラスに移動できます。

capped-memory リソース

capped-memory リソースは、physical、swap、および locked メモリーの制限を設定します。各制限はオプションですが、少なくとも 1 つは設定する必要があります。

- 大域ゾーンから rcapd を使用してゾーンのメモリー上限を設定する場合は、このリソースの値を決定します。capped-memory リソースの physical プロパティーは、ゾーンの max-rss 値として rcapd で使用されます。
- capped-memory リソースの swap プロパティーは、zone.max-swap リソース制御を設定するための望ましい方法です。
- capped-memory リソースの locked プロパティーは、zone.max-locked-memory リソース制御を設定するための望ましい方法です。

注 – 通常はアプリケーションが多量のメモリーをロックすることはありませんが、ゾーンのアプリケーションによってメモリーがロックされることがわかっている場合は、ロックされるメモリーを設定するとよいでしょう。ゾーンの信頼が問題になる場合は、ロックされるメモリーの上限を、システムの物理メモリーの 10 パーセントまたはゾーンの物理メモリー上限の 10 パーセントに設定することもできます。

詳細は、第 10 章「リソース上限デーモンによる物理メモリーの制御 (概要)」、第 11 章「リソース上限デーモンの管理 (タスク)」、および 482 ページの「lx ブランドゾーンの構成方法」を参照してください。

lx ブランドゾーンのゾーンネットワークインタフェース

lx ブランドゾーンでは、共有 IP ネットワーク構成だけがサポートされています。

ネットワーク接続を必要とする各ゾーンには、1 つ以上の専用 IP アドレスが必要です。これらのアドレスは、論理ネットワークインタフェースに関連付けられます。`zonecfg` コマンドで構成されたネットワークインタフェースは、ブート時に自動的に設定され、ゾーンに配置されます。Oracle Solaris 10 10/08 リリース以降では、必要に応じて `defrouter` プロパティーを使用して、ネットワークインタフェースのデフォルトのルーターを設定することもできます。

lx ブランドゾーンでマウントされるファイルシステム

通常、ゾーンでマウントされるファイルシステムには、次のものが含まれます。

- 仮想プラットフォームの初期化時にマウントされる一連のファイルシステム
- ゾーン自体の内部からマウントされる一連のファイルシステム

これには、たとえば次のようなファイルシステムが含まれます。

- `automount` によって引き起こされるマウント
- ゾーン管理者が明示的に実行するマウント

アプリケーション環境内部から実行されるマウントには、いくつかの制限事項があります。これらの制限事項は、ほかのゾーンに悪影響を与えないようにするために、ゾーン管理者がシステムのほかの部分に対するサービスを拒否できないようにします。

一部のファイルシステムについては、ゾーン内部からマウントする場合にセキュリティー制限があります。ほかのファイルシステムは、ゾーン内でマウントされたときに特有の動作を行います。詳細は、380 ページの「ファイルシステムと非大域ゾーン」を参照してください。

lx ブランドゾーンでのゾーン規模のリソース制御

ゾーン規模のリソース制御を設定する場合に望ましい、より簡単な方法は、rctl リソースの代わりにプロパティー名を使用する方法です。これらの制限は、大域ゾーンと非大域ゾーンの両方に対して指定されます。

大域管理者は、rctl リソースを使用して、ゾーン規模の特権付きリソース制御をゾーンに対して設定することもできます。

ゾーン規模のリソース制御は、ゾーン内のすべてのプロセスエンティティーによる総リソース消費を制限します。これらの制限は、大域ゾーンと非大域ゾーンのどちらに対しても、zonecfg コマンドを使用して指定します。手順については、482 ページの「lx ブランドゾーンの構成方法」を参照してください。

現在使用できるリソース制御は次のとおりです。

表 32–1　ゾーン規模のリソース制御

制御名	グローバルプロパティー名	説明	デフォルトの単位	使用される値
zone.cpu-cap		Oracle Solaris 10 5/08 リリースでは、このゾーンに対して CPU リソース量の絶対的な制限を設定します。project.cpu-cap 設定と同様、100 の値は 1 つの CPU の 100% を意味します。125 の値は 125% になります。CPU キャップの使用時は、100% がシステム上の 1 つの CPU の上限となります。	数量 (CPU の数)	
zone.cpu-shares	cpu-shares	このゾーンに対する公平配分スケジューラ (FSS) の CPU 配分	数量 (配分)	
zone.max-locked-memory		ゾーンで使用できるロックされた物理メモリーの合計量。	サイズ (バイト)	capped-memory の locked プロパティー
zone.max-lwps	max-lwps	このゾーンで同時に使用できる LWP の最大数	数量 (LWP 数)	

表 32–1 ゾーン規模のリソース制御 (続き)

制御名	グローバルプロパティー名	説明	デフォルトの単位	使用される値
zone.max-msg-ids	max-msg-ids	このゾーンに許容されるメッセージキュー ID の最大数	数量 (メッセージキュー ID の数)	
zone.max-sem-ids	max-sem-ids	このゾーンに許容されるセマフォー ID の最大数	数量 (セマフォー ID の数)	
zone.max-shm-ids	max-shm-ids	このゾーンに許容される共有メモリー ID の最大数	数量 (共有メモリー ID の数)	
zone.max-shm-memory	max-shm-memory	このゾーンに許容される System V 共有メモリーの合計量	サイズ (バイト)	
zone.max-swap		このゾーンのユーザープロセスのアドレス空間マッピングと tmpfs マウントで消費できるスワップの合計量。	サイズ (バイト)	capped-memory の swap プロパティー

lx ブランドゾーンで構成可能な特権

limitpriv プロパティーは、定義済みのデフォルト特権セット以外の特権マスクを指定する場合に使用します。ゾーンのブート時に、デフォルトの特権セットがブランド構成に含められます。これらの特権は、ゾーン内の特権プロセスがシステムのほかの非大域ゾーン内のプロセスや大域ゾーン内のプロセスに影響を及ぼすことを防ぐため、安全と見なされます。limitpriv プロパティーを使用して、次の操作を実行できます。

- デフォルトの特権セットに追加します。ただし、この種の変更を行うと、あるゾーン内のプロセスがグローバルリソースを制御できるようになって、ほかのゾーン内のプロセスに影響する場合があります。
- デフォルトの特権セットから削除します。ただし、この種の変更を行うと、実行に必要な特権がないため一部のプロセスが正しく動作しなくなる場合があります。

注 – わずかですが、この時点でゾーンのデフォルト特権セットから削除できない特権があります。同じように、特権セットに追加できない特権もあります。

詳細は、470 ページの「lx ブランドゾーンで定義される特権」、397 ページの「非大域ゾーン内の特権」、および privileges(5) のマニュアルページを参照してください。

lx ブランドゾーンの attr リソース

attr リソースタイプを使用して、大域ゾーンにあるオーディオデバイスへのアクセスを可能にすることができます。手順については、483 ページの「lx ブランドゾーンを構成、検証、および確定する方法」の手順 12 を参照してください。

attr リソースタイプを使ってゾーンのコメントを追加することもできます。

デフォルトで構成に含まれているリソース

lx ブランドゾーンで構成されるデバイス

各ゾーンでサポートされるデバイスについては、そのブランドに関するマニュアルページやほかのドキュメントに記載されています。lx ゾーンでは、サポートされていないデバイスや認識されないデバイスの追加は禁止されます。サポートされていないデバイスを追加しようとすると、フレームワークによって検出されます。ゾーン構成が検証不可能であることを示すエラーメッセージが生成されます。

大域ゾーンで稼働中のオーディオデバイスへのアクセスは、attr リソースプロパティーを使用して追加できます。手順については、483 ページの「lx ブランドゾーンを構成、検証、および確定する方法」の手順 12 を参照してください。

lx ブランドゾーンで定義されるファイルシステム

ブランドゾーンに必要なファイルシステムは、ブランド内で定義されます。fs リソースプロパティーを使用して、追加の Oracle Solaris ファイルシステムを lx ブランドゾーンに追加できます。手順については、483 ページの「lx ブランドゾーンを構成、検証、および確定する方法」の手順 9 を参照してください。

注 – ローカルの Linux ファイルシステムは追加できません。Linux サーバーのファイルシステムを NFS マウントすることはできます。

---

lx ブランドゾーンで定義される特権

プロセスは、特権の一部に制限されています。特権を制限することで、ほかのゾーンに影響を及ぼす可能性がある操作がゾーンで実行されないようにします。特権セットにより、特権が付与されたユーザーがゾーン内で実行可能な機能が制限されます。

デフォルトの特権、必須のデフォルト特権、省略可能な特権、および禁止される特権が各ブランドによって定義されます。`limitpriv` プロパティーを使用して、特定の特権の追加や削除を行うこともできます。手順については、483 ページの「lx ブランドゾーンを構成、検証、および確定する方法」の手順 8 を参照してください。表 27–1 に、Solaris の特権すべて、およびゾーン内での各特権のステータスを示します。

特権の詳細は、`ppriv(1)` のマニュアルページおよび『Solaris のシステム管理ガイド (セキュリティサービス)』を参照してください。

zonecfg コマンドを使用した lx ブランドゾーンの作成

`zonecfg` コマンド (`zonecfg(1M)` のマニュアルページを参照) は、ゾーンを構成するために使用されます。このコマンドは、大域ゾーンのリソース管理設定を持続的に指定する場合にも使用できます。

`zonecfg` コマンドは、対話型モード、コマンド行モード、またはコマンドファイルモードで使用できます。このコマンドを使用して、次の操作を実行できます。

- ゾーン構成を作成または削除 (破棄) します
- 特定の構成にリソースを追加します
- 構成に追加したリソースのプロパティーを設定します
- 特定の構成からリソースを削除します
- 構成の照会または確認を行います
- 構成を確定します
- 前の構成に戻します
- ゾーンの名前を変更します
- `zonecfg` のセッションを終了します

`zonecfg` のプロンプトは次のような形式です。

```
zonecfg:zonename>
```

ファイルシステムなど、特定のリソースタイプの構成を行うときは、そのリソースタイプもプロンプトに表示されます。

```
zonecfg:zonename:fs>
```

この章で説明する `zonecfg` のさまざまなコンポーネントの使用手順など、詳細については、482 ページの「lx ブランドゾーンの構成方法」を参照してください。

zonecfg のモード

このユーザーインタフェースでは「有効範囲」という概念が使用されます。有効範囲は、「大域」または「リソース固有」のどちらかです。デフォルトの有効範囲は大域です。

大域有効範囲で add サブコマンドまたは select サブコマンドを使用すると、特定のリソースが選択されます。すると、有効範囲がそのリソースタイプに変わります。

- add サブコマンドの場合、end、cancel のいずれかのサブコマンドを使用すると、リソースの指定が完了します。
- select サブコマンドの場合、end、cancel のいずれかのサブコマンドを使用すると、リソースの変更が完了します。

すると、有効範囲が大域に戻ります。

add、remove、set などのように、有効範囲によって異なる意味を持つサブコマンドもあります。

zonecfg の対話型モード

対話型モードでは、次のサブコマンドがサポートされます。サブコマンドで使用する意味とオプションの詳細については、zonecfg(1M) のマニュアルページでオプション情報を参照してください。破壊的な操作や作業内容の消失を伴うようなサブコマンドの場合、処理を実行する前にユーザーの確認が求められます。-F (強制) オプションを使用すると、この確認手順を省略できます。

help　一般ヘルプまたは特定のリソースに関するヘルプを表示します。

```
zonecfg:lx-zone:net> help
```

create　指定された新しいブランドゾーンに使用するメモリー内構成の構成を開始します。

- -t *template* オプションを使用して、指定したテンプレートと同一の構成を作成します。ゾーン名がテンプレート名から新しいゾーン名に変更されます。Linux ブランドゾーンを作成するには、次のコマンドを使用します。

  ```
  zonecfg:lx-zone> create -t SUNWlx
  ```

- -b オプションを使用して、空の構成を作成し、そのブランドを設定します。

  ```
  zonecfg:lx-zone> create -b
  zonecfg:lx-zone> set brand=lx
  ```

- -F オプションを使用して、既存の構成を上書きします。

export　標準出力または指定された出力ファイルに、コマンドファイルに使用できる形式で構成を出力します。

`add` 大域有効範囲では、指定されたリソースタイプを構成に追加します。

リソース固有の有効範囲では、指定された名前と値を持つプロパティーを追加します。

詳細は、「`lx` ブランドゾーンの構成方法」および `zonecfg`(1M) のマニュアルページを参照してください。

`set` 指定されたプロパティー名を、指定されたプロパティー値に設定します。`zonepath` などの大域的なプロパティーと、リソース固有のプロパティーがあることに注意してください。このコマンドは、大域有効範囲とリソース固有の有効範囲の両方で使用できます。

`select` 大域有効範囲でのみ使用できます。指定されたタイプのリソースのうち、指定されたプロパティー名とプロパティー値の対の条件に一致するものを、変更対象として選択します。有効範囲がそのリソースタイプに変わります。リソースが一意に識別されるように、プロパティーの名前と値の対を十分な数だけ指定する必要があります。

`clear` 省略可能な設定の値をクリアーします。必須の設定はクリアーできません。ただし、必須の設定のいくつかは、新しい値を割り当てることによって変更できます。

`remove` 大域有効範囲では、指定されたリソースタイプを削除します。リソースタイプが一意に識別されるように、プロパティーの名前と値の対を十分な数だけ指定する必要があります。プロパティーの名前と値の対をまったく指定しないと、すべてのインスタンスが削除されます。該当するものが複数ある場合は、`-F` オプションを使用していない限り、確認を求めるメッセージが表示されます。

リソース固有の有効範囲では、指定された名前と値を持つプロパティーを現在のリソースから削除します。

`end` リソース固有の有効範囲でのみ使用できます。リソースの指定を終了します。

次に、`zonecfg` コマンドは、現在のリソースが正しく指定されているかどうかを確認します。

- リソースが正しく指定されている場合は、そのリソースがメモリー内に保持される構成に追加され、有効範囲が大域に戻ります。
- 指定が不完全な場合は、必要な作業を示すエラーメッセージが表示されます。

`cancel` リソース固有の有効範囲でのみ使用できます。リソースの指定を終了し、有効範囲を大域に戻します。リソースの指定が不完全な場合、そのリソースは保持されません。

delete　指定された構成を破棄します。メモリーと安定した記憶領域の両方から構成を削除します。delete に -F (強制) オプションを使用する必要があります。

注意 – この操作は即時に実行されます。確定手順は行われず、削除されたゾーンを元に戻すことはできません。

info　現在の構成または大域のリソースプロパティー zonepath、autoboot、および pool に関する情報を表示します。リソースタイプが指定されている場合は、そのタイプのリソースについてのみ情報を表示します。リソース固有の有効範囲では、このサブコマンドは、追加または変更しようとしているリソースにのみ適用されます。

verify　現在の構成が正しいかどうかを確認します。各リソースに必須プロパティーがすべて指定されていることを確認します。

commit　現在の構成をメモリーから安定した記憶領域に確定します。メモリー内の構成を確定するまでは、revert サブコマンドで変更内容を削除できます。zoneadm で構成を使用するには、その構成を確定する必要があります。zonecfg セッションを完了するときに、この操作の実行が自動的に試みられます。正しい構成のみ確定できるので、確定操作では自動的に確認も行われます。

revert　構成を最後に確定されたときの状態に戻します。

exit　zonecfg のセッションを終了します。exit に -F (強制) オプションを使用できます。

必要な場合は、commit 操作が自動的に試行されます。EOF 文字を使ってセッションを終了することもできることに注意してください。

zonecfg のコマンドファイルモード

コマンドファイルモードでは、ファイルから入力されます。このファイルを生成するには、「zonecfg の対話型モード」で説明されている export サブコマンドを使用します。構成を標準出力に出力するか、-f オプションで指定した出力ファイルに出力することができます。

ブランドゾーン構成データ

ゾーン構成データは 2 種類のエンティティーから成ります。リソースとプロパティーです。各リソースは、タイプのほかにも 1 つ以上のプロパティーを持つことがあります。プロパティーは名前と値から成ります。どのようなプロパティーセットを持つかは、リソースタイプによって異なります。

リソースタイプとプロパティータイプ

リソースタイプとプロパティータイプは次のとおりです。

ゾーン名

ゾーン名は、構成ユーティリティーでゾーンを識別するために使用されます。ゾーン名には次のような規則が適用されます。

- 各ゾーンの名前は一意でなければならない。
- ゾーン名では大文字と小文字が区別される。
- ゾーン名は英数字で始まる必要がある。

  名前には、英数字、下線 (_)、ハイフン (-)、およびピリオド (.) を使用できます。
- 名前の長さは 64 文字以内でなければならない。
- `global` という名前と `SUNW` で始まるすべての名前は、予約されているので使用できない。

`zonepath`

`zonepath` プロパティーは、ゾーンのルートのパスです。各ゾーンには、ルートディレクトリのパスが設定されます。これは、大域ゾーンのルートディレクトリに対する相対パスです。インストール時には、大域ゾーンのディレクトリの可視性が制限されている必要があります。大域ゾーンのディレクトリの所有者は `root`、モードは `700` であることが必要です。

非大域ゾーンのルートパスは 1 つ下のレベルになります。ゾーンのルートディレクトリの所有権とアクセス権は、大域ゾーンのルートディレクトリ (/) と同じになります。ゾーンのディレクトリの所有者は `root` で、モードは `755` であることが必要です。これらのディレクトリは正しいアクセス権を使って自動作成され、ゾーン管理者がこれらのディレクトリを検証する必要はありません。この階層構造により、大域ゾーンのユーザーでも権限を持っていない場合は、非大域ゾーンのファイルシステムと行き来できなくなります。

パス	説明
/home/export/lx-zone	zonecfg zonepath
/home/export/lx-zone/root	ゾーンのルート
/home/export/lx-zone/root/dev	ゾーン用に作成されたデバイス

この問題の詳細については、386 ページの「ファイルシステムの行き来」を参照してください。

注 – zoneadm の move サブコマンドで新しいフルパス zonepath を指定することにより、ゾーンを同じシステム上の別の場所に移動できます。手順については、322 ページの「Solaris 10 11/06: 非大域ゾーンの移動」を参照してください。

---

autoboot

このプロパティーを true に設定すると、大域ゾーンのブート時にこのゾーンが自動的にブートされます。ゾーンサービス svc:/system/zones:default が無効になっている場合、このプロパティーの設定にかかわらず、ゾーンは自動的にはブートしません。svcadm(1M) のマニュアルページに記載されているように、svcadm コマンドを使用してゾーンサービスを有効にできます。

```
global# svcadm enable zones
```

bootargs

このプロパティーは、ゾーンのブート引数を設定するために使用します。reboot、zoneadm boot、または zoneadm reboot コマンドで無効にされた場合を除き、このブート引数が適用されます。495 ページの「ブランドゾーンのブート引数」を参照してください。

pool

このプロパティーは、システム上の特定のリソースプールをゾーンに関連付けるために使用します。1 つのプール内のリソースを複数のゾーンが共有してもかまいません。463 ページの「dedicated-cpu リソースを指定する」も参照してください。

limitpriv

このプロパティーは、デフォルト以外の特権マスクを指定する場合に使用します。397 ページの「非大域ゾーン内の特権」を参照してください。

特権を追加するには、特権名だけを指定するか、特権名の前に priv_ 付けて指定します。特権を除外するには、名前の前にダッシュ (-) または感嘆符 (!) を付けます。複数の特権は、コンマで区切り、引用符 (“) で囲みます。

priv_str_to_set(3C)で説明されているように、特殊な特権セット none、all、および basic は、それぞれの通常の定義に展開されます。ゾーン構成は大域ゾーンで行われるため、特殊な特権セット zone は使用できません。特定の特権を追加または削除してデフォルトの特権セットを変更するのが一般的な使用方法であるため、特殊なセットである default はデフォルトの特権セットにマップされます。limitpriv プロパティーの先頭に default がある場合、デフォルトセットに展開されます。

次のエントリは、システムクロックの設定機能の追加、および raw ICMP (Internet Control Message Protocol) パケット送信機能の削除を実行します

```
global# zonecfg -z userzone
zonecfg:userzone> set limitpriv="default,sys_time,!net_icmpaccess"
```

ゾーンの特権セットに不許可の特権が含まれる場合、必須の特権が欠落している場合、または未知の特権が含まれる場合、ゾーンの検証、準備、またはブートの試行は失敗し、エラーメッセージが表示されます。

scheduling-class　このプロパティーは、ゾーンのスケジューリングクラスを設定します。詳細とヒントについては、464 ページの「ゾーンのスケジューリングクラス」を参照してください。

dedicated-cpu　このリソースは、ゾーンの実行中にシステムのプロセッサの一部をそのゾーン専用に割り当てます。dedicated-cpu リソースは、ncpus の制限を設定し、必要に応じて importance も設定します。詳細は、463 ページの「dedicated-cpu リソースを指定する」を参照してください。

capped-memory　このリソースは、ゾーンのメモリー上限を設定するためのプロパティーをグループ化します。capped-memory リソースは、physical、swap、および locked メモリーの制限を設定します。これらのプロパティーの少なくとも 1 つは指定する必要があります。

fs　各ゾーンでは、インストール済み状態から準備完了状態に移行するときにマウントする各種のファイルシステムを指定できます。ファイルシステムリソースは、ファイルシステムのマウントポイントのパスを指定します。ゾーンでファイルシステムを使用する方法の詳細については、380 ページの「ファイルシステムと非大域ゾーン」を参照してください。

net
: ネットワークインタフェースリソースは、仮想インタフェースの名前です。各ゾーンでは、インストール済み状態から準備完了状態に移行するときに設定するべきネットワークインタフェースを指定できます。

  lx ブランドゾーンでは、共有 IP ネットワーク構成だけがサポートされています。

rctl
: rctl リソースは、ゾーン規模のリソース制御に使用されます。リソース制御は、ゾーンがインストール済み状態から準備完了状態に移行するときに有効になります。

注 – rctl リソースの代わりに zonefig の set *global_property_name* サブコマンドを使用してゾーン規模のリソース制御を構成する方法については、482 ページの「lx ブランドゾーンの構成方法」を参照してください。

---

attr
: この汎用属性は、ユーザーコメントとして使用したり、ほかのサブシステムで使用したりできます。attr の name プロパティーは、英数字で始まる必要があります。name プロパティーには、英数字、ハイフン (-)、およびピリオド (.) を使用できます。zone. で始まる属性名はシステム用に予約されています。

lx ブランドゾーンのリソースタイプのプロパティー

リソースには、構成可能なプロパティーもあります。リソースタイプとそれに関連付けられるプロパティーは次のとおりです。

dedicated-cpu　ncpus、importance

CPU の数を指定し、必要に応じてプールの相対的な重要性も指定します。次の例では、ゾーン my-zone で使用する CPU の範囲を指定します。importance も設定します。

```
zonecfg:my-zone> add dedicated-cpu
zonecfg:my-zone:dedicated-cpu> set ncpus=1-3
zonecfg:my-zone:dedicated-cpu> set importance=2
zonecfg:my-zone:dedicated-cpu> end
```

capped-cpu　ncpus

CPU の数を指定します。次の例では、ゾーン lx-zone で使用できる CPU 数の制限を 3.5 に指定します。

```
zonecfg:lx-zone> add capped-cpu
zonecfg:lx-zone:capped-cpu> set ncpus=3.5
zonecfg:lx-zone:capped-cpu> end
```

capped-memory　physical、swap、locked

このリソースは、ゾーンのメモリー上限を設定するためのプロパティーをグループ化します。次の例では、ゾーン my-zone のメモリー制限を指定します。各制限はオプションですが、少なくとも 1 つは設定する必要があります。

```
zonecfg:my-zone> add capped-memory
zonecfg:my-zone:capped-memory> set physical=50m
zonecfg:my-zone:capped-memory> set swap=100m
zonecfg:my-zone:capped-memory> set locked=30m
zonecfg:my-zone:capped-memory> end
```

fs　dir、special、raw、type、options

次の例では、非大域ゾーンで CD または DVD メディアに読み取り専用のアクセスを行う権限を追加します。ro,nodevices オプション (読み取り専用、デバイスなし) を指定して、非大域ゾーンでファイルシステムをループバックマウントします。

```
zonecfg:lx-zone> add fs
zonecfg:lx-zone:fs> set dir=/cdrom
zonecfg:lx-zone:fs> set special=/cdrom
zonecfg:lx-zone:fs> set type=lofs
zonecfg:lx-zone:fs> add options [ro,nodevices]
zonecfg:lx-zone:fs> end
```

セクション 1M のマニュアルページには、特定のファイルシステムに固有のマウントオプションに関するものがあります。このようなマニュアルページの名前は、mount_*filesystem* という形式です。

net　address、physical、defrouter

次の例では、IP アドレス 192.168.0.1 をゾーンに追加します。物理インタフェースとして bge0 カードを使用し、デフォルトのルーターを設定します。

```
zonecfg:lx-zone> add net
zonecfg:lx-zone:net> set address=192.168.0.1
zonecfg:lx-zone:net> set physical=bge0
zonecfg:lx-zone:net> set defrouter=10.0.0.1
zonecfg:lx-zone:net> end
```

注 – どの物理インタフェースを使用するかを決定するには、システムで ifconfig -a と入力します。出力の各行は、ループバックドライバの行を除き、システムにインストールされているカードの名前で始まります。説明に LOOPBACK が含まれている場合、その行はカードに関するものではありません。

rctl　　name、value

使用可能なゾーン規模のリソース制御については、467 ページの「lx ブランドゾーンでのゾーン規模のリソース制御」を参照してください。

```
zonecfg:lx-zone> add rctl
zonecfg:lx-zone:rctl> set name=zone.cpu-shares
zonecfg:lx-zone:rctl> add value (priv=privileged,limit=10,action=none)
zonecfg:lx-zone:rctl> end

zonecfg:lx-zone> add rctl
zonecfg:lx-zone:rctl> set name=zone.max-lwps
zonecfg:lx-zone:rctl> add value (priv=privileged,limit=100,action=deny)
zonecfg:lx-zone:rctl> end
```

attr　　name、type、value

次の例では、ゾーンに関するコメントを追加します。

```
zonecfg:lx-zone> add attr
zonecfg:lx-zone:attr> set name=comment
zonecfg:lx-zone:attr> set type=string
zonecfg:lx-zone:attr> set value="Production zone"
zonecfg:lx-zone:attr> end
```

export サブコマンドを使用すると、ゾーン構成を標準出力に出力できます。構成は、コマンドファイルに使用できる形式で保存されます。

◆ ◆ ◆ 第 33 章

lx ブランドゾーンの構成 (タスク)

この章では、x64 システムまたは x86 システムに lx ブランドゾーンを構成する方法について説明します。手順は、Oracle Solaris ゾーンの構成手順と基本的に同じです。いくつかのプロパティーは、ブランドゾーンの構成には必要ありません。

lx ブランドゾーンの計画と構成 (タスクマップ)

ゾーンを使用できるようにシステムを設定する前に、まず、情報を収集してゾーンの構成方法を決定する必要があります。次のタスクマップに、lx ゾーンの計画および構成方法の概要を示します。

タスク	説明	参照先
ゾーンの全体的な計画を立てます。	■ ゾーンで実行するアプリケーションを決定します。 ■ ゾーン内のファイルを保持するディスク領域の可用性を評価します。 ■ リソース管理機能も使用している場合は、リソース管理境界に合わせてゾーンを配列する方法を決定します。 ■ リソースプールを使用する場合は、必要に応じてプールを構成します。	461 ページの「システム要件と容量要件」および 150 ページの「ゾーンで使用されるリソースプール」を参照してください。

タスク	説明	参照先
ゾーンの名前とパスを決定します。	命名規則に基づいてゾーンの名前を決定します。ZFS (Zetabyte File System) 上のパスをお勧めします。複製元の zonepath と複製先の zonepath が両方とも ZFS 上にあり、同じプールに含まれる場合、zoneadm cloneコマンドは自動的に ZFS を使用してゾーンを複製します。	474 ページの「リソースタイプとプロパティータイプ」および『Oracle Solaris ZFS 管理ガイド』を参照してください。
ゾーンの IP アドレスを取得または構成します。	構成に基づき、ネットワークアクセスを行う非大域ゾーンごとに 1 つ以上の IP アドレスを取得する必要があります。	262 ページの「ゾーンホスト名の決定およびネットワークアドレスの取得」および『Solaris のシステム管理 (IP サービス)』を参照してください。
ゾーンにファイルシステムをマウントするかどうかを決定します。	アプリケーションの要件を確認します。	詳細は、239 ページの「ゾーンでマウントされるファイルシステム」を参照してください。
ゾーンで使用可能にするべきネットワークインタフェースを決定します。	アプリケーションの要件を確認します。	詳細は、388 ページの「共有 IP ネットワークインタフェース」を参照してください。
非大域ゾーンのデフォルトの特権セットを変更する必要があるかどうかを決定します。	特権セットを確認します。 デフォルトの特権、追加および削除が可能な特権、および現時点では使用できない特権があります。	474 ページの「リソースタイプとプロパティータイプ」および 397 ページの「非大域ゾーン内の特権」を参照してください。
ゾーンを構成します。	zonecfg を使用してゾーンの構成を作成します。	483 ページの「lx ブランドゾーンを構成、検証、および確定する方法」を参照してください。
構成したゾーンを検証および確定します。	指定されたリソースおよびプロパティーが仮想サーバー上で有効かどうかを判定します。	483 ページの「lx ブランドゾーンを構成、検証、および確定する方法」を参照してください。

lx ブランドゾーンの構成方法

次の処理を実行するには、zonecfg コマンド (zonecfg(1M) のマニュアルページを参照) を使用します。

- ゾーン構成を作成します
- 必要な情報がすべて存在することを確認します

- 非大域ゾーン構成を確定します

ヒント – CD または DVD を使用してアプリケーションを lx ブランドゾーンにインストールする予定の場合は、ブランドゾーンを最初に構成するときに add fs を使用して、大域ゾーンの CD または DVD メディアに読み取り専用のアクセスを行う権限を追加します。アクセス権を追加したら、CD または DVD を使用して製品をブランドゾーンにインストールできます。

---

zonecfg ユーティリティーを使用してゾーンを構成する際、revert サブコマンドを使用して、リソースの設定を元に戻すことができます。278 ページの「ゾーン構成を元に戻す方法」を参照してください。

システムに複数のゾーンを構成するスクリプトについては、487 ページの「複数の lx ブランドゾーンを構成するスクリプト」を参照してください。

非大域ゾーンの構成を表示する方法については、489 ページの「ブランドゾーンの構成を表示する方法」を参照してください。

ヒント – ブランドゾーンを構成したあとは、ゾーンの構成のコピーを作成することをお勧めします。このバックアップを使用して、あとでゾーンを復元することができます。スーパーユーザーまたは Primary Administrator として、ゾーン lx-zone の構成をファイルに出力してください。次の例では、lx-zone.config というファイルを使用しています。

```
global# zonecfg -z lx-zone export > lx-zone.config
```

詳細は、434 ページの「非大域ゾーンを個別に復元する方法」を参照してください。

▼ lx ブランドゾーンを構成、検証、および確定する方法

ラベルが有効になっている Trusted Oracle Solaris システムでは、lx ブランドゾーンを使用できません。zoneadm コマンドで構成を検証できません。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 選択したゾーン名を使用して、ゾーン構成を設定します。

この手順例では、lx-zone という名前を使用します。

```
global# zonecfg -z lx-zone
```

このゾーンの初回構成時には、次のシステムメッセージが表示されます。

```
lx-zone: No such zone configured
Use ’create’ to begin configuring a new zone.
```

3 **SUNWlx** テンプレートを使用して、新しい **lx** ゾーン構成を作成します。

```
zonecfg:lx-zone> create -t SUNWlx
```

または、空のゾーンを作成し、そのブランドを明示的に設定することもできます。

```
zonecfg:lx-zone> create -b
zonecfg:lx-zone> set brand=lx
```

4 ゾーンのパス (この手順では **/export/home/lx-zone**) を設定します。

```
zonecfg:lx-zone> set zonepath=/export/home/lx-zone
```

5 **autoboot** 値を設定します。

true に設定すると、大域ゾーンのブート時にこのゾーンが自動的にブートします。ゾーンを自動的にブートするには、ゾーンサービス svc:/system/zones:default も有効になっている必要があります。デフォルト値は false です。

```
zonecfg:lx-zone> set autoboot=true
```

6 ゾーンの持続的なブート引数を設定します。

```
zonecfg:lx-zone> set bootargs="-i=altinit"
```

7 システムでリソースプールが有効な場合、プールをゾーンに関連付けます。

この例では、pool_default という名前のデフォルトプールを使用します。

```
zonecfg:lx-zone> set pool=pool_default
```

リソースプールはオプションのスケジューリングクラス割り当てを保持できるため、プール機能を使用して、システムのデフォルトでないデフォルトスケジューラを非大域ゾーン用に設定できます。詳細は、181 ページの「プールをスケジューリングクラスに対応付ける方法」および 196 ページの「構成の作成」を参照してください。

8 権限のデフォルトセットを修正します。

```
zonecfg:lx-zone> set limitpriv="default,proc_priocntl"
```

proc_priocntl 特権は、リアルタイムクラスのプロセスを実行するために使用されます。

9 **CPU** の配分を **5** に設定します。

```
zonecfg:lx-zone> set cpu-shares=5
```

10 メモリー上限を追加します。

```
zonecfg:lx-zone> add capped-memory
```

a. メモリー上限を設定します。

```
zonecfg:lx-zone:capped-memory> set physical=50m
```

b. スワップメモリーの上限を設定します。

```
zonecfg:lx-zone:capped-memory> set swap=100m
```

c. ロックされたメモリーの上限を設定します。

```
zonecfg:lx-zone:capped-memory> set locked=30m
```

d. 指定を終了します。

```
zonecfg:lx-zone:capped-memory> end
```

11 ファイルシステムを追加します。

```
zonecfg:lx-zone> add fs
```

a. ファイルシステムのマウントポイント (この手順では **/export/linux/local**) を設定します。

```
zonecfg:lx-zone:fs> set dir=/export/linux/local
```

b. 大域ゾーン内の **/opt/local** を、構成中のゾーン内で **/export/linux/local** としてマウントすることを指定します。

```
zonecfg:lx-zone:fs> set special=/opt/local
```

非大域ゾーン内では、/export/linux/local ファイルシステムは読み取りおよび書き込みが可能です。

c. ファイルシステムのタイプ (この手順では **lofs**) を指定します。

```
zonecfg:lx-zone:fs> set type=lofs
```

このタイプは、カーネルとそのファイルシステムとの相互動作の方法を示します。

d. ファイルシステムの指定を終了します。

```
zonecfg:lx-zone:fs> end
```

この手順を複数回実行することで、複数のファイルシステムを追加できます。

12 ネットワーク仮想インタフェースを追加します。

```
zonecfg:lx-zone> add net
```

a. **IP** アドレスを *ip address of zone*/*netmask* の形式で設定します。この手順では *10.6.10.233/24* を使用します。

```
zonecfg:lx-zone:net> set address=10.6.10.233/24
```

b. ネットワークインタフェースの物理デバイスタイプ (この手順では **bge** デバイス) を指定します。

```
zonecfg:lx-zone:net> set physical=bge0
```

c. 指定を終了します。

```
zonecfg:lx-zone:net> end
```

この手順を複数回実行することで、複数のネットワークインタフェースを追加できます。

13 **attr** リソースタイプを使用して、大域ゾーンにあるオーディオデバイスをこのゾーンで使用可能にします。

```
zonecfg:lx-zone> add attr
```

a. 名前を **audio** に設定します。

```
zonecfg:lx-zone:attr> set name=audio
```

b. タイプを **boolean** に設定します。

```
zonecfg:lx-zone:attr> set type=boolean
```

c. 値を **true** に設定します。

```
zonecfg:lx-zone:attr> set value=true
```

d. リソースタイプ **attr** の指定を終了します。

```
zonecfg:lx-zone:attr> end
```

14 ゾーンの構成を検証します。

```
zonecfg:lx-zone> verify
```

15 ゾーンの構成を確定します。

```
zonecfg:lx-zone> commit
```

16 **zonecfg** コマンドを終了します。

```
zonecfg:lx-zone> exit
```

プロンプトで `commit` コマンドを明示的に入力しなくても、`exit` を入力するか EOF が発生すると、`commit` の実行が自動的に試みられます。

参考 コマンド行での複数のサブコマンドの使用

ヒント – zonecfg コマンドは、複数のサブコマンドもサポートします。次に示すように、同じシェル呼び出しで引用符で囲み、セミコロンで区切ります。

```
global# zonecfg -z lx-zone "create -t SUNWlx; set zonepath=/export/home/lx-zone"
```

次に進む手順

確定済みのゾーン構成をインストールする方法については、498 ページの「lx ブランドゾーンのインストールとブート」を参照してください。

複数の lx ブランドゾーンを構成するスクリプト

このスクリプトを使用して、システムで複数のゾーンを構成およびブートできます。スクリプトには、次のパラメータを指定します。

- 作成するゾーンの数
- 接頭辞 *zonename*
- 基本ディレクトリとして使用するディレクトリ

このスクリプトを実行するには、大域ゾーン内の大域管理者になる必要があります。大域管理者は、大域ゾーン内でスーパーユーザー権限を保持するか、Primary Administrator 役割になります。

```
#!/bin/ksh
#
# Copyright 2006 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
#ident    "%Z%%M%   %I%    %E% SMI"
if [[ -z "$1" || -z "$2" || -z "$3" || -z "$4" ]]; then
    echo "usage: $0 <#-of-zones> <zonename-prefix> <basedir> <template zone>"
    exit 2
fi
if [[ ! -d $3 ]]; then
    echo "$3 is not a directory"
    exit 1
fi
state=‘zoneadm -z $4 list -p 2>/dev/null | cut -f 3 -d ":"‘
if [[ -z "$state" || $state != "installed" ]]; then
    echo "$4 must be an installed, halted zone"
    exit 1
fi

template_zone=$4

nprocs=‘psrinfo | wc -l‘
```

```
nzones=$1
prefix=$2
dir=$3

ip_addrs_per_if=‘ndd /dev/ip ip_addrs_per_if‘
if [ $ip_addrs_per_if -lt $nzones ]; then
        echo "ndd parameter ip_addrs_per_if is too low ($ip_addrs_per_if)"
        echo "set it higher with ’ndd -set /dev/ip ip_addrs_per_if <num>"
        exit 1
fi

i=1
while [ $i -le $nzones ]; do
    zoneadm -z $prefix$i clone $template_zone > /dev/null 2>&1
    if [ $? != 0 ]; then
        echo configuring $prefix$i
        F=$dir/$prefix$i.config
        rm -f $F
        echo "create -t SUNWlx" > $F
        echo "set zonepath=$dir/$prefix$i" >> $F
        zonecfg -z $prefix$i -f $dir/$prefix$i.config 2>&1 | \
            sed ’s/^/    /g’
    else
        echo "skipping $prefix$i, already configured"
    fi
    i=‘expr $i + 1‘
done

i=1
while [ $i -le $nzones ]; do
    j=1
    while [ $j -le $nprocs ]; do
        if [ $i -le $nzones ]; then
            if [ ‘zoneadm -z $prefix$i list -p | \
                cut -d’:’ -f 3‘ != "configured" ]; then
                echo "skipping $prefix$i, already installed"
            else
                echo installing $prefix$i
                mkdir -pm 0700 $dir/$prefix$i
                chmod 700 $dir/$prefix$i
                zoneadm -z $prefix$i install -s -d /path/to/ISOs > /dev/null 2>&1 &
                sleep 1    # spread things out just a tad
            fi
        fi
        i=‘expr $i + 1‘
        j=‘expr $j + 1‘
    done
    wait
done

i=1
para=‘expr $nprocs \* 2‘
while [ $i -le $nzones ]; do
    date
    j=1
    while [ $j -le $para ]; do
        if [ $i -le $nzones ]; then
            echo booting $prefix$i
            zoneadm -z $prefix$i boot &
```

```
        fi
        j=‘expr $j + 1‘
        i=‘expr $i + 1‘
    done
    wait
done
```

▼ ブランドゾーンの構成を表示する方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。
2. ゾーンの構成を表示します。

```
global# zonecfg -z zonename info
```

ゾーンの構成を変更する、元に戻す、または削除する

ゾーンの構成を変更する、元に戻す、または削除する手順については、次のセクションを参照してください。

- 274 ページの「ゾーン構成内のリソースタイプを変更する方法」
- 275 ページの「Solaris 10 8/07: ゾーン構成内のプロパティータイプをクリアーする方法」
- 276 ページの「Solaris 10 8/07: ゾーンの名前を変更する方法」
- 278 ページの「ゾーン構成を元に戻す方法」
- 279 ページの「ゾーン構成を削除する方法」

◆ ◆ ◆ 第 34 章

lx ブランドゾーンのインストール、ブート、停止、複製、およびアンインストールについて (概要)

この章では、次の内容について説明します。

- システムへの lx ゾーンのインストール
- ゾーンの停止、リブート、およびアンインストール
- システムでのゾーンの複製

ブランドゾーンのインストールと管理の概要

zoneadm コマンド (zoneadm(1M) のマニュアルページを参照) は、非大域ゾーンをインストールおよび管理するための主要なツールです。zoneadm コマンドを使用する操作は、大域ゾーンから実行する必要があります。zoneadm コマンドを使用すると、次のタスクを実行できます。

- ゾーンを検証します
- ゾーンをインストールします
- ゾーンをブートします
- 稼働中のゾーンに関する情報を表示します
- ゾーンを停止します
- ゾーンをリブートします
- ゾーンをアンインストールします
- 同じシステム上で、ゾーンを別の場所へ再配置します
- 同一システムの既存ゾーンの構成に基づいて、新しいゾーンをプロビジョニングします
- ゾーンを移行します。zonecfg コマンドとともに使用します

ゾーンのインストールおよび検証手順については、第 35 章「lx ブランドゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)」および zoneadm(1M) のマニュアルページを参照してください。zoneadm list コマンドでサポートされるオプションについては、zoneadm(1M) のマニュアルページも参照してください。ゾーンの構成手順については、第 33 章「lx ブランドゾーンの構成 (タス

ク)」および zonecfg(1M) のマニュアルページを参照してください。ゾーンの状態については、222 ページの「非大域ゾーンの状態モデル」に記載されています。

ゾーンの Oracle Solaris 監査レコードの生成を計画している場合は、非大域ゾーンをインストールする前に402 ページの「ゾーン内での Oracle Solaris 監査の使用」を参照してください。

注 – ゾーンのインストール後は、ソフトウェアの構成と管理はすべて、ゾーン管理者がゾーン内から Linux ツールを使用して行う必要があります。

---

lx ブランドゾーンのインストール方法

lx ブランドゾーンのインストールには、tarball、CD-ROM または DVD ディスク、または ISO イメージを使用できます。ディスクまたは ISO イメージからインストールする場合は、Sun パッケージクラスタのカテゴリを指定できます。カテゴリは累積されたものです。クラスタを指定しないと、デフォルトで desktop になります。

表 34–1　パッケージクラスタのカテゴリ

Sun カテゴリ	内容
core	ゾーンの構築に必要なパッケージの最小セット。
server	core のパッケージに httpd、mailman、imapd、spam-assassin などのサーバー向けパッケージを加えたもの。
desktop	server のパッケージに evolution、gimp、mozilla、openoffice などのユーザー向けパッケージを加えたもの。
developer	desktop のパッケージに bison、emacs、gcc、vim-X11 などの開発者パッケージと多くのライブラリ開発パッケージを加えたもの。
all	インストールメディアに含まれているもので、ゾーンの動作に干渉しないことがわかっているものすべて。一部のパッケージは、Linux ゾーンで機能しない場合があります。

構成済みの lx ブランドゾーンをインストールする方法については、499 ページの「lx ブランドゾーンをインストールする方法」を参照してください。

lx ブランドゾーンの構築

このセクションの内容は、既存のゾーンの複製にではなく、初期のゾーン構築だけに適用されます。

非大域ゾーンを構成したあとで、システムの構成にゾーンを安全にインストールできることを確認してください。その後、ゾーンをインストールできます。ゾーンのルートファイルシステムに必要とされるファイルは、システムによりゾーンのルートパス内にインストールされます。Linux ゾーンは、CD、ISO イメージ、または tarball から生成されます。詳細は、499 ページの「lx ブランドゾーンをインストールする方法」を参照してください。

ゾーンの状態がインストール済みから準備完了に移行する際、構成ファイルで指定されたリソースセットが追加されます。システムにより、一意のゾーン ID が割り当てられます。ファイルシステムがマウントされ、ネットワークインタフェースが設定され、デバイスが構成されます。準備完了状態に移行すると、仮想プラットフォームでユーザープロセスを開始する準備が整います。

準備完了状態のゾーンには、稼働中のユーザープロセスは存在しません。準備完了状態のゾーンと稼働中のゾーンの主な違いは、稼働中のゾーンでは 1 つ以上のプロセスが稼働している点です。詳細は、init(1M) のマニュアルページを参照してください。

準備完了状態では、仮想プラットフォームを管理するため、zsched および zoneadmd プロセスが開始されます。

zoneadmd ゾーン管理デーモン

ゾーン管理デーモン zoneadmd は、ゾーンの仮想プラットフォーム管理用の主要なプロセスです。詳細は、284 ページの「zoneadmd デーモン」を参照してください。

zsched ゾーンスケジューリングプロセス

アプリケーション環境を管理するプロセス zsched については、285 ページの「zsched ゾーンスケジューラ」を参照してください。

ブランドゾーンアプリケーション環境

ゾーンアプリケーション環境の作成には、zoneadm コマンドが使用されます。

追加の構成はすべて、ゾーン管理者がゾーン内から Linux ツールを使用して行います。

パスワード

ゾーンを Sun tarball からインストールすると、root (スーパーユーザー) パスワードは root になります。ゾーンを ISO イメージまたは CD からインストールすると、root (スーパーユーザー) パスワードは未設定 (空白) になります。

lx ブランドゾーンの停止、リブート、アンインストール、および複製について

このセクションでは、ゾーンの停止、リブート、アンインストール、およびクローニングの概要について説明します。

ブランドゾーンを停止する

ゾーンのアプリケーション環境および仮想プラットフォームの両方を削除する場合に、zoneadm halt コマンドを使用します。これにより、ゾーンはインストール済みの状態に戻されます。すべてのプロセスが終了し、デバイスが構成解除され、ネットワークインタフェースが破棄され、ファイルシステムのマウントが解除され、カーネルデータ構造が破棄されます。

halt コマンドにより、ゾーン内部の停止処理スクリプトが実行されることはありません。ゾーンの停止処理を行う方法については、319 ページの「zlogin を使用してゾーンを停止処理する方法」を参照してください。

停止操作に失敗する場合は、443 ページの「ゾーンが停止しない」を参照してください。

ブランドゾーンをリブートする

zoneadm reboot コマンドを使用してブランドゾーンをリブートします。ゾーンは停止し、その後再ブートします。ゾーンのリブート時に、ゾーン ID が変更されます。

ブランドゾーンのブート引数

ゾーンでは、次のブート引数を zoneadm boot および reboot コマンドに使用できます。

- -i *altinit*
- -s

次の定義が適用されます。

-i *altinit*　最初のプロセスとなる代替実行可能ファイルを選択します。*altinit* は実行可能ファイルへの有効なパスでなければなりません。デフォルトの最初のプロセスについては、init(1M) のマニュアルページを参照してください。

-s　ゾーンを init のレベル s にブートします。

使用例については、504 ページの「lx ブランドゾーンをブートする方法」および 505 ページの「lx ブランドゾーンをシングルユーザーモードでブートする方法」を参照してください。

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

ブランドゾーンの autoboot

ゾーンの構成内で autoboot リソースプロパティーを true に設定すると、大域ゾーンのブート時にそのゾーンが自動的にブートします。デフォルトの設定は false です。

ゾーンを自動的にブートするには、ゾーンサービス svc:/system/zones:default も有効になっている必要があります。

ブランドゾーンをアンインストールする

zoneadm uninstall コマンドは、ゾーンのルートファイルシステム内のすべてのファイルを削除します。-F (force) オプションを合わせて指定しない限り、処理を続行する前に、コマンドプロンプトにより実行の確認が求められます。実行した操作を元に戻すことはできないため、uninstall コマンドは慎重に使用してください。

lx ブランドゾーンの複製について

クローンを使用すると、システムの既存の構成済みおよびインストール済みゾーンをコピーして、新しいゾーンを同一のシステム上に迅速にプロビジョニングできます。複製処理の詳細については、509 ページの「同一システム上での lx ブランドゾーンの複製」を参照してください。

lx ブランドゾーンのブートとリブート

ゾーンのブートおよびリブートの手順については、504 ページの「lx ブランドゾーンをブートする方法」および 507 ページの「lx ブランドゾーンをリブートする方法」を参照してください。

◆ ◆ ◆ 第 35 章

lx ブランドゾーンのインストール、ブート、停止、アンインストール、および複製 (タスク)

この章では、lx ブランドゾーンのインストールおよびブート方法について説明します。次に示すその他のタスクについても説明します。

- 複製を使用した、同一システムへのゾーンのインストール
- ゾーンの停止、リブート、およびアンインストール
- システムからのゾーンの削除

lx ブランドゾーンのインストール (タスクマップ)

タスク	説明	参照先
Linux アーカイブを入手します。	lx ブランドゾーンをインストールするには、まず Linux アーカイブを入手する必要があります。	498 ページの「Linux アーカイブを入手する方法」
構成済みの lx ブランドゾーンをインストールします。	構成済みの状態にあるゾーンをインストールします。	499 ページの「lx ブランドゾーンをインストールする方法」
(オプション) 使用可能なパッケージの一部をインストールします。	CD または ISO イメージからインストールする場合は、インストールメディア上にあるパッケージの一部をインストールすることができます。	501 ページの「パッケージの一部をインストールする方法」
(オプション) ゾーンでネットワークを使用可能にします。	ネットワークはデフォルトでは使用不可になっているので、この機能が必要な場合は、ネットワークを使用可能にする必要があります。	501 ページの「lx ブランドゾーンでネットワークを使用可能にする方法」
ゾーンの汎用一意識別子 (UUID) を取得します。	ゾーンのインストール時に割り当てられるこの個別の識別子は、ゾーンを識別するための代替手段になります。	502 ページの「インストール済みのブランドゾーンの UUID を取得する方法」

タスク	説明	参照先
(任意) インストール済みのゾーンを準備完了状態に移行します。	ゾーンをすぐにブートして使用する場合、この手順は省略できます。	503 ページの「(オプション) インストール済みの lx ブランドゾーンを準備完了状態に移行する」
lx ブランドゾーンをブートします。	ゾーンをブートすると、ゾーンが稼働状態になります。ゾーンは、準備完了状態またはインストール済み状態からブートできます。	504 ページの「lx ブランドゾーンをブートする方法」
ゾーンをシングルユーザーモードでブートします。	マイルストーン svc:/milestone/single-user:default に対してのみブートします。このマイルストーンは、init のレベル s と同等です。init(1M) および svc.startd(1M) のマニュアルページを参照してください。	298 ページの「ゾーンをシングルユーザーモードでブートする方法」

lx ブランドゾーンのインストールとブート

zoneadm(1M) のマニュアルページの記述に従って zoneadm コマンドを使用し、非大域ゾーンのインストールタスクを実行します。

▼ Linux アーカイブを入手する方法

lx ブランドゾーンをインストールするには、まず Linux アーカイブを入手する必要があります。アーカイブは次の形式で配布されます。

- 圧縮された tar アーカイブ (*tarball*)
- CD-ROM または DVD のディスクセット
- 一連の ISO イメージ

- 次のいずれかの方法で **Linux** ディストリビューションを入手します。
  - CD-ROM または DVD のディスクセットを入手するには、CentOS のサイト http://www.centos.org または Red Hat のサイト http://www.redhat.com にアクセスします。
  - ISO イメージを入手するには、CentOS のサイト http://www.centos.org または Red Hat のサイト http://www.redhat.com にアクセスします。

▼ lx ブランドゾーンをインストールする方法

構成済みの lx ブランドゾーンをインストールする場合に、ここで説明する手順を使用します。ゾーンのインストール後は、ソフトウェアの構成と管理はすべて、ゾーン管理者がゾーン内から Linux ツールを使用して行う必要があります。

異なるディストリビューションパスを使用してゾーンをインストールするコマンド行の例については、例 35–1、例 35–2、および例 35–3 を参照してください。ディスクまたは ISO イメージからインストールする場合は、Sun パッケージクラスタのカテゴリを指定する必要があります。パッケージクラスタのカテゴリについては、492 ページの「lx ブランドゾーンのインストール方法」を参照してください。

ゾーンをインストールする前に検証できます。この手順を省略した場合、ゾーンのインストール時に検証が自動的に実行されます。手順については、292 ページの「(オプション) インストール前に構成済みのゾーンを検証する方法」を参照してください。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

注 – 手順 3 の `zoneadm install` コマンドでは、`zonepath` が ZFS 上にある場合、ゾーンのインストール時に `zonepath` の ZFS ファイルシステム (データセット)が自動的に作成されます。`-x nodataset` パラメータを指定することで、この処理をブロックできます。

---

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **(オプション) DVD または CD からインストールする場合は、システムで `volfs` を使用可能にし、稼働していることを確認します。**

   ```
   global# svcadm enable svc:/system/filesystem/volfs:default

   global# svcs | grep volfs
   ```

   次のような情報が表示されます。

   ```
   online  17:30 svc:/system/filesystem/volfs:default
   ```

3. **`zoneadm` コマンドに `install` オプションとアーカイブのパスを指定して、構成済みゾーン `lx-zone` をインストールします。**

   - ゾーンをインストールします。**`zonepath` が ZFS 上にある場合は、ZFS ファイルシステムを自動的に作成します。**

     ```
     global# zoneadm -z lx-zone install -d archive_path
     ```

システムには次のように表示されます。

```
A ZFS file system has been created for this zone.
```

- **zonepath** が **ZFS** 上にあるゾーンをインストールします。ただし、**ZFS** ファイルシステムの自動作成は行いません。

  ```
  global# zoneadm -z lx-zone install -x nodataset -d archive_path
  ```

ゾーンのルートファイルシステムに必要なファイルおよびディレクトリがパッケージファイルとともにゾーンのルートパスにインストールされる際、さまざまなメッセージが表示されます。

注 – archive_path を指定しない場合、デフォルトは CD です。

---

4 (オプション) エラーメッセージが表示され、ゾーンのインストールに失敗した場合は、次のように入力してゾーンの状態を取得します。

```
global# zoneadm -z lx-zone list -iv
```

- 状態が構成済みであると表示された場合は、メッセージに示された修正を行い、zoneadm install コマンドを再度実行します。
- 状態が不完全であると表示された場合は、最初に次のコマンドを実行します。

  ```
  global# zoneadm -z lx-zone uninstall
  ```

  次にメッセージに示された修正を行い、zoneadm install コマンドを再度実行します。

5 インストールが完了したら、**list** サブコマンドに **-i** オプションおよび **-v** オプションを指定してインストール済みのゾーンを一覧表示し、ステータスを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                          BRAND      IP
 0  global   running      /                             native     shared
 -  lx-zone  installed    /export/home/lx-zone          lx         shared
```

例 35–1 CentOS の圧縮された tar アーカイブを使用するインストールコマンド

```
global# zoneadm -z lx-zone install -d /export/centos_fs_image.tar.bz2
```

例 35–2 CentOS の CD を使用するインストールコマンド

CD または DVD からインストールする場合は、システムで volfs が使用可能になっている必要があります。ソフトウェアクラスタパッケージを指定する必要があります。たとえば、development を使用して完全な環境をインストールするか、特定

のクラスタの名前を指定します。クラスタパッケージを指定しないと、デフォルトで desktop がインストールされます。CD デバイスは /cdrom/cdrom0 です。

```
global# zoneadm -z lx-zone install -d /cdrom/cdrom0 development
```

例 35–3 CentOS の ISO イメージを使用するインストールコマンド

ソフトウェアクラスタパッケージを指定する必要があります。development を使用して完全な環境をインストールするか、特定のクラスタを指定します。クラスタパッケージを指定しないと、デフォルトで desktop がインストールされます。CentOS の ISO イメージは /export/centos_3.7 にあります。

```
global# zoneadm -z lx-zone install -d /export/centos_3.7 development
```

参照 データセットの詳細については、『Oracle Solaris ZFS 管理ガイド』を参照してください。

注意事項 ゾーンのインストールが中断または失敗した場合は、ゾーンの状態は不完全なままになります。uninstall -F を使用して、ゾーンを構成済みの状態にリセットします。

▼ パッケージの一部をインストールする方法

CD または ISO イメージからインストールする場合は、インストールメディア上にあるパッケージの一部をインストールすることができます。使用可能なサブセットは、core、server、desktop、developer、および all です。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **server** パッケージだけをインストールします。

   ```
   global# zoneadm -z lx-zone install -d archive_path server
   ```

▼ lx ブランドゾーンでネットワークを使用可能にする方法

lx ブランドゾーンをインストールしたとき、ネットワークは使用不可になっています。ネットワークを使用可能にするには、ここで説明する手順などを使用します。

この手順を実行するには、ゾーン管理者になる必要があります。

1. ゾーンの **/etc/sysconfig/network** ファイルを編集します。

```
NETWORKING=yes
HOSTNAME=your.hostname
```

2. **NIS** ドメインを設定するには、次のような行を追加します。

```
NISDOMAIN=domain.Sun.COM
```

参考 ネットワークサービスとネームサービスの構成

ネットワークサービスやネームサービスの構成方法については、使用している Linux ディストリビューションのドキュメントを参照してください。

▼ インストール済みのブランドゾーンの UUID を取得する方法

ゾーンのインストール時に、汎用一意識別子 (UUID) がゾーンに割り当てられます。UUID は、`zoneadm` に `list` サブコマンドと `-p` オプションを使うことで取得できます。UUID は、5 番目に表示されるフィールドです。

- インストールされたゾーンの **UUID** を表示します。

```
global# zoneadm list -p
```

次のような情報が表示されます。

```
0:global:running:/::native
    1:centos38:running:/zones/centos38:27fabdc8-d8ce-e8aa-9921-ad1ea23ab063:lx
```

例 **35–4** コマンド内で UUID を使用する方法

```
global# zoneadm -z lx-zone -u 61901255-35cf-40d6-d501-f37dc84eb504 list -v
```

-u *uuid-match* と - z *zonename* の両方が存在する場合、最初に UUID に基づいてマッチングが行われます。指定した UUID のゾーンが見つかった場合はそのゾーンが使用され、`-z` パラメータは無視されます。指定した UUID のゾーンが見つからなかった場合、システムはゾーン名で検索を実行します。

参考 UUID について

ゾーンをアンインストールすることも、同名のゾーンを内容を変えて再インストールすることもできます。ゾーンの内容を変更せずにゾーンの名前を変更することも可能です。こうした理由から、UUID はゾーン名よりも信頼性の高いハンドルです。

参照 詳細は、`zoneadm(1M)` および `libuuid(3LIB)` を参照してください。

▼ インストールした lx ブランドゾーンに不完全のマークを付ける方法

システムに加えられた管理上の変更のためにゾーンが使用不可になるか、矛盾が生じた場合、インストールしたゾーンの状態を不完全に変更できます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは Primary Administrator 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 ゾーン **testzone** に不完全のマークを付けます。

```
global# zoneadm -z testzone mark incomplete
```

3 **list** サブコマンドに **-i** オプションと **-v** オプションを使って、ステータスを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME       STATUS        PATH                    BRAND      IP
0   global     running       /                        native     shared
-   testzone   incomplete    /export/home/testzone   lx         shared
```

参考 ゾーンへの不完全のマーク付け

注 – ゾーンへの不完全のマーク付けは、取り消すことができません。不完全のマークが付けられたゾーンに実行可能なのは、ゾーンをアンインストールして、構成済みの状態に戻す操作だけです。508 ページの「ブランドゾーンをアンインストールする方法」を参照してください。

(オプション) インストール済みの lx ブランドゾーンを準備完了状態に移行する

準備完了状態に移行すると、仮想プラットフォームでユーザープロセスを開始する準備が整います。準備完了状態のゾーンには、内部で実行中のユーザープロセスは存在しません。

ゾーンをすぐにブートして使用する場合、この手順は省略できます。ゾーンのブート時に、準備完了状態への移行が自動的に行われます。

296 ページの「(オプション) インストール済みのゾーンを準備完了状態に移行する方法」を参照してください。

▼ lx ブランドゾーンをブートする方法

ゾーンをブートすると、ゾーンが稼働状態になります。ゾーンは、準備完了状態またはインストール済み状態からブートできます。ブートしたインストール済み状態のゾーンは、準備完了状態から稼働状態に透過的に移行します。稼働状態のゾーンに対してはゾーンへのログインが可能です。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

ヒント – ラベルが有効になっている Trusted Oracle Solaris システムでは、ブランドゾーンをブートできません。

---

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. **zoneadm** コマンドを **-z** オプション、ゾーン名 **(lx-zone)**、および **boot** サブコマンドとともに使用することで、ゾーンをブートします。

   ```
   global# zoneadm -z lx-zone boot
   ```

3. ブートが完了したら、**list** サブコマンドに **-v** オプションを指定してステータスを確認します。

   ```
   global# zoneadm list -v
   ```

   次のような情報が表示されます。

   ```
   ID  NAME     STATUS       PATH                  BRAND      IP
    0  global   running      /                     native     shared
    1  lx-zone  running      /export/home/lx-zone  lx         shared
   ```

例 35–5 ゾーンのブート引数を指定する

`-i altinit` オプションを使用してゾーンをブートします。

```
global# zoneadm -z lx-zone boot -- -i /path/to/process
```

注意事項 ゾーン構成で指定された IP アドレス用のネットマスクをシステムが検出できなかったことを示すメッセージが表示された場合は、444 ページの「ゾーンブート時に netmasks の警告が表示される」を参照してください。このメッセージは単なる警告であり、コマンドは成功しています。

▼ lx ブランドゾーンをシングルユーザーモードでブートする方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 ゾーンをシングルユーザーモードでブートします。

```
global# zoneadm -z lx-zone boot -- -s
```

次に進む手順

ゾーンにログインする方法については、515 ページの「lx ブランドゾーンへのログイン」を参照してください。

lx ブランドゾーンの停止、リブート、アンインストール、複製、および削除 (タスクマップ)

タスク	説明	参照先
ゾーンを停止します。	停止手順を実行して、ゾーンのアプリケーション環境と仮想プラットフォームの両方を削除します。この手順により、ゾーンが準備完了状態からインストール済み状態に戻されます。ゾーンの完全な停止処理を行う方法については、518 ページの「zlogin を使用して lx ブランドゾーンを停止処理する方法」を参照してください。	506 ページの「lx ブランドゾーンを停止する方法」
ゾーンをリブートします。	リブートの手順を実行すると、ゾーンが停止してから再びブートします。	507 ページの「lx ブランドゾーンをリブートする方法」

タスク	説明	参照先
ゾーンをアンインストールします。	この手順は、ゾーンのルートファイルシステム内のすべてのファイルを削除します。「この手順は、十分注意して実行する必要があります。」実行した操作を元に戻すことはできません。	508 ページの「ブランドゾーンをアンインストールする方法」
同一システムの既存ゾーンの構成に基づいて、新しい非大域ゾーンをプロビジョニングします。	ゾーンのクローニングは、ゾーンのインストールより高速な代替手段です。ただし、インストールの前にはやはり新規ゾーンを構成してください。	509 ページの「同一システム上での lx ブランドゾーンの複製」
システムから非大域ゾーンを削除します。	この手順を実行すると、システムからゾーンが完全に削除されます。	511 ページの「システムから lx ブランドゾーンを削除する」

lx ブランドゾーンの停止、リブート、およびアンインストール

▼ lx ブランドゾーンを停止する方法

lx ブランドゾーンのアプリケーション環境と仮想プラットフォームの両方を削除する場合に、この停止手順を使用します。ゾーンの完全な停止処理を行う方法については、「zlogin を使用して lx ブランドゾーンを停止処理する方法」を参照してください。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 システムで稼働中のゾーンの一覧を表示します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                    BRAND      IP
 0  global   running      /                       native     shared
 1  lx-zone  running      /export/home/lx-zone    lx         shared
```

3 **zoneadm** コマンドを **-z** オプション、ゾーン名 (**lx-zone** など)、および **halt** サブコマンドとともに使用することで、指定されたゾーンを停止します。

```
global# zoneadm -z lx-zone halt
```

4 システム内のゾーンの一覧を再度表示して、**lx-zone** が停止していることを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME       STATUS        PATH                  BRAND      IP
0  global     running       /                     native     shared
-  lx-zone    installed     /export/home/lx-zone  lx         shared
```

5 ゾーンを再び起動する場合は、次のコマンドをブートします。

```
global# zoneadm -z lx-zone boot
```

注意事項 ゾーンが正しく停止しない場合は、443 ページの「ゾーンが停止しない」でトラブルシューティングのヒントを参照してください。

▼ lx ブランドゾーンをリブートする方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 システムで稼働中のゾーンの一覧を表示します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                  BRAND      IP
 0  global   running      /                     native     shared
 1  lx-zone  running      /export/home/lx-zone  lx         shared
```

3 **zoneadm** コマンドを **-z reboot** オプションとともに使用することで、ゾーン **lx-zone** をリブートします。

```
global# zoneadm -z lx-zone reboot
```

4 システム内のゾーンの一覧を再度表示して、**lx-zone** がリブートしたことを確認します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                   BRAND      IP
 0  global   running      /                      native     shared
 2  lx-zone  running      /export/home/lx-zone   lx         shared
```

ヒント – lx-zone のゾーン ID が変更されていることに注目してください。通常、リブートするとゾーン ID は変更されます。

▼ ブランドゾーンをアンインストールする方法

注意 – この手順は、ゾーンのルートファイルシステム内のすべてのファイルを削除します。実行した操作を元に戻すことはできません。

ゾーンは稼働状態であっていけません。uninstall 操作は、稼働中のゾーンに対しては無効です。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. システム内のゾーンの一覧を表示します。

   ```
   global# zoneadm list -v
   ```

   次のような情報が表示されます。

   ```
   ID  NAME       STATUS        PATH                  BRAND      IP
   0  global     running       /                     native     shared
   -  lx-zone    installed     /export/home/lx-zone  lx         shared
   ```

3. **zoneadm** コマンドを **-z uninstall** オプションとともに使用することで、ゾーン **lx-zone** を削除します。
   -F オプションを使用すると、処理を強制的に実行できます。このオプションが指定されていない場合、システムにより確認を求めるメッセージが表示されます。

   ```
   global# zoneadm -z lx-zone uninstall -F
   ```

   zonepath として独自の ZFS ファイルシステムを保持しているゾーンをアンインストールすると、その ZFS ファイルシステムは破棄されます。

4. システム内のゾーンの一覧を再度表示して、**lx-zone** が一覧に含まれていないことを確認します。

   ```
   global# zoneadm list -v
   ```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH              BRAND      IP
0  global   running      /                 native     shared
```

注意事項 ゾーンのアンインストールが中断した場合、ゾーンの状態は不完全なままになります。zoneadm uninstall コマンドを使用して、ゾーンを構成済みの状態にリセットしてください。

実行した操作を元に戻すことはできないため、uninstall コマンドは慎重に使用してください。

同一システム上での lx ブランドゾーンの複製

複製操作は、複製元の zonepath から複製先の zonepath にデータをコピーすることにより、システム上に新しいゾーンをプロビジョニングするのに使用されます。

▼ lx ブランドゾーンを複製する方法

新規ゾーンをインストールする前に、そのゾーンを構成する必要があります。zoneadm create サブコマンドに渡されるパラメータは、クローンするゾーンの名前です。このソースゾーンを停止する必要があります。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 複製元のゾーン (この手順では **lx-zone**) を停止します。

```
global# zoneadm -z lx-zone halt
```

3 複製元ゾーン **lx-zone** の構成をファイル (たとえば、**master**) にエクスポートすることにより、新規ゾーンの構成を開始します。

```
global# zonecfg -z lx-zone export -f /export/zones/master
```

注 – 既存の構成を変更する代わりに、266 ページの「ゾーンの構成方法」で説明されている手順を使って、新規ゾーン構成を作成することもできます。この方法を使用する場合は、ゾーンを作成したあとで手順 6 に進みます。

---

4 **master** ファイルを編集します。少なくとも、新規ゾーンに別の **zonepath** と **IP** アドレスを設定する必要があります。

5 *master* ファイル内のコマンドを使って、新規ゾーン **zone1** を作成します。

```
global# zonecfg -z zone1 -f /export/zones/master
```

6 **lx-zone** を複製して、新規ゾーン **zone1** をインストールします。

```
global# zoneadm -z zone1 clone lx-zone
```

システムには次のように表示されます。

```
Cloning zonepath /export/home/lx-zone...
```

7 システム内のゾーンの一覧を表示します。

```
global# zoneadm list -iv
  ID  NAME          STATUS          PATH                    BRAND      IP
   0  global        running         /                       native     shared
   -  lx-zone       installed       /export/home/lx-zone    lx         shared
   -  zone1         installed       /export/home/zone1      lx         shared
```

▼ 既存のスナップショットからゾーンを複製する方法

最初にゾーンを複製したときに作成された既存のスナップショットから、元のゾーンを何度も複製することができます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 ゾーン **zone2** を構成します。

3 既存のスナップショットを使用して **new-zone2** を作成することを指定します。

```
global# zoneadm -z zone2 clone -s zeepool/zones/lx-zone@SUNWzone1 lx-zone
```

システムには次のように表示されます。

```
Cloning snapshot zeepool/zones/lx-zone@SUNWzone1
```

zoneadm コマンドは、スナップショット SUNWzone*1* のソフトウェアを検証し、スナップショットを複製します。

4 システム内のゾーンの一覧を表示します。

```
global# zoneadm list -iv
  ID NAME            STATUS         PATH                    BRAND      IP
   0 global          running        /                       native     shared
```

```
- lx-zone          installed      /zeepool/zones/lx-zone  lx         shared
- zone1            installed      /zeepool/zones/zone1    lx         shared
- zone2            installed      /zeepool/zones/zone1    lx         shared
```

▼ ZFSクローンの代わりにコピーを使用する方法

ZFSファイルシステム上のゾーンの自動複製を防止し、代わりにzonepathをコピーするように指定する場合は、ここで説明する手順を使用します。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたはPrimary Administrator役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solarisのシステム管理(基本編)』の「Solaris管理ツールをRBACと組み合わせて使用する(作業マップ)」を参照してください。

2. ZFSクローンは行わず、ZFS上のzonepathをコピーするように指定します。

```
global# zoneadm -z zone1 clone -m copy lx-zone
```

システムからlxブランドゾーンを削除する

このセクションでは、システムからゾーンを完全に削除する手順を説明します。

▼ lxブランドゾーンを削除する方法

1. ゾーンlx-zoneを停止処理します。

```
global# zlogin lx-zone shutdown -y -g0 -i0
```

2. lx-zoneのルートファイルシステムを削除します。

```
global# zoneadm -z lx-zone uninstall -F
```

3. lx-zoneの構成を削除します。

```
global# zonecfg -z lx-zone delete -F
```

4. システム内のゾーンの一覧を表示し、lx-zoneが一覧に含まれていないことを確認します。

```
global# zoneadm list -iv
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH   BRAND      IP
 0  global   running      /      native     shared
```

◆ ◆ ◆ 第 36 章

lx ブランドゾーンへのログイン (タスク)

この章では、次の内容について説明します。

- ゾーンへのログインの概要
- インストール済み lx ブランドゾーンの内部構成の完了
- 大域ゾーンからゾーンへのログイン
- ゾーンの停止処理
- zonename コマンドを使用した現在のゾーンの名前の出力

zlogin コマンドの概要

zlogin コマンドを使用して、大域ゾーンから稼働状態または準備完了状態にある任意のゾーンにログインします。

注 – 稼働状態にないゾーンへのログインに使用できるのは、zlogin コマンドと -C オプションの組み合わせだけです。

-C オプションを使用してゾーンコンソールに接続しない限り、zlogin を使用してゾーンにログインすると、新しいタスクが開始されます。1 つのタスクを 2 つのゾーンで実行することはできません。

517 ページの「非対話型モードを使用して lx ブランドゾーンにアクセスする方法」に記述されているとおり、ゾーン内での実行を指示するコマンドを指定することで、zlogin コマンドを非対話型モードで使用できます。ただし、このコマンドおよびこのコマンドの処理対象となるファイルは、いずれも NFS 上に存在していてはなりません。開かれているファイルのいずれか、またはそのアドレス空間のいずれかの部分が NFS 上に存在する場合、コマンドは失敗します。アドレス空間には、コマンドの実行可能ファイル自体またはリンクされたライブラリが含まれます。

大域ゾーンを操作する大域管理者が使用できるのは、zlogin コマンドだけです。詳細は、zlogin(1) のマニュアルページを参照してください。

lx ブランドゾーンへのログイン方法

ゾーンコンソールログインおよびユーザーログインの方法の概要については、309 ページの「非大域ゾーンへのログイン方法」を参照してください。

ログインで問題が発生し、`zlogin` コマンドまたは `zlogin` コマンドと `-C` オプションを使用してゾーンにアクセスできない場合は、フェイルセーフモードを使用します。このモードについては、309 ページの「フェイルセーフモード」を参照してください。

ゾーンへのリモートログインについては、310 ページの「リモートログイン」を参照してください。

対話型モードでは、ゾーン内部で使用する新しい仮想端末が割り当てられます。非対話型モードは、ゾーンを管理するシェルスクリプトを実行する場合に使用します。詳細は、310 ページの「対話型モードと非対話型モード」を参照してください。

ブランドゾーンへのログイン手順 (タスクマップ)

タスク	説明	参照先
ゾーンにログインします。	ゾーンへのログインには、コンソールを使用する、対話型モードを使って仮想端末を割り当てる、またはゾーン内で実行するコマンドを指定する方法があります。実行するコマンドを指定する場合、仮想端末は割り当てられません。ゾーンへの接続が拒否された場合は、フェイルセーフモードを使用してログインすることもできます。	515 ページの「lx ブランドゾーンへのログイン」
ブランドゾーンから抜けます。	ブランドゾーンへの接続を切り離します。	517 ページの「lx ブランドゾーンから抜ける方法」
ブランドゾーンを停止処理します。	`shutdown` ユーティリティーまたはスクリプトを使用して、ブランドゾーンを停止処理します。	518 ページの「zlogin を使用して lx ブランドゾーンを停止処理する方法」

lx ブランドゾーンへのログイン

zlogin コマンドを使用して、大域ゾーンから稼働状態または準備完了状態にある任意のゾーンへログインします。詳細は、zlogin(1) のマニュアルページを参照してください。

次の手順で説明されているように、ゾーンへのログインはさまざまな方法で実行できます。310 ページの「リモートログイン」で説明されているように、リモートでログインすることも可能です。

▼ lx ブランドゾーンコンソールへのログイン方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **zlogin** コマンドを **-C** オプションと **lx-zone** などのゾーン名とともに使用します。

```
global# zlogin -C lx-zone
[Connected to zone 'lx-zone' console]
```

注 – zoneadm boot コマンドの実行後、すぐに zlogin セッションを開始すると、ゾーンからのブートメッセージが表示されます。

```
INIT: version 2.85 booting
                    Welcome to CentOS
                    Press 'I' to enter interactive startup.
    Configuring kernel parameters:  [  OK  ]
    Setting hostname lx-zone:  [  OK  ]
    [...]
    CentOS release 3.6 (Final)
    Kernel 2.4.21 on an i686
```

---

3 ゾーンコンソールが表示されたら、**root** でログインし、**Return** キーを押します。プロンプトが表示されたら **root** のパスワードを入力します。

```
lx-zone console login: root
Password:
```

注 – ゾーンを Sun tarball からインストールした場合、root (スーパーユーザー) パスワードは root です。ゾーンを ISO イメージまたは CD からインストールした場合、root (スーパーユーザー) パスワードは未設定 (空白) です。

▼ 対話型モードを使用してブランドゾーンにアクセスする方法

対話型モードでは、ゾーン内部で使用する新しい仮想端末が割り当てられます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。
役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 大域ゾーンからゾーン (例: **lx-zone**) にログインします。

```
global# zlogin lx-zone
```

次のような情報が表示されます。

```
[Connected to zone 'lx-zone' pts/2]
Last login: Wed Jul  3 16:25:00 on console
Sun Microsystems Inc. SunOS 5.10 Generic July 2006
```

3 **exit** と入力して、接続を閉じます。
次のようなメッセージが表示されます。

```
[Connection to zone 'lx-zone' pts/2 closed]
```

▼ 稼働中の環境を確認する方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。
役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 ゾーン (例: **lx-zone**) にログインします。

```
global# zlogin lx-zone
```

3 **Oracle Solaris** オペレーティングシステムの下で **Linux** 環境が稼働していることを確認します。

```
[root@lx-zone root]# uname -a
```

次のような情報が表示されます。

```
Linux lx-zone 2.4.21 BrandZ fake linux i686 i686 i386 GNU/Linux
```

▼ 非対話型モードを使用して lx ブランドゾーンにアクセスする方法

ゾーン内部で実行されるコマンドを指定すると、非対話型モードが有効になります。非対話型モードでは、新しい仮想端末は割り当てられません。

コマンドおよびコマンドの処理対象のファイルは、いずれも NFS 上に存在してはならないことに注意してください。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 大域ゾーンから **lx-zone** ゾーンにログインして、コマンド名を入力します。
   command には、ゾーン内で実行するコマンドの名前を指定します。

   ```
   global# zlogin lx-zone command
   ```

例 **36–1** ゾーン lx_master でコマンド uptime を使用する

```
global#  zlogin lx_master uptime
    21:16:01  up  2:39,  0 users,  load average: 0.19, 0.13, 0.11
    fireball#
```

▼ lx ブランドゾーンから抜ける方法

- 非大域ゾーンへの接続を切り離すには、次のいずれかの方法を使用します。
  - ゾーンの非仮想コンソールを終了するには、次の操作を行います。

    ```
    zonename# exit
    ```

  - ゾーンの仮想コンソールへの接続を切り離すには、次のようにチルダ (~) 文字とピリオドを使用します。

    ```
    zonename# ~.
    ```

    画面には、次のようなメッセージが表示されます。

    ```
    [Connection to zone 'lx-zone' pts/6 closed]
    ```

参照 `zlogin` コマンドのオプションの詳細については、`zlogin(1)` のマニュアルページを参照してください。

▼ フェイルセーフモードを使用して lx ブランドゾーンに入る方法

ゾーンへの接続が拒否された場合、zlogin コマンドと -S オプションを使用して、ゾーン内の最小環境に入ることができます。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは Primary Administrator 役割になります。
役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 大域ゾーンから、**zlogin** コマンドと **-S** オプションを使用してゾーン (例: **lx-zone**) にアクセスします。

```
global# zlogin -S lx-zone
```

▼ zlogin を使用して lx ブランドゾーンを停止処理する方法

注 – 大域ゾーンで init 0 を実行して Oracle Solaris システムの完全な停止処理を実行すると、システム上のそれぞれの非大域ゾーンでも init 0 が実行されます。init 0 は、ローカルユーザーとリモートユーザーに対してシステムが停止する前にログオフするよう警告しません。

ゾーンを正しく停止処理するには、次の手順を実行します。停止処理スクリプトを実行せずにゾーンを停止する方法については、299 ページの「ゾーンの停止方法」を参照してください。

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1 スーパーユーザーまたは Primary Administrator 役割になります。
役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 停止処理を行うゾーン (例: **lx-zone**) にログインし、ユーティリティーの名前として **shutdown** を、状態として **init 0** を指定します。

```
global# zlogin lx-zone shutdown -y -g0 -i0
```

サイトによっては、特定の環境に合わせた独自の停止処理スクリプトが存在する場合もあります。

参考　非対話型モードでの shutdown の使用

現時点では、非対話型モードで shutdown コマンドを使って、ゾーンをシングルユーザー状態にすることはできません。詳細は、6214427 を参照してください。

516 ページの「対話型モードを使用してブランドゾーンにアクセスする方法」の説明に従って、対話型ログインを使用できます。

◆ ◆ ◆ 第 37 章

lx ブランドゾーンの移動と移行 (タスク)

この章では、次の操作を行う方法について説明します。

- 既存の lx ブランドゾーンを同じマシンの新しい場所に移動する
- 実際の移行を実行する前に、lx ブランドゾーンの移行で何が発生するかを検証する
- 既存の lx ブランドゾーンを新しいマシンに移行する

lx ブランドゾーンの移動

zonepath を変更してゾーンを同じシステムの新しい場所に移動する場合に、ここで説明する手順を使用します。ゾーンは、停止する必要があります。新規 zonepath がローカルファイルシステムに存在する必要があります。474 ページの「リソースタイプとプロパティータイプ」に説明されている、 zonepath の通常の基準が適用されます。

▼ ゾーンを移動する方法

1. スーパーユーザーまたは **Primary Administrator** 役割になります。

   役割については、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 移動するゾーン (この手順では **db-zone**) を停止します。

   ```
   global# zoneadm -z db-zone halt
   ```

3. **zoneadm** コマンドを **move** サブコマンドとともに使用して、ゾーンを新規の **zonepath** である **/export/zones/db-zone** に移動します。

   ```
   global# zoneadm -z db-zone move /export/zones/db-zone
   ```

4 パスを確認します。

```
global# zoneadm list -iv
  ID NAME             STATUS         PATH                    BRAND      IP
   0 global           running        /                       native     shared
   - lx-zone          installed      /export/home/lx-zone    lx         shared
   - db-zone          installed      /export/zones/db-zone   lx         shared
```

別のマシンへの lx ブランドゾーンの移行

lx ブランドゾーンの移行について

zonecfg および zoneadm コマンドを使用して、既存の非大域ゾーンをあるシステムから別のシステムに移行できます。ゾーンは停止され、現在のホストから切り離されます。zonepath はターゲットホストに移動され、そこで接続されます。

lx ブランドゾーンの移行には、次の要件が適用されます。

- 移行先システムの大域ゾーンで、移行元ホストと同じ Oracle Solaris リリースが稼働している必要があります。
- ゾーンが正常に動作することを保証するため、移行元ホストにインストールされているものと同じ必須オペレーティングシステムパッケージおよびパッチが、移行先システムにもインストールされている必要があります。
- ブランドは、移行元ホストと移行先システムで同じでなければなりません。
- 移行先システムは、サポートされている次の i686 プロセッサタイプのいずれかを備えている必要があります。
  - Intel
    - Pentium Pro
    - Pentium II
    - Pentium III
    - Celeron
    - Xeon
    - Pentium 4
    - Pentium M
    - Pentium D
    - Pentium Extreme Edition
    - Core
    - Core 2

    AMD
    - Opteron
    - Athlon XP
    - Athlon 64

- Athlon 64 X2
- Athlon FX
- Duron
- Sempron
- Turion 64
- Turion 64 X2

zoneadm detach プロセスにより、別のシステムでゾーンを接続するのに必要な情報が作成されます。 zoneadm attach プロセスは、ターゲットマシンがゾーンのホストとして機能するための適正な構成を保持していることを確認します。新規ホストで zonepath を使用可能にする方法は複数存在するため、あるシステムから別のシステムへの zonepath の実際の移動は、大域管理者が手動で行います。

新規システムへの接続時に、ゾーンはインストール済みの状態になります。

▼ lx ブランドゾーンを移行する方法

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 移行するゾーン (この手順では **lx-zone**) を停止します。

   ```
   host1# zoneadm -z lx-zone halt
   ```

3. ゾーンを切り離します。

   ```
   host1# zoneadm -z lx-zone detach
   ```

   切り離されたゾーンは、現在、構成済みの状態にあります。

4. **lx-zone** の **zonepath** を新規ホストに移動します。
   詳細は、525 ページの「zonepath を新規ホストに移動する方法」を参照してください

5. 新規ホスト上でゾーンを構成します

   ```
   host2# zonecfg -z lx-zone
   ```

   次のシステムメッセージが表示されます

   ```
   lx-zone: No such zone configured
   Use 'create' to begin configuring a new zone.
   ```

6. 新規ホスト上にゾーン **lx-zone** を作成するには、**zonecfg** コマンドに **-a** オプションおよび新規ホストの **zonepath** を指定します。

   ```
   zonecfg:lx-zone> create -a /export/zones/lx-zone
   ```

7 構成を表示します。

```
zonecfg:lx-zone> info
zonename: lx-zone
zonepath: /export/zones/lx-zone
brand: lx
autoboot: false
bootargs:
pool:
limitpriv:
net:
         address: 192.168.0.90
         physical: bge0
```

8 (オプション) 構成に必要な調整を加えます。

たとえば、新規ホストではネットワーク物理デバイスが異なる場合があります。また、構成に含まれるデバイスの名前が新規ホストでは異なることもあります。

```
zonecfg:lx-zone> select net physical=bge0
zonecfg:lx-zone:net> set physical=e1000g0
zonecfg:lx-zone:net> end
```

9 構成を確定して終了します。

```
zonecfg:lx-zone> commit
zonecfg:lx-zone> exit
```

10 新規ホスト上でゾーンを接続します。

- 妥当性検査を使用して、ゾーンを接続します。

  ```
  host2# zoneadm -z lx-zone attach
  ```

  次の条件のいずれかまたは両方に当てはまる場合、実行が必要な操作がシステム管理者に通知されます。

  - 必須パッケージおよびパッチが新規マシンに存在しない。
  - ソフトウェアレベルがマシン間で異なる。

- 妥当性検査を実行せずに、接続操作を強制的に実行します。

  ```
  host2# zoneadm -z lx-zone attach -F
  ```

注意 – -F オプションを使用すると、妥当性検査を実行せずに attach が強制的に実行されます。これは、クラスタ環境やバックアップ/復元操作など、特定の場合に役立ちますが、システムがゾーンのホストとして動作するよう正しく構成されている必要があります。構成が不正な場合、あとで未定義の動作が実行される可能性があります。

▼ zonepath を新規ホストに移動する方法

zonepath のアーカイブの作成には、いくつもの方法があります。たとえば、cpio または pax コマンドを使用できます。詳細は、cpio(1) および pax(1) のマニュアルページを参照してください。

アーカイブを新規ホストに転送する方法も、複数存在します。zonepath を転送元ホストから転送先ホストに転送するメカニズムは、ローカルの構成によって異なります。SAN などのいくつかの場合には、zonepath データを実際には移動できないこともあります。SAN の場合は、zonepath が新規ホストに表示されるように、再構成が実行されるだけです。それ以外の場合は、zonepath をテープに書き込み、それを新規サイトに送付することもあります。

これらの理由のために、この手順は自動化されていません。システム管理者は、zonepath を新規ホストに移動する最適な手法を選択する必要があります。

1 スーパーユーザーまたは **Primary Administrator** 役割になります。

役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2 **zonepath** を新規ホストに移動します。この手順で説明した方法を使用することも、別の方法を選んで使用することもできます。

例 37–1 tar コマンドを使用した zonepath のアーカイブおよび移動

1. host1 上で zonepath の tar ファイルを作成し、 sftp コマンドを使って host2 に転送します。

```
host1# cd /export/zones
host1# tar cf lx-zone.tar lx-zone
host1# sftp host2
Connecting to host2...
Password:
sftp> cd /export/zones
sftp> put lx-zone.tar
Uploading lx-zone.tar to /export/zones/lx-zone.tar
sftp> quit
```

2. host2 上で tar ファイルを展開します。

```
host2# cd /export/zones
host2# tar xf lx-zone.tar
```

詳細は、sftp(1) および tar(1) を参照してください。

注意事項 次の情報のトラブルシューティングについては、444 ページの「zoneadm 接続操作の問題解決」を参照してください。

- パッチおよびパッケージが同期しない。

- オペレーティングシステムのリリースが一致しない。

新しいマシンのプロセッサタイプがサポートされていることを確認する必要があります。詳細は、522 ページの「lx ブランドゾーンの移行について」を参照してください。

Oracle Solaris 10 5/08: 移行を行う前の lx ブランドゾーンの移行の検証について

「no execute」(実行しない) オプションである -n を使用することで、ゾーンを新しいマシンに移行する前に試行を行うことができます。

-n オプションを指定して zoneadm detach サブコマンドを使用すると、実際にゾーンを切り離さずに実行中のゾーンでマニフェストを生成できます。移行元のシステムのゾーンの状態は変わりません。ゾーンのマニフェストは標準出力に送信されます。大域管理者は、この出力をファイルに送ったり、移行先ホストですぐに検証されるようにリモートコマンドにパイプしたりできます。-n オプションを指定して zoneadm attach サブコマンドを使用すると、このマニフェストを読み取り、実際に接続を行わずに、移行先のマシンがゾーンのホストとして機能するための適正な構成を保持しているかどうかを確認できます。

試行接続を行う前に、新規ホストで移行先システムのゾーンを構成する必要はありません。

▼ Oracle Solaris 10 5/08: 移行を行う前に lx ブランドゾーンの移行を検証する方法

この手順を実行するには、大域ゾーン内で大域管理者になる必要があります。

1. スーパーユーザーまたは **Primary Administrator** 役割になります。
   役割の作成と作成した役割のユーザーへの割り当てについては、『Solaris のシステム管理 (基本編)』の「Solaris 管理ツールを RBAC と組み合わせて使用する (作業マップ)」を参照してください。

2. 次のいずれかを実行します。

   - **lx-zone** という名前の移行元ホストでマニフェストを生成し、移行先ホストをすぐに検証するリモートコマンドにその出力をパイプします。

     ```
     global# zoneadm -z lx-zone detach -n | ssh remotehost zoneadm attach -n -
     ```

     行の最後にあるハイフン (–) は、パスとして標準入力を指定します。

- **lx-zone** という名前の移行元ホストでマニフェストを生成し、その出力をファイルに送ります。

  ```
  global# zoneadm -z lx-zone detach -n
  ```

  525 ページの「zonepath を新規ホストに移動する方法」の説明に従って、そのマニフェストを新しいホストシステムにコピーし、検証を行います。

  ```
  global# zoneadm attach -n path_to_manifest
  ```

  パスを – にすると標準入力を指定できます。

◆ ◆ ◆ 第 38 章

38 lx ブランドゾーンでのアプリケーションの管理と実行 (タスク)

この章では、lx ブランドゾーンでのアプリケーションの実行について説明します。

サポートされている構成の保守について

サポートされている CentOS または Red Hat Enterprise Linux ディストリビューションを使用してゾーンをインストールすると、サポートされているゾーンが作成されます。このゾーンに別のバージョンからパッケージを追加すると、サポート不可能なブランドゾーンが作成される可能性があります。

ディストリビューションのアップグレードとパッケージの追加

▼ CentOS 3.*x* ディストリビューションをアップグレードする方法

この手順を実行するには、lx ブランドゾーン内のゾーン管理者になる必要があります。

- **yum upgrade** または **up2date** を使用して、CentOS 3.*x* ディストリビューションを別のバージョンにアップグレードします。

  手順については、`http://www.centos.org` で利用可能なドキュメントを参照してください。

▼ Red Hat 3.*x* ディストリビューションをアップグレードする方法

この手順を実行するには、lx ブランドゾーン内のゾーン管理者になる必要があります。

- **up2date** を使用して、**Red Hat Enterprise Linux 3.*x*** ディストリビューションを別のバージョンにアップグレードします。

  手順については、http://www.redhat.com で利用可能なドキュメントを参照してください。

▼ パッケージをアップグレードする方法

この手順を実行するには、lx ブランドゾーン内のゾーン管理者になる必要があります。

- パッケージを更新するには、次のいずれかの方法を使用します。
  - **yum update** *package_name*
  - **rpm -U** *package_name*

参考 yum と rpm の使用

yum:

- Fedora Documentation のサイト
- yum.conf(5)
- yum(8)

rpm:

- https://access.redhat.com/kb/FAQ_35_198.shtm の「How do I install or upgrade an RPM package?」を参照してください。
- rpm(8)

lx ブランドゾーンにアプリケーションをインストールする方法

アプリケーションをインストールするには、Linux システムでの手順と同様に、CD をマウントし、インストールプログラムを実行します。このセクションでは、lx ブランドゾーンへのアプリケーションの一般的なインストールについて説明します。

ヒント – CD または DVD を使用してアプリケーションを lx ブランドゾーンにインストールする予定の場合は、ブランドゾーンを最初に構成するときに、大域ゾーンの CD または DVD メディアに読み取り専用のアクセスを行う権限を追加します。531 ページの「CD を使用して MATLAB 7.2 をインストールする方法」の手順 7 を参照してください。

MATLAB について

MATLAB は、計算負荷の高いタスクをすばやく実行できる対話型環境を提供する高級言語です。この製品は The MathWorks によって開発されました。詳細は、`http://www.mathworks.com` を参照してください。

▼ CD を使用して MATLAB 7.2 をインストールする方法

1. **MATLAB 7.2 の CD を入手します。**
   MATLAB/Simulink パッケージには 3 枚の CD があります。単純な MATLAB のインストールには、ディスク 1 と 3 だけが必要です。

2. **lx ブランドゾーンを作成し、インストールします。**手順については、**483 ページの「lx ブランドゾーンを構成、検証、および確定する方法」**および**498 ページの「lx ブランドゾーンのインストールとブート」**を参照してください。

3. 大域ゾーンでボリューム管理ファイルシステムが稼働していない場合は、起動します。

   ```
   global# svcadm volfs enable
   ```

4. メディアを挿入します。

5. ドライブにメディアが入っているかどうかを確認します。

   ```
   global# volcheck
   ```

6 **CD** が自動マウントされているかどうかをテストします。

```
global# ls /cdrom
```

次のような情報が表示されます。

```
cdrom   cdrom1   mathworks_2006a1
```

7 **ro,nodevices** オプション (読み取り専用、デバイスなし) を指定して、非大域ゾーンでファイルシステムをループバックマウントします。

```
global# zonecfg -z lx-zone
zonecfg:lx-zone> add fs
zonecfg:lx-zone:fs> set dir=/cdrom
zonecfg:lx-zone:fs> set special=/cdrom
zonecfg:lx-zone:fs> set type=lofs
zonecfg:lx-zone:fs> add options [ro,nodevices]
zonecfg:lx-zone:fs> end
zonecfg:lx-zone> commit
zonecfg:lx-zone> exit
```

8 非大域ゾーンをリブートします。

```
global# zoneadm -z lx-zone reboot
```

9 **zoneadm list** コマンドに **-v** オプションを指定して、ステータスを確認します。

```
global# zoneadm list -v
```

次のような情報が表示されます。

```
ID  NAME     STATUS       PATH                           BRAND      IP
 0  global   running      /                              native     shared
 1  lx-zone  running      /export/home/lx-zone           lx         shared
```

10 **lx** ゾーンにログインします。

```
global# zlogin lx-zone
```

11 **CD-ROM** がマウントされているかを確認します。

```
lx-zone# ls /cdrom
```

次のような内容が表示されます。

```
cdrom   cdrom1   mathworks_2006a1
```

12 **MATLAB** のドキュメントに従って、ライセンスファイルを作成します。

13 製品のインストールガイドで説明されているとおりに、製品をインストールします。

```
lx-zone# /mnt/install
```

14 ゾーンを終了します。

```
lx-zone# exit
```

ヒント – /cdrom ファイルシステムを非大域ゾーンに残すこともできます。マウントするときには、常に CD-ROM ドライブの現在の内容が反映されます。つまり、ドライブが空の場合は、ディレクトリは空になります。

15 (オプション) 非大域ゾーンから **/cdrom** ファイルシステムを削除する場合は、次の手順を使用します。

```
global# zonecfg -z lx-zone
zonecfg:lx-zone> remove fs dir=/cdrom
zonecfg:lx-zone> commit
zonecfg:lx-zone> exit
```

▼ ISO イメージを使用して MATLAB 7.2 をインストールする方法

始める前に　この方法では、かなり多くのディスク容量が消費されることに注意してください。

1 **MATLAB 7.2** の **CD** を入手します。

MATLAB/Simulink パッケージには 3 枚の CD があります。単純な MATLAB のインストールには、ディスク 1 と 3 だけが必要です。

2 **lx** ブランドゾーンを作成し、インストールします。手順については、**483** ページの「**lx** ブランドゾーンを構成、検証、および確定する方法」および **498** ページの「**lx** ブランドゾーンのインストールとブート」を参照してください。

3 各 **CD** のデータを **.iso** ファイルにコピーします。

```
global# /usr/bin/dd if=/dev/rdsk/c1d0s2 of=disk1.iso
```

これにより、最初の CD のデータがファイル disk1.iso にコピーされます。disk3.iso などの別のファイル名を使用して、3 枚目の CD についても繰り返します。

4 大域ゾーンから、最初の **.iso** ファイルを **lx** ゾーンに **lofi** でマウントします。

```
global# lofiadm -a /zpool/local/disk1.iso
global# mount -F hsfs /dev/lofi/1 /zones/lx-zone/root/mnt
```

5 **lx** ゾーンにログインします。

```
global# zlogin lx-zone
```

6 **X** 転送を使用して、表示先をデスクトップに変更します。

```
lx-zone# ssh -X root@lx-zone
```

7 **MATLAB** のドキュメントに従って、ライセンスファイルを作成します。

8 製品のインストールガイドで説明されているとおりに、製品をインストールします。

```
lx-zone# /mnt/install
```

9 CD 3 の挿入を求めるメッセージが表示されたら、大域ゾーンの端末ウィンドウに戻り、最初のファイルの代わりに **disk3.iso** ファイルをマウントします。

```
global# umount /zones/lx-zone/root/mnt
    global# lofiadm -d /dev/lofi/1
    global# lofiadm -a /zpool/local/disk3.iso
    global# mount -F hsfs /dev/lofi/1 /zones/lx-zone/root/mnt
```

インストールが終了します。

lx ブランドゾーンのバックアップ

ゾーンのバックアップについては、404 ページの「ゾーンがインストールされている Oracle Solaris システムのバックアップについて」、405 ページの「非大域ゾーン内でバックアップするデータの決定」、407 ページの「非大域ゾーンの復元について」、および434 ページの「非大域ゾーンの復元」を参照してください。

lx ブランドゾーンでサポートされていない機能

lx ブランドゾーンでは、共有 IP ネットワーク構成だけがサポートされています。

Linux ゾーンでは、chroot コマンドはサポートされていません。プロセスに対して使用すると、そのプロセスは実行に必要な Oracle Solaris ライブラリを見つけることができなくなります。

ラベルが有効になっている Trusted Oracle Solaris システムに lx ブランドゾーンを構成してインストールすることはできますが、このシステム構成で lx ブランドゾーンをブートすることはできません。

zonecfg コマンドの fs リソースプロパティーを使用してローカルの Linux ファイルシステムを追加することはできません。

用語集

bless Perl において、このキーワードはオブジェクトを作成するときに使用されます。

blessed Perl において、この用語はクラスのメンバーシップを示すのに使用されます。

FSS 公平配分スケジューラを参照してください。

Oracle Solaris コンテナ 完全なアプリケーション実行環境。リソース管理と Oracle Solaris ゾーンソフトウェア区分技術は、どちらもこのコンテナの一部です。

Oracle Solaris ゾーン Oracle Solaris コンテナを参照してください。オペレーティングシステムサービスを仮想化し、アプリケーションを実行するための分離およびセキュリティー保護された環境を提供するソフトウェア区分技術。

RSS 常駐セットサイズを参照してください。

WSS 作業セットサイズを参照してください。

アカウンティングの拡張 Oracle Solaris オペレーティングシステムで、タスクまたはプロセスに基づくリソース消費量を柔軟に記録できる方法。

完全ルートゾーン 非大域ゾーンの一種で、`inherit-pkg-dir` リソースを持たないもの。

局所有効範囲 制御値を超えようとしているプロセスに対して実行される局所アクション。

公平配分スケジューラ 公平さを基準に CPU 時間を割り当てるスケジューリングクラス (FSS とも呼ぶ)。配分は、システムの CPU リソースのうちプロジェクトに割り当てる部分を定義します。

作業セットサイズ 作業セットのサイズ。作業セットとは、プロジェクトの作業負荷がその処理サイクル中にアクティブに使用するページのことです。

作業負荷 アプリケーションまたはアプリケーショングループのプロセスすべての合計。

上限 システムリソース使用率に対する規制。

上限制御 システムリソース使用率を規制するプロセス。

常駐セットサイズ 常駐セットのサイズ。常駐セットとは、物理メモリーに常駐するページのことです。

スキャナ まれにしか使用されないページを識別し、そのページを物理メモリー外部の領域に再配置するカーネルスレッド。

静的プール構成 管理者が、リソースプール機能に関してシステムを構成する方法を表現したもの。

ゾーン管理者	Zone Management プロファイルを保持する管理者。ゾーン管理者の特権は、非大域ゾーンに対してのみ有効です。 大域管理者も参照してください。
ゾーン状態	非大域ゾーンのステータス。ゾーン状態は、構成済み、不完全、インストール済み、準備完了、稼働、または停止処理のいずれかになります。
疎ルートゾーン	非大域ゾーンの一種で、`inherit-pkg-dir` リソースを持ち、オブジェクトの共有を最適化します。
大域管理者	スーパーユーザー権限または Primary Administrator の役割を持つ管理者。大域ゾーンにログインすると、大域管理者はシステム全体を監視したり制御したりできます。 ゾーン管理者も参照してください。
大域ゾーン	すべての Oracle Solaris システムに含まれるゾーン。非大域ゾーンを使用しているときには、大域ゾーンはシステムのデフォルトゾーンであると同時に、システム規模の管理制御に使用されるゾーンでもあります。 非大域ゾーンも参照してください。
大域有効範囲	システム上のすべてのリソース制御のリソース制御値に適用されるアクション。
タスク	リソース管理において、長時間にわたる作業の集合を表すプロセスの集まり。各タスクは 1 つのプロジェクトに関連付けられます。
デフォルトプール	プールが有効に設定される際にシステムにより作成されるプール。 リソースプールも参照してください。
デフォルトプロセッサセット	プールが有効に設定される際にシステムにより作成されるプロセッサセット。 プロセッサセットも参照してください。
動的構成	ある時点における、指定されたシステムのリソースプールフレームワーク内部のリソース配置に関する情報。
動的再構成	SPARC ベースのシステムで、システムの稼働中にハードウェアを再構成する機能。DR とも呼ばれます。
ネームサービスデータベース	このドキュメントの「プロジェクトとタスク (概要)」の章では、LDAP コンテナと NIS マップの両方を指して使用されます。
ヒープ	プロセス内で割り当てられたスクラッチメモリー。
非大域ゾーン	Oracle Solaris オペレーティングシステムの単一インスタンス内に作成された仮想オペレーティングシステム環境。Oracle Solaris ゾーンソフトウェア区分技術を使用して、オペレーティングシステムサービスが仮想化されます。
非大域ゾーン管理者	ゾーン管理者を参照してください。
プール	リソースプールを参照してください。

プールデーモン	リソースの動的割り当てが必要な場合にアクティブになる poold システムデーモン。
ブランドゾーン	実行時の動作の代替セットを含むコンテナを作成するためのフレームワーク。
プロジェクト	ネットワーク全体の関連作業に対する管理識別子。
プロセッサセット	互いに素である CPU のグループ。各プロセッサセットには、0 以上のプロセッサを含めることができます。プロセッサセットは、リソースプール構成内でリソース要素として表されます。これは pset とも呼ばれます。 素も参照してください。
ページアウト	ページを物理メモリー外部の領域に再配置すること。
ページイン	1 回に 1 ページをファイルから物理メモリーに読み出すこと。
メモリー上限実行しきい値	リソース上限デーモンが上限を制限するときのシステム上における物理メモリーの使用効率 (パーセンテージ)。
素	セットのメンバーが重複しないセットのタイプ。
リソース	アプリケーションの動作を変更する目的で操作されるコンピューティングシステムの側面。
リソース管理	アプリケーションが利用可能なシステムリソースをどのように使用するかを制御する機能。
リソース上限デーモン	リソース上限が定義されたプロジェクト内で動作するプロセスが消費する物理メモリーを規制するデーモン。
リソース消費者	基本的には Oracle Solaris のプロセス。プロジェクトやタスクなどのプロセスモデルエンティティーにより、集計済みのリソース消費に関して考察できます。
リソース制御	プロセスごと、タスクごと、またはプロジェクトごとのリソース消費量に対する制限。
リソースセット	プロセスをバインド可能なリソース。たいていの場合、カーネルサブシステムにより構築され、ある種の区分化を提供するオブジェクトを指して使用されます。リソースセットの例には、スケジューリングクラスやプロセッサセットが含まれます。
リソースパーティション	リソースの排他的な一部。リソースのパーティションすべての合計は、実行中の単一の Oracle Solaris インスタンスで利用可能なリソースの総量を表します。
リソースプール	マシンのリソース区分化に使用する構成メカニズム。リソースプールは、区分化可能なリソースグループ間の関係を表します。
ロックされたメモリー	ページング不可能なメモリー。

索引

P