

Oracle® Solaris 11.1 パッケージリポジト リのコピーおよび作成

このソフトウェアおよび関連ドキュメントの使用と開示は、ライセンス契約の制約条件に従うものとし、知的財産に関する法律により保護されています。ライセンス契約で明示的に許諾されている場合もしくは法律によって認められている場合を除き、形式、手段に関係なく、いかなる部分も使用、複写、複製、翻訳、放送、修正、ライセンス供与、送信、配布、発表、実行、公開または表示することはできません。このソフトウェアのリバース・エンジニアリング、逆アセンブル、逆コンパイルは互換性のために法律によって規定されている場合を除き、禁止されています。

ここに記載された情報は予告なしに変更される場合があります。また、誤りが無いことの保証はいたしかねます。誤りを見つけた場合は、オラクル社までご連絡ください。

このソフトウェアまたは関連ドキュメントを、米国政府機関もしくは米国政府機関に代わってこのソフトウェアまたは関連ドキュメントをライセンスされた者に提供する場合は、次の通知が適用されます。

U.S. GOVERNMENT END USERS:

Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

このソフトウェアもしくはハードウェアは様々な情報管理アプリケーションでの一般的な使用のために開発されたものです。このソフトウェアもしくはハードウェアは、危険が伴うアプリケーション（人的傷害を発生させる可能性があるアプリケーションを含む）への用途を目的として開発されていません。このソフトウェアもしくはハードウェアを危険が伴うアプリケーションで使用する場合、安全に使用するために、適切な安全装置、バックアップ、冗長性（redundancy）、その他の対策を講じることは使用者の責任となります。このソフトウェアもしくはハードウェアを危険が伴うアプリケーションで使用したこと起因して損害が発生しても、オラクル社およびその関連会社は一切の責任を負いかねます。

OracleおよびJavaはOracle Corporationおよびその関連企業の登録商標です。その他の名称は、それぞれの所有者の商標または登録商標です。

Intel, Intel Xeonは、Intel Corporationの商標または登録商標です。すべてのSPARCの商標はライセンスをもとに使用し、SPARC International, Inc.の商標または登録商標です。AMD, Opteron, AMDロゴ、AMD Opteronロゴは、Advanced Micro Devices, Inc.の商標または登録商標です。UNIXは、The Open Groupの登録商標です。

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。

目次

はじめに	5
1 Image Packaging System パッケージリポジトリ	9
ローカル IPS リポジトリ	9
リポジトリホストシステムの準備	10
システム要件	10
リポジトリ管理の特権	10
ZFS データセットの作成	11
2 IPS パッケージリポジトリのコピー	13
インターネットからのリポジトリのコピー	13
ローカルリポジトリ用のインフラストラクチャーの作成	13
リポジトリのコピー	13
ファイルからのリポジトリのコピー	15
パッケージリポジトリファイルの取得	15
リポジトリファイルのコンテンツを使用可能にする	15
リポジトリファイルのコピー	16
イメージのアンマウント	16
検索インデックスの構築とリポジトリのスナップショットの作成	17
3 リポジトリへのアクセスの提供	19
ファイルインタフェースを使用したパッケージの取得	19
NFS 共有の構成	19
ファイルリポジトリ URI への発行元の起点の設定	20
HTTP インタフェースを使用したパッケージの取得	20
リポジトリサーバーサービスの構成	20
リポジトリサービスの開始	21

HTTP リポジトリ URI への発行元の起点の設定	21
4 ローカル IPS パッケージリポジトリの保守	23
ローカルリポジトリの更新	23
リポジトリプロパティの確認および設定	24
ローカルリポジトリのカスタマイズ	26
複数の集積サーバーインスタンスを使用した複数のリポジトリの提供	27
集積サーバーの Apache 構成	28
集積サーバー用のキャッシュの構成	28
Web プロキシの背後での集積サーバーの実行	30
Apache 構成の例	31

はじめに

『Oracle Solaris 11.1 パッケージリポジトリのコピーおよび作成』は、Oracle Solaris Image Packaging System (IPS) 機能を使用してソフトウェアパッケージリポジトリを作成する方法について説明します。IPS ツールによって、独自のパッケージのために既存のパッケージのコピーまたは独自のリポジトリの作成が簡単に実行でき、リポジトリ内のパッケージを簡単に更新することができます。リポジトリのユーザーにはファイルインタフェースまたはHTTP インタフェースを提供できます。

対象読者

本書は、ソフトウェアのインストールおよび管理を行うか、ソフトウェアのインストールおよび管理を行うほかのユーザーを支援するシステム管理者を対象としています。

内容の紹介

- **第1章「Image Packaging System パッケージリポジトリ」**ではローカル IPS パッケージリポジトリを提供するメリットについて説明し、リポジトリ用の ZFS ファイルシステムを作成する方法を示します。
- **第2章「IPS パッケージリポジトリのコピー」**では、ファイルからのリポジトリのコピーおよびインターネットロケーションからのリポジトリのコピーについて説明します。
- **第3章「リポジトリへのアクセスの提供」**では、クライアントがリポジトリのパッケージを表示し、リポジトリからパッケージをインストールできるようにする方法について説明します。
- **第4章「ローカル IPS パッケージリポジトリの保守」**では、次のタスクを実施する方法について説明します。
 - 更新されたパッケージをリポジトリに追加する
 - リポジトリのプロパティの値を変更する
 - パッケージを別のソースからリポジトリに追加する
 - 1つのサーバー上にある複数のリポジトリへのアクセスを設定する
 - リポジトリ集積サーバーを構成する

関連ドキュメント

- 『Oracle Solaris 11.1 の管理: ZFS ファイルシステム』
- 『Oracle Solaris 11.1 でのサービスと障害の管理』の第1章「サービスの管理(概要)」では、Oracle Solaris サービス管理機能(SMF)について説明します
- Apache HTTP サーババージョン 2.2 ドキュメント
- 『Oracle Solaris 11.1 での Image Packaging System を使用したソフトウェアのパッケージ化および配布』
- 『Oracle Solaris 11.1 ソフトウェアパッケージの追加および更新』
- 『Oracle Solaris 11.1 システムのインストール』

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、このドキュメント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.
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:
<i>aabbcc123</i>	プレースホルダ: 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 <i>filename</i> と入力します。

表 P-1 表記上の規則 (続き)

字体	説明	例
<i>AaBbCc123</i>	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

次の表に、Oracle Solaris OS に含まれるシェルで使用する、UNIX 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示します。コマンド例では、シェルプロンプトはコマンドが一般ユーザーまたは特権ユーザーのどちらによって実行されるべきかを示しています。

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#

Image Packaging System パッケージリポジトリ

Oracle Solaris 11 ソフトウェアは Image Packaging System (IPS) パッケージの形態で配布されます。IPS パッケージは、IPS 発行元が提供する IPS パッケージリポジトリに格納されます。

このガイドでは、IPS パッケージリポジトリを作成する方法について説明します。この章では、ローカルの IPS パッケージリポジトリを内部使用の目的で作成する理由を示します。

ローカル IPS リポジトリ

ローカル IPS リポジトリは次の理由で必要になる場合があります。

- パフォーマンスとセキュリティ。クライアントシステムがインターネットにアクセスして新規ソフトウェアパッケージを取得したり、既存のパッケージを更新したりすることを望まない。
- レプリケーション。今日実行したのと同じインストールを来年も確実に実行できるようにしたい。
- カスタムパッケージ。独自の IPS パッケージを Oracle Solaris パッケージと同じリポジトリに含めたい。

IPS は起点リポジトリとミラーリポジトリの2種類のリポジトリをサポートします。上記のパフォーマンスおよびセキュリティーの目標を達成するには、作成するローカルリポジトリは起点リポジトリとする必要があります。起点リポジトリは、1つ以上のパッケージのすべてのメタデータ(カタログ、マニフェスト、検索インデックスなど)とコンテンツ(ファイル)を含みます。ミラーリポジトリはパッケージのコンテンツ(ファイル)のみを含みます。ミラーリポジトリからパッケージをインストールおよび更新するクライアントは、起点リポジトリからメタデータをダウンロードする必要があります。IPS クライアントがパッケージのコンテンツをミラーからダウンロードする場合でも、クライアントは発行元のカタログを取得するために起点にアクセスします。

このドキュメントで説明するリポジトリコピー方法は、どちらも起点リポジトリを作成します。起点リポジトリは、`pkgrecv` コマンドを使用してパッケージリポジトリをコピーするときに暗黙に作成され、Oracle 提供のリポジトリ ISO ファイルが起点リポジトリを提供します。

リポジトリホストシステムの準備

このセクションでは、IPS パッケージリポジトリをホストするシステムの要件と推奨事項について説明します。このセクションでは、パッケージリポジトリの作成と構成を行うために必要な特権についても説明します。

システム要件

IPS パッケージリポジトリをホストするシステムは、x86 ベースまたは SPARC ベースのいずれかのシステムとすることができます。

オペレーティングシステム IPS リポジトリサーバーは、Oracle Solaris 11 11/11 OS またはそれ以降のバージョンの Oracle Solaris OS を実行している必要があります。Oracle Solaris 11 11/11 を実行しているリポジトリサーバーは、すべての Oracle Solaris 11 更新パッケージをサポートします。

ディスク容量 Oracle Solaris 11.1 リリースリポジトリのコピーをホストするには、リポジトリサーバーに 15G バイトの空き容量が必要です。

1 つのシステムが複数の IPS リポジトリをホストする場合、各リポジトリを別個の ZFS ファイルシステムにすることで、各リポジトリを別々にロールバックおよび復元できます。

リポジトリ管理の特権

パッケージリポジトリの作成と構成を行うために必要な特権を得るには、次のいずれかの方法を使用します。

権利プロファイル `profiles` コマンドを使用して、自分に割り当てられている権利プロファイルを一覧表示します。

ZFS ファイルシステム管理

「ZFS ファイルシステム管理」権利プロファイルを持っている場合は、`pfexec` コマンドを使用して `zfs` コマンドを実行できます。

```
$ pfexec zfs create rpool/export/repoSolaris11
```

ソフトウェアインストール

「ソフトウェアインストール」権利プロファイルを持っている場合は、`pfexec` コマンドを使用して `pkg` コマンドを実行できます。

```
$ pfexec pkg set-publisher \  
-g http://localhost:80/ solaris
```

サービス管理

「サービス管理」権利プロファイルを持っている場合は、サービスコマンドを実行できます。この場合、`pfexec` コマンドは必要ありません。

```
$ svcadm enable application/pkg/server
```

役割

`roles` コマンドを使用して、自分に割り当てられている役割を一覧表示します。root 役割を持つ場合、root パスワードで `su` コマンドを使用して、root 役割になることができます。

sudo コマンド

サイトのセキュリティポリシーに応じて、自分のユーザーパスワードで `sudo` コマンドを使用し、特権コマンドを実行できる場合があります。

ZFS データセットの作成

推奨されるベストプラクティスは、ローカルパッケージリポジトリ用に別個の ZFS ファイルシステムを作成することです。別個の ZFS ファイルシステムを使用すると、次のメリットを得ることができます。

- 高いパフォーマンスの達成。
- 異なるファイルシステム特性の設定。
- 指定したファイルシステムの直接のスナップショットと回復。

現在の ZFS データセットを表示するには `zfs list` コマンドを使用します。

```
$ zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
rpool                                75.2G  108G   5.00G  /rpool
rpool/ROOT                           23.0G  108G    31K   legacy
rpool/ROOT/solaris                   44.8G  108G   3.52G  /
rpool/dump                           1.97G  108G   1.97G  -
rpool/export                         43.0G  108G   30.5G  /export
rpool/export/home                    12.6G  108G    32K   /export/home
rpool/export/home/bob                12.6G  108G   12.6G  /export/home/bob
rpool/swap                           2.09G  108G   1.97G  -
```

ルートプール内にパッケージリポジトリ用の ZFS ファイルシステムを作成します。

```
$ pfexec zfs create rpool/export/repoSolaris11
$ zfs list
NAME                                USED  AVAIL  REFER  MOUNTPOINT
rpool                               75.2G  108G   5.00G   /rpool
rpool/export/repoSolaris11         31K   108G   31K    /export/repoSolaris11
...
```

ヒント-リポジトリを更新するときに高いパフォーマンスを得るには、`atime`を`off`に設定します。

```
$ pfexec zfs set atime=off rpool/export/repoSolaris11
```

`atime` プロパティは、ファイルが読み取られるときにファイルのアクセス時間が更新されるかどうかを制御します。このプロパティをオフにすると、ファイルを読み取るときに書き込みトラフィックが生成されなくなります。

IPS パッケージリポジトリのコピー

この章では、Oracle Solaris 11.1 リリースの IPS パッケージリポジトリのコピーを作成するための 2 つの方法について説明します: メディアまたは Oracle Solaris 11.1 ダウンロードサイトからのリポジトリファイルを使用するか、リポジトリをインターネットから取得することができます。

インターネットからのリポジトリのコピー

このセクションでは、Oracle Solaris 11.1 リリースパッケージリポジトリをインターネット上の場所からコピーすることによって、リポジトリのローカルコピーを作成する方法について説明します。

ローカルリポジトリ用のインフラストラクチャーの作成

リポジトリをコピーするために必要な pkg(5) リポジトリインフラストラクチャーを作成します。pkg(5) および pkgrepo(1) のマニュアルページを参照してください。

```
$ pkgrepo create /export/repoSolaris11
```

リポジトリのコピー

リポジトリをコピーするには、pkgrecv コマンドを使用します。この操作はネットワークパフォーマンスに影響することがあります。この操作が完了するために必要な時間は、ネットワーク帯域幅と接続速度に依存します。Oracle Solaris 11.1 リリースリポジトリをコピーする際、約 7G バイトのデータが転送されます。

ヒント-高いパフォーマンスを得るには、大量のメモリーを使用しているアプリケーションを閉じ、ZFS ストレージプールの容量が 80% 未満であることを確認してください。

プールの容量を表示するには `zpool list` コマンドを使用します。

```
$ zpool list
NAME      SIZE  ALLOC  FREE  CAP  DEDUP  HEALTH  ALTROOT
rpool    186G  75.2G  111G  40%  1.00x  ONLINE  -
```

次のコマンドは、`-s` オプションで指定したパッケージリポジトリからすべてのパッケージの最新バージョンを取得し、`-d` オプションで指定したリポジトリに入れます。詳細は、`pkgrecv(1)` のマニュアルページを参照してください。

```
$ pkgrecv -s http://pkg.oracle.com/solaris/release/ -d /export/repoSolaris11 '*'
Processing packages for publisher solaris ...
Retrieving and evaluating 4400 package(s)...
Download Manifests (4400/4400)
PROCESS                ITEMS          GET (MB)        SEND (MB)
developer/build/cmake  446/4400       332.1/4589.7   1000.2/14511.8
...
Completed                4400/4400     4589.7/4589.7  14511.8/14511.8
```

リポジトリがコピーされた後、プロセスは完了作業を実行します。「Completed」の行が表示された後、プロンプトが戻されるまで数分待ちます。

後でこのリポジトリを更新する場合、変更内容のみがコピーされ、プロセスに要する時間がずっと少なくなることがあります。

`pkgrecv` 処理が中断された場合、`-c` オプションを使用して、すでにダウンロードされたコンテンツを取得し、コンテンツのダウンロードを再開します。転送が中断された場合、次の例に示すように、`cache_dir` の値が情報メッセージ内で提供されます。

```
PROCESS                ITEMS          GET (MB)        SEND (MB)
...
pkgrecv: http protocol error: code: 503 reason: Service Unavailable
URL: 'http://pkg.oracle.com/solaris/release/file/file_hash

pkgrecv: Cached files were preserved in the following directory:
/var/tmp/pkgrecv-f0GaIg
Use pkgrecv -c to resume the interrupted download.
$ pkgrecv -c /var/tmp/pkgrecv-f0GaIg \
-s http://pkg.oracle.com/solaris/release/ -d /export/repoSolaris11 '*'
Processing packages for publisher solaris ...
Creating Plan
Retrieving and evaluating 156 package(s)...
PROCESS                ITEMS          GET (MB)        SEND (MB)
desktop/compiz         1/156          0/395.0         0/1100.2
```

ファイルからのリポジトリのコピー

このセクションでは、メディア上にあるか、または Oracle Solaris 11.1 ダウンロードサイトで入手可能なリポジトリファイルから、Oracle Solaris 11.1 リリースパッケージリポジトリのローカルコピーを作成する方法について説明します。

パッケージリポジトリファイルの取得

システムインストールイメージをダウンロードした場所と同じ場所から Oracle Solaris 11.1 IPS パッケージリポジトリ `.iso` ファイルをダウンロードするか、メディアパケットからリポジトリ DVD を見つけます。リポジトリは2つのファイルで構成され、合計で約7Gバイトです。

リポジトリ `.iso` ファイルに加えて、2つのファイルが提供されます。

- チェックサムファイル。ダウンロードページの上部にある「MD5 checksum」のリンクをクリックします。チェックサムは、2つのリポジトリファイルと、これら2つのファイルの連結用に提供されています。次のコマンドの出力と、チェックサムファイルの該当する値を比較して、ダウンロードに成功したことを確認します。

```
$ digest -a md5 iso_file
```

- README ファイル。README ファイルには、このセクションにある情報のほかに、リポジトリを USB または DVD メディアにコピーする方法などの追加情報が格納されています。

前のステップで作成したファイルシステムにリポジトリファイルをコピーします。これらのファイルを1つのファイルに連結します。

```
$ cat sol-11_1-repo-full.iso-a sol-11_1-repo-full.iso-b > \  
sol-11_1-repo-full.iso  
$ ls /export/repoSolaris11  
sol-11_1-repo-full.iso
```

リポジトリファイルのコンテンツを使用可能にする

リポジトリ `.iso` ファイルの内容を使用可能にします。

```
$ pfexec mount -F hsfs /export/repoSolaris11/sol-11_1-repo-full.iso /mnt  
$ ls /mnt  
COPYRIGHT  NOTICES  README  repo
```

`mount` コマンドからエラーメッセージを受け取った場合、`.iso` ファイルへの完全な絶対パスを指定したことを確認してください。

作業を確認します。

```
$ df -k /mnt
Filesystem                1K-blocks    Used Available Use% Mounted on
/export/repoSolaris11/sol-11_1-repo-full.iso  6778178  6778178      0 100% /mnt
```

リポジトリサーバーシステムが再起動するたびに .iso イメージを再マウントする必要があります。システムが再起動するたびに .iso を再マウントする必要があることを回避するには、次のセクションに記載されているようにリポジトリファイルをコピーします。

リポジトリファイルのコピー

リポジトリアクセスのパフォーマンスを向上させ、システムが再起動するたびに .iso イメージを再マウントする必要があることを回避するには、/mnt/repo/ から ZFS ファイルシステムにリポジトリファイルをコピーします。このコピーは rsync または tar を使用して実行できます。

- rsync コマンドを使用するときに、repo ディレクトリ内のファイルおよびサブディレクトリをコピーする場合は、/mnt/repo/ ではなく /mnt/repo/ (末尾のスラッシュ文字をつける) を指定するようにしてください。rsync(1) のマニュアルページを参照してください。

```
$ rsync -aP /mnt/repo/ /export/repoSolaris11
```

- 次の例に示す方法で tar コマンドを使用すると、マウント済みファイルシステムからリポジトリの ZFS ファイルシステムへのリポジトリの移動が速くなる場合があります。

```
$ cd /mnt/repo; tar cf - . | (cd /export/repoSolaris11; tar xfp -)
$ cd /export/repoSolaris11
```

作業を確認します。

```
$ ls /export/repoSolaris11
pkg5.repository          README
publisher                 sol-11_1-repo-full.iso
$ df -k /export/repoSolaris11
Filesystem                1K-blocks    Used Available Use% Mounted on
rpool/export/repoSolaris11 191987712 13733450 75787939 16% /export/repoSolaris11
```

イメージのアンマウント

イメージをアンマウントします。

```
$ pfexec umount /mnt
```

検索インデックスの構築とリポジトリのスナップショットの作成

リポジトリ作成コマンドは、デフォルトで検索インデックスを構築しません。クライアントがローカルリポジトリ内のパッケージを検索できるようにするには、次のコマンドを使用して、リポジトリ内のパッケージをカタログ化し、検索インデックスを更新します。

```
$ pkgrepo -s /export/repoSolaris11 refresh
Initiating repository refresh.
```

このパッケージリポジトリ用に別個の ZFS ファイルシステムを作成したので、そのリポジトリファイルシステムの ZFS スナップショットを作成できます。スナップショットがあれば、後日そのファイルシステムを簡単にロールバックしたり、そのファイルシステムをすばやくコピーしたりできます。リポジトリの内容を変更するときは必ず、リポジトリファイルシステムのスナップショットを作成するようにしてください。新しいリポジトリのスナップショットを作成するには、次のコマンドを使用します。

```
$ pfexec zfs snapshot rpool/export/repoSolaris11@initial
```

`zfs clone` コマンドを使用すると、リポジトリファイルシステムのコピーを作成できます。たとえば、あるリポジトリをこの元のパッケージセットで保守したり、新しいバージョンのパッケージで別のコピーを定期的に更新したりできます。

ZFS クローン、スナップショット、およびロールバックの詳細は、『[Oracle Solaris 11.1 の管理: ZFS ファイルシステム](#)』を参照してください。

リポジトリへのアクセスの提供

この章では、ファイルインタフェースを使用するか HTTP インタフェースを使用することによって、クライアントがローカルリポジトリ内のパッケージを取得できるようにする方法について説明します。両方のアクセスの種類について1つのリポジトリを設定できます。

ファイルインタフェースを使用したパッケージの取得

このセクションでは、ローカルネットワーク上のディレクトリからローカルリポジトリパッケージを提供する方法について説明します。

NFS 共有の構成

クライアントが NFS 経由でローカルリポジトリにアクセスできるようにするには、`sharenfs` プロパティを設定して共有を作成および公開します。

```
$ pfexec zfs create -o mountpoint=/export/repoSolaris11 rpool/repoSolaris11
$ pfexec zfs set share=name=s11repo,path=/export/repoSolaris11,prot=nfs rpool/repoSolaris11
name=s11repo,path=/export/repoSolaris11,prot=nfs
$ pfexec zfs set sharenfs=on rpool/repoSolaris11
```

次のいずれかのテストを使用して、共有が公開されたことを確認します。

- 共有ファイルシステムテーブル内のリポジトリを検索します。

```
$ grep repo /etc/dfs/sharetab
/export/repoSolaris11 s11repo nfs sec=sys,rw
```

- リポジトリがリモートシステムからアクセス可能かどうかを確認します。

```
$ dfshares solaris
RESOURCE          SERVER ACCESS  TRANSPORT
solaris:/export/repoSolaris11  solaris -      -
```

ファイルリポジトリ URI への発行元の起点の設定

クライアントシステムでローカルファイルリポジトリからパッケージを取得できるようにするには、発行元の起点を設定します。次のコマンドを使用して、新しいリポジトリ内にあるパッケージの発行元の名前を確認します。

```
$ pkgrepo info -s /export/repoSolaris11
PUBLISHER PACKAGES STATUS          UPDATED
solaris   4400    online          2012-07-25T23:40:03.496688Z
```

各クライアントで次のコマンドを実行して、solaris 発行元の起点をリセットします。

```
$ pfmexec pkg set-publisher -G '*' -M '*' -g /net/host1/export/repoSolaris11/ solaris
-G '*'    solaris 発行元についての既存の起点をすべて削除します。
-M '*'    solaris 発行元についての既存のミラーをすべて削除します。
-g       新しく追加されたローカルリポジトリの URI を solaris 発行元の新しい起点として追加します。
```

発行元の構成方法の詳細は、『[Oracle Solaris 11.1 ソフトウェアパッケージの追加および更新](#)』の「[発行元の構成](#)」を参照してください。

HTTP インタフェースを使用したパッケージの取得

このセクションでは、パッケージ集積サーバーを使用してローカルリポジトリパッケージを提供する方法について説明します。

異なるポートで実行されている複数の pkg.depotd デーモンを使用して複数のリポジトリを提供する方法については、[27 ページ](#)の「[複数の集積サーバーインスタンスを使用した複数のリポジトリの提供](#)」を参照してください。異なるプレフィックスを使用して1つのドメイン名で複数のリポジトリを実行する方法については、[32 ページ](#)の「[1つのドメインでの複数リポジトリ](#)」を参照してください。

リポジトリサーバーサービスの構成

クライアントが HTTP 経由でローカルリポジトリにアクセスできるようにするには、application/pkg/server サービス管理機能 (SMF) サービスを有効にします。

```
$ svccfg -s application/pkg/server setprop pkg/inst_root=/export/repoSolaris11
$ svccfg -s application/pkg/server setprop pkg/readonly=true
```

作業を確認します。

```
$ svcprop -p pkg/inst_root application/pkg/server
/export/repoSolaris11
```

リポジトリをクライアントに提供するには、`pkg.depotd` を使用します。デフォルトでは、`pkg.depotd` はポート 80 で接続を待機します。`pkg/port` プロパティをリセットすることでポートを変更できます。

```
$ svccfg -s application/pkg/server setprop pkg/port=port_number
```

`application/pkg/server` プロパティの完全なリストについては、`pkg.depotd(1M)` のマニュアルページを参照してください。

複数のサービスプロパティを設定するには、次のコマンドを使用してすべてのプロパティを一度に編集します。

```
$ svccfg -s pkg/server editprop
```

変更するすべての行の先頭にあるコメントマーカー (#) を削除してください。

リポジトリサービスの開始

`pkg.depotd` リポジトリサービスを再起動します。

```
$ svcadm refresh application/pkg/server
$ svcadm enable application/pkg/server
```

リポジトリサービスが動作しているかどうかを確認するには、`localhost` の場所を指定してブラウザウィンドウを開きます。デフォルトでは、`pkg.depotd` はポート 80 で接続を待機します。ポートを変更した場合、`localhost:port_number` の場所を指定してブラウザウィンドウを開きます。

HTTP リポジトリ URI への発行元の起点の設定

クライアントシステムでローカルファイルリポジトリからパッケージを取得できるようにするには、発行元の起点を設定します。次のコマンドを使用して、新しいリポジトリ内にあるパッケージの発行元の名前を確認します。

```
$ pkgrepo info -s /export/repoSolaris11
PUBLISHER PACKAGES STATUS          UPDATED
solaris   4400      online          2012-07-25T23:40:03.496688Z
```

各クライアントで次のコマンドを実行して、`solaris` 発行元の起点をリセットします。

```
$ pfxexec pkg set-publisher -G '*' -M '*' -g http://localhost:port_number/ solaris
```

- G '*' solaris 発行元についての既存の起点をすべて削除します。
- M '*' solaris 発行元についての既存のミラーをすべて削除します。
- g 新しく追加されたローカルリポジトリの URI を solaris 発行元の新しい起点として追加します。

発行元の構成方法の詳細は、『[Oracle Solaris 11.1 ソフトウェアパッケージの追加および更新](#)』の「[発行元の構成](#)」を参照してください。

ローカル IPS パッケージリポジトリの保守

この章では、IPS リポジトリ内のパッケージを更新する方法、リポジトリのプロパティを設定または更新する方法、および別のソースからパッケージをリポジトリに追加する方法について説明します。

この章では、キャッシュおよび負荷分散を含む集積サーバーの Apache 構成についても説明します。

ローカルリポジトリの更新

新しいパッケージをローカルリポジトリに転送する前に、コピーする予定のパッケージが構築される対象のバージョンと同じかそれよりも新しい Oracle Solaris 11 OS をリポジトリサーバーが実行しているようにします。たとえば、サーバーが Oracle Solaris 11 OS を実行していて、リポジトリを Oracle Solaris 11.1 リポジトリに更新する場合、リポジトリを更新する前にサーバーを Oracle Solaris 11.1 に更新します。システムの更新については、『[Oracle Solaris 11.1 ソフトウェアパッケージの追加および更新](#)』の「[イメージの更新](#)」を参照してください。

ローカル IPS パッケージリポジトリを作成する際、`pkgrecv` コマンドまたは `.iso` ファイルのいずれを使用した場合であっても、リポジトリを更新するには `pkgrecv(1)` コマンドを使用します。更新されたパッケージのみが更新されます。[13 ページ](#) の「[リポジトリのコピー](#)」にあるパフォーマンスのヒントを参照してください。

リポジトリの内容を変更する前に、リポジトリファイルシステムのスナップショットを作成するようにしてください。そのスナップショットにわかりやすい名前 (SRU (Support Repository Update) 番号を含めるなど) を付けてください。ZFS のスナップショットとロールバックの詳細は、『[Oracle Solaris 11.1 の管理: ZFS ファイルシステム](#)』を参照してください。

```
$ pkgrecv -s http://pkg.oracle.com/solaris/support/ -d /export/repoSolaris11 \  
--key /var/pkg/ssl/Oracle_Solaris_11_Support.key.pem \  
--cert /var/pkg/ssl/Oracle_Solaris_11_Support.certificate.pem '*'
```

この更新を定期的に行う場合、PKG_SRC および PKG_DEST 環境変数を使用する場合もあります。

```
$ export PKG_SRC=http://pkg.oracle.com/solaris/support/
$ export PKG_DEST=/export/repoSolaris11
$ pkgrecv '*'
```

リポジトリを更新した後、次のコマンドを実行して、リポジトリ内で検出された新しいパッケージをカタログ化し、すべての検索インデックスを更新します。

```
$ pkgrepo rebuild -s /export/repoSolaris11
```

HTTP インタフェースを介してリポジトリを提供する場合は、SMF サービスを再起動します。

```
$ svcadm restart application/pkg/server:default
```

複数のリポジトリを作成した場合は、サービスを再起動するときに適切なサービスインスタンスを必ず指定するようにしてください。

リポジトリプロパティの確認および設定

このセクションでは IPS リポジトリに関する情報を表示する方法と、リポジトリおよび発行元プロパティを設定する方法について説明します。[pkgrepo\(1\)](#) マニュアルページを参照してください。

次のコマンドは、ローカルリポジトリによって認識されているパッケージ発行元の一覧を表示します。STATUS 列を見れば、発行元のパッケージデータが現在処理中であるかどうかを判別できます。

```
$ pkgrepo info -s /export/repoSolaris11
PUBLISHER PACKAGES STATUS          UPDATED
solaris   4400    online          2012-07-25T23:40:03.496688Z
```

次のコマンドは、ローカルリポジトリに関するプロパティ情報を表示します。

```
$ pkgrepo get -s /export/repoSolaris11
SECTION  PROPERTY  VALUE
publisher prefix    solaris
repository description This\ repository\ serves\ a\ copy\ of\ the\ Oracle\ Solaris\ 11.1\
Package\ Repository.
repository name    Oracle\ Solaris\ 11.1\ Package\ Repository
repository version 4
```

発行元のプレフィックスの値は、次の場合に `solaris` が使用されることを指定しています。

- 複数の発行元のパッケージが存在し、`pkg` コマンドのパッケージ名に発行元が指定されない場合
- パッケージがリポジトリに公開されて、発行元が指定されない場合

バージョン4リポジトリはデフォルトで作成されます。バージョン4リポジトリは複数の発行元についてのパッケージの保管をサポートします。

`set` サブコマンドを使用して、新しいプロパティ値を指定します。

```
$ pkgrepo set -s /export/repoSolaris11 \
repository/description="Local copy of the Oracle Solaris 11.1 repository" \
repository/name="Oracle Solaris 11.1 Package Repository"
$ pkgrepo get -s /export/repoSolaris11
SECTION  PROPERTY  VALUE
publisher prefix      solaris
repository description Local\ copy\ of\ the\ Oracle\ Solaris\ 11.1\ repository
repository name      Oracle\ Solaris\ 11.1\ Package\ Repository
repository version   4
```

次のコマンドは、ローカルリポジトリ内の `solaris` 発行元に関するプロパティ情報を表示します。丸括弧は、特定の値が値のリストである可能性があることを示しています。複数のローカルリポジトリを作成する場合は、構成中にリポジトリ名、パス、ポート番号、および Apache プロキシベースを必ず変更するようにしてください。31 ページの「プレフィックスを使用した単純なプロキシ構成」には `pkg/proxy_base` の設定方法が記載されています。

```
$ pkgrepo get -p solaris -s /export/repoSolaris11
PUBLISHER SECTION  PROPERTY  VALUE
solaris publisher alias
solaris publisher prefix      solaris
solaris repository collection-type core
solaris repository description Local\ copy\ of\ the\ Oracle\ Solaris\ 11.1\ repository
solaris repository legal-uris ()
solaris repository mirrors   ()
solaris repository name      Oracle\ Solaris\ 11.1\ Package\ Repository
solaris repository origins   ()
solaris repository refresh-seconds ""
solaris repository registration-uri ""
solaris repository related-uris ()
```

`collection-type` `core` コレクションタイプは、このリポジトリにはリポジトリ内のパッケージによって宣言されたすべての依存関係が含まれていることを示します。

`legal-uris` `legal-uris` は、リポジトリに関する法的情報を提供するドキュメントの場所のリストです。

`origins` `origins` は、このリポジトリのパッケージのメタデータとコンテンツの完全なコピーを含むリポジトリの場所のリストです。

`related-uris` `related-uris` は、ユーザーが関心を持っている可能性があるパッケージを含むリポジトリの場所のリストです。

ほかの発行元およびリポジトリプロパティの説明については、[pkgrepo\(1\)](#) のマニュアルページを参照してください。

次のコマンドは、`pkg.oracle.com` リポジトリ内の指定された `section/property` に関する情報を表示します。

```
$ pkgrepo get -p solaris -s http://pkg.oracle.com/solaris/release \
repository/name repository/description
PUBLISHER SECTION PROPERTY VALUE
solaris repository description Local\ copy\ of\ the\ Oracle\ Solaris\ 11.1\ repository
solaris repository name Oracle\ Solaris\ 11.1\ Package\ Repository
```

ローカルリポジトリ内の `solaris` 発行元には、リポジトリの説明とリポジトリ名のプロパティ値が設定されないことに注意します。発行元のプロパティ値を設定するには、上記のように `set` サブコマンドを使用し、発行元名も指定します。発行元の `repository/name` の値は、ブラウザインタフェースのページ先頭付近およびページタイトルとして表示されます。発行元の `repository/description` の値は、ブラウザインタフェース内で名前のすぐ下にあるバージョン情報のセクションに表示されません。

```
$ pkgrepo set -p solaris -s /export/repoSolaris11 \
repository/description="Local copy of the Oracle Solaris 11.1 repository" \
repository/name="Oracle Solaris 11.1 Package Repository"
$ pkgrepo get -p solaris -s /export/repoSolaris11
PUBLISHER SECTION PROPERTY VALUE
solaris publisher alias
solaris publisher prefix solaris
solaris repository collection-type core
solaris repository description Local\ copy\ of\ the\ Oracle\ Solaris\ 11.1\ repository
solaris repository legal-uris ()
solaris repository mirrors ()
solaris repository name Oracle\ Solaris\ 11.1\ Package\ Repository
solaris repository origins ()
solaris repository refresh-seconds
solaris repository registration-uri ""
solaris repository related-uris ()
```

ローカルリポジトリのカスタマイズ

ソースリポジトリのサブセットとなるリポジトリを作成できます。次のコマンドは、`group/feature/amp` パッケージのすべてのバージョンとそのバージョンのすべての依存関係を `amprepo` リポジトリにコピーします。`amprepo` リポジトリは、`pkgrepo create` コマンドを使用して事前に作成されたものです。

```
$ pkgrecv -s http://pkg.oracle.com/solaris/release/ -d /export/amprepo \
-m all-versions -r group/feature/amp
```

別の発行元からパッケージをリポジトリに追加できます。次の `pkgrecv` コマンドは、`ISVproducts.p5p` パッケージアーカイブのすべてのパッケージをローカルリポジトリに追加します。`pkg list` 出力に発行元が表示されます。これは、この発行元が、このイメージ内の検索順序で最上位にランクされている発行元でないためです。

```
$ pkg list -g /tmp/ISVproducts.p5p
NAME (PUBLISHER)      VERSION      IFO
isvtool (isvpub)     1.0         ---
$ pkgrecv -s /tmp/ISVproducts.p5p -d /export/repoSolaris11 '*'
Processing packages for publisher isvpub ...
Retrieving and evaluating 1 package(s)...
PROCESS      ITEMS      GET (MB)      SEND (MB)
Completed    1/1        0.0/0.0       0.0/0
$ pkg list -g /export/repoSolaris11 isvtool
NAME (PUBLISHER)      VERSION      IFO
isvtool (isvpub)     1.0         ---
```

パッケージアーカイブは、発行元の情報と、その発行元によって提供された1つ以上のパッケージを含むファイルです。『Oracle Solaris 11.1 での Image Packaging System を使用したソフトウェアのパッケージ化および配布』の「パッケージアーカイブとして発行する」を参照してください。

複数の集積サーバーインスタンスを使用した複数のリポジトリの提供

このセクションでは、20 ページの「HTTP インタフェースを使用したパッケージの取得」で提供されている情報を拡張して、同じリポジトリサーバー上の異なるポートで実行されている複数の `pkg.depotd` デーモンを使用して複数のリポジトリを提供できるようにする方法について説明します。

この例では、`repoSolaris11` リポジトリのほかに `dev_repo` リポジトリが存在しています。`repoSolaris11` リポジトリには、ポート 80 を使用して `http://localhost/` からアクセスできます。

発行元のプレフィックスが `dev_repo` リポジトリに設定されているようにします。

```
$ pkgrepo set -s /export/dev_repo publisher/prefix=dev
pkg/server サービスの新しいインスタンスを追加します。
$ svccfg -s pkg/server add dev
$ svccfg -s pkg/server:dev setprop pkg/port=81
$ svccfg -s pkg/server:dev setprop pkg/inst_root=/export/dev_repo
```

新しいインスタンスが追加されているかどうか確認します。

```
$ svccfg -s pkg/server list
:properties
default
dev
```

新しいサービスを起動します。

```
$ svcadm refresh application/pkg/server:dev
$ svcadm enable application/pkg/server:dev
```

<http://localhost:81/> でリポジトリを参照します。

異なるプレフィックスを使用して1つのドメイン名で複数のリポジトリを実行する方法については、[32 ページの「1つのドメインでの複数リポジトリ」](#)を参照してください。

集積サーバーの Apache 構成

このセクションでは、次のメリットを得るために Web サーバーインスタンスの背後で集積サーバーを実行することについて説明します。

- コンテンツキャッシュおよび負荷分散によってパフォーマンスを向上します。
- 1つのドメイン名で複数のリポジトリのホスティングを可能にします。

集積サーバー用のキャッシュの構成

キャッシュプロキシの背後に集積サーバーを設定するには最低限の構成が必要です。後で説明するカタログ属性ファイルとリポジトリ検索結果を除けば、提供するすべてのファイルは固有であるため、必要な場合は無制限にキャッシュしても安全です。また、キャッシュ内のファイルが誤って無効にならないようにするために、すべての集積応答には適切な HTTP ヘッダーが含まれています。

Apache をキャッシュプロキシとして構成することについては、Apache の [Caching Guide](#) を参照してください。

CacheRoot ディレクティブを使用して、キャッシュされたファイルを格納するディレクトリを指定します。指定されたディレクトリが Apache プロセスによって書き込みできることを確認してください。Apache がこのディレクトリに書き込みできない場合、明示的なエラーメッセージは出力されません。

```
CacheRoot /tank/proxycache
```

Apache では、特定のディレクトリについてのキャッシュを有効にできます。次のディレクティブに示すように、リポジトリサーバーはおそらくサーバーのすべてのコンテンツをキャッシュすることが必要になります。

```
CacheEnable disk /
```

CacheMaxFileSize ディレクティブを使用して、キャッシュするファイルの最大サイズを設定します。Apache のデフォルトである 1M バイトは、ほとんどのリポジトリで小さすぎる可能性があります。次のディレクティブでは、キャッシュするファイルの最大サイズを 1G バイトに設定します。

```
CacheMaxFileSize 1000000000
```

基礎となるファイルシステムについて最適なパフォーマンスを得るには、ディスク上のキャッシュのディレクトリ構造を調節します。ZFS データセットでは、ディレクトリレベルが複数ある方が、1つのディレクトリ内のファイル数よりもパフォーマンスに影響を及ぼします。したがって、ディレクトリレベルを1つにして、各ディレクトリ内に多数のファイルを持たせるように構成します。ディレクトリ構造を制御するには CacheDirLevels および CacheDirLength ディレクティブを使用します。CacheDirLevels を 1 に設定します。CacheDirLength の値は、ディレクトリ数とディレクトリあたりのファイル数がうまく均衡するように設定します。次のように値を 2 に設定すると、4096 個のディレクトリが生成されます。詳細については、Apache の [Disk-based Caching](#) のドキュメントを参照してください。

```
CacheDirLevels 1
CacheDirLength 2
```

カタログ属性ファイルに対するキャッシュの考慮事項

リポジトリカタログ属性ファイル (catalog.attrs) には、リポジトリカタログの現行ステータスが含まれています。このファイルは、キャッシュを正当化するほどの十分な大きさになることがあります。ただし、バックエンドリポジトリのカタログが変更されると、このファイルは無効になります。次の2つの方法のいずれかを使用して、この問題に対処することができます。

- このファイルをキャッシュしません。この対処方法は、追加トラフィックが重要な考慮事項とならない高帯域環境においてリポジトリサーバーが実行する場合に最適です。次に示す httpd.conf ファイルの一部は、catalog.attrs ファイルをキャッシュしないことを指定する方法を示しています。

```
<LocationMatch ".*/catalog.attrs">
    Header set Cache-Control no-cache
</LocationMatch>
```

- バックエンドリポジトリのカタログが更新されるたびにこのファイルをキャッシュから除去します。

検索に対するキャッシュの考慮事項

パッケージリポジトリを検索すると、要求に基づくカスタム応答が生成されません。したがって、検索結果はキャッシュにあまり適していません。集積サーバーは、検索結果がキャッシュ内で無効にならないように適切な HTTP ヘッダーを設定します。ただし、キャッシュ作成からの帯域幅の節約量は小さいことが予想されます。次に示す httpd.conf ファイルの一部は、検索結果をキャッシュしないように指定する方法を示しています。

```
<LocationMatch ".*search/\d/.*">
    Header set Cache-Control no-cache
</LocationMatch>
```

Web プロキシの背後での集積サーバーの実行

pkg(5) 集積サーバーによって、ローカルネットワーク内またはインターネット上のリポジトリへのアクセスを簡単に提供できます。ただし、集積サーバーは、1つのドメイン名または詳細なプレフィックスで複数のリポジトリを提供することをサポートしません。1つのドメイン名で複数のリポジトリをホストするには、Web プロキシの背後に集積サーバーを実行します。Web プロキシの背後に集積サーバーを実行すると、複数の集積の間での負荷分散を有効にしたり、コンテンツキャッシュを有効にしたりすることによって、サーバーのパフォーマンスを向上することもできます。

このセクションの例では、Apache Web サーバーをプロキシソフトウェアとして使用します。Oracle Solaris 11.1 OS では、web/server/apache-22 パッケージに Apache Web サーバーが含まれ、/etc/apache2/2.2 に基本的な httpd.conf ファイルが含まれています。Apache Web サーバーをアクティブ化するには、svc:/network/http:apache22 サービスを有効にします。詳細は、[Apache HTTP サーババージョン 2.2 ドキュメント](#)を参照してください。

これらの例で示す原則を、任意のプロキシサーバーソフトウェアに適用できるようにする必要があります。

一般的に推奨される Apache 構成設定

次の設定はパフォーマンスとセキュリティーに影響します。

Apache DEFLATE フィルタを有効にします。

HTTP クライアントはサーバーに対し、HTTP 要求内の圧縮されたデータを受け入れることを通知できます。Apache DEFLATE フィルタを有効にすると、カタログやマニフェストなどのメタデータの有線送信サイズを著しく削減できます。カタログやマニフェストなどのメタデータは、通常 90% 圧縮されます。

```
AddOutputFilterByType DEFLATE text/html application/javascript text/css text/plain
```

エンコードされたスラッシュをデコードしません。

パッケージは URL エンコードされたスラッシュを含むことができます。これらのスラッシュがディレクトリ区切り文字として解釈されないようにするには、これらをデコードしないように Apache に指示します。

```
AllowEncodedSlashes NoDecode
```

注 - この設定を省略すると、検索機能に著しいマイナスの影響が生じます。

パイプライン要求の増加を許可します。

`MaxKeepAliveRequests` の値を増加することで、クライアントが接続をクローズすることなく多数のパイプライン要求を作成できるようになります。Apache のデフォルトの 100 では少なすぎます。

```
MaxKeepAliveRequests 10000
```

応答の最大待機時間を設定します。

プロキシタイムアウトは、バックエンド集積からの応答を Apache が待機する時間を設定します。ほとんどの操作では、30 秒あれば十分です。結果の件数が非常に多くなる検索は、時間がかかり長くなる可能性があります。このような検索に対応するために、タイムアウト値を大きくする場合もあります。

```
ProxyTimeout 30
```

フォワードプロキシを無効にします。

フォワードプロキシが無効になっていることを確認します。

```
ProxyRequests Off
```

Apache 構成の例

このセクションでは、複数のリポジトリを使用した、負荷分散されていない設定と負荷分散された設定を示します。

プレフィックスを使用した単純なプロキシ構成

この例では、負荷分散されていない集積サーバーの基本構成を示します。この例では `http://pkg.example.com/myrepo` を `internal.example.com:10000` に接続します。

この例で説明していない、必要なその他のプロパティの設定については、[27 ページの「複数の集積サーバーインスタンスを使用した複数のリポジトリの提供」](#)を参照してください。

集積サーバーは、集積サーバーへのアクセスが可能な URL を指定する `pkg/proxy_base` 設定を使用して構成する必要があります。 `pkg/proxy_base` を設定するには次のコマンドを使用します。

```
$ svccfg -s pkg/server add repo
$ svccfg -s pkg/server:repo "setprop pkg/proxy_base = astring: http://pkg.example.com/myrepo"
$ svcadm refresh pkg/server:repo
$ svcadm enable pkg/server:repo
```

ネットワーク処理を実行するとき、pkg(5)クライアントは集積サーバーに対して20個の並列接続を開きます。集積スレッドの数が、任意の時点で予想されるサーバーへの接続と一致するようにします。集積ごとのスレッド数を設定するには次のコマンドを使用します。

```
$ svccfg -s pkg/server:repo "setprop pkg/threads = 200"
$ svcadm refresh pkg/server:repo
$ svcadm restart pkg/server:repo
```

URLの正規化を抑制するにはnocanonを使用します。この設定は検索を正しく機能させる上で重要です。また、バックエンド接続の数は、集積サーバーが提供するスレッド数に制限します。次に示すhttpd.confファイルの一部は、1つの集積サーバーのプロキシを設定する方法を示しています。

```
Redirect /myrepo http://pkg.example.com/myrepo/
ProxyPass /myrepo/ http://internal.example.com:10000/ nocanon max=200
```

1つのドメインでの複数リポジトリ

集積サーバーをプロキシの背後で実行するもっとも重要な理由は、1つのドメイン名において異なるプレフィックスを使用して複数のリポジトリを簡単に実行することです。31ページの「プレフィックスを使用した単純なプロキシ構成」の例は、複数のリポジトリをサポートするように簡単に拡張できます。

次の例では、1つのドメイン名の3種類のプレフィックスが3種類のパッケージリポジトリに接続されます。

- http://pkg.example.com/repo_one は internal.example.com:10000 に接続されます
- http://pkg.example.com/repo_two は internal.example.com:20000 に接続されます
- http://pkg.example.com/xyz/repo_three は internal.example.com:30000 に接続されます

pkg(5)集積サーバーはSMF管理対象サービスです。したがって、同じホスト上で複数の集積サーバーを実行するには、単純に新しいサービスインスタンスを作成します。

```
$ svccfg -s pkg/server add repo1
$ svccfg -s pkg/server:repo1 setprop pkg/property=value
$ ...
```

前の例と同様、各集積サーバーは200個のスレッドを実行します。

```
Redirect /repo_one http://pkg.example.com/repo_one/
ProxyPass /repo_one/ http://internal.example.com:10000/ nocanon max=200
```

```
Redirect /repo_two http://pkg.example.com/repo_two/
ProxyPass /repo_two/ http://internal.example.com:20000/ nocanon max=200

Redirect /xyz/repo_three http://pkg.example.com/xyz/repo_three/
ProxyPass /xyz/repo_three/ http://internal.example.com:30000/ nocanon max=200
```

負荷分散の構成

集積サーバーを Apache ロードバランサの背後で実行させることが必要な場合もあります。この例では `http://pkg.example.com/myrepo` を `internal1.example.com:10000` および `internal2.example.com:10000` に接続します。

31 ページの「プレフィックスを使用した単純なプロキシ構成」で示すように、適切な `proxy_base` 設定を使用して集積サーバーを構成します。

バックエンド接続の数は、ロードバランサ設定において各集積が実行するスレッド数を集積の数で割った数に制限します。それ以外の場合、Apache は 1 つの集積に対して使用可能な数よりも多くの接続を開くことで接続が停止し、パフォーマンスを低下させる可能性があります。 `max=` パラメータを使用して、各集積への並列接続の最大数を指定します。次の例では、それぞれ 200 個のスレッドを実行する 2 つの集積を示します。集積のスレッド数を設定する方法の例については、31 ページの「プレフィックスを使用した単純なプロキシ構成」を参照してください。

```
<Proxy balancer://pkg-example-com-myrepo>
  # depot on internal1
  BalancerMember http://internal1.example.com:10000 retry=5 max=100

  # depot on internal2
  BalancerMember http://internal2.example.com:10000 retry=5 max=100
</Proxy>

Redirect /myrepo http://pkg.example.com/myrepo/
ProxyPass /myrepo/ balancer://pkg-example-com-myrepo/ nocanon
```

完全な負荷分散の例

次の例は、負荷分散に対応した集積サーバーおよび負荷分散に対応しない集積サーバーの設定をホストするリポジトリサーバーのために、`httpd.conf` ファイルに追加する必要があるすべてのディレクティブを含んでいます。

次の例では、1 つのドメイン名の 2 種類のプレフィックスが 3 種類のパッケージリポジトリに接続されます。

- `http://pkg.example.com/repo_one` は `internal1.example.com:10000` および `internal2.example.com:10000` に接続されます
- `http://pkg.example.com/repo_two` は `internal1.example.com:20000` に接続されます

```
AddOutputFilterByType DEFLATE text/html application/javascript text/css text/plain
AllowEncodedSlashes NoDecode
```

```
MaxKeepAliveRequests 10000

ProxyTimeout 30

ProxyRequests Off

<Proxy balancer://pkg-example-com-repo_one>
    # depot on internal1
    BalancerMember http://internal1.example.com:10000 retry=5 max=100

    # depot on internal2
    BalancerMember http://internal2.example.com:10000 retry=5 max=100
</Proxy>

Redirect /repo_one http://pkg.example.com/repo_one/
ProxyPass /repo_one/ balancer://pkg-example-com-repo_one/ nocanon
Redirect /repo_two http://pkg.example.com/repo_two/
ProxyPass /repo_two/ http://internal.example.com:20000/ nocanon max=200
```