Oracle® Solaris 11.2 パッケージリポジトリのコピーと作成

Part No: E53761-02
2014 年 9 月

ORACLE

目次

例目次

このドキュメントの使用法

- **概要** – 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の作成、コピー、アクセスできるようにするための設定、更新、および保守を行う方法を説明します。
- **対象読者** – 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。
- **必要な知識** – Oracle Solaris Service Management Facility (SMF) の経験、および NFS と Web サーバーの管理経験。

製品ドキュメントライブラリ

この製品に関する最新情報および既知の問題については、ドキュメントライブラリ (http://www.oracle.com/pls/topic/lookup?ctx=E56342) に記載されています。

Oracle サポートへのアクセス

Oracle ユーザーは My Oracle Support から電子サポートにアクセスできます。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。

フィードバック

このドキュメントに関するフィードバックを http://www.oracle.com/goto/docfeedback からお聞かせください。

♦ ♦ ♦ 第 1 章

1 Image Packaging System のパッケージリポジトリ

Oracle Solaris 11 ソフトウェアは、Image Packaging System (IPS) パッケージで配布されます。IPS パッケージは、IPS パブリッシャーが提供する 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、HTTP または HTTPS インタフェースを提供できます。また、このガイドでは、リポジトリを自動的に更新する方法、およびリポジトリをクローニングする方法を説明し、Apache Web サーバーの構成 (キャッシュ、負荷分散、HTTPS アクセスの構成など) を示します。

この章では、次の内容について説明します。

- ローカルの IPS パッケージリポジトリを内部使用のために作成する理由
- パッケージリポジトリを作成するためのベストプラクティス
- リポジトリをホストするためのシステム要件

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ー。**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。
- **変更管理。**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。システムを更新できるバージョンを簡単に管理したい。
- **カスタムパッケージ。**カスタム IPS パッケージを配布したい。

ローカルの IPS パッケージリポジトリの作成および使用のベストプラクティス

次のベストプラクティスを利用して、リポジトリの可用性を維持し、エラーを最小限にします。

すべての Support Repository Update (SRU) のすべての内容を含めます。

ローカルリポジトリをすべてのサポート更新で常に最新にします。サポート更新には、セキュリティー更新およびその他の重要な修正が含まれています。Oracle Solaris OS のパッケージリポジトリの各マイナーリリースおよび更新は、フルセットのパッケージとしてリリースされます。SRU は、変更されたパッケージのみの疎アップデートとしてリリースされます。

- パッケージのサブセットをサポート更新からリポジトリに追加しないでください。サポート更新のすべての内容をローカルリポジトリに追加します。
- サポート更新をスキップしないでください。適用可能なすべてのサポート更新を各リポジトリに適用してください。
- Oracle パブリッシャーによって配布されたパッケージを削除しないでください。
- `svc:/application/pkg/mirror` サービス管理機能 (SMF) サービスを使用して、Oracle サポートリポジトリからローカルのマスターリポジトリを自動的に更新します。手順については、23 ページの「インターネットからリポジトリを自動的にコピーする方法」を参照してください。

リポジトリ内の最新のバージョンより前のバージョンに更新するには、インストールする `entire` incorporation パッケージのバージョンを指定します。『Oracle Solaris 11.2 ソフトウェアの追加と更新 』の第 4 章「Oracle Solaris イメージの更新またはアップグレード」を参照してください。

リポジトリを更新するたびに検証します。

リポジトリの内容またはプロパティーの値を変更するたびに、`pkgrepo verify` コマンドを使用します。`pkgrepo verify` コマンドは、リポジトリの内容の次の属性が正しいことを検証します。

- ファイルのチェックサム。
- ファイルのアクセス権。`pkg5srv` ユーザーがリポジトリの内容を読み取ることができるように、リポジトリファイル、ディレクトリ、およびリポジトリへのパスがチェックされます。
- パッケージマニフェストのアクセス権。
- パッケージマニフェストの内容。
- パッケージ署名。

リポジトリを共有の場所に作成します。

共有の場所とは、どのブート可能環境 (BE) にも含まれていない場所です。共有の場所の例として、`/var/share` および `/export` があります。リポジトリを共有の場所に作成すると、次の利点があります。

- リポジトリをほかの既存の BE から簡単に利用できます。
- アップグレードまたは既存の BE をクローニングして新しい BE を作成した場合、リポジトリに複数のコピーがあることによって領域が無駄に使用されることはありません。
- ほかの BE に適用したリポジトリ更新を再適用するために、時間と入出力リソースを浪費することはありません。

非大域ゾーンを使用している場合、非大域ゾーンに構成されているパブリッシャーのすべての場所は、そのパブリッシャーが大域ゾーンに構成されていなくても、大域ゾーンからアクセスできる必要があります。

各リポジトリを独自の ZFS ファイルシステムに作成します。

別個の ZFS ファイルシステムを使用すると、次のことが可能になります。

- 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。たとえば、リポジトリを更新する場合、パフォーマンスの改善のために `atime` を `off` に設定します。`atime` プロパティーは、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。このプロパティーをオフにすると、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。
- リソースの使用量の管理。大型のリポジトリの更新によってプール内のすべての領域が消費されないように、各リポジトリデータセットに適切なディスク割り当てを指定します。このベストプラクティスは、23 ページの「インターネットからリポジトリを自動的にコピーする方法」に記述されているように更新を自動的に実行している場合に特に重要です。
- スナップショットの作成。

リポジトリを更新するたびにスナップショットを作成します。

次の利点を得られるように、リポジトリを更新するたびにリポジトリのファイルシステムのスナップショットを作成します。

- スナップショットから以前のバージョンのリポジトリにロールバックします。
- ユーザーへの影響を最小化するために、スナップショットからリポジトリを更新します。

高可用性を提供します。

- リポジトリのクローンを別の場所に保持します。手順については、39 ページの「複数の同一のローカルリポジトリの保守」を参照してください。

- キャッシュ、負荷分散、および複数リポジトリの提供を実行する Web サーバーを構成します。詳細については、第5章「Web サーバーの背後での集積サーバーの実行」を参照してください。

ローカルリポジトリをセキュアにします。

手順については、60 ページの「HTTPS でのリポジトリアクセスの構成」を参照してください。

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

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 11.2 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 16G バイトの空き容量が必要です。

ベストプラクティスとしては、すべてのサポート更新によってローカルリポジトリを最新にしているため、サポート更新のために 10-15G バイトの追加領域を毎年使用するように計画します。その他のソフトウェア (Oracle Solaris Studio、Oracle Solaris Cluster など) も、パッケージリポジトリで追加の領域を必要とします。

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。プロファイルおよび役割の詳細 (必要なプロファイルまたは役割を判別する方法を含む) は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』を参照してください。

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。次のプロファイルは、ローカルのパッケージリポジトリを保守するために役立ちます。

ZFS ファイルシステム管理

この権利プロファイルでは、ユーザーは `zfs` コマンドを実行できます。

ソフトウェアのインストール

この権利プロファイルでは、ユーザーは `pkg` コマンドを実行できます。

サービス管理

この権利プロファイルでは、ユーザーは SMF コマンド (`svccfg` など) を実行できます。

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` 役割を持っている場合は、`su` コマンドと `root` パスワードを使用して、`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ーポリシーに応じて、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。

♦ ♦ ♦ 第 2 章

2 IPS パッケージリポジトリのコピー

この章では、Oracle Solaris 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 つの方法を説明します。メディアまたは Oracle Solaris ダウンロードサイトのリポジトリファイルを使用するか、リポジトリの内容をインターネットから手動または自動で取得できます。どの場合でも、最初にローカルのパッケージリポジトリ用に個別の ZFS ファイルシステムを共有の場所に作成します。リポジトリを作成したら、リポジトリを検証してスナップショットを作成します。

この章では、リポジトリのコピーに関連するパフォーマンスおよびトラブルシューティングの情報も提供します。

リポジトリのコピーのパフォーマンスに関する考慮事項

Oracle Solaris ダウンロードサイトからリポジトリファイルをダウンロードする場合、または22 ページの「インターネットからのリポジトリのコピー」に示す `pkgrecv` コマンドを使用してインターネット上の場所からリポジトリの内容を取得する場合は、転送のパフォーマンス改善のため、次の構成を検討してください。

- 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ます。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。
- プロキシを使用している場合は、プロキシのパフォーマンスを確認します。
- 大量のメモリーを使用するアプリケーションを閉じます。
- 一時ディレクトリ内に十分な空き領域があることを確認します。`pkgrecv` コマンドは、操作中に、`$TMPDIR` を一時ストレージのディレクトリとして使用します。`TMPDIR` が設定されていない場合、`pkgrecv` はこの一時ストレージに `/var/tmp` を使用します。実行している `pkgrecv` 操作のサイズに対する十分な空き領域が、`$TMPDIR` または `/var/tmp` にあることを確認します。
- `pkgrecv` コマンドを使用して大きなリポジトリをコピーしている場合は、`--clone` オプションの使用を検討してください。`--clone` オプションを使用すると、操作がより速くなり、メモリー

使用量が少なくなります。40 ページの「ローカルの IPS パッケージリポジトリをクローニングする方法」を参照してください。

- pkgrecv コマンドを使用して大型リポジトリを作成または更新する場合は、宛先リポジトリ用に SSD を使用することを検討してください。パッケージの取得が完了したら、必要に応じて、リポジトリを移動できます。

ローカルのパッケージリポジトリのトラブルシューティング

次の方法を使用すると、問題を回避、または発生する可能性がある問題の原因を見つけることができます。

- リポジトリのソースファイルを検証します。.zip ファイルを使用してリポジトリを作成する場合は、17 ページの「zip ファイルからリポジトリをコピーする方法」の説明に従ってチェックサムを使用して、システム上のファイルが正しいことを確認します。
- インストールされているリポジトリを検証します。pkgrepo verify コマンドを使用して、インストールされているリポジトリをチェックします。

  次の権限の問題が pkgrepo verify によって報告されます。

  - ファイルのアクセス権。ファイルシステムベースのリポジトリでディレクトリおよびファイルの権限に関する問題を回避するには、pkg5srv のユーザーにリポジトリを読み取るための権限があることを確認します。
  - ディレクトリのアクセス権。リポジトリ内のすべてのディレクトリに実行権があることを確認します。

  pkgrepo verify コマンドでほかのタイプのエラーが報告された場合は、pkgrepo fix コマンドを使用してそのエラーの修正を試みてください。詳細は、pkgrepo(1) のマニュアルページを参照してください。
- パブリッシャーの起点を確認します。各イメージに各パブリッシャーの起点を適切に設定していることを確認してください。インストール済みのパッケージの更新、インストール済みのパッケージに依存するパッケージのインストール、または非大域ゾーンのインストールを行うには、少なくともパブリッシャーの起点として設定したリポジトリに、パブリッシャーを設定しているイメージにインストールされているのと同じソフトウェアが含まれている必要があります。29 ページの「ユーザーがファイルインタフェースを使用してパッケージを取得できるようにする方法」のステップ 3 を参照してください。パブリッシャーの設定、およびパッケージのインストールの問題のトラブルシューティングについては、『Oracle Solaris 11.2 ソフトウェアの追加と更新 』を参照してください。

- Web サーバーの構成を確認します。リポジトリにアクセスするために Apache Web サーバーを構成する場合は、エンコードされたスラッシュをデコードしないように Web サーバーを構成します。54 ページの「必要な Apache 構成の設定」の手順を参照してください。エンコードされたスラッシュをデコードすると、パッケージが見つからないというエラーになることがあります。
- 非大域ゾーンからのみアクセスできるリポジトリは作成しないでください。非大域ゾーンに構成されているパブリッシャーのすべての場所は、そのパブリッシャーが大域ゾーンに構成されていなくても、大域ゾーンからアクセスできる必要があります。

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、Oracle Solaris パッケージリポジトリのローカルコピーを 1 つ以上のリポジトリファイルから作成する方法を説明します。リポジトリファイルは、メディア上にあるか、Oracle Solaris のダウンロードサイトで入手できる場合があります。リポジトリファイルは、`zip` ファイルまたは `iso` ファイルである場合があります。

▼ zip ファイルからリポジトリをコピーする方法

1. **新しいリポジトリ用の ZFS ファイルシステムを作成します。**

   共有の場所にリポジトリを作成します。リポジトリのファイルシステムを作成する場合は、`atime` を `off` に設定します。10 ページの「ローカルの IPS パッケージリポジトリの作成および使用のベストプラクティス」を参照してください。

   ```
   $ zfs create -o atime=off rpool/export/IPSpkgrepos
   $ zfs create rpool/export/IPSpkgrepos/Solaris
   $ zfs get atime rpool/export/IPSpkgrepos/Solaris
   NAME                              PROPERTY  VALUE  SOURCE
   rpool/export/IPSpkgrepos/Solaris  atime     off    inherited from rpool/export/IPSpkgrepos
   ```

2. **パッケージリポジトリファイルを取得します。**

   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 IPS パッケージリポジトリ `.zip` 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。`.zip` ファイルとともに、`install-repo.ksh` スクリプトおよび `.txt` ファイル (`README` およびチェックサムファイル) をダウンロードします。

   ```
   $ ls
   ```

```
install-repo.ksh           sol-11_2-ga-repo-3of4.zip
README-zipped-repo.txt     sol-11_2-ga-repo-4of4.zip
sol-11_2-ga-repo-1of4.zip  sol-11_2-ga-repo.txt
sol-11_2-ga-repo-2of4.zip
```

3. **スクリプトファイルが実行可能であることを確認します。**

```
$ chmod +x install-repo.ksh
```

4. **リポジトリのインストールスクリプトを実行します。**

リポジトリのインストールスクリプト `install-repo.ksh` は、各リポジトリの `.zip` ファイルを指定されたディレクトリに圧縮解除します。このスクリプトは、オプションで次の追加のタスクを実行します。

- ダウンロードされた `.zip` ファイルのチェックサムを検証します。チェックサムを検証するための `-c` オプションを指定しない場合は、リポジトリのインストールスクリプトを実行する前に、チェックサムを手動で検証します。次の `digest` コマンドを実行して、`.md5` ファイルの該当するチェックサムと出力を比較します。

```
$ digest -a md5 file
```

- 指定された宛先にリポジトリがすでにある場合は、リポジトリの内容を既存の内容に追加します。
- 最終的なリポジトリを確認します。リポジトリを検証するための `-v` オプションを指定しない場合は、リポジトリのインストールスクリプトを実行したあとに、`pkgrepo` コマンドの `info`、`list` および `verify` サブコマンドを使用して、リポジトリを検証します。
- マウントおよび配布を行うために、ISO イメージファイルを作成します。`-I` オプションを使用して `.iso` ファイルを作成すると、`.iso` ファイルおよび `.iso` ファイルの使用方法を説明する `README` ファイルが、指定された宛先ディレクトリに作成されます。

5. **リポジトリの内容を検証します。**

前のステップで `-v` オプションを指定しなかった場合は、`pkgrepo` コマンドの `info`、`list`、および `verify` サブコマンドを使用して、リポジトリが正常にコピーされたことを確認します。`pkgrepo verify` コマンドでエラーが報告された場合は、`pkgrepo fix` コマンドを使用してそのエラーの修正を試みてください。詳細は、pkgrepo(1) のマニュアルページを参照してください。

6. **新しいリポジトリのスナップショットを作成します。**

```
$ zfs snapshot rpool/export/IPSpkgrepos/Solaris@sol-11_2_0
```

例 2-1 zip ファイルからの新しいリポジトリの作成

この例では、zip ファイルを展開するまで、リポジトリは存在していません。スクリプトには次のオプションを指定できます。

`-s` オプション。.zip ファイルがあるディレクトリへのフルパスを指定します。デフォルト: 現在のディレクトリ。

`-d` 必須。リポジトリを作成するディレクトリへのフルパスを指定します。

`-i` オプション。このリポジトリを生成するために使用するファイルを指定します。ソースディレクトリには、複数のセットの .zip ファイルを含めることができます。デフォルト: ソースディレクトリで使用可能なもっとも新しいイメージ。

`-c` オプション。.zip ファイルのチェックサムと指定されたファイルのチェックサムを比較します。-c を引数なしで指定した場合、使用されるデフォルトファイルは、ソースディレクトリの -i のイメージの .md5 ファイルです。

`-v` オプション。最終的なリポジトリを確認します。

`-I` オプション。リポジトリの ISO イメージをソースディレクトリに作成します。また、ソースディレクトリに mkiso.log ログファイルを残します。

`-h` オプション。使用方法に関するメッセージを表示します。

```
$ ./install-repo.ksh -d /export/IPSpkgrepos/Solaris -c -v -I
Comparing checksums of downloaded files...done. Checksums match.
Uncompressing sol-11_2-ga-repo-1of4.zip...done.
Uncompressing sol-11_2-ga-repo-2of4.zip...done.
Uncompressing sol-11_2-ga-repo-3of4.zip...done.
Uncompressing sol-11_2-ga-repo-4of4.zip...done.
Repository can be found in /export/IPSpkgrepos/Solaris.
Initiating repository verification.
Building ISO image...done.
ISO image and instructions for using the ISO image are at:
/tank/downloads/sol-11_2-ga-repo.iso
/tank/downloads/README-repo-iso.txt
$ ls /export/IPSpkgrepos/Solaris
COPYRIGHT         NOTICES           pkg5.repository   publisher         README-iso.txt
```

リポジトリの再構築および検証には時間がかかることがありますが、「Repository can be found in」というメッセージが表示されたら、リポジトリの内容を取得できます。

例 2-2 zip ファイルから既存のリポジトリへの追加

この例では、リポジトリの zip ファイルの内容が、既存のパッケージリポジトリの内容に追加されます。

```
$ pkgrepo -s /export/IPSpkgrepos/Solaris info
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4764     online           2014-03-18T05:30:57.221021Z
$ ./install-repo.ksh -d /export/IPSpkgrepos/Solaris -c -v -I
IPS repository exists at destination /export/IPSpkgrepos/Solaris
Current version: 0.175.2.0.0.35.0
Do you want to add to this repository? (y/n) y
Comparing checksums of downloaded files...done. Checksums match.
Uncompressing sol-11_2-ga-repo-1of4.zip...done.
Uncompressing sol-11_2-ga-repo-2of4.zip...done.
Uncompressing sol-11_2-ga-repo-3of4.zip...done.
Uncompressing sol-11_2-ga-repo-4of4.zip...done.
Repository can be found in /export/IPSpkgrepos/Solaris.
Initiating repository rebuild.
Initiating repository verification.
Building ISO image...done.
ISO image and instructions for using the ISO image are at:
/tank/downloads/sol-11_2-ga-repo.iso
/tank/downloads/README-repo-iso.txt
$ pkgrepo -s /export/IPSpkgrepos/Solaris info
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-06-02T18:11:55.640930Z
```

▼ iso ファイルからリポジトリをコピーする方法

1. **新しいリポジトリ用の ZFS ファイルシステムを作成します。**

   共有の場所にリポジトリを作成します。リポジトリのファイルシステムを作成する場合は、`atime` を `off` に設定します。10 ページの「ローカルの IPS パッケージリポジトリの作成および使用のベストプラクティス」を参照してください。

   ```
   $ zfs create -o atime=off rpool/export/IPSpkgrepos
   $ zfs create rpool/export/IPSpkgrepos/Solaris
   $ zfs get atime rpool/export/IPSpkgrepos/Solaris
   NAME                              PROPERTY  VALUE  SOURCE
   rpool/export/IPSpkgrepos/Solaris  atime     off    inherited from rpool/export/IPSpkgrepos
   ```

2. **リポジトリパッケージのイメージファイルを取得します。**

   例2-1「zip ファイルからの新しいリポジトリの作成」の説明に従って `-I` オプションを使用して、リポジトリの `.zip` ファイルから `.iso` ファイルを作成します。

3. **イメージファイルをマウントします。**

   リポジトリの `.iso` ファイルをマウントして、内容にアクセスします。

   ```
   $ mount -F hsfs /path/sol-11_2-repo.iso /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ることを回避するには、次のステップに記載されているようにリポジトリファイルの内容をコピーします。

4. **リポジトリの内容を新しい場所にコピーします。**

   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 コマンドまたは tar コマンドを使用して実行できます。

   - **rsync コマンドを使用します。**

     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/IPSpkgrepos/Solaris
     ```

   - **tar コマンドを使用します。**

     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コピーが速くなる場合があります。

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

5. **イメージファイルをアンマウントします。**

   /mnt ディレクトリから移動したことを確認します。

   ```
   $ umount /mnt
   ```

6. **新しいリポジトリの内容を確認します。**

   pkgrepo コマンドの info、list、および verify サブコマンドを使用して、リポジトリが正常にコピーされたことを確認します。pkgrepo verify コマンドでエラーが報告された場合は、pkgrepo fix コマンドを使用してそのエラーの修正を試みてください。詳細は、pkgrepo(1) のマニュアルページを参照してください。

7. **新しいリポジトリのスナップショットを作成します。**

   ```
   $ zfs snapshot rpool/export/IPSpkgrepos/Solaris@sol-11_2_0
   ```

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 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。最初の手順は、コマンド行からコピーコマンドを発行する方法を示しています。2 番目の手順は、SMF サービスを使用して、リポジトリを自動的にコピーおよび更新する方法を示しています。

▼ インターネットからリポジトリを明示的にコピーする方法

1. **新しいリポジトリ用の ZFS ファイルシステムを作成します。**

   共有の場所にリポジトリを作成します。リポジトリのファイルシステムを作成する場合は、`atime` を `off` に設定します。10 ページの「ローカルの IPS パッケージリポジトリの作成および使用のベストプラクティス」を参照してください。

```
$ zfs create -o atime=off rpool/export/IPSpkgrepos
$ zfs create rpool/export/IPSpkgrepos/Solaris
$ zfs get atime rpool/export/IPSpkgrepos/Solaris
NAME                              PROPERTY  VALUE  SOURCE
rpool/export/IPSpkgrepos/Solaris  atime     off    inherited from rpool/export/IPSpkgrepos
```

2. **必要なリポジトリインフラストラクチャーを作成します。**

   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。前の方法で使用したイメージファイルには、リポジトリインフラストラクチャーが含まれているため、このステップは必要ありません。この方法の説明に従って `pkgrecv` コマンドを使用してリポジトリの内容をコピーする場合は、リポジトリのインフラストラクチャーを作成して、リポジトリの内容をそのインフラストラクチャーにコピーする必要があります。`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/IPSpkgrepos/Solaris
```

3. **リポジトリの内容を新しい場所にコピーします。**

   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。15 ページの「リポジトリのコピーのパフォーマンスに関する考慮事項」も参照してください。あとでこのリポジトリを更新する場合、変更内容のみが転送され、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。

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すべての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。セキュアなサイトからコピーする場合は、必要な SSL 証明書および鍵がインストールされていることを確認し、必要な証明書および鍵のオプションを指定します。

```
$ pkgrecv -s https://pkg.oracle.com/solaris/support -d /export/IPSpkgrepos/Solaris \
--key /path-to-ssl_key --cert /path-to-ssl_cert '*'
```

-m および --clone オプションの詳細は、pkgrecv(1)のマニュアルページを参照してください。このためには -m latest オプションを使用しないでください。極端な疎リポジトリを使用すると、ユーザーがそのイメージを更新しようとしたときに、エラーが発生する場合があります。

4. **新しいリポジトリの内容を確認します。**

   pkgrepo コマンドの info、list、および verify サブコマンドを使用して、リポジトリが正常にコピーされたことを確認します。pkgrepo verify コマンドでエラーが報告された場合は、pkgrepo fix コマンドを使用してそのエラーの修正を試みてください。詳細は、pkgrepo(1) のマニュアルページを参照してください。

5. **新しいリポジトリのスナップショットを作成します。**

   ```
   $ zfs snapshot rpool/export/IPSpkgrepos/Solaris@sol-11_2_0
   ```

▼ インターネットからリポジトリを自動的にコピーする方法

デフォルトでは、svc:/application/pkg/mirror SMF サービスは、このイメージに定義されている solaris パブリッシャーの起点から /var/share/pkg/repositories/solaris に対して、pkgrecv 操作を定期的に実行します。この pkgrecv 操作は、各月である 1 日の午前 2 時 30 分に開始されます。このデフォルトの動作を変更するには、この手順の説明に従ってサービスを構成します。

このサービスの正常な各実行の終わりに、リポジトリカタログがリフレッシュされます。検索インデックスを構築するためにリポジトリをリフレッシュする必要はありません。

このサービスは定期的に実行されるため、リポジトリが作成されて随時更新されます。このドキュメントに示す手動のリポジトリ更新手順を使用する必要はありません。

ほかのシステムは、solaris パブリッシャーの起点を、この自動更新のリポジトリまたはこのリポジトリのクローンに設定できます。インターネットパブリッシャーの起点を持ち、mirror サービスを実行して更新を自動的に受け取る必要があるシステムは 1 つだけです。

1. **パブリッシャーの起点を設定します。**

   デフォルトでは、mirror サービスは、/ をルートに持つイメージに構成されている solaris パブリッシャーからパッケージを転送します。mirror サービスの構成にパブリッシャーの起点は直接指定できませんが、この情報を取得するイメージルートを構成できます。そのイメージルートで、pkg set-publisher を使用して、ミラーリポジトリのための pkgrecv の転送のソースとして使用するパブリッシャーの起点を構成します。

   a. **(オプション) イメージのルートを設定します。**

      ミラーサービスに使用するパブリッシャーの構成がこのイメージで使用するパブリッシャーの構成と異なる場合は、次の例に示すように、ユーザーイメージをどの BE にも含まれていない共有の場所に作成し、mirror サービスの config/ref_image プロパティーの値をその新しいイメージにリセットします。mirror サービスは、config/ref_image イメージからのパブリッシャー構成を使用します。

      ```
      $ svccfg -s pkg/mirror:default setprop config/ref_image = /var/share/pkg/
      mirror_svc_ref_image
      $ pkg image-create /var/share/pkg/mirror_svc_ref_image
      ```

   b. **(オプション) パブリッシャーを設定します。**

      solaris パブリッシャーに加えて、ほかのパブリッシャーからのパッケージでミラーリポジトリを更新する場合は、ha-cluster および solarisstudio パブリッシャーの追加を示す次の例のように、mirror サービスの config/publishers プロパティーの値をリセットします。

      ```
      $ svccfg -s pkg/mirror:default setprop config/publishers = solaris,ha-
      cluster,solarisstudio
      ```

   c. **パブリッシャーの起点を設定します。**

      このサービスは定期的に実行されるため、パブリッシャーの起点を定期的な更新が提供されるリポジトリに設定してください。Oracle 製品の場合は、パブリッシャーの起点をサポートリポジトリに設定して、Support Repository Updates (SRU) を取得する場合があります。次の例では、パブリッシャーに代替のイメージルートを構成する場合にのみ、-R オプションが必要となります。起点 URI によっては、-k および -c オプションは必要がない場合があります。

```
$ pkg -R /var/share/pkg/mirror_svc_ref_image set-publisher \
-g https://pkg.oracle.com/solaris/support/ -k ssl_key -c ssl_cert solaris
$ pkg -R /var/share/pkg/mirror_svc_ref_image set-publisher \
-g https://pkg.oracle.com/ha-cluster/support/ -k ssl_key -c ssl_cert ha-cluster
$ pkg -R /var/share/pkg/mirror_svc_ref_image set-publisher \
-g https://pkg.oracle.com/solarisstudio/support/ -k ssl_key -c ssl_cert solarisstudio
```

次のいずれかのコマンドを使用して、新しいイメージに構成されているパブリッシャーを検証します。

```
$ pkg -R /var/share/pkg/mirror_svc_ref_image publisher
$ pkg -R /var/share/pkg/mirror_svc_ref_image publisher solaris ha-cluster solarisstudio
```

2. **(オプション) ミラーサービスのその他のプロパティーを構成します。**

   `mirror` サービスのほかのプロパティー (サービスが実行される時間、ミラーリポジトリの場所など) の変更が必要になる場合があります。

   ミラー化しているパブリッシャーの起点の更新が予期する時間に近くなるように、サービスの実行時間の変更が必要になる場合があります。サービスの実行時間を変更するには、`config/crontab_period` プロパティーの値を変更します。

   ミラーリポジトリの場所を変更するには、`config/repository` プロパティーの値を変更します。ミラーリポジトリの場所を変更する場合は、そのリポジトリを共有の場所に配置します。10 ページの「ローカルの IPS パッケージリポジトリの作成および使用のベストプラクティス」を参照してください。デフォルトの場所 (`/var/share/pkg/repositories/solaris`) は、どの BE にも含まれていない共有の場所です。

3. **ミラーサービスを有効にします。**

   `svcs mirror` コマンドを使用して、`mirror` サービスの状態を確認します。

   ■ **サービスが無効だが、このサービスを使用する必要がある。**

   a. **構成を変更した場合は、サービスインスタンスをリフレッシュします。**

      前の手順の `svccfg setprop` コマンドに示したように、`mirror` サービスの構成を変更した場合は、サービスをリフレッシュして、変更された値を実行中のスナップショットにコミットします。`svcprop -p config mirror` コマンドの出力に必要な値が表示されない場合は、`svccfg -s mirror:default listprop config` コマンドの出力に必要な値が表示されることを確認します。`svcadm refresh mirror:default` または `svccfg -s mirror:default refresh` を使用して、サービスの実行中のスナップショットに変

更された値をコミットします。svcprop -p config mirror コマンドを再度使用して、サービスが意図したとおりに構成されていることを確認します。

b. **サービスインスタンスを有効にします。**

次のコマンドを使用して、ミラーサービスを有効にします。

```
$ svcadm enable mirror:default
```

svcs mirror コマンドを使用して、mirror サービスがオンラインであることを確認します。サービスは、config/crontab_period プロパティーに設定されている時間に実行されます。

- **サービスはオンラインであり、サービスをすぐに実行する必要がある。**

  サービスがオンラインである場合、サービスをすぐに実行するには、サービスをリフレッシュします。svc-pkg-mirror メソッドおよび pkgrecv コマンドの pkg5srv ユーザーによる実行を確認できます。

- **サービスはオンラインであるが、このサービスを使用しない。**

  svcadm disable mirror コマンドを使用して、このサービスを無効にします。マスターリポジトリを保守するために、このサービスを 1 つのシステムのみで実行する場合があります。ほかのシステムでは、このサービスを無効にします。

- **サービスは保守状態であるか、機能低下している。**

  svcs -xvL mirror コマンドを使用して詳細情報を取得し、問題を診断および修正します。

4. **リポジトリの内容を確認します。**

mirror サービスの実行が完了したら、pkgrepo コマンドの info、list、および verify サブコマンドを使用して、リポジトリが正常にコピーまたは更新されたことを確認します。pkgrepo verify コマンドでエラーが報告された場合は、pkgrepo fix コマンドを使用してそのエラーの修正を試みてください。詳細は、pkgrepo(1) のマニュアルページを参照してください。

サービスが実行される時間を確認するには、mirror サービスの config/crontab_period プロパティーの値をチェックします。サービスの実行中、svcs -p mirror コマンドは、サービスの状態を online* として表示し、このサービスによって開始したプロセスを表示します。サービスの状態が online として表示され、サービスに関連付けられているプロセスがなくなるまで待ってから、リポジトリを確認します。

5. **新しいリポジトリのスナップショットを作成します。**

```
$ zfs snapshot rpool/VARSHARE/pkg/repositories/solaris@sol-11_2_0
```

次の手順 同時に複数のパブリッシャーから内容をコピーする必要が場合があります。1 つの `config/publishers` プロパティーに複数のパブリッシャーを設定する代わりに、`pkg/mirror` サービスの複数のインスタンスを作成できます。たとえば、`config/publishers` プロパティーには、`default` インスタンスの場合は `solaris`、新しい `pkg/mirror:ha-cluster` インスタンスの場合は `ha-cluster`、新しい `pkg/mirror:solarisstudio` インスタンスの場合は `solarisstudio` を設定できます。同様に、`config/crontab_period` は、各インスタンスに対して別々に設定できます。この手順に示すように、各パブリッシャーの内容を 1 つのリポジトリに格納するか、各 `pkg/mirror` インスタンスに別個の `config/repository` 値を設定できます。

参照 SMF コマンドの詳細は、『Oracle Solaris 11.2 でのシステムサービスの管理 』を参照してください。

♦ ♦ ♦ 第 3 章

3 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 つのリポジトリを構成できます。

ユーザーがファイルインタフェースを使用してパッケージを取得できるようにする

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。

▼ ユーザーがファイルインタフェースを使用してパッケージを取得できるようにする方法

1. **NFS 共有を構成します。**

   クライアントが NFS を使用してローカルリポジトリにアクセスできるようにするには、NFS 共有を作成して公開します。

   ```
   $ zfs share -o share.nfs=on rpool/export/IPSpkgrepos%ipsrepo
   ```

   設定可能なその他の `share.nfs` プロパティーなどの詳細は、`zfs_share`(1M) のマニュアルページを参照してください。

2. **共有が公開されたことを確認します。**

   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/IPSpkgrepos     ipsrepo 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判別します。**

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

3. **パブリッシャーの起点を設定します。**

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、パブリッシャーの起点を設定します。

a. **パブリッシャーの名前を判別します。**

次のコマンドを使用して、リポジトリ内のパブリッシャーの名前を判別します。

```
$ pkgrepo info -s /export/IPSpkgrepos/Solaris
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-04-02T18:11:55.640930Z
```

b. **このパブリッシャーの起点の適合性を確認します。**

インストール済みのパッケージの更新、インストール済みのパッケージに依存するパッケージのインストール、または非大域ゾーンのインストールを行うには、少なくともパブリッシャーの起点として設定したリポジトリに、パブリッシャーを設定しているイメージにインストールされているのと同じソフトウェアが含まれている必要があります。リポジトリにはより古いソフトウェアまたはより新しいソフトウェアを含めることもできますが、イメージにインストールされているのと同じソフトウェアが含まれている必要があります。

次のコマンドは、指定されたリポジトリがこのイメージに適合しないパブリッシャーの起点であることを示しています。

```
$ pkg list entire
NAME (PUBLISHER)      VERSION                    IFO
entire                0.5.11-0.175.2.0.0.36.0    i--
$ pkgrepo list -Hs http://pkg.oracle.com/solaris/release entire@0.5.11-0.175.2.0.0.36.0
pkgrepo list: The following pattern(s) did not match any packages:
        entire@0.5.11-0.175.2.0.0.36.0
```

次のコマンドは、指定されたリポジトリがこのイメージに適合するパブリッシャーの起点であることを示しています。

```
$ pkgrepo list -Hs /export/IPSpkgrepos/Solaris entire@0.5.11-0.175.2.0.0.36.0
solaris       entire       0.5.11,5.11-0.175.2.0.0.36.0:20140401T190148Z
```

c. **パブリッシャーの起点を設定します。**

前の手順のリポジトリの場所およびパブリッシャー名を使用して、次のコマンドを実行し、パブリッシャーの起点を設定します。

```
$ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/Solaris/ solaris
```

`-G '*'` `solaris` パブリッシャーについての既存の起点をすべて削除します。

`-M '*'` `solaris` パブリッシャー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` パブリッシャーの新しい起点として追加します。

パブリッシャーの構成の詳細は、『Oracle Solaris 11.2 ソフトウェアの追加と更新 』の「パブリッシャーの構成」を参照してください。

ほかのイメージでパブリッシャーの起点をリセットした場合は、適合性テストを再度実行します。ほかのイメージには別のバージョンのソフトウェアがインストールされている可能性があるため、このリポジトリを使用できないことがあります。ほかのシステム上のイメージでパブリッシャーの起点をリセットした場合は、`-g` 引数にフルパスを使用します。

ユーザーが HTTP インタフェースを使用してパッケージを取得できるようにする

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。

▼ ユーザーが HTTP インタフェースを使用してパッケージを取得できるようにする方法

パッケージ集積サーバー `pkg.depotd` は、パッケージリポジトリ内に含まれているデータへのネットワークアクセスを提供します。`svc:/application/pkg/server` SMF サービスは、`pkg.depotd` デーモンを呼び出します。クライアントが HTTP を使用してローカルリポジトリにアクセスできるようにするために、この手順では `pkg/server` サービスを構成する方法を示しています。サービ

スの `default` インスタンスを構成できます。この手順は、新しいインスタンスを作成および構成する方法を示しています。

1. **集積サーバーのインスタンスを作成します。**

   `add` サブコマンドを使用して、`solaris` という名前の `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 solaris
   ```

2. **リポジトリへのパスを設定します。**

   サービスのこのインスタンスが、リポジトリのデータを検索できるパスを設定します。

   ```
   $ svccfg -s pkg/server:solaris setprop pkg/inst_root=/export/IPSpkgrepos/Solaris
   ```

3. **(オプション) ポート番号を設定します。**

   集積サーバーインスタンスが受信パッケージ要求を待機するポート番号を設定します。デフォルトでは、`pkg.depotd` はポート 80 で接続を待機します。ポートを変更するには、`pkg/port` プロパティーをリセットします。

   ```
   $ svccfg -s pkg/server:solaris setprop pkg/port=81
   ```

4. **(オプション) ほかのプロパティーを設定します。**

   `pkg/server` プロパティーの完全な一覧については、`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ーを設定するには、次のコマンドを使用してすべてのプロパティーを一度に編集します。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。

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

5. **リポジトリサービスを起動します。**

   パッケージ集積サーバーサービスを再起動します。

   ```
   $ svcadm refresh pkg/server:solaris
   $ svcadm enable pkg/server:solaris
   ```

6. **リポジトリサーバーが動作しているかどうかをテストします。**

   リポジトリサービスが動作しているかどうかを判別するには、`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。

7. **パブリッシャーの起点を設定します。**

   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、パブリッシャーの起点を設定します。

   a. **パブリッシャーの名前を判別します。**

      次のコマンドを使用して、リポジトリ内のパブリッシャーの名前を判別します。

```
$ pkgrepo info -s /export/IPSpkgrepos/Solaris
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-04-02T18:11:55.640930Z
```

   b. **このパブリッシャーの起点の適合性を確認します。**

      インストール済みのパッケージの更新、インストール済みのパッケージに依存するパッケージのインストール、または非大域ゾーンのインストールを行うには、少なくともパブリッシャーの起点として設定したリポジトリに、パブリッシャーを設定しているイメージにインストールされているのと同じソフトウェアが含まれている必要があります。リポジトリにはより古いソフトウェアまたはより新しいソフトウェアを含めることもできますが、イメージにインストールされているのと同じソフトウェアが含まれている必要があります。

      次のコマンドは、指定されたリポジトリがこのイメージに適合しないパブリッシャーの起点であることを示しています。

```
$ pkg list entire
NAME (PUBLISHER)      VERSION                    IFO
entire                0.5.11-0.175.2.0.0.36.0    i--
$ pkgrepo list -Hs http://pkg.oracle.com/solaris/release entire@0.5.11-0.175.2.0.0.36.0
pkgrepo list: The following pattern(s) did not match any packages:
        entire@0.5.11-0.175.2.0.0.36.0
```

      次のコマンドは、指定されたリポジトリがこのイメージに適合するパブリッシャーの起点であることを示しています。

```
$ pkgrepo list -Hs http://localhost:81/ entire@0.5.11-0.175.2.0.0.36.0
solaris       entire       0.5.11,5.11-0.175.2.0.0.36.0:20140401T190148Z
```

   c. **パブリッシャーの起点を設定します。**

      パブリッシャーの起点に次のいずれかの値を設定します。

      - `pkg/inst_root` の場所。

```
$ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/Solaris/ solaris
```

      - `pkg/port` の場所。

```
$ pkg set-publisher -G '*' -M '*' -g http://localhost:81/ solaris
```

`-G '*'` solaris パブリッシャーについての既存の起点をすべて削除します。

`-M '*'` solaris パブリッシャー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 パブリッシャーの新しい起点として追加します。

パブリッシャーの構成の詳細は、『Oracle Solaris 11.2 ソフトウェアの追加と更新 』の「パブリッシャーの構成」を参照してください。

ほかのイメージでパブリッシャーの起点をリセットした場合は、適合性テストを再度実行します。ほかのイメージには別のバージョンのソフトウェアがインストールされている可能性があるため、このリポジトリを使用できないことがあります。

参照

- 47 ページの「Web サーバーアクセスを使用した複数のリポジトリの提供」では、複数の場所または単一の場所から複数のリポジトリを提供する方法を説明しています。
- 58 ページの「1 つのドメインでの複数リポジトリ」では、異なる接頭辞を付けて 1 つのドメイン名の配下で複数のリポジトリを実行する方法を説明しています。
- 60 ページの「HTTPS でのリポジトリアクセスの構成」では、セキュアなリポジトリアクセスを構成する方法を説明しています。

♦ ♦ ♦ 第 4 章

4 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ー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。

ローカルリポジトリの更新

このセクションで説明する手順は、IPS パッケージリポジトリを更新するための次のベストプラクティスを示しています。

- そのリリースのすべてのサポート更新で各リポジトリを常に最新にします。サポート更新には、セキュリティー更新およびその他の重要な修正が含まれています。
  - サポート更新から特定の修正を選択して適用しようとしないでください。パッケージのサブセットをサポート更新からリポジトリに追加しないでください。サポート更新のすべての内容をローカルリポジトリに追加します。`pkgrecv` コマンドのデフォルトの動作では、すべてのパッケージのすべてのバージョンが取得されます。
  - サポート更新をスキップしないでください。適用可能なすべてのサポート更新を各リポジトリに適用してください。

  リポジトリ内の最新のバージョンより前のバージョンに更新するには、インストールする `entire` incorporation パッケージのバージョンを指定します。『Oracle Solaris 11.2 ソフトウェアの追加と更新 』の第 4 章「Oracle Solaris イメージの更新またはアップグレード」を参照してください。
- リポジトリのコピーを更新します。この方法では、リポジトリの更新中にシステムがリポジトリにアクセスしません。リポジトリの更新、スナップショットのクローニング、更新の実行、および元のリポジトリと更新済みクローンの置き換えを行う前に、リポジトリのスナップショットを作成します。

同じ内容のパッケージリポジトリのコピーを複数保守している場合は、次の手順を使用して、それらの同一のリポジトリのいずれかを更新します。このマスターリポジトリからその他のリポジトリを更新する手順については、39 ページの「複数の同一のローカルリポジトリの保守」を参照してください。

▼ ローカルの IPS パッケージリポジトリを更新する方法

注記 - `svc:/application/pkg/mirror` SMF サービスを使用してリポジトリを定期的に更新している場合は、この手順を実行する必要はありません。`mirror` サービスを使用する手順については、23 ページの「インターネットからリポジトリを自動的にコピーする方法」を参照してください。

1. **パッケージリポジトリの ZFS スナップショットを作成します。**

   更新するリポジトリの最新のスナップショットがあることを確認します。

```
$ zfs list -t all -r rpool/export/IPSpkgrepos/Solaris
NAME                                          USED  AVAIL  REFER  MOUNTPOINT
rpool/export/IPSpkgrepos/Solaris             17.6G  78.4G    34K  /export/IPSpkgrepos/Solaris
rpool/export/IPSpkgrepos/Solaris@initial         0      -  17.6G  -
```

   リポジトリのスナップショットがすでにある場合は、`zfs diff` コマンドを使用して、そのスナップショットがリポジトリデータセットと同じであるかどうかを確認します。

```
$ zfs diff rpool/export/IPSpkgrepos/Solaris@initial
$
```

   `zfs diff` コマンドで出力が生成されない場合、そのスナップショットは親データセットと同じであり、そのスナップショットを更新に使用できます。

   `zfs diff` コマンドの出力があった場合、またはリポジトリのスナップショットがない場合は、22 ページの「インターネットからリポジトリを明示的にコピーする方法」のステップ 6 に示すように、新しいスナップショットを作成します。この新しいスナップショットを更新に使用します。

2. **パッケージリポジトリの ZFS クローンを作成します。**

   リポジトリのスナップショットをクローニングして、更新可能なリポジトリのコピーを作成します。

```
$ zfs clone rpool/export/IPSpkgrepos/Solaris@initial rpool/export/IPSpkgrepos/Solaris_tmp
$ zfs list -r rpool/export/IPSpkgrepos/Solaris/
NAME                                          USED  AVAIL  REFER  MOUNTPOINT
rpool/export/IPSpkgrepos/Solaris             17.6G  78.4G    34K  /export/IPSpkgrepos/Solaris
rpool/export/IPSpkgrepos/Solaris@initial         0      -  17.6G  -
rpool/export/IPSpkgrepos/Solaris_tmp          76K  78.4G  17.6G  /export/IPSpkgrepos/
Solaris_tmp
```

3. **パッケージリポジトリの ZFS クローンを更新します。**

   ファイルまたは HTTP の場所から元のリポジトリを作成したように、ファイルまたは HTTP の場所からリポジトリを更新できます。

   ■ **zip ファイルから更新します。**

   例2-2「zip ファイルから既存のリポジトリへの追加」を参照してください。指定した宛先にパッケージリポジトリがすでにある場合は、zip ファイルの内容が既存のリポジトリの内容に追加されます。

   ■ **ISO ファイルから更新します。**

   a. **ISO イメージをマウントします。**

   ```
   $ mount -F hsfs ./sol-11_2-incr-repo.iso /mnt
   ```

   b. **ISO ファイルの内容をリポジトリのクローンにコピーします。**

   20 ページの「iso ファイルからリポジトリをコピーする方法」に示す rsync または tar のどちらかを使用します。

   ```
   $ rsync -aP /mnt/repo/ /export/IPSpkgrepos/Solaris_tmp
   ```

   c. **ISO イメージをアンマウントします。**

   ■ **リポジトリから更新します。**

   別のリポジトリの内容をリポジトリクローンにコピーします。セキュアなサイトからコピーする場合は、必要な SSL 証明書および鍵がインストールされていることを確認し、必要な証明書および鍵のオプションを指定します。

   ```
   $ pkgrecv -s https://pkg.oracle.com/solaris/support \
   -d /export/IPSpkgrepos/Solaris_tmp \
   --key /path-to-ssl_key --cert /path-to-ssl_cert '*'
   ```

   pkgrecv コマンドの詳細は、pkgrecv(1) のマニュアルページを参照してください。変更されたパッケージのみが更新されるため、リポジトリを更新する時間は、元のリポジトリを取り込むためにかかる時間より大幅に短くなることがあります。15 ページの「リポジトリのコピーのパフォーマンスに関する考慮事項」のパフォーマンスのヒントを参照してください。

   pkgrecv 操作が中断された場合は、38 ページの「中断されたパッケージの受信の再開」の手順に従います。

4. **稼働中のリポジトリを更新済みのクローンに置き換えます。**

```
$ svcadm disable -st pkg/server:solaris
$ zfs promote rpool/export/IPSpkgrepos/Solaris_tmp
$ zfs rename rpool/export/IPSpkgrepos/Solaris rpool/export/IPSpkgrepos/Solaris_old
$ zfs rename rpool/export/IPSpkgrepos/Solaris_tmp rpool/export/IPSpkgrepos/Solaris
```

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

5. **更新されたリポジトリを検証します。**

`pkgrepo verify` コマンドを使用して、更新済みのリポジトリを検証します。`pkgrepo verify` および `pkgrepo fix` コマンドの詳細は、`pkgrepo(1)` のマニュアルページを参照してください。

6. **新しいパッケージをカタログ化して検索インデックスを更新します。**

新しく更新されたリポジトリで見つかった新しいパッケージをすべてカタログ化し、すべての検索インデックスを更新します。

```
$ pkgrepo refresh -s rpool/export/IPSpkgrepos/Solaris
```

7. **パッケージリポジトリの新しく更新されたクローンの ZFS スナップショットを作成します。**

```
$ zfs snapshot rpool/export/IPSpkgrepos/Solaris@S11U2SRU1
```

8. **SMF サービスを再起動します。**

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。サービスを再起動するときは、適切なサービスインスタンスを指定してください。

```
$ svcadm restart pkg/server:solaris
```

9. **古いリポジトリを削除します。**

更新されたリポジトリが正しく動作していることを確認したら、古いリポジトリを削除できます。

```
$ zfs destroy rpool/export/IPSpkgrepos/Solaris_old
```

中断されたパッケージの受信の再開

`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: 'https://pkg.oracle.com/solaris/support/file/file_hash

pkgrecv: Cached files were preserved in the following directory:
        /var/tmp/pkgrecv-fOGaIg
Use pkgrecv -c to resume the interrupted download.
$ pkgrecv -c /var/tmp/pkgrecv-fOGaIg \
-s https://pkg.oracle.com/solaris/support -d /export/IPSpkgrepos/Solaris_tmp \
--key /path/to/ssl_key --cert /path/to/ssl_cert '*'
Processing packages for publisher solaris ...
Retrieving and evaluating 156 package(s)...
```

複数の同一のローカルリポジトリの保守

次の目的を達成するために、内容が同じパッケージリポジトリのコピーを複数保持する場合があります。

- 異なるノードにコピーを保持することによって、リポジトリの可用性が向上します。
- ユーザーが多数の場合、またはユーザーが離れた場所に分散している場合に、リポジトリアクセスのパフォーマンスを向上させます。

いずれかのパッケージリポジトリを更新するには、36 ページの「ローカルの IPS パッケージリポジトリを更新する方法」の手順を使用します。その後、40 ページの「ローカルの IPS パッケージリポジトリをクローニングする方法」の手順を使用して、最初に更新したリポジトリからその他の同一のリポジトリを更新します。これらの 2 つの手順は非常に似ていますが、大きな違いは `pkgrecv` コマンドの使用方法です。クローンの手順で示した `pkgrecv` の操作では、ソースリポジトリファイルが厳密にコピーされ、次のような状態になります。

- クローニングされたリポジトリのカタログのタイムスタンプは、ソースリポジトリのカタログのタイムスタンプとまったく同じです。リポジトリが負荷分散されている場合、ロードバランサがクライアントをあるノードから別のノードに切り替えるときの問題を避けるために、すべてのリポジトリのカタログがまったく同じである必要があります。負荷分散の詳細は、58 ページの「負荷分散の構成」を参照してください。
- 宛先リポジトリに存在するがソースリポジトリに存在しないパッケージは、宛先リポジトリから削除されます。目的が疎リポジトリのみの厳密なコピーを作成することである場合を除き、クローン操作で疎リポジトリをソースとして使用しないでください。

▼ ローカルの IPS パッケージリポジトリをクローニングする方法

これらの手順の詳細は、36 ページの「ローカルの IPS パッケージリポジトリを更新する方法」を参照してください。

1. **宛先リポジトリをコピーします。**
   宛先リポジトリの最新のスナップショットがあることを確認します。このスナップショットの ZFS クローンを作成します。

2. **宛先リポジトリのコピーを更新します。**
   `pkgrecv` コマンドを使用して、これまで更新されていたローカルのパッケージリポジトリを宛先リポジトリのコピーにクローニングします。`pkgrecv` のクローン操作の詳細は、`pkgrecv(1)`のマニュアルページを参照してください。

   ```
   $ pkgrecv -s /net/host1/export/IPSpkgrepos/Solaris \
   -d /net/host2/export/IPSpkgrepos/Solaris_tmp --clone
   ```

3. **稼働中の宛先リポジトリを更新されたクローンに置き換えます。**

4. **更新されたリポジトリを検証します。**
   `pkgrepo verify` コマンドを使用して、更新された宛先リポジトリを検証します。

5. **新しく更新されたリポジトリのスナップショットを作成します。**

6. **SMF サービスを再起動します。**
   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。サービスを再起動するときは、適切なサービスインスタンスを指定してください。

7. **古いリポジトリを削除します。**
   更新されたリポジトリが正しく動作していることを確認したら、古いリポジトリを削除します。

参照 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、次の関連するドキュメントを参照してください。

- 47 ページの「Web サーバーアクセスを使用した複数のリポジトリの提供」では、異なるポートで実行される複数の `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を説明しています。

- 58 ページの「1 つのドメインでの複数リポジトリ」では、異なる接頭辞を付けて 1 つのドメイン名の配下で複数のリポジトリを実行する方法を説明しています。

リポジトリプロパティーの確認および設定

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、およびリポジトリのプロパティー値を変更する方法について説明します。

リポジトリ全体に適用されるプロパティーの表示

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パブリッシャーの一覧を表示します。STATUS 列は、パブリッシャーのパッケージデータが現在処理中であるかどうかを示しています。

```
$ pkgrepo info -s /export/IPSpkgrepos/Solaris
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4506     online           2013-07-11T23:32:46.379726Z
```

次のコマンドは、リポジトリ全体に適用されるプロパティー情報を表示します。リポジトリのプロパティーの完全な一覧とその説明 (それらの値の指定を含む) 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。

```
$ pkgrepo get -s /export/IPSpkgrepos/Solaris
SECTION    PROPERTY                     VALUE
publisher  prefix                       solaris
repository check-certificate-revocation False
repository signature-required-names     ()
repository trust-anchor-directory       /etc/certs/CA/
repository version                      4
```

publisher/prefix

デフォルトのパブリッシャーの名前。リポジトリには複数のパブリッシャーからのパッケージを含めることができますが、デフォルトのパブリッシャーとして設定できるのは 1 つのパブリッシャーのみです。このデフォルトのパブリッシャー名は、次の目的に使用されます。

- pkg コマンドでパッケージ FMRI にパブリッシャーが指定されていない場合に、パッケージを識別するため
- パッケージがリポジトリに公開される場合 (pkgsend(1) コマンドを使用)、およびパッケージマニフェストにパブリッシャーが指定されていない場合に、パッケージにパブリッシャーを割り当てるため

repository/check-certificate-revocation

証明書を確認するフラグ。True と設定すると、pkgrepo verify は証明書が発行時よりもあとに失効していないかどうかを調べます。この値は、『Oracle Solaris 11.2 ソフトウェアの追加と更新 』の「追加のイメージのプロパティー」および pkg(1)のマニュアルページに記述されている check-certificate-revocation イメージプロパティーの値と一致している必要があります。

repository/signature-required-names

パッケージの署名の検証中に、証明書の共通名として表示される必要のある名前の一覧です。このリストは pkgrepo verify コマンドで使用します。この値は、『Oracle Solaris 11.2 ソフトウェアの追加と更新 』の「署名付きパッケージのイメージプロパティー」および pkg(1)のマニュアルページに記述されている signature-required-names イメージプロパティーの値と一致している必要があります。

repository/trust-anchor-directory

このリポジトリ内のパッケージのトラストアンカーを含むディレクトリの絶対パス名。デフォルトは /etc/certs/CA/ です。この値は、『Oracle Solaris 11.2 ソフトウェアの追加と更新 』の「追加のイメージのプロパティー」および pkg(1)のマニュアルページに記述されている trust-anchor-directory イメージプロパティーの値と一致している必要があります。

repository/version

リポジトリの形式のバージョン。この値は、44 ページの「リポジトリのプロパティー値の変更」に示す pkgrepo set コマンドでは設定できません。この値は pkgrepo create コマンドを使用して設定できます。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パブリッシャーについてのパッケージの保管をサポートします。

リポジトリパブリッシャーのプロパティーの表示

次のコマンドは、ローカルリポジトリ内の solaris パブリッシャーに関するプロパティー情報を表示します。丸括弧は、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。

```
$ pkgrepo get -p solaris -s /export/IPSpkgrepos/Solaris
PUBLISHER SECTION    PROPERTY         VALUE
solaris   publisher  alias
solaris   publisher  prefix           solaris
solaris   repository collection-type  core
solaris   repository description      ""
solaris   repository legal-uris       ()
solaris   repository mirrors          ()
solaris   repository name             ""
solaris   repository origins          ()
solaris   repository refresh-seconds  ""
solaris   repository registration-uri ""
solaris   repository related-uris     ()
```

`publisher/prefix`

`-p` オプションで指定されたパブリッシャーの名前。`-p` オプションを指定しない場合、前のセクションで説明したように、この値はこのリポジトリのデフォルトパブリッシャーの名前です。

`repository/collection-type`

このリポジトリ内のパッケージのタイプ。値が `core` である場合、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ます。値が `supplemental` の場合、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とは限りません。

`repository/description`

このリポジトリの目的と内容。このリポジトリを HTTP インタフェースから利用できる場合、この値はメインページの上部近くにある「バージョン情報」セクションに表示されます。

`repository/legal-uris`

リポジトリに関する使用条件情報を提供するドキュメントの場所のリスト。

`repository/mirrors`

このリポジトリと同じパッケージの内容を含むリポジトリの場所の一覧。

`repository/name`

このリポジトリの名前。このリポジトリを HTTP インタフェースから利用できる場合、この値はメインページの上部およびウィンドウのタイトルに表示されます。

`repository/origins`

このリポジトリと同じパッケージの内容およびメタデータを含むリポジトリの場所の一覧。

`repository/refresh-seconds`

このリポジトリ内の更新されたパッケージデータのチェックを次に実行するまでにクライアントが待機する秒数。

`repository/registration-uri`

このリポジトリへのアクセスのための資格を取得するために使用する必要のあるリソースの場所。

`repository/related-uris`

関連する可能性のあるその他のパッケージを含むリポジトリの場所のリスト。

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 This\ repository\ serves\ the\ Oracle\ Solaris\ 11\ Package\
 repository.
solaris   repository name        Oracle\ Solaris\ 11\ Package\ Repository
```

リポジトリのプロパティー値の変更

42 ページの「リポジトリパブリッシャーのプロパティーの表示」は、ローカルリポジトリ内の `solaris` パブリッシャーには、リポジトリ名および説明のプロパティー値が設定されないことを示しています。このリポジトリを HTTP インタフェースから利用でき、ブラウザを使用してこのリポジトリの内容を表示する場合、デフォルト名は表示されますが、説明は表示されません。これらの値を設定すると、パブリッシャーの `repository/name` 値は、ページの上部近く、およびページのタイトルとして表示されます。パブリッシャーの `repository/description` 値は、名前のすぐ下にある「バージョン情報」セクションに表示されます。これらの値を設定するときは、`-p` オプションを使用して 1 つ以上のパブリッシャーを指定する必要があります。このリポジトリに複数のパブリッシャーの内容が含まれている場合は、各パブリッシャーに異なる値を設定するか、`-p all` を指定できます。

```
$ pkgrepo set -p solaris -s /export/IPSpkgrepos/Solaris \
repository/description="Local copy of the Oracle Solaris 11 repository." \
repository/name="Oracle Solaris 11"
$ pkgrepo get -p solaris -s /export/IPSpkgrepos/Solaris repository/name repository/description
PUBLISHER SECTION    PROPERTY         VALUE
solaris   repository description      Local\ copy\ of\ the\ Oracle\ Solaris\ 11\ repository.
solaris   repository name             Oracle\ Solaris\ 11
```

ローカルリポジトリのカスタマイズ

`pkgrecv` コマンドを使用すると、パッケージおよびそのパブリッシャーのデータをリポジトリに追加できます。`pkgrepo` コマンドを使用すると、リポジトリからパッケージおよびパブリッシャーを削除できます。

リポジトリへのパッケージの追加

リポジトリにパブリッシャーを追加できます。たとえば、`solaris`、`ha-cluster`、および `solarisstudio` パッケージを 1 つのリポジトリに保持できます。

カスタムパッケージを追加する場合は、カスタムパブリッシャー名でそれらのパッケージを公開します。カスタムパッケージを既存のパブリッシャー (solaris など) として公開しないでください。パブリッシャーが指定されていないパッケージを公開すると、それらのパッケージはそのリポジトリのデフォルトパブリッシャーに追加されます。カスタムパッケージを正しいデフォルトパブリッシャーでテストリポジトリに公開します。その後、pkgrecv コマンドを使用して、それらのパッケージおよびそのパブリッシャー情報を本稼動リポジトリに追加します。手順については、『Oracle Solaris 11.2 での Image Packaging System を使用したソフトウェアのパッケージ化と配布 』の「パッケージを発行する」を参照してください。

次の例では、isvpub パブリッシャーのデータおよび ISVproducts.p5p パッケージアーカイブからのすべてのパッケージをローカルリポジトリに追加しています。*パッケージアーカイブ*は、パブリッシャーの情報と、そのパブリッシャー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 11.2 での Image Packaging System を使用したソフトウェアのパッケージ化と配布 』の「パッケージアーカイブファイルとしての配布」を参照してください。pkgrepo のほとんどの操作は、パッケージアーカイブでは使用できません。パッケージアーカイブには、パッケージが含まれていますが、リポジトリ構成は含まれていません。ただし、pkgrepo list および pkgrepo contents コマンドは、パッケージアーカイブに対して動作します。pkgrepo contents コマンドについては、46 ページの「リポジトリ内のパッケージの検査」で説明しています。

pkgrepo list 出力にパブリッシャーが表示されます。これは、このパブリッシャーが、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パブリッシャーでないためです。

```
$ pkgrepo -s /tmp/ISVproducts.p5p list
PUBLISHER NAME                                    O VERSION
isvpub    isvtool                                   1.1,5.11:20131120T021902Z
isvpub    isvtool                                   1.0,5.11:20131120T010105Z
```

次の pkgrecv コマンドは、ソースリポジトリからすべてのパッケージを取得します。取得するパッケージの名前を指定する場合、または '*' 以外のパターンを指定する場合は、-r オプションを指定して、必要なすべての依存するパッケージを取得してください。

```
$ pkgrecv -s /tmp/ISVproducts.p5p -d /export/IPSpkgrepos/Solaris '*'
Processing packages for publisher isvpub ...
Retrieving and evaluating 2 package(s)...
PROCESS        ITEMS      GET (MB)      SEND (MB)
Completed        2/2       0.0/0.0          0.0/0
```

リポジトリの内容を変更したら、そのリポジトリをリフレッシュして、このリポジトリ用に構成されているパッケージ集積サーバーのサービスインスタンスを再起動します。

```
$ pkgrepo -s /export/IPSpkgrepos/Solaris refresh -p isvpub
Initiating repository refresh.
```

```
$ svcadm refresh pkg/server:solaris
$ svcadm restart pkg/server:solaris
```

次の pkgrepo info コマンドは 1 つのパッケージを表示します。これは、取得した 2 つのパッケージが同じパッケージの異なるバージョンであったためです。pkgrepo list コマンドは両方のパッケージを表示します。

```
$ pkgrepo -s /export/IPSpkgrepos/Solaris info
PUBLISHER PACKAGES STATUS           UPDATED
solaris   4768     online           2014-01-02T19:19:06.983979Z
isvpub    1        online           2014-03-20T23:24:37.196773Z
$ pkgrepo -s /export/IPSpkgrepos/Solaris list -p isvpub
PUBLISHER NAME                                    O VERSION
isvpub    isvtool                                   1.1,5.11:20131120T021902Z
isvpub    isvtool                                   1.0,5.11:20131120T010105Z
```

pkg set-publisher コマンドを使用して、isvpub パブリッシャーに新しいリポジトリの場所を追加します。

このリポジトリを HTTP インタフェースから利用でき、ブラウザを使用してこのリポジトリの内容を表示する場合は、その場所のパブリッシャーを指定するとこの新しいパッケージを表示できます。たとえば、http://localhost:81/isvpub/ を指定できます。

リポジトリ内のパッケージの検査

44 ページの「リポジトリへのパッケージの追加」に示す pkgrepo info および pkgrepo list コマンドのほかに、pkgrepo contents コマンドを使用すると、リポジトリ内のパッケージの内容を検査できます。

単一のパッケージの場合、pkgrepo contents コマンドの出力は、pkg contents -m コマンドの出力と同じです。pkgrepo contents コマンドは指定したリポジトリ内で一致する各パッケージの出力を表示し、pkg contents コマンドはこのイメージにインストール可能な一致するパッケージのバージョンのみの出力を表示します。-t オプションを指定すると、pkgrepo contents コマンドは指定されたアクションのみを表示します。

次の例では、パッケージの 1 つのバージョンのみが指定されたリポジトリに存在するため、このパッケージのバージョンを指定する必要はありません。このパッケージには、Oracle Database 12 のインストールおよび操作に必要な一連の Oracle Solaris パッケージを提供する depend アクションが含まれています。

```
$ pkgrepo -s http://pkg.oracle.com/solaris/release/ \
```

```
contents -t depend oracle-rdbms-server-12cR1-preinstall
depend fmri=x11/library/libxi type=group
depend fmri=x11/library/libxtst type=group
depend fmri=x11/session/xauth type=group
depend fmri=compress/unzip type=require
depend fmri=developer/assembler type=require
depend fmri=developer/build/make type=require
```

リポジトリからのパッケージの削除

Oracle パブリッシャーによって配布されたパッケージを削除しないでください。『Oracle Solaris 11.2 ソフトウェアの追加と更新 』には、目的のパッケージのみをインストールし、それ以外のパッケージをインストールしない方法が示されています。

`pkgrepo remove` コマンドを使用すると、Oracle パブリッシャーから配布されたものではないを削除できます。`pkgrepo remove-publisher` コマンドを使用すると、パブリッシャーおよびそのパブリッシャーによって配布されたすべてのパッケージを削除できます。詳細は、`pkgrepo`(1)のマニュアルページを参照してください。36 ページの「ローカルの IPS パッケージリポジトリを更新する方法」の説明に従って、これらの操作はリポジトリのコピーに対して実行してください。

Web サーバーアクセスを使用した複数のリポジトリの提供

このセクションの手順では、31 ページの「ユーザーが HTTP インタフェースを使用してパッケージを取得できるようにする」に示す情報を拡張して、複数のリポジトリの提供をサポートする方法について説明します。

次の 2 種類の方法では、HTTP アクセスを使用して複数の IPS パッケージリポジトリを提供します。どちらの方法も、一意のリポジトリパスを使用して `pkg/server` サービスの追加のインスタンスを作成することから開始します。

- 複数の場所。ユーザーは別々の場所でページを表示することによって各リポジトリにアクセスします。
- 単一の場所。ユーザーは 1 つの場所からすべてのリポジトリにアクセスします。

複数のリポジトリへのアクセスを提供することに加えて、単一のリポジトリは、44 ページの「リポジトリへのパッケージの追加」に示すように、複数のパブリッシャーからのパッケージを提供できます。

▼ 別々の場所から複数のリポジトリを提供する方法

この例では、Solaris リポジトリのほかに SolarisStudio リポジトリが存在しています。pkg/server サービスの solaris インスタンスに指定されているように、Solaris リポジトリには、ポート 81 を使用して http://localhost/ からアクセスできます。31 ページの「ユーザーが HTTP インタフェースを使用してパッケージを取得できるようにする」を参照してください。

1. **新しい集積サーバーインスタンスを作成します。**
   svccfg コマンドの add サブコマンドを使用して、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 studio
   ```

2. **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。**

   ```
   $ svcs pkg/server
   STATE  STIME    FMRI
   online 14:54:16 svc:/application/pkg/server:default
   online 14:54:20 svc:/application/pkg/server:studio
   online 14:54:20 svc:/application/pkg/server:solaris
   ```

3. **リポジトリへのパスを設定します。**
   サービスのこのインスタンスが、リポジトリのデータを検索できるパスを設定します。

   ```
   $ svccfg -s pkg/server:studio setprop pkg/inst_root=/export/IPSpkgrepos/SolarisStudio
   ```

4. **(オプション) 新しいインスタンスのポート番号を設定します。**

   ```
   $ svccfg -s pkg/server:studio setprop pkg/port=82
   ```

5. **(オプション) Apache のプロキシベースを設定します。**
   pkg/proxy_base の設定例については、57 ページの「プレフィックスを使用した単純なプロキシの構成」を参照してください。

6. **リポジトリ名および説明を設定します。**
   44 ページの「リポジトリのプロパティー値の変更」に示すように、リポジトリ名および説明が設定されていることを確認します。

7. **リポジトリサービスを開始します。**
   パッケージ集積サーバーサービスを再起動します。

   ```
   $ svcadm refresh pkg/server:studio
   $ svcadm enable pkg/server:studio
   ```

8. **リポジトリサーバーが動作しているかどうかをテストします。**

   `http://localhost:82/` の場所でブラウザウィンドウを開きます。

   ポート番号を設定していない場合、デフォルトは 80 です。`http://localhost:80/` または `http://localhost/` でリポジトリを表示します。

   そのポート番号が別の `pkg/server` インスタンスでも使用されている場合は、その場所にパブリッシャー名を付加して新しいパッケージを表示します。たとえば、`http://localhost:81/solarisstudio/` でリポジトリを表示します。

9. **パブリッシャーの起点を設定します。**

   パブリッシャーの起点に次のいずれかの値を設定します。

   - `pkg/inst_root` の場所。

     ```
     $ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/SolarisStudio/ \
     solarisstudio
     ```

   - `pkg/port` の場所。

     ```
     $ pkg set-publisher -G '*' -M '*' -g http://localhost:82/ solarisstudio
     ```

参照 異なるプレフィックスを使用して 1 つのドメインでの複数のリポジトリの実行 (`http://pkg.example.com/solaris` および `http://pkg.example.com/studio` など) については、58 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。

▼ 単一の場所から複数のリポジトリを提供する方法

この手順のステップの多くは、前の手順のステップと同じです。詳細は、前の手順を参照してください。

1. **新しい集積サーバーインスタンスを作成します。**

2. **リポジトリへのパスを設定します。**

   特定の `pkg/depot` インスタンスによって管理される各 `pkg/server` インスタンスには、一意の `pkg/inst_root` 値が必要です。

3. **新しいインスタンスの `readonly` プロパティーを確認します。**

pkg/readonly プロパティーのデフォルト値は true です。この値が変更されている場合は、値を true にリセットします。

```
$ svcprop -p pkg/readonly pkg/server:studio
true
```

4. **新しいインスタンスの standalone プロパティーを設定します。**

   デフォルトでは、pkg/standalone プロパティーの値は true です。pkg/standalone プロパティーが false に設定されている pkg/server インスタンスは、pkg/depot サービスインスタンスによって同じ場所から提供できます。

```
$ svccfg -s pkg/server:studio
svc:/application/pkg/server:studio> setprop pkg/standalone=false
svc:/application/pkg/server:studio> refresh
svc:/application/pkg/server:studio> select solaris
svc:/application/pkg/server:solaris> setprop pkg/standalone=false
svc:/application/pkg/server:solaris> refresh
svc:/application/pkg/server:solaris> exit
$
```

   pkg/standalone プロパティーが false に設定されている pkg/server の各インスタンスで、pkg/inst_root プロパティーの値が一意であることを確認します。

5. **(オプション) pkg/depot インスタンスのポート番号を設定します。**

   デフォルトでは、svc:/application/pkg/depot:default サービスのポート番号は 80 です。このポート番号は、この pkg/depot インスタンスによって管理される pkg/server インスタンスのポート番号と同じにできます。ポート番号を変更するには、pkg/depot:default の config/port プロパティーを設定します。

6. **pkg/depot インスタンスを再起動します。**

```
$ svcadm refresh pkg/depot:default
$ svcadm restart pkg/depot:default
```

7. **リポジトリサーバーが動作しているかどうかをテストします。**

   ユーザーが http://localhost:80/ を開くと、solaris パブリッシャーがリストされている http://localhost/solaris リポジトリ、および solarisstudio パブリッシャーがリストされている http://localhost/studio リポジトリが表示されます。

   1 つのリポジトリが複数のパブリッシャーのパッケージを提供する場合は、すべてのパブリッシャーがリストされます。たとえば、http://localhost/solaris リポジトリが solaris および isvpub パブリッシャーとともに表示される場合があります。

8. **パブリッシャーの起点を設定します。**

   パブリッシャーの起点に次のいずれかの値を設定します。

   - 一意の pkg/inst_root の場所。

     ```
     $ pkg set-publisher -G '*' -M '*' -g /export/IPSpkgrepos/SolarisStudio/ \
     solarisstudio
     ```

   - config/port の値および pkg/server インスタンス名によって定義される場所。

     ```
     $ pkg set-publisher -G '*' -M '*' -g http://localhost:80/studio/ solarisstudio
     ```

次の手順 35 ページの「ローカルリポジトリの更新」および44 ページの「ローカルリポジトリのカスタマイズ」で説明されているように、pkg/depot インスタンスによって管理されるリポジトリの内容を変更する場合、次の両方の手順を実行します。

- リポジトリに対して pkgrepo refresh を実行します。
- pkg/depot インスタンスで svcadm restart を実行します。

各インスタンスが 1 つ以上のリポジトリをホストする pkg/depot サービスに追加のインスタンスを作成できます。

pkg/server および pkg/depot のサービスインスタンスを構成するのではなく、スタンドアロンの構成を生成する場合は、pkg.depot-config(1M)のマニュアルページを参照してください。

♦ ♦ ♦ 第 5 章

5 Web サーバーの背後での集積サーバーの実行

Apache 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と、次の利点があります。

- 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。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。
- パフォーマンスと可用性が向上します。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および可用性を向上できます。
- セキュアなリポジトリサーバーを提供できます。Secure Sockets Layer (SSL) プロトコルを有効にした、クライアント証明書をサポートする Apache インスタンスの背後で集積サーバーを実行します。

集積サーバーの Apache 構成

この章の例では、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。`svc:/network/http:apache22` サービスを有効にして、Apache Web サーバーをアクティブ化します。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。

Oracle Solaris 11.2 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。通常、次のコマンドを使用すると `httpd.conf` ファイルを見つけることができます。

```
$ pkg search -Hl -o path ':file:path:*httpd.conf'
etc/apache2/2.2/httpd.conf
etc/apache2/2.2/original/httpd.conf
```

必要な Apache 構成の設定

パッケージ集積サーバーを Apache Web サーバーインスタンスの背後で実行する場合は、エンコードされているスラッシュをデコードしないように、次の設定を `httpd.conf` ファイルに含めます。

```
AllowEncodedSlashes NoDecode
```

スラッシュは階層パッケージ名を表すために使用されるため、パッケージ名にスラッシュがエンコードされた URL を含めることができます。たとえば、パッケージ名 `pkg://solaris/developer/build/make` は、Web サーバーでは `http://pkg.oracle.com/solaris/release/manifest/0/developer%2Fbuild%2Fmake` になります。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、`%2F` でエンコードされたスラッシュをデコードしないように Apache に指示します。

この設定を省略すると、`404 Not Found` エラーになることがあり、検索機能に悪影響を及ぼす場合があります。

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。

メタデータの転送サイズを減らします。

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
```

追加のパイプライン化された要求を許可します。

`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。

```
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、時間がかな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
```

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。カタログ属性ファイル (56 ページの「カタログ属性ファイルに対するキャッシュの考慮事項」を参照) およびリポジトリの検索結果 (56 ページの「検索に対するキャッシュの考慮事項」を参照) を除き、提供されるすべてのファイルは一意であるため、必要な場合に応じて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す。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>
```

プレフィックスを使用した単純なプロキシの構成

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ーの設定については、47 ページの「Web サーバーアクセ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、集積サーバーを構成します。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
```

nocanon を使用して、URL の正規化を抑止します。この設定は、検索を正常に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
```

Oracle Solaris の SSL カーネルプロキシ、および SSL を使用した Web サーバー通信の暗号化と高速化については、『Oracle Solaris 11.2 でのネットワークのセキュリティー保護 』の第 3 章「Web サーバーと Secure Sockets Layer プロトコル」を参照してください。

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。57 ページの「プレフィックスを使用した単純なプロキシの構成」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。負荷分散の利点の 1 つは、リポジトリの可用性が向上することです。このセクションでは、負荷分散の 2 つの例を示します。

リポジトリが負荷分散されている場合、ロードバランサがクライアントをあるノードから別のノードに切り替えるときの問題を避けるために、すべてのリポジトリのカタログがまったく同じである必

要があります。カタログがまったく同じであることを確認するには、39 ページの「複数の同一のローカルリポジトリの保守」の説明に従って、負荷分散に関係しているリポジトリをクローニングします。

負荷分散に対応した 1 つのリポジトリサーバー

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。

57 ページの「プレフィックスを使用した単純なプロキシの構成」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 つの集積を示しています。集積のスレッド数を設定する方法の例については、57 ページの「プレフィックスを使用した単純なプロキシの構成」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
```

負荷分散に対応した 1 つのリポジトリサーバーと負荷分散に対応しない 1 つのリポジトリサーバー

この例は、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
```

HTTPS でのリポジトリアクセスの構成

クライアントは、HTTP を介してパッケージを提供するように構成されているリポジトリからパッケージをダウンロードできます。場合によっては、アクセスを制限する必要があります。アクセスを制限する 1 つの方法は、クライアント証明書をサポートする、SSL を有効にした Apache インスタンスの背後で集積サーバーを実行することです。

SSL の使用には次の利点があります。

- クライアントとサーバー間での暗号化されたパッケージデータの転送を保証する
- クライアントがサーバーに提示する証明書に基づく、リポジトリへのアクセス権の付与を可能にする

セキュアなリポジトリサーバーを設定するには、カスタム証明書チェーンを作成する必要があります。

1. 証明書チェーンの先頭である認証局 (CA) を作成します。

2. リポジトリへのアクセスが許可されるクライアントに、この CA から証明書を発行します。

CA のコピーが 1 つリポジトリサーバーに格納されます。クライアントがサーバーに証明書を提示すると、そのクライアント証明書がサーバーの CA に対して検証され、アクセスを許可するかどうかが判断されます。

このセクションでは、証明書チェーンを作成し、Apache のフロントエンドを構成して、クライアント証明書を検証するための次の手順について説明します。

- キーストアの作成
- クライアント証明書の認証局の作成
- Apache 構成ファイルへの SSL 構成の追加
- 自己署名付きサーバー認証局の作成
- PKCS12 キーストアの作成

Oracle Solaris での Apache Web サーバーの権限の詳細は、『Oracle Solaris 11.2 でのユーザーとプロセスのセキュリティー保護 』の「拡張特権を使用したリソースのロックダウン」を参照してください。

キーストアの作成

証明書および鍵を管理するには、キーストアを作成します。キーストアには、CA、CA 鍵、およびクライアント証明書と鍵が格納されます。

キーストアの管理に使用するツールは `pktool` です。詳細は、`pktool`(1) のマニュアルページを参照してください。

`pktool` のデフォルトのキーストアの場所は `/var/user/`*username* です。ここで、*username* は現在のシステムユーザーの名前です。キーストアを複数のユーザーが管理する場合は、このキーストアのデフォルトの場所が問題になることがあります。また、IPS パッケージリポジトリの管理では、証明書の混乱を避けるために専用のキーストアを使用してください。IPS パッケージリポジトリの `pktool` キーストアのカスタムの場所を設定するには、環境変数 `SOFTTOKEN_DIR` を設定します。複数のキーストアを管理するために、必要に応じて、`SOFTTOKEN_DIR` 変数をリセットします。

次のコマンドを使用して、キーストアのディレクトリを作成します。複数のユーザーがキーストアを管理する必要がある場合は、所有者、グループ、および権限を適切に設定します。

```
$ mkdir /path-to-keystore
```

```
$ export SOFTTOKEN_DIR=/path-to-keystore
```

キーストアへのアクセスは、pktool コマンドを呼び出すたびに入力する必要があるパスフレーズによって保護されます。新しく作成されたキーストアのデフォルトのパスフレーズは changeme です。よりセキュアなパスフレーズにするために、changeme パスフレーズを変更してください。

次のコマンドを使用して、キーストアのパスフレーズ (PIN) を設定します。

```
$ pktool setpin
Enter token passphrase: changeme
Create new passphrase:
Re-enter new passphrase:
Passphrase changed.
$ ls /path-to-keystore
pkcs11_softtoken
```

クライアント証明書の認証局の作成

CA は証明書チェーンの最上位の証明書です。CA は、クライアント証明書の生成、およびクライアントによるリポジトリへのアクセスのために提示する証明書の検証を行うために必要となります。

サードパーティーの CA は、VeriSign など信頼できる少数の企業によって管理されます。この信頼できる管理によって、クライアントはいずれかの CA に対してサーバーの識別情報の検証が可能になります。このセクションの例には、リポジトリサーバーの識別情報の検証は含まれていません。この例は、クライアント証明書の検証のみを示しています。このため、この例では、自己署名付き証明書を使用して CA を作成しており、サードパーティーの CA を使用しません。

CA では共通名 (CN) が必要です。1 つのリポジトリのみを実行する場合は、CN に「Oracle Software Delivery」などの組織の名前を設定できます。複数のリポジトリがある場合は、各リポジトリに独自の CA が必要です。この場合は、CA を作成しているリポジトリを一意に識別する名前を CN に設定します。たとえば、リリースリポジトリとサポートリポジトリがある場合、リリースリポジトリへのアクセスが許可されるのはリリース CA からの証明書のみであり、サポートリポジトリへのアクセスが許可されるのはサポート CA からの証明書のみです。

キーストア内の証明書を識別するには、証明書に説明ラベルを設定します。証明書ラベルは *CN*_ca に設定することをお勧めします。ここで、*CN* は証明書の CN です。

次のコマンドを使用して、CA 証明書を作成します。ここで、*name* は証明書の CN であり、*CAlabel* は証明書のラベルです。

```
$ pktool gencert label=CAlabel subject="CN=name" serial=0x01
```

CA はキーストアに格納されます。次のコマンドを使用して、キーストアの内容を表示します。

```
$ pktool list
```

65 ページの「Apache 構成ファイルへの SSL 構成の追加」の説明に従って Apache を構成する場合は、キーストアから CA 証明書を抽出する必要があります。次のコマンドを使用して、`ca_file.pem` という名前のファイルに CA 証明書を抽出します。

```
$ pktool export objtype=cert label=CAlabel outformat=pem \
outfile=ca_file.pem
```

リポジトリへのアクセスに使用されるクライアント証明書の作成

CA を生成したら、クライアント証明書を生成できます。

証明書署名要求を生成する

クライアント証明書を生成するには、証明書署名要求 (CSR) を生成します。CSR には、サーバーにセキュアに渡す必要があるすべての情報が含まれています。

ユーザーが発行した有効な証明書をクライアントが持っているかどうかを確認するだけの場合は、情報をエンコードする必要はありません。クライアントが証明書をサーバーに提示すると、サーバーは証明書を CA に対して検証し、そのクライアント証明書がユーザーによって生成されたかどうかを検証します。ただし、SSL ではその CSR の `subject` が要求されます。サーバーにほかの情報を渡す必要がない場合は、`subject` に証明書が発行された国を設定できます。たとえば、`subject` に `C=US` を設定できます。

クライアントのユーザー名を証明書にエンコードして、サーバーがクライアントを識別できるようにすることをお勧めします。ユーザー名は、リポジトリへのアクセス権の付与対象となるユーザーの名前です。CN はこのために使用できます。64 ページの「証明書の鍵を抽出する」の説明に従って、最終的な証明書の鍵を見つけて抽出できるように、この CSR にラベルを指定します。

次のコマンドを使用して CSR を生成します。

```
$ pktool gencsr subject="C=US,CN=username" label=label format=pem \
outcsr=cert.csr
```

次の OpenSSL コマンドを使用して、`cert.csr` ファイル内の CSR を検査します。

```
$ openssl req -text -in cert.csr
```

CSR に署名する

証明書を作成するには、CA によって CSR が署名されている必要があります。CSR に署名するには、次の情報を提供します。

- 62 ページの「クライアント証明書の認証局の作成」に示すように、`gencert` コマンドを使用して CA を作成したときに `subject` に使用したものと同じ文字列を証明書の `issuer` に設定します。
- 16 進数のシリアル番号を設定します。この例では、CA のシリアル番号は `0x01` と指定されているため、最初のクライアント証明書のシリアル番号は `0x02` になります。新しいクライアント証明書を生成するたびにシリアル番号を増分します。

  各 CA およびその子孫クライアントの証明書には、独自の一連のシリアル番号があります。キーストアに複数の CA が構成されている場合は、クライアント証明書のシリアル番号を正しく設定してください。
- `signkey` をキーストア内の CA のラベルに設定します。
- `outcert` を証明書ファイルの名前に設定します。アクセス対象のリポジトリのあとに証明書および鍵を指定することをお勧めします。

次のコマンドを使用して CSR に署名します。

```
$ pktool signcsr signkey=CAlabel csr=cert.csr  \
serial=0x02 outcert=reponame.crt.pem issuer="CN=name"
```

証明書は *reponame*`.crt.pem` ファイルに作成されます。次の OpenSSL コマンドを使用して証明書を検査します。

```
$ openssl x509 -text -in reponame.crt.pem
```

証明書の鍵を抽出する

キーストアからこの証明書の鍵を抽出します。63 ページの「証明書署名要求を生成する」で CSR を生成するために `gencsr` を実行したときに指定したものと同じラベル値を `label` に設定します。次のコマンドを使用して、キーストアから鍵をエクスポートします。

```
$ pktool export objtype=key label=label outformat=pem \
outfile=reponame.key.pem
```

SSL 保護されたリポジトリにアクセスする必要があるクライアントシステムに、証明書および鍵を転送します。

クライアントシステムから保護されたリポジトリにアクセスできるようにする

SSL 保護されたリポジトリにアクセスするには、クライアントシステムは証明書および鍵のコピーを保持し、パブリッシャーの構成に証明書および鍵を指定する必要があります。

証明書 (*reponame*`.crt.pem`) および鍵 (*reponame*`.key.pem`) を各クライアントシステムにコピーします。たとえば、各クライアントの `/var/pkg/ssl` ディレクトリにこれらをコピーできます。

次のコマンドを使用して、生成された証明書および鍵をパブリッシャーの構成に指定します。

```
$ pkg set-publisher -k reponame.key.pem -c reponame.crt.pem \
-p https://repolocation
```

SSL 認証は、HTTPS のリポジトリ URI でのみサポートされます。ファイルリポジトリの URI では、SSL 認証はサポートされません。

Apache 構成ファイルへの SSL 構成の追加

リポジトリでクライアント証明書ベースの認証を使用するには、53 ページの「集積サーバーの Apache 構成」の説明に従って、最初に汎用の集積サーバーの Apache の構成を設定します。`httpd.conf` ファイルの最後に次の SSL 構成を追加します。

```
# Let Apache listen on the standard HTTPS port
Listen 443

# VirtualHost configuration for request on port 443
<VirtualHost 0.0.0.0:443>
        # DNS domain name of the server, needs to match your server certificate
        ServerName pkg-sec.example.com

        # enable SSL
        SSLEngine On

        # Location of the server certificate and key.
        # You either have to get one from a certificate signing authority like
        # VeriSign or create your own CA for testing purposes (see "Creating a
        # Self-Signed CA for Testing Purposes")
        SSLCertificateFile /path/to/server.crt
        SSLCertificateKeyFile /path/to/server.key
```

```
        # Intermediate CA certificate file. Required if your server certificate
        # is not signed by a top-level CA directly but an intermediate authority
        # Comment out this section if you are using a test certificate or your
        # server certificate doesn't require it.
        # For more info:
        # http://httpd.apache.org/docs/2.2/mod/mod_ssl.html#sslcertificatechainfile
        SSLCertificateChainFile /path/to/ca_intermediate.pem

        # CA certs for client verification.
        # This is where the CA certificate created in step 3 needs to go.
        # If you have multiple CAs for multiple repos, just concatenate the
        # CA certificate files
        SSLCACertificateFile /path/to/ca_cert.pem

        # If the client presents a certificate, verify it here. If it doesn't,
        # ignore.
        # This is required to be able to use client-certificate based and
        # anonymous SSL traffic on the same VirtualHost.
        # This statement could also go into the <Location> tags but putting it
        # here avoids re-negotiation which can cause security issues with older
        # servers/clients:
        # http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2009-3555
        SSLVerifyClient optional

        <Location /repo>
                SSLVerifyDepth 1
                # This is the SSL requirement for this location.
                # Requirements can be made based on various information encoded
                # in the certificate. Two variants are the most useful for use
                # with IPS repositories:
                # a) SSLRequire ( %{SSL_CLIENT_I_DN_CN} =~ m/reponame/ )
                #    only allow access if the CN in the client certificate matches
                #    "reponame", useful for different certificates for different
                #    repos
                #
                # b) SSLRequire ( %{SSL_CLIENT_VERIFY} eq "SUCCESS" )
                #    grant access if clients certificate is signed by one of the
                #    CAs specified in SSLCACertificateFile
                SSLRequire ( %{SSL_CLIENT_VERIFY} eq "SUCCESS" )

                # proxy request to depot running at internal.example.com:12345
                ProxyPass http://internal.example.com:12345 nocanon max=500
        </Location>
</VirtualHost>
```

自己署名付きサーバー認証局の作成

テスト目的では、サードパーティーの CA ではなく、自己署名付きサーバー認証局 (CA) を使用できます。Apache の自己署名付きサーバー CA を作成する手順は、62 ページの「クライ

アント証明書の認証局の作成」で説明されているクライアント証明書のために CA を作成する手順と非常に似ています。

次のコマンドを使用して、サーバー CA を作成します。subject をサーバーの DNS 名に設定します。

```
$ pktool gencert label=apacheCA subject="CN=apachetest" \
serial=0x01
```

次のコマンドを使用して、サーバー CA の CSR を作成します。サーバーに複数の名前でアクセスできる場合、または IP アドレスで直接利用できるようにする場合は、OpenSSL のドキュメントの Subject Alternative Name の説明に従って、subjectAltNames ディレクティブを使用します。

```
$ pktool gencsr label=apache subject="CN=pkg-sec.internal.example.com" \
altname="IP=192.168.1.1,DNS=pkg-sec.internal.example.com" \
format=pem outcsr=apache.csr
```

次のコマンドを使用して CSR に署名します。SSLCertificateFile には server.crt を使用します。

```
$ pktool signcsr signkey=apacheCA csr=apache.csr serial=0x02 \
outcert=server.crt issuer="CN=apachetest"
```

次のコマンドを使用して鍵を抽出します。SSLCertificateKeyFile には server.key を使用します。

```
$ pktool export objtype=key label=apache outformat=pem \
outfile=server.key
```

クライアントがこのサーバー鍵を受け入れるようにするには、クライアントシステムの受入済み CA ディレクトリに CA 証明書 (apacheCA) を追加し、ca-certificates サービスを再起動して、OpenSSL に必要なリンクを作成します。

次のコマンドを使用して CA 証明書を抽出します。

```
$ pktool export label=apacheCA objtype=cert outformat=pem \
outfile=test_server_ca.pem
```

CA 証明書をクライアントマシンの CA 証明書ディレクトリにコピーします。

```
$ cp /path-to/test_server_ca.pem /etc/certs/CA/
```

CA 証明書サービスを再起動します。

```
$ svcadm refresh ca-certificates
```

続行する前に、新しい CA 証明書がリンクされていることを確認します。リフレッシュしたあと、ca-certificate サービスは /etc/openssl/certs ディレクトリにリンクを再構築します。次のコマンドを実行して、新しい CA 証明書がリンクされているかどうかを確認します。

```
$ ls -l /etc/openssl/certs | grep test_server_ca.pem
lrwxrwxrwx   1 root     root          40 May  1 09:51 e89d96e0.0 -> ../../certs/CA/
test_server_ca.pem
```

ハッシュ値 e89d96e0.0 は、証明書の subject に基づいているため、異なる場合があります。

Firefox でセキュアなリポジトリにアクセスするための PKCS12 キーストアの作成

63 ページの「リポジトリへのアクセスに使用されるクライアント証明書の作成」で作成した PEM 証明書は、pkg クライアントでのセキュアなリポジトリへのアクセスに使用できます。ただし、ブラウザユーザーインタフェース (BUI) にアクセスするには、Firefox がインポートできる形式に証明書および鍵を変換する必要があります。Firefox は PKCS12 キーストアを受け入れます。

次の OpenSSL コマンドを使用して、Firefox の PKCS12 キーストアを作成します。

```
$ openssl pkcs12 -export -in /path-to/certificate.pem \
-inkey /path-to/key.pem -out name.p12
```

作成した PKCS12 キーストアをインポートするには、次の Firefox のメニュー、タブ、およびボタンを選択します: 「編集」>「設定」>「詳細」>「暗号化」>「証明書を表示」>「認証局証明書」>「インポート」。

一度に 1 つの証明書をインポートします。

完全にセキュアなリポジトリの例

この例では、repo1、repo2、および repo3 という名前の 3 つのセキュアなリポジトリを構成します。repo1 および repo2 リポジトリは、専用の証明書を使用して構成します。このため、repo1 の証明書は repo2 では機能せず、repo2 の証明書は repo1 では機能しません。repo3 リポジトリは、どちらの証明書も受け入れるように構成します。

この例では、Apache インスタンスの適切なサーバー証明書がすでに利用可能であることを想定しています。Apache インスタンスのサーバー証明書がない場合は、66 ページの「自己署名付きサーバー認証局の作成」のテスト証明書の作成の手順を参照してください。

3 つのリポジトリが、`https://pkg-sec.example.com/repo1`、`https://pkg-sec.example.com/repo2`、および `https://pkg-sec.example.com/repo3` に設定されます。これらのリポジトリは、`http://internal.example.com` でポート 10001、10002、および 10003 にそれぞれ設定されている集積サーバーを指しています。`SOFTTOKEN_DIR` 環境変数が、61 ページの「キーストアの作成」の説明に従って正しく設定されていることを確認します。

▼ セキュアなリポジトリを構成する方法

1. **repo1 の CA 証明書を作成します。**

```
$ pktool gencert label=repo1_ca subject="CN=repo1" serial=0x01
$ pktool export objtype=cert label=repo1_ca outformat=pem \
outfile=repo1_ca.pem
```

2. **repo2 の CA 証明書を作成します。**

```
$ pktool gencert label=repo2_ca subject="CN=repo2" serial=0x01
$ pktool export objtype=cert label=repo2_ca outformat=pem \
outfile=repo2_ca.pem
```

3. **結合した CA 証明書ファイルを作成します。**

```
$ cat repo1_ca.pem > repo_cas.pem
$ cat repo2_ca.pem >> repo_cas.pem
$ cp repo_cas.pem /path-to-certs
```

4. **ユーザー myuser がリポジトリ repo1 にアクセスできる 1 つのクライアント証明書/鍵のペアを作成します。**

```
$ pktool gencsr subject="C=US,CN=myuser" label=repo1_0001 format=pem \
outcsr=repo1_myuser.csr
$ pktool signcsr signkey=repo1_ca csr=repo1_myuser.csr  \
serial=0x02 outcert=repo1_myuser.crt.pem issuer="CN=repo1"
$ pktool export objtype=key label=repo1_0001 outformat=pem \
outfile=repo1_myuser.key.pem
$ cp repo1_myuser.key.pem /path-to-certs
$ cp repo1_myuser.crt.pem /path-to-certs
```

5. **ユーザー myuser がリポジトリ repo2 にアクセスできる 1 つのクライアント証明書/鍵のペアを作成します。**

```
$ pktool gencsr subject="C=US,CN=myuser" label=repo2_0001 format=pem \
```

```
outcsr=repo2_myuser.csr
$ pktool signcsr signkey=repo2_ca csr=repo2_myuser.csr  \
serial=0x02 outcert=repo2_myuser.crt.pem issuer="CN=repo2"
$ pktool export objtype=key label=repo2_0001 outformat=pem \
outfile=repo2_myuser.key.pem
$ cp repo2_myuser.key.pem /path-to-certs
$ cp repo2_myuser.crt.pem /path-to-certs
```

6. **Apache を構成します。**

   `httpd.conf` ファイルの最後に次の SSL 構成を追加します。

```
# Let Apache listen on the standard HTTPS port
Listen 443

<VirtualHost 0.0.0.0:443>
        # DNS domain name of the server
        ServerName pkg-sec.example.com

        # enable SSL
        SSLEngine On

        # Location of the server certificate and key.
        # You either have to get one from a certificate signing authority like
        # VeriSign or create your own CA for testing purposes (see "Creating a
        # Self-Signed CA for Testing Purposes")
        SSLCertificateFile /path/to/server.crt
        SSLCertificateKeyFile /path/to/server.key

        # Intermediate CA certificate file. Required if your server certificate
        # is not signed by a top-level CA directly but an intermediate authority.
        # Comment out this section if you don't need one or if you are using a
        # test certificate
        SSLCertificateChainFile /path/to/ca_intermediate.pem

        # CA certs for client verification.
        # This is where the CA certificate created in step 3 needs to go.
        # If you have multiple CAs for multiple repos, just concatenate the
        # CA certificate files
        SSLCACertificateFile /path/to/certs/repo_cas.pem

        # If the client presents a certificate, verify it here. If it doesn't,
        # ignore.
        # This is required to be able to use client-certificate based and
        # anonymous SSL traffic on the same VirtualHost.
        SSLVerifyClient optional

        <Location /repo1>
                SSLVerifyDepth 1
                SSLRequire ( %{SSL_CLIENT_I_DN_CN} =~ m/repo1/ )
                # proxy request to depot running at internal.example.com:10001
                ProxyPass http://internal.example.com:10001 nocanon max=500
        </Location>
```

```
        <Location /repo2>
                SSLVerifyDepth 1
                SSLRequire ( %{SSL_CLIENT_I_DN_CN} =~ m/repo2/ )
                # proxy request to depot running at internal.example.com:10002
                ProxyPass http://internal.example.com:10002 nocanon max=500
        </Location>

        <Location /repo3>
                SSLVerifyDepth 1
                SSLRequire ( %{SSL_CLIENT_VERIFY} eq "SUCCESS" )
                # proxy request to depot running at internal.example.com:10003
                ProxyPass http://internal.example.com:10003 nocanon max=500
        </Location>

</VirtualHost>
```

7. **repo1 へのアクセスをテストします。**

```
$ pkg set-publisher -k /path-to-certs/repo1_myuser.key.pem \
-c /path-to-certs/repo1_myuser.crt.pem \
-p https://pkg-sec.example.com/repo1/
```

8. **repo2 へのアクセスをテストします。**

```
$ pkg set-publisher -k /path-to-certs/repo2_myuser.key.pem \
-c /path-to-certs/repo2_myuser.crt.pem \
-p https://pkg-sec.example.com/repo2/
```

9. **repo3 へのアクセスをテストします。**

repo1 証明書を使用して、repo3 へのアクセスをテストします。

```
$ pkg set-publisher -k /path-to-certs/repo1_myuser.key.pem \
-c /path-to-certs/repo1_myuser.crt.pem \
-p https://pkg-sec.example.com/repo3/
```

repo2 証明書を使用して、repo3 へのアクセスをテストします。

```
$ pkg set-publisher -k /path-to-certs/repo2_myuser.key.pem \
-c /path-to-certs/repo2_myuser.crt.pem \
-p https://pkg-sec.example.com/repo3/
```

索引

S

W

Z