Image Packaging System のマニュアルページ

Part No: E25753
2012 年 3 月、E25753-03

120411@25097

目次

はじめに

このドキュメントは、Oracle Solaris 11 システムインストールツールのマニュアルページを提供します。

概要

次に、マニュアルページの各セクションと、そこに含まれている情報について簡単について説明します。

- セクション 1 では、オペレーティングシステムで利用可能なコマンドについて説明します。
- セクション 1M では、システムの保守および管理目的で主に使用するコマンドについて説明します。
- セクション 5 には、文字セット表などのその他のドキュメントが含まれています。

以下に、マニュアルページの一般的な形式を示します。一般に、各マニュアルセクションのマニュアルページはこの順序に従いますが、必要な見出しのみが含まれています。たとえば、レポートするバグがない場合は、「使用上の留意点」セクションはありません。一般的なマニュアルページの詳細については、`man` コマンドを参照してください。

名前
: このセクションでは、ドキュメント化されたコマンドまたは関数の名前に続いて、その動作について簡単な説明を示します。

形式
: このセクションでは、コマンドまたは関数の構文を示します。コマンドまたはファイルが標準パスに存在しない場合は、そのフルパス名が表示されます。異なる引数順序が必要でないかぎり、オプションおよび引数はアルファベット順 (1 番目に単一文字の引数、次に引数が付いたオプション) で表示されます。

  このセクションでは、次の特殊文字が使用されています。

  [ ]
  : 角括弧。角括弧で囲まれたオプションまたは引数は省略可能です。角括弧を省略した場合は、引数を指定する必要があります。

. . .	省略記号。前の引数に複数の値を指定したり、前の引数を複数回指定したりできます (例: *filename* . . .)。
|	区切り文字。この文字で区切られた引数のうち 1 つのみを一度に指定できます。
{ }	中括弧。中括弧で囲まれたオプション・引数は相互に依存しており、1 つの組として取り扱う必要があります。

プロトコル　このセクションは、サブセクション 3R でのみ発生し、プロトコルの説明ファイルを示します。

機能説明　このセクションでは、サービスの機能および動作を定義します。つまり、コマンドの動作について簡潔に説明しています。オプションについて説明したり、例を示したりはしていません。対話型のコマンド、サブコマンド、要求、マクロ、および関数については、「使用法」で説明されています。

オプション　このセクションでは、各オプションの動作についての簡潔なサマリーとともに、コマンドのオプションを一覧表示します。オプションは文字順および「形式」セクションで表示される順序で一覧表示されます。オプションに指定可能な引数については「オプション」で説明され、該当する場合はデフォルト値が示されます。

オペランド　このセクションでは、コマンドのオペランドを一覧表示し、コマンドの動作への影響について説明します。

出力　このセクションでは、コマンドで生成される出力 (標準出力、標準エラー、または出力ファイル) について説明します。

戻り値　マニュアルページに値を戻す関数について記載されている場合、このセクションでは、これらの値を一覧表示し、値が戻される条件について説明します。関数が定数値 (0 や -1 など) のみを戻すことができる場合は、これらの値がタグ付きの段落に一覧表示されます。それ以外の場合は、単一の段落で各関数の戻り値について説明します。void が宣言された関数は値を戻さないため、「戻り値」では説明しません。

エラー　大部分の関数は失敗時に、エラーの原因を示すエラーコードをグローバル変数 `errno` に配置します。このセクションでは、関数が生成できるすべてのエラーコードをアルファベット順に一覧表示し、各エラーが発生する条件について説明します。複数の条件で同じエラーが発生する可能性がある場合は、エラーコードの下にある個別の段落で各条件について説明します。

使用法	このセクションでは、詳細な説明を必要とする特別な規則、機能、およびコマンドを一覧表示します。ここで一覧表示するサブセクションを使用して、組み込み機能について説明します。 コマンド 修飾子 変数 式 入力文法
使用例	このセクションでは、コマンドまたは関数の使用例または使用方法を示します。可能なかぎり、コマンド行エントリおよびマシン応答を含む完全な例が表示されます。例が示される場合は、常にプロンプトは `example%` と表示され、ユーザーがスーパーユーザーになる必要がある場合は、`example#` と表示されます。例に続いて、説明、変数の置換規則、または戻り値が表示されます。ほとんどの例は、「形式」、「機能説明」、「オプション」、および「使用法」に記載されたコンセプトを例証しています。
環境変数	このセクションでは、コマンドまたは関数によって影響を受ける環境変数を一覧表示したあとに、その影響について簡潔に説明します。
終了ステータス	このセクションでは、コマンドが呼び出し元のプログラムまたはシェルに戻す値、およびそれらの値が戻される条件を一覧表示します。通常、正常に完了した場合はゼロが戻され、さまざまなエラー状況の場合はゼロ以外が戻されます。
ファイル	このセクションでは、マニュアルページで参照されるすべてのファイル名、対象のファイル、およびコマンドによって作成または要求されるファイルを一覧表示します。各ファイルのあとに、説明的なサマリーまたは説明が続きます。
属性	このセクションでは、属性タイプおよび対応する値を定義して、コマンド、ユーティリティー、およびデバイスドライバの特性を一覧表示します。詳細については、`attributes(5)` のマニュアルページを参照してください。
関連項目	このセクションでは、その他のマニュアルページ、Oracle のドキュメント、および社外出版物への参照を一覧表示します。
診断	このセクションでは、エラーが発生する条件についての簡単な説明とともに、診断メッセージを一覧表示します。

警告	このセクションでは、作業状況に著しく影響を与える可能性のある特別な条件についての警告を一覧表示します。これは診断の一覧ではありません。
注意事項	このセクションでは、ページのどこにも属さない追加情報を一覧表示します。これはユーザーへの余談という形式を取り、特別に関心がある点について扱います。重大な情報については、ここでは扱いません。
使用上の留意点	このセクションでは、既知のバグについて説明し、可能な場合は回避方法を示します。

参 照

ユーザーコマンド

名前　packagemanager – Image Packaging System の GUI

形式

```
/usr/bin/packagemanager [options]

/usr/bin/packagemanager [-hiRu] [--help]
    [--info-install file] [--update-all]
    [--image-dir dir]

/usr/bin/packagemanager [file]
```

機能説明

packagemanager は、 pkg(5) (Image Packaging System ソフトウェア) のグラフィカルユーザーインタフェースです。

パッケージマネージャーを使用すると、次の作業を実行できます。

- パッケージの検索、インストール、および削除
- パブリッシャーの追加、削除、および変更
- ブート環境の作成、削除、および管理

file オペランドが指定されていて、その接尾辞が .p5i である場合、packagemanager は Web インストールモードで起動し、1 つ以上のパブリッシャーおよびパブリッシャーごとに数多くのパッケージが追加されます。

オプション　サポートしているオプションは、次のとおりです。

-h or --help
使用方法に関するメッセージを表示します。

-i または --info-install *file*
.p5i ファイルを指定して、packagemanager を Web インストールモードで実行できます。*file* の接尾辞は .p5i である必要があります。

-R or --image-dir *dir*
自動的に検出されたイメージではなく、*dir* をルートとするイメージに対して処理を行います。

-U または --update-all
利用可能な更新があるすべてのインストール済みパッケージを更新します。

注 – package/pkg、package/pkg/package-manager 、または package/pkg/update-manager を更新する必要がある場合、packagemanager は最初にこれらのパッケージを更新してから、残りの更新を実行するために再起動されます。

オペランド

file　Web インストールファイル。このファイルの接尾辞は .p5i である必要があります。Web インストールについての詳細は、パッケージマネージャーのオンラインヘルプを参照してください。

使用例

例1 現在のイメージで操作する

現在のイメージで packagemanager を起動します。

```
$ packagemanager
```

例2 指定されたイメージで操作する

/aux0/example_root に格納されたイメージで packagemanager を起動します。

```
$ packagemanager -R /aux0/example_root
```

例3 Webインストールモードで起動する

Webインストールモードで packagemanager を起動します。

```
$ packagemanager ~/test.p5i
```

終了ステータス

次の終了ステータスが返されます。

0 すべてが動作しました。

1 エラーが発生した。

2 無効なコマンド行オプションが指定された。

ファイル

pkg(5) イメージはより大きなファイルシステム内に任意に配置できるため、トークン $IMAGE_ROOT を使用して相対パスが区別されます。標準のシステムインストールでは、$IMAGE_ROOT は / と同等です。

$IMAGE_ROOT/var/pkg	完全または部分的なイメージのメタデータディレクトリ。
$IMAGE_ROOT/.org.opensolaris,pkg	ユーザーイメージのメタデータディレクトリ。 特定のイメージのメタデータ内にある特定のファイルおよびディレクトリには、修復時や復旧時に役立つ情報が含まれています。トークン $IMAGE_META は、メタデータが含まれる最上位ディレクトリを参照するために使用されます。通常、 $IMAGE_META は前述の 2 つのパスのいずれかです。
$IMAGE_META/gui-cache	プログラムの起動やパブリッシャーの切り替えの速度を上げるために packagemanager で保持されるキャッシュ済みメタデータの場所。

$IMAGE_META ディレクトリ階層内のその他のパスは非公開であり、変更される可能性があります。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg/package-manager
インタフェースの安定性	不確実

関連項目

pm-updatemanager(1)、pkg(1)、pkg(5)

パッケージマネージャーのオンラインヘルプ

http://hub.opensolaris.org/bin/view/Project+pkg/

注意事項

packagemanager は、イメージのファイルおよびディレクトリで操作するために十分な権限で起動する必要があります。

名前

pkg – Image Packaging System 取得クライアント

形式

```
/usr/bin/pkg [options] command [cmd_options] [operands]

/usr/bin/pkg refresh [--full] [publisher ...]

/usr/bin/pkg install [-nvq] [-g path_or_uri ...] [--accept]
    [--licenses] [--no-be-activate] [--no-index] [--no-refresh]
    [--no-backup-be | --require-backup-be] [--backup-be-name name]
    [--deny-new-be | --require-new-be] [--be-name name]
    [--reject pkg_fmri_pattern ...] pkg_fmri_pattern ...

/usr/bin/pkg uninstall [-nvq] [--no-be-activate] [--no-index]
    [--no-backup-be | --require-backup-be] [--backup-be-name name]
    [--deny-new-be | --require-new-be] [--be-name name]
    pkg_fmri_pattern ...

/usr/bin/pkg update [-fnvq] [-g path_or_uri ...] [--accept]
    [--licenses] [--no-be-activate] [--no-index] [--no-refresh]
    [--no-backup-be | --require-backup-be] [--backup-be-name name]
    [--deny-new-be | --require-new-be] [--be-name name]
    [--reject pkg_fmri_pattern ...] [pkg_fmri_pattern ...]

/usr/bin/pkg list [-Hafnsuv] [-g path_or_uri ...]
    [--no-refresh] [pkg_fmri_pattern ...]

/usr/bin/pkg info [-lr] [-g path_or_uri ...] [--license]
    [pkg_fmri_pattern ...]

/usr/bin/pkg contents [-Hmr] [-a attribute=pattern ...]
    [-g path_or_uri ...] [-o attribute ...] [-s sort_key]
    [-t action_type ...] [pkg_fmri_pattern ...]

/usr/bin/pkg search [-HIaflpr] [-o attribute ...]
    [-s repo_uri] query

/usr/bin/pkg verify [-Hqv] [pkg_fmri_pattern ...]

/usr/bin/pkg fix [--accept] [--licenses] [pkg_fmri_pattern ...]

/usr/bin/pkg revert [-nv] [--no-be-activate]
    [--no-backup-be | --require-backup-be] [--backup-be-name name]
    [--deny-new-be | --require-new-be] [--be-name name]
    (--tagged tag-name ... | path-to-file ...)

/usr/bin/pkg mediator [-aH] [-F format] [mediator ...]

usr/bin/pkg set-mediator [-nv] [-I implementation]
    [-V version] [--no-be-activate]
    [--no-backup-be | --require-backup-be] [--backup-be-name name]
    [--deny-new-be | --require-new-be] [--be-name name]
    mediator ...

/usr/bin/pkg unset-mediator [-nvIV] [--no-be-activate]
    [--no-backup-be | --require-backup-be] [--backup-be-name name]
```

```
    [--deny-new-be | --require-new-be] [--be-name name]
    mediator ...

/usr/bin/pkg variant [-H] [variant.variant_name ...]

/usr/bin/pkg change-variant [-nvq] [-g path_or_uri ...]
    [--accept] [--licenses] [--no-be-activate]
    [--no-backup-be | --require-backup-be] [--backup-be-name name]
    [--deny-new-be | --require-new-be] [--be-name name]
    variant_name=value ...

/usr/bin/pkg facet [-H] [facet_name ...]

/usr/bin/pkg change-facet [-nvq] [-g path_or_uri ...]
    [--accept] [--licenses] [--no-be-activate]
    [--no-backup-be | --require-backup-be] [--backup-be-name name]
    [--deny-new-be | --require-new-be] [--be-name name]
    facet_name=[True|False|None] ...

/usr/bin/pkg avoid [pkg_fmri_pattern ...]

/usr/bin/pkg unavoid [pkg_fmri_pattern ...]

/usr/bin/pkg freeze [-n] [-c reason] [pkg_fmri_pattern] ...

/usr/bin/pkg unfreeze [-n] [pkg_name_pattern] ...

/usr/bin/pkg property [-H] [propname ...]

/usr/bin/pkg set-property propname propvalue

/usr/bin/pkg add-property-value propname propvalue

/usr/bin/pkg remove-property-value propname propvalue

/usr/bin/pkg unset-property propname ...

/usr/bin/pkg publisher [-HPn] [publisher ...]

/usr/bin/pkg set-publisher [-Ped] [-k ssl_key] [-c ssl_cert]
    [-g origin_to_add | --add-origin origin_to_add ...]
    [-G origin_to_remove | --remove-origin origin_to_remove ...]
    [-m mirror_to_add | --add-mirror mirror_to_add ...]
    [-M mirror_to_remove | --remove-mirror mirror_to_remove ...]
    [--enable] [--disable] [--no-refresh]
    [--reset-uuid] [--non-sticky] [--sticky]
    [--search-after publisher] [--search-before publisher]
    [--search-first]
    [--approve-ca-cert path_to_CA]
    [--revoke-ca-cert hash_of_CA_to_remove]
    [--unset-ca-cert hash_of_CA_to_remove]
    [--set-property name_of_property=value]
    [--add-property-value name_of_property=value_to_add]
    [--remove-property-value name_of_property=value_to_remove]
    [--unset-property name_of_property_to_delete]
    publisher
```

```
/usr/bin/pkg set-publisher -p repo_uri
    [-Ped] [-k ssl_key] [-c ssl_cert]
    [--non-sticky] [--sticky]
    [--search-after publisher] [--search-before publisher]
    [--search-first]
    [--approve-ca-cert path_to_CA]
    [--revoke-ca-cert hash_of_CA_to_remove]
    [--unset-ca-cert hash_of_CA_to_remove]
    [--set-property name_of_property=value]
    [--add-property-value name_of_property=value_to_add]
    [--remove-property-value name_of_property=value_to_remove]
    [--unset-property name_of_property_to_delete]
    [publisher]

/usr/bin/pkg unset-publisher publisher ...

/usr/bin/pkg history [-Hl] [-t [time | time-time],...]
    [-o column,...] [-n number]

/usr/bin/pkg purge-history

/usr/bin/pkg rebuild-index

/usr/bin/pkg update-format

/usr/bin/pkg version

/usr/bin/pkg help

/usr/bin/pkg image-create [-FPUfz] [--force]
    [--full | --partial | --user] [--zone]
    [-k ssl_key] [-c ssl_cert]
    [--no-refresh] [--variant variant_name=value ...]
    [-g path_or_uri | --origin path_or_uri ...]
    [-m uri | --mirror uri ...]
    [--set-property name_of_property=value]
    [--facet facet_name=(True|False) ...]
    [(-p | --publisher) [name=]repo_uri] dir
```

機能説明

pkg は Image Packaging System 用の取得クライアントです。有効な構成では、pkg を呼び出すことにより、パッケージをインストールする場所を作成し、イメージを呼び出し、パッケージをイメージにインストールすることができます。パッケージはパブリッシャーによって公開され、パブリッシャーはそれらのパッケージを 1 つ以上のリポジトリで、またはパッケージアーカイブで入手可能にすることができます。pkg はパブリッシャーのリポジトリまたはパッケージアーカイブからパッケージを取得し、パッケージをイメージにインストールします。

パブリッシャーの名前によって、人、人のグループ、または組織が 1 つ以上のパッケージのソースとして識別されます。パブリッシャーの名前の競合を避け、パブリッシャーを識別しやすくするために、パッケージを公開するエンティティーを表すドメイン名をパブリッシャーの名前として使用することがベストプラクティスです。

リポジトリは、クライアントがパッケージの内容 (プログラムやドキュメントなど、パッケージの内部に含まれるファイル) とメタデータ (パッケージの名前や説明など、パッケージについての情報) を公開および取得できる場所です。たとえば、example.org という名前のパブリッシャーは、http://example.org/repository という URI にリポジトリを配置することができます。

pkg では、パッケージをアンインストールしたり、(利用可能なパッケージの一覧などの) パブリッシャーメタデータを更新したり、イメージにインストールされたパッケージを検証したり、イメージからさまざまなトークンを照会したりすることもできます。これらのクエリーは pkg(5) リポジトリで構成することもできます。

イメージには 3 つの種類があります。完全なシステムを提供できるフルイメージ、フルイメージ (親イメージ) にリンクされるがそれ自身では完全なシステムを提供しない部分イメージ、およびユーザーイメージです。

オプション

サポートしているオプションは、次のとおりです。

-R *dir*
: *dir* をルートとするイメージを操作します。ディレクトリが指定されなかったか、または環境に基づいて決定される場合、デフォルトは / です。詳細は、「環境変数」のセクションを参照してください。

--help or -?
: 使用方法に関するメッセージを表示します。

サブコマンド

サポートされているサブコマンドは次のとおりです。

refresh [--full] [*publisher* ...]
: 指定されたパブリッシャーごとに、クライアントの利用可能パッケージの一覧およびパブリッシャーメタデータを更新します。パブリッシャーを指定しない場合、すべてのパブリッシャーを対象に操作が実行されます。

  --full を指定すると、増分更新を試みる代わりにすべてのパブリッシャーメタデータを強制的に完全取得し、操作中に使用されるすべてのプロキシでキャッシュデータを無視するように要求します。このオプションはトラブルシューティング目的に用意されており、通常時は使用しないでください。

install [-nvq] [-g *path_or_uri* ...] [--accept] [--licenses] [--no-be-activate] [--no-index] [--no-refresh] [--no-backup-be | --require-backup-be] [--backup-be-name *name*] [--deny-new-be | --require-new-be] [--be-name *name*] [--reject *pkg_fmri_pattern* ...] *pkg_fmri_pattern* ...
: パッケージをインストールし、イメージにインストールされたパッケージで許容されている *pkg_fmri_pattern* と一致する最新バージョンにパッケージを更新します。パッケージの最新バージョンを明示的に要求するには、*pkg_fmri_pattern* のバージョン部分に latest を使用します。たとえば、vim@latest のように指定します。

インストールプロセスの間に、一部の構成ファイルの名前変更または置換が行われる場合があります。どのファイルを保持するかをパッケージシステムが決定する方法と、パッケージ操作中にファイルが保持されるしくみについては、pkg(5) のマニュアルページのファイルアクションに関する項目を参照してください。

パッケージが回避リストにある場合は、インストールすると回避リストから削除されます。

-g を指定すると、パッケージデータの取得元である指定のパッケージリポジトリまたはアーカイブが、イメージ内のソースのリストに一時的に追加されます。指定されたソースからのパッケージが、イメージ内に構成されているパブリッシャーからも入手できる場合でも、クライアントは指定されたソースのみからパッケージの内容を取得します。どのバージョンのパッケージを使用するかを決定するときは、イメージ内に構成されているけれども指定されたソースに見つからないパブリッシャーが優先されます。インストールまたは更新のあとに、パブリッシャーによって提供され、イメージ内に見つからないパッケージがある場合は、起点なしでイメージ構成に追加されます。このオプションは複数回指定できます。

-n を指定すると、パッケージの変更は行わずに試しに操作を実行します。

-q を指定すると、要求された操作の実行中、進捗状況メッセージを表示しません。

-v を指定すると、要求された操作の実行中に詳細な進捗状況メッセージを出力し、詳細な計画情報 (ファセット、メディエータ、バリアントの変更など) を表示します。このオプションを複数回指定して、表示される計画情報の量を増やすことができます。

--accept は、更新またはインストールされるパッケージのライセンス条項に同意することを示します。このオプションを指定しないと、パッケージのライセンスに同意が必要になった場合、インストール操作は失敗します。

--licenses を指定すると、この操作の一環としてインストールまたは更新されるパッケージのすべてのライセンスを表示します。

--no-backup-be を指定すると、バックアップブート環境を作成しません。

--no-be-activate を指定すると、ブート環境が作成される場合に、それを次回ブート時にアクティブ BE として設定しません。詳細は、beadm(1M) を参照してください。

--no-index が指定されている場合、操作が正常に完了したあとに検索インデックスを更新しません。

--no-refresh を指定すると、入手可能パッケージやその他のメタデータの最新リストを取得するために、イメージのパブリッシャー用のリポジトリへのアクセスを試みません。

--backup-be-name を指定すると、指定された引数を使用して、作成されたバックアップブート環境に名前を付けます。--backup-be-name を使用すると --require-backup-be が暗黙的に指定されます。beadm(1M) も参照してください。

--be-name を指定すると、新たに作成されたブート環境の名前を、指定された引数になるように変更します。--be-name の使用は、暗黙的に --require-new-be を示します。beadm(1M) も参照してください。

--require-backup-be を指定すると、新しいブート環境が作成されない場合に常にバックアップブート環境を作成します。このオプションを指定しないと、イメージポリシーに基づいてバックアップブート環境が作成されます。バックアップブート環境がいつ自動的に作成されるかについての説明は、次の「イメージプロパティー」の be-policy を参照してください。

--require-new-be を指定すると、常に新しいブート環境を作成します。このオプションを指定しないと、イメージポリシーに基づいてブート環境が作成されます。ブート環境がいつ自動的に作成されるかについての説明は、次の「イメージプロパティー」の be-policy を参照してください。このオプションを --require-backup-be と組み合わせることはできません。

--deny-new-be を指定すると、新しいブート環境を作成しません。新しいブート環境が必要な場合、この操作は実行されません。

--reject を指定すると、指定されたパターンと一致する名前を持つパッケージはインストールされません。一致するパッケージがすでにインストールされている場合、それらはこの操作の一環として削除されます。グループ依存関係のターゲットである拒否対象パッケージは回避リストに登録されます。

uninstall [-nvq] [--no-be-activate] [--no-index] [--no-backup-be | --require-backup-be] [--backup-be-name *name*] [--deny-new-be | - -require-new-be] [--be-name *name*] *pkg_fmri_pattern* ...

pkg_fmri_pattern に一致するインストール済みパッケージを削除します。

パッケージがグループ依存関係の対象である場合、パッケージをアンインストールするとそのパッケージは回避リストに登録されます。後述する avoid サブコマンドを参照してください。

その他すべてのオプションの使用法および効果については、前述した install コマンドを参照してください。

update [-fnvq] [-g *path_or_uri* ...] [--accept] [--licenses] [--no-be-activate] [--no-index] [--no-refresh] [--no-backup-be | --require-backup-be] [--backup-be-name *name*] [--deny-new-be | --require-new-be] [--be-name *name*] [--reject *pkg_fmri_pattern* ...] [*pkg_fmri_pattern* ...]

引数を指定しないか、または提供されたパターンの 1 つがアスタリスク (*) である場合、現在のイメージ内のすべてのインストール済みパッケージを、インストール済みパッケージおよびパブリッシャー構成によってシステムに適用される

制約で許可される最新のバージョンに更新します。パッケージの最新バージョンを明示的に要求するには、*pkg_fmri_pattern* のバージョン部分に `latest` を使用します。たとえば、`vim@latest` のように指定します。

pkg_fmri_pattern を指定した場合、`update` は、*pkg_fmri_pattern* に一致するインストール済みパッケージを、パターンによって許可され、さらにインストール済みパッケージおよびパブリッシャー構成によってシステムに課せられる制約によって許可される、最新のバージョンに置き換えます。すでにインストールされているものより古い、または新しいバージョンを指定することで、特定のパッケージのインプレースダウングレードまたはアップグレードを実行できます。パッケージの名前変更または廃止の境界をまたがった特定パッケージの更新はサポートされていません。

保持される構成ファイルのうち、`update` によってダウングレードされるパッケージの一部であり、元のバージョンがインストールされたあとに変更されたファイルは、拡張子 `.update` を使用して名前が変更されます。どのファイルを保持するかをパッケージシステムが決定する方法と、パッケージアップグレード中にファイルが保持されるしくみについては、`pkg`(5) のマニュアルページのファイルアクションに関する項目を参照してください。

`-f` オプションを指定すると、すべてのインストール済みパッケージを更新するときにクライアントに対する最新状態チェックを実行しません。

その他すべてのオプションの使用法および効果については、前述した `install` コマンドを参照してください。

`list [-Hafnsuv] [-g` *path_or_uri* `...] [--no-refresh] [`*pkg_fmri_pattern* `...]`

引数が指定されていない場合、バージョンやインストール状態などの情報を含む、現在のイメージ内のパッケージのリストを表示します。引数が指定されている場合、指定されたパッケージの情報を表示します。デフォルトでは、異なるアーキテクチャーまたはゾーンタイプのパッケージバリアントは除外されます。通常の出力は 3 列形式です。

```
NAME (PUBLISHER)            VERSION          IFO
system/core-os              0.5.11-0.169     i--
x11/wm/fvwm (fvwm.org)      2.6.1-3          i--
```

最初の列にはパッケージの名前が表示されます。パッケージのインストール元 (または、インストールされていない場合は提供元) であるパブリッシャーがパブリッシャー検索順で先頭でない場合、パッケージ名のあとに一覧表示されるパブリッシャー名は括弧で囲まれています。2 番目の列にはパッケージのリリースバージョンとブランチバージョンが表示されます。リリースバージョンとブランチバージョンについて、およびバリアントについては、`pkg`(5) のマニュアルページを参照してください。

最後の列には、パッケージのステータスを示す一連のフラグが表示されます。

- I 列の i は、パッケージがインストールされていることを示します。
- F 列の f は、パッケージが凍結されていることを示します。
- O 列の o は、パッケージが廃止されていることを示します。O 列の r は、パッケージの名前が変更されたことを示します (廃止の形態の 1 つです)。

-H が指定されている場合は、リストのヘッダーを省略します。

-a を指定すると、インストール済みパッケージと、インストールのために入手可能なパッケージの最新バージョンを一覧表示します。インストール済みの incorporation によって、またイメージのバリアントによって許可されている場合、そのパッケージはインストールのために入手可能であるとみなされます。1 つ以上のパターンを指定した場合、指定されたパターンに一致し、インストール済みの incorporation およびイメージのバリアントによって許可されている最新バージョンが一覧表示されます。-a を指定しない場合、インストール済みパッケージのみを一覧表示します。

-f および -a を指定した場合、incorporation の制約またはインストール状態に関係なく、すべてのバリアントについてすべてのパッケージのすべてのバージョンを一覧表示します。これらのオプションを使用するとき、パッケージの最新バージョンを明示的に一覧表示するには、*pkg_fmri_pattern* のバージョン部分に latest を使用します。たとえば、vim@latest のように指定します。

-g を指定すると、指定されたパッケージリポジトリまたはアーカイブを、操作のためのパッケージデータのソースとして使用します。このオプションは複数回指定できます。-n を指定しない場合、-g を使用すると -a が暗黙的に指定されます。

-n を指定すると、インストール状態に関係なく、すべての既知のパッケージの最新バージョンを表示します。

-s を指定すると、パッケージ名とサマリーを示す 1 行の短縮形式を表示します。このオプションは -a、-n、-u、または -v とともに使用できます。

-u を指定すると、新しいバージョンが入手可能なパッケージのみを一覧表示します。このオプションは -g とともに使用できません。

-v を指定すると、パブリッシャーと完全バージョンを含む、フルパッケージ FMRI をすべて最初の列に表示します (VERSION 列は消えます)。このオプションは -a、-n、または -u とともに使用できます。

--no-refresh を指定すると、入手可能なパッケージの最新リストを取得するために、イメージのパブリッシャー用のリポジトリへのアクセスを試みません。

`info [-lr] [-g` *path_or_uri* `...] [--license] [`*pkg_fmri_pattern* `...]`
パッケージについての情報を人間が判読できる形式で表示します。複数の FMRI パターンを指定できます。パターンを指定しない場合、イメージ内のすべてのインストール済みパッケージについての情報を表示します。

`-g` を指定すると、指定されたパッケージリポジトリまたはアーカイブを、操作のためのパッケージデータのソースとして使用します。このオプションは複数回指定できます。`-g` を使用すると `-r` が暗黙的に指定されます。

`-l` を指定すると、インストール済みパッケージの情報のみを表示します。これはデフォルトです。

`-r` を指定すると、最新の入手可能バージョンに基づいてパッケージを照合し、イメージの構成済みパブリッシャーのリポジトリから、現在インストールされていないパッケージの情報を取得します (必要な場合)。このオプションを使用するときは、少なくとも 1 つのパッケージを指定する必要があります。`-r` を指定しない場合、デフォルトでインストール済みパッケージのみが表示されます。

`--license` を指定すると、パッケージのライセンステキストを表示します。このオプションは `-l` または `-r` と組み合わせることができます。

`contents [-Hmr] [-a` *attribute=pattern* `...] [-g` *path_or_uri* `...] [-o` *attribute*`,...] [-s` *sort_key*`] [-t` *action_type* `...] [`*pkg_fmri_pattern* `...]`
パッケージの内容 (アクション属性) を表示します。オプションまたはオペランドが指定されていない場合、現在のイメージにインストールされているアクションの `path` 属性の値をアルファベット順で並べ替えて表示します。アクションとそれらの属性については、`pkg`(5) のマニュアルページのアクションに関する項目を参照してください。下の疑似属性名のリストも参照してください。

`-a` を指定すると、名前がオプション引数で指定され、値がオプション引数 (属性名と等号に続く部分) の (glob) パターンに一致する属性を持つアクションに出力を限定します。複数の `-a` オプションを指定した場合、それらのいずれかに一致するアクションが表示されます。

`-g` が指定されている場合、指定されたパッケージリポジトリまたはアーカイブからこのイメージにインストール可能なパッケージの情報を表示します。インストール可能なパッケージには、現在インストールされているパッケージと、バリアントやファセットの制限などのこのイメージへのインストールの条件を満たすその他のパッケージが含まれます。このオプションは複数回指定できます。`-g` を使用すると `-r` が暗黙的に指定されます。

`-m` が指定された場合、このイメージにインストールできないアクションを含めて、指定されたパッケージのすべてのアクションのすべての属性を表示します。

`-o` が指定されている場合、リストの先頭に表示されている値に従って並べ替えられた属性を表示します。`-o` オプションは複数回指定できます。または、属性名をコンマで区切ることにより、1 つの `-o` オプションの引数として複数の属性を指定できます。要求された属性を持つアクションのみが表示されます。

-r が指定されている場合、このイメージに構成された発行元のリポジトリからこのイメージにインストール可能なパッケージの最新バージョンの情報を表示します。インストール可能なパッケージには、現在インストールされているパッケージと、バリアントやファセットの制限などのこのイメージへのインストールの条件を満たすその他のパッケージが含まれます。このオプションを使用するときは、少なくとも 1 つのパッケージを指定する必要があります。

-s を指定すると、指定されたアクション属性によってアクションをソートします。このオプションを指定しない場合、デフォルトではパスによって、または -o オプションで最初に指定された属性によってソートします。-s オプションは複数回指定できます。

-t を指定すると、指定されたタイプのアクションのみを一覧表示します。複数のタイプをコンマ区切りリストで指定できます。このオプションは複数回指定できます。

-H が指定されている場合は、リストのヘッダーを省略します。

pkg_fmri_pattern が指定された場合、それらの名前付きパッケージの情報のみを表示します。

利便性のために、いくつかの特殊な疑似属性名を使用できます。

action.hash	アクションがペイロードを伝送する場合、アクションのハッシュの値です。
action.key	アクションのキー属性の値です。たとえば、file アクションの場合、キー属性はファイルのパスです。キー属性のないアクションもあります。
action.name	アクションの名前です。たとえば、ファイルアクションの場合、これは file です。
action.raw	一致するアクションのすべての属性。
pkg.fmri	アクションを包含しているパッケージのフル形式 FMRI (たとえば、pkg://solaris/web/amp@0.5.11,5.11-0.169:20110705T153434Z) です。
pkg.name	アクションを包含しているパッケージの名前 (たとえば、web/amp) です。
pkg.publisher	アクションを包含しているパッケージの発行元 (たとえば、solaris) です。
pkg.shortfmri	アクションを包含しているパッケージの短縮形式 FMRI (たとえば、pkg://solaris/web/amp@0.5.11,5.11-0.169) です。

関連するサブコマンドは contents および search であり、どちらもパッケージの内容についてシステムをクエリーします。contents サブコマンドは、1 つまたは複数のインストールされているか、インストール可能なパッケージ内のアクションを、指定されたオプションに基づいて出力をフィルタ処理して表示します。search サブコマンドは逆方向からクエリーを行い、ユーザーが指定したトークンを含むすべてのパッケージの名前を表示します。

各サブコマンドで実行できるクエリーの一部は、他方でも実行できます。サブコマンドの選択は慎重に行ってください。クエリーによっては、もう一方のほうがより自然に実行できる場合があります。

search [-HIaflpr] [-o *attribute*,...] [-s *repo_uri*] *query*

query の一致を検索し、結果を表示します。どのトークンがインデックス化されるかはアクションに依存しますが、コンテンツハッシュとパス名を含めることができます。アクションとそれらの属性については、pkg(5) のマニュアルページのアクションに関する項目を参照してください。上記の pkg contents および下記の -o の疑似属性名のリストも参照してください。

デフォルトでは、クエリーは完全一致する一連の条件として解釈されます。? および * 文字を glob(3C) 形式のワイルドカードとして使用でき、より柔軟なクエリー一致が可能になります。

-H を指定するとヘッダーを省略します。

-I を指定すると、大文字と小文字を区別する検索を使用します。

デフォルトでは、-a を一緒に指定すると、検索を実行し、一致したアクションについての情報を表示します。

search はデフォルトで、現在インストールされているバージョンよりも古いパッケージ、および現在の incorporation によって除外されているパッケージバージョンからの結果を取り除きます。パッケージバージョンに関係なくすべての結果を表示するには -f を使用します。

-l を指定すると、イメージのインストール済みパッケージを検索します。

-o を使用して結果の列を制御できます。-o オプションは複数回指定できます。または、属性名をコンマで区切ることにより、1 つの -o オプションの引数として複数の属性を指定できます。前述した疑似属性に加えて、検索結果用の次の属性が定義されています。

search.match	検索クエリーに一致した文字列に対応します。
search.match_type	検索クエリーに一致した文字列を含んでいた属性に対応します。

-p を指定すると、一部のアクションが各クエリー条件に一致するパッケージを表示します。このオプションを使用することは、クエリーの各条件を山括弧 (<>) で囲むことと等価です。<> 演算子についての説明は後述します。

デフォルトでは、-r を一緒に指定すると、イメージのパブリッシャーに対応するリポジトリを検索します。

-s を指定すると、指定された URI に位置する pkg(5) リポジトリを検索します。これは複数回指定できます。パッケージアーカイブはサポートされていません。

-l と -r (または -s) の両方を同時に指定できます。この場合、ローカル検索とリモート検索の両方が実行されます。

単純なトークン一致およびワイルドカード検索に加えて、より複雑なクエリー言語がサポートされています。単一引用符または二重引用符 (' または ") を使用することにより、語句を検索できます。pkg が実際に ' または " を認識するように、必ずシェルを考慮に入れてください。

AND と OR を使用する論理検索がサポートされています。フィールド (構造化された) クエリーがサポートされています。これらのための構文は *pkg_name*:*action_type*:*key*:*token* です。欠けているフィールドは、暗黙的にワイルドカード化されます。:basename:pkg の検索は、basename というキーを持ち、トークン pkg に一致するすべてのパッケージ内のすべてのアクションタイプに一致します。pkg_name および token フィールドでは明示的ワイルドカードがサポートされます。action_type および key は正確に一致する必要があります。

そのアクションを包含するパッケージにアクションを変換するには、<> を使用します。-a オプションを使用する場合、token を検索した結果は token に一致したアクションについての情報である一方で、<token> を検索した結果は、token に一致したアクションを包含しているパッケージの一覧です。

verify [-Hqv] [*pkg_fmri_pattern* ...]
現在のイメージ内でパッケージのインストールを検証します。関連するパブリッシャーの現在の署名ポリシーが ignore でない場合、各パッケージの署名がポリシーに基づいて検証されます。署名ポリシーが適用されるしくみについては、後述する「イメージプロパティー」の signature-policy で説明します。

-H を指定すると、検証の出力からヘッダーを省略します。

-q を指定すると何も出力しませんが、致命的なエラーがある場合はエラーを返します。

-v を指定すると、パッケージに関する情報メッセージを含めます。

fix [--accept] [--licenses] [*pkg_fmri_pattern* ...]
pkg verify で報告されたエラーをすべて修正します。インストール済みパッケージの内容は、独自の内容解析に基づいて検証されるため、ほかのプログラムの場合とは異なる結果が返されることがあります。

--accept は、更新またはインストールされるパッケージのライセンス条項に同意することを示します。このオプションを指定しないと、パッケージのライセンスに同意が必要になった場合、操作は失敗します。

--licenses を指定すると、この操作の一環としてインストールまたは更新されるパッケージのすべてのライセンスを表示します。

revert [-nv] [--no-be-activate] [--no-backup-be | --require-backup-be] [--backup-be-name *name*] [--deny-new-be | --require-new-be] [--be-name *name*] (--tagged *tag-name* ... | *path-to-file* ...)

ファイルを配布時の状態に戻します。特定の値でタグ付けされたすべてのファイル、または個別ファイルを元に戻すことができます。ファイルの所有権および保護も復元されます。

注意 – 一部の編集可能ファイルをデフォルト値に戻すと、システムがブート不可になったり、その他の異常動作の原因になったりする可能性があります。

その他すべてのオプションの使用法および効果については、前述した install コマンドを参照してください。

mediator [-aH] [-F *format*] [*mediator* ...]

すべてのメディエータの、または引数が使用されている場合は指定されたメディエータのみの、現在選択されているバージョンと実装を表示します。

-a を指定すると、現在インストールされているパッケージに設定可能なメディエーションを一覧表示します。

-F が指定されている場合は、代替の出力形式を指定します。現時点では tsv (タブ区切り値) のみが有効です。

-H が指定されている場合は、リストのヘッダーを省略します。

set-mediator [-nv] [-I *implementation*] [-V *version*] [--no-be-activate] [--no-backup-be | --require-backup-be] [--backup-be-name *name*] [--deny-new-be | - -require-new-be] [--be-name *name*] *mediator* ...

現在のイメージ内の指定されたメディエータのバージョンと実装を設定します。

-I は、使用するメディエート対象インタフェースの実装を設定します。デフォルトでは、バージョンが指定されない場合、すべての実装バージョンが許可されます。バージョンなしで実装を指定するには、アット記号 (@) を付加します。

-V は、使用するメディエート対象インタフェースのバージョンを設定します。

指定されたメディエータのバージョンと実装のどちらかまたは両方が現在入手できない場合、指定されたメディエータを使用するリンクはすべて削除されます。

その他すべてのオプションの使用方法および効果については、前述の install コマンドを参照してください。

`unset-mediator [-nvIV] [--no-be-activate ] [--no-backup-be | --require-backup-be] [--backup-be-name` *name*`] [--deny-new-be | --require-new-be] [--be-name` *name*`]` *mediator* `...`
指定されたメディエータのバージョンと実装をシステムデフォルトに戻します。

`-I` は、メディエート対象インタフェースの実装のみを元に戻します。

`-V` は、メディエート対象インタフェースのバージョンのみを元に戻します。

その他すべてのオプションの使用方法および効果については、前述の `install` コマンドを参照してください。

`variant [-H] [variant.`*variant_name* `...]`
引数が指定されていない場合、このイメージに設定されたすべてのバリアントの現在値を表示します。引数が指定されている場合、このイメージに設定され、指定されている各 `variant.`*variant_name* の値を表示します。

`-H` が指定されている場合は、リストのヘッダーを省略します。

バリアントの詳細については、`pkg`(5) のマニュアルページのファセットとバリアントに関する項目を参照してください。

`change-variant [-nvq] [-g` *path_or_uri* `...] [--accept] [--licenses] [--no-be-activate] [--no-backup-be | --require-backup-be] [--backup-be-name` *name*`] [--deny-new-be | - -require-new-be] [--be-name` *name*`]` *variant_name*=*value* `...`
現在のイメージに設定され、指定されているバリアントの値を変更します。

オプションの使用法および効果については、前述の `install` コマンドを参照してください。

バリアントの値を変更すると、パッケージの内容が削除、更新、またはインストールされることがあります。バリアントの値を変更すると、新しいイメージ構成を満たすために、パッケージ全体がインストール、更新、または削除されることもあります。バリアントの詳細については、`pkg`(5) のマニュアルページのファセットとバリアントに関する項目を参照してください。

`facet [-H] [`*facet_name* `...]`
引数が指定されていない場合、`pkg change-facet` コマンドを使用してこのイメージに明示的に設定されたすべてのファセットの現在の値を表示します。引数が指定されている場合、このイメージに設定され、指定された各 *facet_name* の値を表示します。

`-H` が指定されている場合は、リストのヘッダーを省略します。

ファセットの詳細については、`pkg`(5) のマニュアルページのファセットとバリアントに関する項目を参照してください。

`change-facet [-nvq] [-g` *path_or_uri* `...] [--accept] [--licenses] [--no-be-activate] [--no-backup-be | --require-backup-be] [--backup-be-name` *name*`] [--deny-new-be | - -require-new-be] [--be-name` *name*`]` *facet_name*`=[True|False|None] ...`

現在のイメージに設定され、指定されたファセットの値を変更します。

ファセットは `True` または `False` に設定できます。ファセットを `None` に設定すると、`True` のデフォルト値がそのファセットに適用されるため、ファセットに依存するすべてのアクションがインストールされます。アクションについては、`pkg(5)` のマニュアルページのアクションに関する項目を参照してください。

オプションの使用法および効果については、前述の `install` コマンドを参照してください。

ファセットの値を変更すると、パッケージの内容が削除、更新、またはインストールされることがあります。ファセットの値を変更すると、新しいイメージ構成を満たすために、パッケージ全体がインストール、更新、または削除されることもあります。ファセットの詳細については、`pkg(5)` のマニュアルページのファセットとバリアントに関する項目を参照してください。

`avoid [`*pkg_fmri_pattern* `...]`

指定されたパターンに現在一致するパッケージ名を回避リストに登録することにより、それらがグループ依存関係のターゲットである場合にそれらを回避します。現在インストールされていないパッケージのみを回避できます。パッケージが現在グループ依存関係のターゲットである場合、パッケージをアンインストールするとそのパッケージは回避リストに登録されます。

引数を指定しない場合、回避対象の各パッケージを、そのパッケージにグループ依存関係を持つパッケージとともに表示します。

回避リストに登録されているパッケージは、要求された依存関係を満たすために必要であればインストールされます。その依存関係が削除された場合、パッケージはアンインストールされます。

`unavoid [`*pkg_fmri_pattern* `...]`

指定されたパッケージを回避リストから削除します。回避リストに登録されており、インストール済みパッケージのグループ依存関係に一致するパッケージは、このサブコマンドを使用して削除できません。グループ依存性に一致するパッケージを回避リストから削除するには、パッケージをインストールします。

引数を指定しない場合、回避対象パッケージの一覧を表示します。

`freeze [-n] [-c` *reason*`] [`*pkg_fmri_pattern*`] ...`

指定されたパッケージを指定されたバージョンに凍結します。バージョンを指定しない場合、パッケージがインストールされている必要があり、そのインストール済みバージョンで凍結されます。凍結されているパッケージをインストールまたは更新するときは、凍結された時点のバージョンと一致する

バージョンである必要があります。たとえば、パッケージが 1.2 で凍結された場合、1.2.1、1.2.9、1.2.0.0.1 などのバージョンに更新することはできます。そのパッケージは 1.3 または 1.1 で終了することはできません。*pkg_fmri_pattern* に指定された発行元を使用して、一致するパッケージを検索します。ただし、パブリッシャー情報は凍結の一環として記録されません。パッケージは発行元ではなくバージョンのみに関して凍結されます。すでに凍結されているパッケージを凍結すると、新しく指定されたバージョンによって凍結バージョンが置き換えられます。

パッケージを提供しない場合、現在凍結されているパッケージについての情報 (パッケージ名、バージョン、パッケージがいつ凍結されたか、関連付けられた理由があればその理由) が表示されます。

パッケージを凍結しても、そのパッケージを削除できなくなるわけではありません。パッケージが削除される場合に警告は表示されません。

`-c` を指定すると、凍結されるパッケージとともに理由を記録します。凍結が原因でインストールまたは更新に失敗する場合、その理由が示されます。

`-n` を指定すると、操作を試しに実行し、凍結されるパッケージの一覧を表示しますが、実際にはどのパッケージも凍結しません。

`unfreeze [-n] [`*pkg_name_pattern* `] ...`
凍結によって適用される制約を、指定されたパッケージから削除します。バージョンを提供しても無視されます。

`-n` を指定すると、凍結解除を試しに実行し、凍結解除されるパッケージの一覧を表示しますが、実際にはどのパッケージも凍結解除しません。

`property [-H] [`*propname* `...]`
イメージプロパティー情報を表示します。引数を指定しない場合、すべてのイメージプロパティーの名前および値を表示します。プロパティー名の特定のリストが要求された場合、それらのプロパティーの名前および値を表示します。イメージプロパティーの説明については、下の「イメージプロパティー」を参照してください。

`-H` が指定されている場合は、リストのヘッダーを省略します。

`set-property` *propname propvalue*
既存のイメージプロパティーを更新するか、または新しいイメージプロパティーを追加します。

`add-property-value` *propname propvalue*
既存のイメージプロパティーに値を追加するか、または新しいイメージプロパティーを追加します。

`remove-property-value` *propname propvalue*
既存のイメージプロパティーから値を削除します。

`unset-property` *propname* `...`
既存のイメージプロパティーを削除します。

`publisher [-HPn] [`*publisher* `...]`
発行元情報を表示します。引数を指定しない場合、すべてのパブリッシャー、それらの起点 URI、およびミラーの一覧を、検索の優先順に従って表示します。特定のパブリッシャーが要求された場合、それらのパブリッシャーの詳細な構成を表示します。

`-H` が指定されている場合は、リストのヘッダーを省略します。

`-P` を指定すると、パブリッシャー検索順の先頭のパブリッシャーのみを表示します。

`-n` を指定すると、有効になっているパブリッシャーのみを表示します。

`set-publisher [-Ped] [-k` *ssl_key*`] [-c` *ssl_cert*`] [-g` *origin_to_add* `| --add-origin` *origin_to_add* `...] [-G` *origin_to_remove* `| --remove-origin` *origin_to_remove* `...] [-m` *mirror_to_add* `| - -add-mirror` *mirror_to_add* `...] [-M` *mirror_to_remove* `| --remove-mirror` *mirror_to_remove* `...] [--enable ] [--disable] [--no-refresh] [--reset-uuid ] [--non-sticky] [--sticky] [--search-after` *publisher*`] [--search-before` *publisher*`] [--search-first] [--approve-ca-cert` *path_to_CA*`] [--revoke-ca-cert` *hash_of_CA_to_remove*`] [--unset-ca-cert` *hash_of_CA_to_remove*`] [--set-property` *name_of_property*`=`*value*`] [--add-property-value` *name_of_property*`=`*value_to_add*`] [--remove-property-value` *name_of_property*`=` *value_to_remove*`] [--unset-property` *name_of_property_to_delete* `] `*publisher*
既存のパブリッシャーを更新するか、パッケージパブリッシャーを追加します。検索順に影響するオプションを指定しない場合、新しいパブリッシャーは検索順の末尾に付加され、最後に検索されます。

`-P` または `--search-first` を指定すると、指定されたパブリッシャーを検索順の先頭に設定します。新しいパッケージをインストールするとき、このパブリッシャーが最初に検索されます。インストール済みパッケージの更新は、そのパブリッシャーが sticky であるかぎり、そのパッケージを最初に提供した同じパブリッシャーから取得されます。`-P` または `--search-first` を `-p` とともに使用すると、追加されたパブリッシャーのみが検索順序の先頭に配置されます。

`--non-sticky` は、このパブリッシャーよりも上位にランクされるパブリッシャーが、このパブリッシャーから最初にインストールされたパッケージに更新を提供できることを指定します。

`--sticky` は、このパブリッシャーからインストールされたパッケージへの更新を引き続きこのパブリッシャーから取得する必要があることを指定します。これはデフォルトの動作です。

--search-before は、変更されるパブリッシャーが指定されたパブリッシャーの前に検索されるように、パブリッシャー検索順序を変えます。-p とともに使用すると、--search-before は追加されるパブリッシャーのみに適用されます。

--search-after は、変更されるパブリッシャーが指定されたパブリッシャーのあとに検索されるように、パブリッシャー検索順序を変えます。-p とともに使用すると、--search-after は追加されるパブリッシャーのみに適用されます。

--approve-ca-cert を指定すると、指定された証明書を、信頼できる CA 証明書として追加します。ユーザーが承認した CA 証明書の PEM 表現のハッシュは、pkg publisher コマンドの詳細出力に一覧表示されます。

--revoke-ca-cert を指定すると、指定された PEM 表現のハッシュを持つ証明書を失効済みとして扱います。ユーザーが失効させた CA 証明書のハッシュは、pkg publisher コマンドの詳細出力に一覧表示されます。

--unset-ca-cert を指定すると、指定されたハッシュを持つ証明書を、承認済み証明書のリストおよび失効済み証明書のリストから削除します。

--set-property は、既存のパブリッシャープロパティーを更新するか、新しいパブリッシャープロパティーを追加します。

--add-property-value は、既存のパブリッシャープロパティーに値を追加するか、新しいパブリッシャープロパティーを追加します。

--remove-property-value は、既存のパブリッシャープロパティーから値を削除します。

--unset-property は、既存のパブリッシャープロパティーを削除します。

-c はクライアント SSL 証明書を、-k は SSL 鍵をそれぞれ指定します。

-g (--add-origin) は、指定された URI またはパスを、指定されたパブリッシャーの起点として追加します。これはパッケージのリポジトリまたはアーカイブの場所にしてください。

-G (--remove-origin) は、指定されたパブリッシャーの起点のリストから URI またはパスを削除します。特殊値 * を使用して、すべての起点を削除することができます。

--no-refresh を指定すると、入手可能パッケージやその他のメタデータの最新リストを取得するために、イメージのパブリッシャー用のリポジトリへのアクセスを試みません。

--reset-uuid は、このイメージをそのパブリッシャーに対して識別する新しい一意識別子を選択します。

-m (--add-mirror) は、指定されたパブリッシャーのミラーとして URI を追加します。

-M (--remove-mirror) は、指定されたパブリッシャーのミラーのリストから URI を削除します。特殊値 * を使用して、すべてのミラーを削除することができます。

-p は、指定されたリポジトリ URI からパブリッシャー構成情報を取得します。パブリッシャーを指定した場合、一致するパブリッシャーのみが追加または更新されます。パブリッシャーを指定しない場合、すべてのパブリッシャーが必要に応じて追加または更新されます。このオプションは -g、 --add-origin、 --G、 --remove-origin、 -m、 --add-mirror、 -M、 --remove-mirror、 --disable、 --enable、 --no-refresh、 または --reset-uuid の各オプションとは組み合わせることができません。

-e (--enable) はパブリッシャーを有効にします。-d (--disable) はパブリッシャーを無効にします。無効にされたパブリッシャーは、パッケージリストの生成時に、または特定のパッケージ操作 (インストール、アンインストール、および更新) で使用されません。ただし、無効なパブリッシャーのプロパティーを設定または表示することはできます。発行元が 1 つだけの場合は、無効にすることはできません。

/usr/bin/pkg set-publisher -p *repo_uri* [-Ped] [-k *ssl_key*] [-c *ssl_cert*] [--non-sticky] [--sticky] [--search-after *publisher*] [--search-before *publisher*] [--search-first] [--approve-ca-cert *path_to_CA*] [--revoke-ca-cert *hash_of_CA_to_remove*] [--unset-ca-cert *hash_of_CA_to_remove*] [--set-property *name_of_property*= *value*] [--add-property-value *name_of_property* =*value_to_add*] [--remove-property-value *name_of_property=value_to_remove*] [--unset-property *name_of_property_to_delete*] [*publisher*]

-p は、指定されたリポジトリ URI からパブリッシャー構成情報を取得します。パブリッシャーを指定した場合、一致するパブリッシャーのみが追加または更新されます。パブリッシャーを指定しない場合、すべてのパブリッシャーが必要に応じて追加または更新されます。-p オプションと一緒に使用できるほかのオプションについては、上記の pkg set-publisher を参照してください。-p オプションは、-g、 --add-origin、 -G、 --remove-origin 、 -m、 --add-mirror、 -M、 --remove-mirror、 --disable、 --enable、 --no-refresh、 または --reset-uuid オプションと組み合わせることはできません。

unset-publisher *publisher* ...

指定されたパブリッシャーに関連付けられた構成を削除します。

history [-Hl] [-t [*time* | *time-time*],...] [-o *column*,...] [-n *number*]

該当するイメージのコマンド履歴を表示します。

-H が指定されている場合は、リストのヘッダーを省略します。

-t を指定すると、%Y-%m-%dT%H:%M:%S 形式のタイムスタンプのコンマ区切りリストでログレコードを表示します (strftime(3C) を参照)。日時の範囲を指定するには、開始と終了のタイムスタンプの間にハイフン (-) を使用します。キーワード now は、現在の日時の別名として使用できます。指定されたタイ

ムスタンプに、重複したタイムスタンプまたは重複する日付範囲が含まれる場合、重複した各履歴イベントの 1 つのインスタンスのみが出力されます。

-l を指定すると、長い形式でログレコードを表示します。これには標準形式に加えて、コマンドの結果、コマンドが完了した日時、使用されたクライアントのバージョンと名前、操作を実行したユーザーの名前、およびコマンドの実行中に発生したすべてのエラーが含まれます。

-n は、最新のものから順に指定された数のエントリのみを表示します。

-o は、列名を指定するコンマ区切りリストを使用して出力を表示します。有効な列名は次のとおりです。

be	この操作が開始されたブート環境の名前。
be_uuid	この操作が開始されたブート環境の UUID。
client	クライアントの名前。
client_ver	クライアントのバージョン。
command	この操作のために使用されたコマンド行。
finish	この操作が完了した日時。
id	この操作を開始したユーザー ID。
new_be	この操作によって作成された新しいブート環境。
new_be_uuid	この操作によって作成された新しいブート環境の UUID。
operation	操作の名前。
outcome	この操作の結果のサマリー。
reason	この操作の結果に関する追加情報。
snapshot	この操作中に作成されたスナップショット。これは、操作が正常に完了したあとにスナップショットが自動削除されなかった場合にのみ記録されます。
start	この操作が開始した日時。
time	この操作の実行にかかった合計時間。1 秒未満の操作については 0:00:00 と表示されます。
user	この操作を開始したユーザー名。

command または reason 列を指定する場合、出力フィールドの区切りを維持するためには、それらの列が -o リストの最終項目である必要があります。同じ history コマンドでこれら 2 つの列を表示することはできません。

ブート環境がシステムに存在しなくなった場合、be または new_be の値のあとにアスタリスク (*) が表示されます。

be および new_be の値は、be_uuid または new_be_uuid フィールドを使用して現在のブート環境名を検索することによって取得されます。その後、ブート環境の名前が変更されたあとにその環境が削除された場合、be および new_be に表示される値は、pkg 操作の時点で記録された値です。

purge-history
既存の履歴情報をすべて削除します。

rebuild-index
pkg search によって使用されるインデックスを再構築します。これは復旧操作であり、一般的に使用することは想定されていません。

update-format
イメージの形式を現在のバージョンに更新します。この操作が完了したあとは、古いバージョンの pkg(5) システムと組み合わせてイメージを使用することはできなくなります。

version
pkg(1) のバージョンを識別する一意な文字列を表示します。この文字列は、バージョン間で何らかの方法で比較可能であることは保証されていません。

image-create [-FPUfz] [--force] [--full | --partial | --user] [--zone] [-k *ssl_key*] [-c *ssl_cert*] [--no-refresh] [--variant *variant_name*=*value* ...] [-g *path_or_uri* | --origin *path_or_uri* ...] [-m *uri* | - -mirror *uri* ...] [--set-property *name_of_property* =*value*][--facet *facet_name*=(True|False) ...] [(-p | --publisher) [*name*=] *repo_uri*] *dir*
dir によって指定された場所に、パッケージ操作に適したイメージを作成します。デフォルトのイメージタイプはユーザーであり、-U (--user) オプションによって指定されます。イメージタイプはフルイメージ (--F または --full)、または指定された *dir* パスを包含するフルイメージにリンクされた部分イメージ (-P または --partial) に設定できます。-g または --origin を使用して追加の起点を指定できます。--m または --mirror を使用して追加のミラーを指定できます。

パッケージリポジトリの URI は、-p または --publisher オプションを使用して提供する必要があります。パブリッシャーの名前も提供した場合、イメージの作成時にそのパブリッシャーのみが追加されます。パブリッシャーの名前を提供しない場合、指定されたリポジトリによって認識されているすべてのパブリッシャーがイメージに追加されます。このパブリッシャーに関連付けられたカタログは、初期作成操作に続いて取得が試みられます。

クライアント SSL 認証を使用するパブリッシャーの場合、クライアント鍵およびクライアント証明書は -c および -k オプションを通して登録できます。この鍵と証明書は、イメージ作成中に追加されるすべてのパブリッシャーのために使用されます。

イメージが非大域ゾーンコンテキストの内部で実行される予定の場合、-z (--zone) オプションを使用して適切なバリアントを設定できます。

-f (--force) を指定すると、強制的に既存のイメージを上書きしてイメージを作成します。このオプションは慎重に使用してください。

--no-refresh を指定すると、入手可能パッケージやその他のメタデータの最新リストを取得するために、イメージのパブリッシャー用のリポジトリへのアクセスを試みません。

--variant は、指定されたバリアントを指定された値に設定します。バリアントの詳細については、pkg(5) のマニュアルページのファセットとバリアントに関する項目を参照してください。

--facet は、指定されたファセットを指定された値に設定します。ファセットの詳細については、pkg(5) のマニュアルページのファセットとバリアントに関する項目を参照してください。

--set-property が指定されている場合、指定されたイメージプロパティーに指示された値を設定します。イメージプロパティーの説明については、下の「イメージプロパティー」を参照してください。

イメージプロパティー

次のプロパティーはイメージの特性を定義します。これらのプロパティーは、イメージの目的、内容、および動作に関する情報を格納します。イメージ内のこれらのプロパティーの現在の値を表示するには、pkg property コマンドを使用します。これらのプロパティーの値を変更するには、pkg set-property コマンドおよび pkg unset-property コマンドを使用します。

be-policy
(文字列) パッケージ操作中にいつブート環境が作成されるかを指定します。次の値が許可されます。

default　デフォルトの BE 作成ポリシー create-backup を適用します。

always-new　次のブート時にアクティブに設定されている新しい BE でパッケージ操作を実行するため、すべてのパッケージ操作に対してリブートを必要とします。明示的に要求されないかぎり、バックアップ BE は作成されません。

このポリシーはもっとも安全ですが、リブートしないとパッケージを追加できないため、ほとんどのサイトの要求よりも厳格です。

create-backup
リブートを必要とするパッケージ操作で、新しい BE が作成され、次のブート時にアクティブに設定されます。パッケージが変更されるか、カーネルに影響する可能性のあるコンテンツがインストールされて操作がライブ BE に影響する場合、バックアップ BE は作成されますがアクティブには設定されません。バックアップ BE を明示的に要求することもできます。

このポリシーは、新しくインストールされたソフトウェアによりシステムが不安定になっている場合にのみ潜在的に危険です。この可能性はありますが、比較的まれです。

`when-required`
リブートを必要とするパッケージ操作で、新しい BE が作成され、次のブート時にアクティブに設定されます。明示的に要求されないかぎり、バックアップ BE は作成されません。

パッケージの変更によりライブ BE へのこれ以上の変更が不可能になると、フォールバックできる新しい BE がなくなる可能性があるため、このポリシーはもっとも大きな危険を伴います。

`ca-path`
(文字列) SSL 操作用の CA 証明書が格納されたディレクトリを指すパス名。このディレクトリの形式は、ベースとなる SSL 実装に固有です。信頼できる CA 証明書のために別の場所を使用するには、別のディレクトリを指すようにこの値を変更します。CA ディレクトリの要件については、`SSL_CTX_load_verify_locations`(3openssl) の `CApath` に関する項目を参照してください。

デフォルト値: `/etc/openssl/certs`

`check-certificate-revocation`
(ブール型) `True` に設定されている場合、パッケージクライアントは、署名検証のために使用される証明書の CRL 配布ポイントへのアクセスを試み、発行時よりもあとに証明書が失効していないかどうかを調べます。

デフォルト値: `False`

`flush-content-cache-on-success`
(ブール型) `True` に設定されている場合、パッケージクライアントは、インストールまたは更新操作の完了時にその内容キャッシュ内のファイルを削除します。更新操作の場合、内容はソース BE からのみ削除されます。出力先 BE でパッケージ操作が次に発生したとき、このオプションが変更されていなければ、パッケージクライアントはその内容キャッシュをフラッシュします。

このプロパティーを使用して、ディスク容量の限られたシステムで内容キャッシュを小さく保つことができます。このプロパティーを使用すると、操作が完了するまでの時間が長くなる可能性があります。

デフォルト値: `True`

`mirror-discovery`
(ブール型) このプロパティーは、mDNS および DNS-SD を使用してリンクローカル内容ミラーを検出するようにクライアントに命令します。このプロパティーを `True` に設定すると、クライアントはミラーを動的に検出し、そのミラーからパッケージ内容のダウンロードを試みます。mDNS を介してその内容を通知するミラーの実行方法については、`pkg.depotd`(1M) を参照してください。

デフォルト値: `False`

`send-uuid`
(ブール型) ネットワーク操作の実行時にイメージの汎用一意識別子 (UUID) を送信します。ユーザーはこのオプションを無効にできますが、一部のネットワークリポジトリは UUID を供給しないクライアントとのやり取りを拒否する場合があります。

デフォルト値: `True`

`signature-policy`
(文字列) イメージ内のパッケージのインストール、更新、修正、または検証時にマニフェストに対してどのチェックが実行されるかを決定します。パッケージに適用される最終的なポリシーは、イメージポリシーと発行元ポリシーの組み合わせに依存します。この組み合わせの厳格さは、少なくとも、この 2 つのポリシーが個別に適用された場合の厳格な方と同じです。デフォルトでは、パッケージクライアントは証明書が失効済みかどうかをチェックしません。そのようなチェック (クライアントから外部 Web サイトへのアクセスが必要な場合がある) を有効にするには、`check-certificate-revocation` イメージプロパティーを `True` に設定します。次の値が許可されます。

`ignore`	すべてのマニフェストの署名を無視します。
`verify`	署名が含まれているすべてのマニフェストが有効に署名されていることを確認しますが、インストール済みパッケージがすべて署名されている必要はありません。これがデフォルト値です。
`require-signatures`	新しくインストールされたすべてのパッケージに、有効な署名が少なくとも 1 つ含まれている必要があります。インストール済みパッケージに有効な署名が含まれていない場合は、`pkg fix` および `pkg verify` コマンドでも警告が表示されます。
`require-names`	`require-signatures` と同じ要件に従いますが、`signature-required-names` プロパティーで一覧表示される文字列が、署名の信頼のチェーンを検証するために使用される証明書の共通名としても表示される必要があります。

`signature-required-names`
(文字列のリスト) 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。

`trust-anchor-directory`
(文字列) イメージの信頼アンカーを格納するディレクトリのパス名です。このパスはイメージに対して相対的です。デフォルト値は `ignore` です。

use-system-repo
(ブール型) このプロパティーではシステムリポジトリを、イメージおよびパブリッシャー構成のソースとして、および提供されたパブリッシャーと通信するためのプロキシとしてイメージで使用するべきかどうかを指定します。デフォルト値は `False` です。システムリポジトリについては、`pkg.sysrepo`(1M) を参照してください。

パブリッシャープロパティー

次のプロパティーは、特定の発行元の署名ポリシーを定義します。同じ名前のイメージプロパティーはイメージの署名ポリシーを定義します。特定の発行元のこれらのプロパティーの現在の値を表示するには、`pkg publisher` *publisher_name* コマンドを使用します。これらの発行元の署名ポリシープロパティーの値を変更するには、`pkg set-publisher` コマンドの `--set-property` オプションと `--unset-property` オプションを使用します。

signature-policy
(文字列) このプロパティーの機能は、特定のパブリッシャーからのパッケージのみに適用されることを除いて、同じ名前のイメージプロパティーと同じです。

signature-required-names
(文字列のリスト) このプロパティーの機能は、特定のパブリッシャーからのパッケージのみに適用されることを除いて、同じ名前のイメージプロパティーと同じです。

使用例

例1 パブリッシャーを構成してイメージを作成する

パブリッシャーを `example.com` として新しいフルイメージを作成し、`/aux0/example_root` に格納します。

```
$ pkg image-create -F -p example.com=http://pkg.example.com:10000 \
/aux0/example_root
```

例2 追加の起点とミラーを指定してイメージを作成する

パブリッシャーを `example.com` として新しいフルイメージを作成し、1 つのミラーと 2 つの起点を追加し、`/aux0/example_root` に格納します。

```
$ pkg image-create -F -p example.com=http://pkg.example.com:10000 \
-g http://alternate1.example.com:10000/ \
-g http://alternate2.example.com:10000/ \
-m http://mirror.example.com:10000/ \
/aux0/example_root
```

例3 パブリッシャーを構成せずにイメージを作成する

パブリッシャーを構成せずに、新しいフルイメージを `/aux0/example_root` に作成します。

```
$ pkg image-create -F /aux0/example_root
```

例4 パッケージのインストール

widget パッケージの最新バージョンを現在のイメージにインストールします。

```
$ pkg install application/widget
```

例5 パッケージの指定された内容を一覧表示する

system/file-system/zfs パッケージの内容を一覧表示します。アクション名、ファイルのモード (定義されている場合)、サイズ (定義されている場合)、パス、およびターゲット (リンクの場合) を表示します。すべてのアクションで利用可能な action.name 属性を指定するとすべてのアクションの行が表示されますが、ここでは望ましくないため、dir、file、link、および hardlink の各タイプにアクションを限定します。

```
$ pkg contents -t dir,file,link,hardlink \
-o action.name,mode,pkg.size,path,target system/file-system/zfs
ACTION.NAME MODE PKG.SIZE PATH                TARGET
dir         0755          etc
dir         0755          etc/fs
dir         0755          etc/fs/zfs
link                      etc/fs/zfs/mount    ../../../usr/sbin/zfs
link                      etc/fs/zfs/umount   ../../../usr/sbin/zfs
dir         0755          etc/zfs
dir         0755          kernel
dir         0755          kernel/drv
dir         0755          kernel/drv/amd64
file        0755  1706744 kernel/drv/amd64/zfs
file        0644      980 kernel/drv/zfs.conf
dir         0755          kernel/fs
dir         0755          kernel/fs/amd64
hardlink                  kernel/fs/amd64/zfs  ../../../kernel/drv/amd64/zfs
...
```

例6 2つのパッケージの指定された内容を一覧表示する

web/browser/firefox および mail/thunderbird の内容を一覧表示します。path 属性の末尾が .desktop または .png であるアクションのパッケージ名属性およびパス属性のみに表示を限定します。

```
$ pkg contents -o pkg.name,path -a path=\*.desktop \
-a path=\*.png web/browser/firefox mail/thunderbird
PKG.NAME            PATH
web/browser/firefox usr/share/applications/firefox.desktop
mail/thunderbird    usr/share/applications/thunderbird.desktop
web/browser/firefox usr/share/pixmaps/firefox-icon.png
mail/thunderbird    usr/share/pixmaps/thunderbird-icon.png
...
```

例 7 パッケージを検索する

パッケージデータベースからトークン bge を検索します。

```
$ pkg search bge
INDEX       ACTION VALUE                      PACKAGE
driver_name driver bge                        pkg:/driver/network/bge@0.5.11-0.169
basename    file   kernel/drv/sparcv9/bge     pkg:/driver/network/bge@0.5.11-0.169
basename    file   kernel/drv/amd64/bge       pkg:/driver/network/bge@0.5.11-0.169
pkg.fmri    set    solaris/driver/network/bge pkg:/driver/network/bge@0.5.11-0.169
```

このトークンはパッケージ driver/network/bge 内に、/kernel/drv/*arch*/bge を表すファイルアクションのベース名として、およびドライバ名として存在します。

例 8 指定されたパッケージに依存するパッケージを検索する

package/pkg に依存するインストール済みパッケージを検索します。

```
$ pkg search -l 'depend::package/pkg'
INDEX       ACTION VALUE                    PACKAGE
incorporate depend package/pkg@0.5.11-0.169 pkg:/consolidation/ips/ips-incorporation
                                                @0.5.11-0.169
require     depend package/pkg@0.5.11-0.169 pkg:/system/install@0.5.11-0.169
require     depend package/pkg@0.5.11-0.169 pkg:/package/pkg/system-repository@0.5.11-0.169
```

例 9 依存関係を検索する

インストール済みパッケージ内のすべての incorporate 依存関係を検索します。

```
$ pkg search -l 'depend:incorporate:'
INDEX       ACTION VALUE                            PACKAGE
incorporate depend pkg:/BRCMbnx@0.5.11,5.11-0.133  pkg:/consolidation/osnet/osnet-
                                                       incorporation@0.5.11-0.169
incorporate depend pkg:/BRCMbnxe@0.5.11,5.11-0.133 pkg:/consolidation/osnet/osnet-
                                                       incorporation@0.5.11-0.169
...
```

例 10 発行元を追加

新しいパブリッシャー example.com を追加し、リポジトリの場所を http://www.example.com/repo に設定します。

```
$ pkg set-publisher -g http://www.example.com/repo example.com
```

例 11 鍵と証明書を指定してパブリッシャーを追加する

新しいパブリッシャー example.com を追加し、セキュリティー保護されたリポジトリの場所を https://secure.example.com/repo に設定し、鍵および証明書をディレクトリ /root/creds に格納します。

```
$ pkg set-publisher -k /root/creds/example.key \
-c /root/creds/example.cert -g https://secure.example.com/repo \
example.com
```

例 12 パブリッシャーを追加して自動構成する

自動構成を使用して新しいパブリッシャーを追加し、リポジトリの場所を /export/repo に構成します。

```
$ pkg set-publisher -p /export/repo
```

例 13 パブリッシャーを追加して手動構成する

手動構成を使用して新しいパブリッシャー example.com を追加し、リポジトリの場所を /export/repo/example.com に構成します。

```
$ pkg set-publisher -g /export/repo example.com
```

例 14 すべての署名付きパッケージを検証する

すべての署名付きパッケージを検証するようにイメージを構成します。

```
$ pkg set-property signature-policy verify
```

例 15 すべてのパッケージで署名を必須にする

すべてのパッケージで署名を必須とするように、また、いずれかの証明書で文字列 example.com が共通名として信頼チェーンに出現することを必須とするようにイメージを構成します。

```
$ pkg set-property signature-policy require-names example.com
```

例 16 指定したパブリッシャーからのすべてのパッケージで署名を必須にする

パブリッシャー example.com からインストールされるすべてのパッケージで署名を必須とするようにイメージを構成します。

```
$ pkg set-publisher --set-property signature-policy=require-signatures \
example.com
```

例 17 信頼チェーンで指定された文字列を必須にする

有効とみなされるために署名の信頼チェーンに存在する必要がある文字列 foo をイメージの共通名の一覧に追加します。

```
$ pkg add-property-value signature-require-names foo
```

例 18 指定されたパブリッシャーの信頼チェーンから文字列を削除する

パブリッシャー example.com の署名を検証するために存在する必要がある文字列 foo を共通名の一覧から削除します。

```
$ pkg set-publisher --remove-property-value signature-require-names=foo \
example.com
```

例 19　信頼できる CA 証明書を追加する

/tmp/example_file.pem に格納された証明書を、パブリッシャー example.com の信頼できる CA 証明書として追加します。

```
$ pkg set-publisher --approve-ca-cert /tmp/example_file.pem \
example.com
```

例 20　証明書を失効させる

a12345 というハッシュを持つ、パブリッシャー example.com 用の証明書を失効させ、example.com からのパッケージの署名をその証明書で検証しないようにします。

```
$ pkg set-publisher --revoke-ca-cert a12345 example.com
```

例 21　証明書を抹消する

証明書 a12345 がユーザーによって追加または失効させられたことを pkg から抹消します。

```
$ pkg set-publisher --unset-ca-cert a12345 example.com
```

例 22　パッケージをダウングレードする

インストール済みパッケージ foo@1.1 を古いバージョンにダウングレードします。

```
$ pkg update foo@1.0
```

例 23　競合するパッケージインストールを切り替える

2 つのパッケージが競合している場合に、どちらのパッケージがインストールされるかを切り替えます。パッケージ A がパッケージ B または C のどちらかに依存し、B および C は相互排他であるとします。A と B がインストールされる場合に、次のコマンドを使用して、A をアンインストールせずに B の代わりに C を使用するように切り替えます。

```
$ pkg install --reject B C
```

例 24　パッケージアーカイブ内のパッケージを一覧表示する

パッケージアーカイブ内のすべてのパッケージのすべてのバージョンを一覧表示します。

```
$ pkg list -f -g /my/archive.p5p
```

例 25　パッケージリポジトリ内のパッケージを一覧表示する

リポジトリ内のすべてのパッケージのすべてのバージョンを一覧表示します。

```
$ pkg list -f -g http://example.com:10000
```

例 26 パッケージアーカイブ内のパッケージについての情報を表示する

パッケージアーカイブ内のパッケージの最新バージョンのパッケージ情報を表示します。パッケージが現在インストールされていてもいなくてもかまいません。

```
$ pkg info -g /my/archive.p5p pkg_name
```

例 27 パッケージアーカイブ内のパッケージの内容を表示する

パッケージアーカイブ内のパッケージの内容を表示します。パッケージは現在インストールされていません。

```
$ pkg contents -g /my/archive.p5p pkg_name
```

例 28 パブリッシャーの起点とミラーをすべて削除する

パブリッシャーのすべての起点およびミラーを削除し、新しい起点を追加します。

```
$ pkg set-publisher -G '*' -M '*' -g http://example.com:10000 \
example.com
```

環境変数

PKG_IMAGE
: パッケージ操作に使用するイメージが含まれるディレクトリ。-R を指定した場合は無視されます。

PKG_CLIENT_CONNECT_TIMEOUT
: トランスポート操作中に接続しようとするのを待機する秒数 (試行ごと)。これが経過するとクライアントは操作を中止します。値 0 は無制限に待機することを意味します。

  デフォルト値: 60

PKG_CLIENT_LOWSPEED_TIMEOUT
: トランスポート操作中に lowspeed 制限 (1024 バイト/秒) を下回っていられる秒数。これが経過するとクライアントは操作を中止します。値 0 は、操作を中止しないことを意味します。

  デフォルト値: 30

PKG_CLIENT_MAX_CONSECUTIVE_ERROR
: 一時的なトランスポートエラーの最大数。これを上回るとクライアントは操作を中止します。値 0 は、操作を中止しないことを意味します。

  デフォルト値: 4

PKG_CLIENT_MAX_REDIRECT
: トランスポート操作中に許可される HTTP または HTTPS リダイレクトの最大数。これを上回ると接続が中止されます。値 0 は、操作を中止しないことを意味します。

  デフォルト値: 5

PKG_CLIENT_MAX_TIMEOUT
ホストあたりのトランスポート試行の最大数。これを上回るとクライアントは操作を中止します。値 0 は、操作を中止しないことを意味します。

デフォルト値: 4

http_proxy、https_proxy
HTTP または HTTPS プロキシサーバー。

終了ステータス

次の終了ステータスが返されます。

0 コマンドが成功しました。

1 エラーが発生した。

2 無効なコマンド行オプションが指定された。

3 複数の操作が要求されましたが、それらの一部のみが成功しました。

4 変更が行われませんでした - 何もしません。

5 要求された操作はライブイメージでは実行できません。

6 インストールまたは更新中のパッケージのライセンスが受け入れられなかったため、要求された操作を完了できません。

7 イメージは現在別のプロセスによって使用されているため、変更できません。

99 予期しない例外が発生しました。

ファイル

pkg(5) イメージは、より大きなファイルシステム内の任意の場所に置くことができます。次のファイル説明で、トークン $IMAGE_ROOT は相対パスを区別するために使用されています。一般的なシステムインストールでは、 $IMAGE_ROOT は / と等価です。

$IMAGE_ROOT/var/pkg	完全または部分的なイメージのメタデータディレクトリ。
$IMAGE_ROOT/.org.opensolaris,pkg	ユーザーイメージのメタデータディレクトリ。

特定のイメージのメタデータ内のファイルやディレクトリによっては、修復および復旧中に使用される情報が含まれている可能性があります。トークン $IMAGE_META は、メタデータが含まれる最上位ディレクトリを参照します。$IMAGE_META は通常、上に示した 2 つのパスのいずれかです。

$IMAGE_META/lost+found	パッケージ操作中に移動された、競合するディレクトリおよびファイルの場所。削除されたディレクトリのパッケージ化されない内容の場所。

`$IMAGE_META/publisher` パブリッシャーごとに 1 つのディレクトリが含まれます。各ディレクトリにはパブリッシャー固有のメタデータが格納されます。

`$IMAGE_META` ディレクトリ階層内のほかのパスは非公開であり、変更される可能性があります。

属性

次の属性については、`attributes(5)` を参照してください。

属性タイプ	属性値
使用条件	`package/pkg`
インタフェースの安定性	不確実

関連項目

`pkgsend(1)`、`pkg.depotd(1m)`、`glob(3C)`、`pkg(5)`、`beadm(1M)`

`http://hub.opensolaris.org/bin/view/Project+pkg/`

名前

pkgdepend – Image Packaging System 依存関係アナライザ

形式

```
/usr/bin/pkgdepend [options] command [cmd_options] [operands]

/usr/bin/pkgdepend generate [-IMm] -d dir [-d dir]
    [-D name=value] [-k path] manifest_file

/usr/bin/pkgdepend resolve [-moSv] [-d output_dir]
    [-s suffix] manifest_file ...
```

機能説明

pkgdepend は、パッケージの依存関係を生成および解決するために使用されます。パッケージは、ほかのパッケージのファイルに依存することがあります。pkgdepend は通常、ファイル依存関係の生成と、ファイルからパッケージへの解決という 2 つのパスで使用されます。

generate サブコマンドはパッケージの内容を検査し、そのパッケージにどのような外部ファイルが必要かを判定します。

resolve サブコマンドは、generate ステップからファイルのリストを取得し、これらの依存ファイルを含むパッケージの名前を判定するためにパッケージの参照セットを検索します。依存ファイルが検索されるパッケージの参照セットは、パブリッシャーのシステムに現在インストールされているパッケージです。

依存関係情報のソースとして、提供されたファイルの次のいくつかのコンポーネントが使用されます。

ELF
: 提供されたファイル内の ELF ヘッダーが依存関係情報のために分析され、-k および -D オプションが指定されていると取得された情報が変更されます。ELF 依存関係についての詳細は、ldd(1) および『リンカーとライブラリ』を参照してください。

スクリプト
: インタプリタを参照している #! 行を含むシェルスクリプトによって、そのインタプリタを提供しているパッケージへの依存関係が生成されます。

Python
: Python スクリプトはまず、スクリプトとして分析されます。さらに、そのスクリプトでインポートが宣言されていると、それらも依存関係情報のソースとして機能することがあります。

ハードリンク
: マニフェスト内のハードリンクによって、リンクターゲットを提供しているパッケージへの依存関係が生成されます。

SMF
: require_all 依存関係を含む提供された SMF サービスマニフェストによって、これらの FMRI を提供する SMF マニフェストを提供しているパッケージへの依存関係が生成されます。

オプション

サポートしているオプションは、次のとおりです。

-R *dir*　　*dir* をルートとするイメージを操作します。ディレクトリが指定されなかったか、または環境に基づいて決定される場合、デフォルトは / です。詳細は、「環境変数」のセクションを参照してください。

--help or -?　　使用方法に関するメッセージを表示します。

サブコマンド　　サポートされているサブコマンドは次のとおりです。

generate [-IMm] -d *dir* [-d *dir*] [-D *name=value*] [-k *path*] *manifest_file*
manifest_file で指定されたマニフェストのファイルへの依存関係を生成します。

-I が指定されている場合は、*manifest_file* 内で満たされている依存関係を表示します。-I と pkgdepend resolve の結果を使用しないでください。

-M が指定されている場合は、分析できなかったファイルタイプのリストを表示します。

-m が指定されている場合は、元のマニフェストを繰り返してから、検出された依存関係をすべて追加します。

-d が指定されている場合は、マニフェストのファイルを検索するディレクトリのリストに *dir* を追加します。

-D が指定されるたびに、ELF ファイル依存関係のための実行パス内のトークン *name* を展開する方法として *value* を追加します。

-k が指定されるたびに、カーネルモジュールを検索する実行パスのリストに *path* を追加します。-k 引数を使用すると、デフォルトのパス (/kernel と /usr/kernel) が削除されます。

-k オプションで指定されるような実行パスは、アクションまたはマニフェスト属性 pkg.depend.runpath を使用して、アクションごとまたはマニフェストごとにも指定できます。pkg.depend.runpath 属性の値は、使用するパスのコロンで区切られた文字列です。

-k の使用は、マニフェストまたはアクションで設定された任意の pkg.depend.runpath 属性によって上書きされます。

pkg.depend.runpath 属性値の 1 つのコンポーネントとして、特殊なトークン $PKGDEPEND_RUNPATH を使用すると、分析対象のファイルのための標準のシステム実行パスを含めることができます。

場合によっては、依存関係が自動的に生成されることを回避したいことがあります。たとえば、あるパッケージによって、一連のモジュールをインポートするサンプルの Python スクリプトが提供される場合、そのサンプルスクリプトによってインポートされるこれらのモジュールは、サンプルスクリプトを提供しているパッケージの依存関係ではありません。指定されたファイルに対する依存関

係の生成を回避するには、アクションまたはマニフェスト属性 `pkg.depend.bypass-generate` を使用します。

`pkg.depend.bypass-generate` 値は、ファイル名に一致する Python の正規表現です。これらの正規表現は、ファイルパスの先頭と最後に暗黙的に固定されます。次の例で指定されている値は `this/that` に一致しますが、`something/this/that/the/other` には一致しません。

```
pkg.depend.bypass-generate=this/that
```

Python 正規表現の構文については、コマンド `pydoc re` を使用するか、または `http://docs.python.org/dev/howto/regex.html` にあるより完全なドキュメントを参照してください。

`pkgdepend generate` 入力マニフェストに SMF マニフェストファイルが含まれる場合、それらの SMF マニフェストファイルによって宣言されている SMF サービスまたはインスタンスが `pkgdepend` の出力に含まれます。これらの SMF サービスまたはインスタンスは、`org.opensolaris.smf.fmri` の名前で `set` アクションの形式で含まれます。

`resolve [-moSv] [-d` *output_dir*`] [-s` *suffix*`]` *manifest_file* `...`

ファイルへの依存関係を、これらのファイルを提供するパッケージへの依存関係に変換します。依存関係は、まずコマンド行に指定されたマニフェストに基づいて解決され、次にシステムにインストールされているパッケージに基づいて解決されます。デフォルトでは、各マニフェストの依存関係は *manifest_file*`.res` という名前のファイルに格納されます。

`-m` が指定されている場合は、解決された依存関係を追加する前に、`generate` ステップによって生成された依存関係をすべて削除して、マニフェストを繰り返します。

`-o` が指定されている場合は、結果を標準出力に書き込みます。このオプションは、人による使用を目的にしています。この出力をファイルに追加すると、無効なマニフェストが生成されることがあります。マニフェスト処理のパイプラインでは、`-o` の代わりに `-d` または `-s` オプションを使用することを強くお勧めします。

`-d` が指定されている場合は、指定された各マニフェストの解決された依存関係を *output_dir* 内の個別のファイルに書き込みます。デフォルトでは、各ファイルには、そのファイルに書き込まれた依存関係のソースだったマニフェストと同じベース名が付けられます。

`-s` が指定されている場合は、出力ファイルごとに、解決された依存関係のソースだったファイルのベース名に *suffix* を追加します。「.」が *suffix* に付加されます (指定されていない場合)。

`-S` が指定されている場合は、コマンド行で指定されたマニフェストに対してのみ解決し、システムにインストールされているマニフェストに対しては解決しません。

-v が指定されている場合は、追加のパッケージ依存関係デバッグ用のメタデータを含めます。

使用例

例 1 依存関係を生成する

foo (このコンテンツディレクトリは ./bar/baz 内に存在する) に書き込まれているマニフェストの依存関係を生成し、その結果を foo.fdeps に格納します。

```
$ pkgdepend generate -d ./bar/baz foo > foo.fdeps
```

例 2 依存関係を解決する

foo.fdeps と bar.fdeps における互いに対するファイル依存関係と、システムに現在インストールされているパッケージに対するファイル依存関係を解決します。

```
$ pkgdepend resolve foo.fdeps bar.fdeps
$ ls *.res
foo.fdeps.res    bar.fdeps.res
```

例 3 2 つのマニフェストの依存関係を生成および解決する

2 つのマニフェスト (foo と bar) へのファイル依存関係を生成し、すべての情報を元のマニフェスト内に保持します。次に、これらのファイル依存関係を解決し、結果のマニフェストを ./res 内に格納します。これらの結果のマニフェストは、pkgsend publish で使用できます。

```
$ pkgdepend generate -d /proto/foo -m foo > ./deps/foo
$ pkgdepend generate -d /proto/bar -m bar > ./deps/bar
$ pkgdepend resolve -m -d ./res ./deps/foo ./deps/bar
$ ls ./res
foo    bar
```

例 4 ELF ファイル依存関係のためのトークンに値を追加する

コンテンツディレクトリが / 内に存在する foo に書き込まれているマニフェストの依存関係を生成するときに、ELF ファイルの実行パス内のすべての PLATFORM トークンを sun4v と sun4u に置き換えます。

```
$ pkgdepend generate -d / -D 'PLATFORM=sun4v' -D 'PLATFORM=sun4u' foo
```

例 5 カーネルモジュールディレクトリを指定する

コンテンツディレクトリが / 内に存在する foo に書き込まれているマニフェストの依存関係を生成するときに、カーネルモジュールを検索するディレクトリとして /kmod を指定します。

```
$ pkgdepend generate -d / -k /kmod foo
```

例6 依存関係の生成をバイパスする

指定された Python スクリプトの標準の Python 実行パスに opt/python を追加し、opt/python/foo/file.py として提供されるファイルの test という名前のすべての Python モジュールに対する依存関係の生成をバイパスします。

usr/lib/python2.6/vendor-packages/xdg で提供されたすべてのファイルに対する依存関係の生成を回避します。

```
$ cat manifest.py
set name=pkg.fmri value=pkg:/mypackage@1.0,1.0
set name=pkg.summary value="My test package"
dir path=opt mode=0755 group=sys owner=root
dir path=opt/python mode=0755 group=sys owner=root
dir path=opt/python/foo mode=0755 group=sys owner=root
file NOHASH path=opt/python/__init__.py mode=0644 group=sys owner=root
file NOHASH path=opt/python/foo/__init__.py mode=0644 group=sys owner=root
#
# Add runpath and bypass-generate attributes:
#
file NOHASH path=opt/python/foo/file.py mode=0644 group=sys owner=root \
    pkg.depend.bypass-generate=.*/test.py.* \
    pkg.depend.bypass-generate=.*/testmodule.so \
    pkg.depend.bypass-generate=.*/test.so \
    pkg.depend.bypass-generate=usr/lib/python2.6/vendor-packages/xdg/.* \
    pkg.depend.runpath=$PKGDEPEND_RUNPATH:/opt/python

$ pkgdepend generate -d proto manifest.py
```

環境変数

PKG_IMAGE　パッケージ操作に使用するイメージを含むディレクトリを指定します。-R が指定されている場合は、この値は無視されます。

終了ステータス

次の終了ステータスが返されます。

0　すべてが動作しました。

1　エラーが発生した。

2　無効なコマンド行オプションが指定された。

99　予期しない例外が発生しました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目 pkg(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

名前　pkgdiff – パッケージマニフェストの比較

形式

```
/usr/bin/pkgdiff [-i attribute ...] [-o attribute]
    [-v name=value ...] file1 file2
```

機能説明　`pkgdiff` は 2 つのパッケージマニフェストを比較し、それらの違いを報告します。`pkgdiff` は、比較の前に、各マニフェストとアクションを一貫した順序にソートします。

出力の形式は次のとおりです。

+ complete_action　このアクションは *file2* に含まれていますが、*file1* には含まれていません。

- complete_action　このアクションは *file1* に含まれていますが、*file2* には含まれていません。

actionname keyvalue [*variant values, if any*]

- attribute1=value1　この *attribute,value* は *file1* に含まれていますが、*file2* には含まれていません。

+ attribute2=value2　この *attribute,value* は *file2* に含まれていますが、*file1* には含まれていません。

バリアントは異なるが、タイプとキー属性値が同じアクションは、比較の目的では別のアクションとして扱われます。そのため、属性を変更したアクションは属性変更としてではなく、完全な形式で表示されます。

オプション　サポートしているオプションは、次のとおりです。

`-i` *attribute*　比較中に *attribute* が存在した場合は、無視します。`-i` *hash* により、ファイルのハッシュ値を無視できます。このオプションを `-o` オプションとともに使用することはできません。このオプションは繰り返すことができます。

`-o` *attribute*　*attribute* の違いのみを報告します。このオプションを `-i` オプションとともに使用することはできません。このオプションは、アクション時に *attribute* に影響を与えないアクション変更をすべて省略します。

`-v` *name=value*　このバリアント値の違いのみを計算します。たとえば、`arch=sparc` の違いのみを計算します。このバリアントタグは、比較前にすべてのアクションについて削除されます。バリアントあたり 1 つの値のみを指定できます。このオプションは、異なるバリアントに対して繰り返すことができます。

終了ステータス　次の終了ステータスが返されます。

`0`　違いは見つかりませんでした。

1　違いが見つかりました。

>1　エラーが発生した。

99　予期しない例外が発生しました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目

pkg(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

名前　pkgfmt – パッケージマニフェストの整形

形式　`/usr/bin/pkgfmt [-c|-d|-u] [`*package-manifest-file*`]`

機能説明　`-c` または `-d` オプションが指定されない `pkgfmt` は、80 文字での行の折り返し、タイプによるアクションのソート、属性のソートなどを含む、一貫した方法でパッケージマニフェストを整形します。アクションに解析されない行 (マクロ、コメント、変換など) は、ソートされた順序では表示されません。

引数が指定されない場合、`pkgfmt` は `stdin` を EOF まで読み取ったあと、整形されたマニフェストを `stdout` に書き込みます。コマンド行で指定されたマニフェストはすべて、その場所で整形されます。

`-c` オプションが指定された `pkgfmt` は、マニフェストが `pkgfmt` 形式で整形されているかどうかをチェックします。`-d` オプションを指定すると、ファイルが正しく整形されていない場合にその違いが表示されます。

オプション　サポートしているオプションは、次のとおりです。

`-c`　マニフェストが `pkgfmt` 形式で整形されているかどうかをチェックします。

`-d`　統合された形式で整形されたバージョンとのマニフェストの違いを表示します。

`-u`　80 文字で行を折り返しません。このオプションは、パッケージマニフェストに従来のテキスト処理ツールを適用する場合に有効です。

終了ステータス　次の終了ステータスが返されます。

`0`　コマンドが成功しました。

`1`　`-c` または `-d` オプションが指定され、1 つ以上のマニフェストが `pkgfmt` の正常な形式でないか、またはエラーが発生しました。

`2`　無効なコマンド行オプションが指定された。

`99`　予期しない例外が発生しました。

属性　次の属性については、`attributes`(5) を参照してください。

属性タイプ	属性値
使用条件	`package/pkg`
インタフェースの安定性	不確実

関連項目　pkg(5)

`http://hub.opensolaris.org/bin/view/Project+pkg/`

名前

pkglint – Image Packaging System パッケージ lint

形式

```
/usr/bin/pkglint [-c dir] [-r uri] [-p regexp]
    [-f rcfile] [-b build_no] [-v]
    [-l uri] | manifest ...

/usr/bin/pkglint -L [-v]
```

機能説明

pkglint は、必要に応じて別のリポジトリを参照しながら、1 つ以上のパッケージマニフェストに対して一連のチェックを実行します。

pkglint は、パッケージ公開前の、パッケージ作成処理中に使用するべきです。pkglint は、マニフェストに対して、pkgsend(1) または pkg.depotd(1M) の通常動作中に実行するには負荷が大きすぎる可能性のある徹底的なテストを実行します。pkglint のチェックには、重複したアクション、欠落した属性、および異常なファイルアクセス権に対するテストが含まれます。

lint のためのマニフェストは、コマンド行で、スペースで区切られたローカルファイルのリストとして渡すことができます。または、リポジトリからマニフェストを取得できます。

リポジトリからマニフェストを取得する場合、pkglint は最初の実行で、指定されたキャッシュディレクトリ内に pkg(5) ユーザーイメージを作成して設定します。-r オプションが指定されている場合は、参照リポジトリのために *cache_dir*/ref_image という名前のユーザーイメージが作成されます。-l オプションが指定されている場合は、lint リポジトリのために *cache_dir*/lint_image という名前のユーザーイメージが作成されます。これらのイメージに内容はインストールされません。これらのイメージは、リポジトリからマニフェストを取得するために pkglint でのみ使用されます。

pkglint の以降の呼び出しではキャッシュディレクトリを再利用できるため、-r または -l 引数はすべて省略できます。

pkglint では、キャッシュディレクトリ内のパブリッシャーを構成するためのサポートが制限されています。これらのイメージに対して、より複雑なパブリッシャーの構成を実行するには、pkg(1) を使用します。

pkglint では、パッケージ作成者は、特定のマニフェストまたはアクションに対するチェックをバイパスできます。True に設定された属性 pkg.linted を含むマニフェストまたはアクションでは、そのマニフェストまたはアクションに対する lint 出力は生成されません。

pkglint チェックの名前の部分文字列を使用して、よりきめ細かな pkg.linted 設定を行うことができます。たとえば、pkg.linted.*check.id* が True に設定されていると、特定のマニフェストまたはアクションに対する *check.id* という名前を持つすべてのチェックがバイパスされます。

pkglint の動作は、pkglintrc ファイルを指定することによって構成できます。デフォルトでは、pkglint は、/usr/share/lib/pkg/pkglintrc および $HOME/.pkglintrc 内の構成オプションを検索します。別の構成ファイルを指定するには、-f オプションを使用します。

lint の実行中に検出されたエラーまたは警告はすべて stderr に出力されます。

オプション

サポートしているオプションは、次のとおりです。

-b *build_no*
: lint および参照リポジトリからの lint 中に使用されるパッケージのリストを絞り込むために使用されるビルド番号を指定します。-b オプションが指定されていない場合は、最新バージョンのパッケージが使用されます。version.pattern 構成プロパティーも参照してください。

-c *cache_dir*
: lint および参照リポジトリからのパッケージのメタデータをキャッシュするために使用されるローカルディレクトリを指定します。

-l *lint_uri*
: lint リポジトリの場所を表す URI を指定します。HTTP ベースとファイルシステムベースの両方の公開がサポートされています。-l を指定する場合は、-c も指定する必要があります。

-L
: 既知の lint チェックと除外された lint チェックを一覧表示してから終了します。各チェックの短縮名と説明を表示します。-v フラグと組み合わせる場合は、説明の代わりに、このチェックを実装するメソッドを表示します。

-f *config_file*
: *config_file* 構成ファイルを使用して pkglint セッションを構成します。

-p *regexp*
: lint リポジトリからチェックされるパッケージのリストを絞り込むために使用される正規表現を指定します。参照リポジトリからはすべてのマニフェストがロードされ (-b が指定されている場合は、その値に一致すると仮定)、このパターンは無視されます。

-r *repo_uri*
: 参照リポジトリの場所を表す URI を指定します。-r を指定する場合は、-c も指定する必要があります。

-v
: pkglint を冗長モードで実行し、構成ファイル内の log_level 設定をすべて上書きします。

--help or -?
: 使用方法に関するメッセージを表示します。

ファイル

pkglintrc 構成ファイルは、次のキーと値の引数を取ります。

log_level
: lint メッセージを発行する最小のレベル。このレベルより低い lint メッセージは破棄されます。デフォルト値は INFO です。

ログレベルを最小のレベルからもっとも重大なレベルの順に示すと、DEBUG、INFO、WARNING、ERROR、CRITICAL になります。

`do_pub_checks` — True の場合は、公開されたパッケージにとってのみ意味がある可能性のあるチェックを実行します。デフォルト値は True です。

`pkglint.ext.*` — pkglint のプラグインメカニズムを使用すると、実行時に lint モジュールを追加できます。pkglint.ext. で始まるキーはすべて、完全に指定された Python モジュールである必要のある値を取ります。詳細は、「開発者」のセクションを参照してください。

`pkglint.exclude` — 実行されるチェックのセットから省略する、完全に指定された Python モジュール、クラス、または関数名のスペースで区切られたリスト。

`use_progress_tracker` — True の場合は、lint の実行中にマニフェストに対する処理を繰り返すときに進捗トラッカーを使用します。デフォルト値は True です。

`version.pattern` — lint の対象となるビルド番号を指定するときに使用されるバージョンのパターン (-b)。構成ファイルで指定されていない場合、-b オプションはパターン *,5.11-0. を使用します。これは、ブランチ接頭辞が 0 である 5.11 ビルドのすべてのコンポーネントに一致します。

開発者

pkglint によって実行されるチェックのセットを拡張するには、pkg.lint.base.Checker とそのサブクラス ManifestChecker 、ActionChecker、および ContentChecker をサブクラス化します。これらのクラスを含む Python モジュール名を、構成ファイル内の新しい pkglint.ext. キーに追加します。

これらの新しいサブクラスのインスタンスは、起動時に pkglint によって作成されます。lint セッション中に、特殊なキーワード引数 pkglint_id を持つ、各サブクラスの内部のメソッドが呼び出されます。これらのメソッドには、スーパークラス内の対応する check() メソッドと同じ署名が含まれているべきです。また、これらのメソッドには、pkglint -L によって出力される説明として使用される pkglint_desc 属性も割り当てられていなくてはいけません。

パラメータは Checker サブクラスから使用できます。これにより、これらのサブクラスは自身の動作を調整できます。推奨されるパラメータの命名規則は、*pkglint_id*.*name* です。パラメータ値は構成ファイル内に格納するか、または LintEngine.get_param() メソッドを使用して取得されるマニフェストまたはアクションでアクセスすることができます。マニフェストからパラメータにアクセスす

る場合は、pkglint パラメータが既存のどのアクションまたはマニフェスト値とも重複しないようにするために、キー名に接頭辞 pkg.lint が付加されます。

使用例

例 1 特定のリポジトリに対する最初の実行

特定のリポジトリに対する pkglint セッションのはじめての実行。

```
$ pkglint -c /space/cache -r http://localhost:10000 mymanifest.mf
```

例 2 同じリポジトリに対するそれ以降の実行

例 1 で使用されている同じリポジトリに対するそれ以降の実行。

```
$ pkglint -c /space/cache mymanifest-fixed.mf
```

例 3 絞り込まれたマニフェストセットでの lint リポジトリの使用

lint リポジトリでの pkglint セッションの実行と、チェックするマニフェストのサブセットの指定。

```
$ pkglint -c /space/othercache -l http://localhost:10000 \
-p '.*firefox.*'
```

例 4 ビルドの指定

冗長モードでの特定のビルドに対する pkglint セッションの実行。

```
$ pkglint -c /space/cache -r http://localhost:10000 \
-l http://localhost:12000 -b 147 -v
```

例 5 構成ファイルの変更

新しい lint モジュールを含む構成ファイル (一部のチェックを除外)。

```
$ cat ~/.pkglintrc
[pkglint]

log_level = DEBUG
# log_level = INFO

pkglint.ext.mycheck = org.timf.mychecks
pkglint.ext.opensolaris = pkg.lint.opensolaris
pkglint.exclude: pkg.lint.opensolaris.OpenSolarisActionChecker
pkg.lint.pkglint.PkgActionChecker.unusual_perms pkg.lint.pkglint.PkgManifestChecker
pkg.lint.opensolaris.OpenSolarisManifestChecker
```

終了ステータス

次の終了ステータスが返されます。

0 コマンドが成功しました。

1 1 つ以上の lint チェックが出力を発行しました。

2　無効なコマンド行オプションが指定された。

99　予期しない例外が発生しました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目

pkg(1)、pkg.depotd(1m)、pkgsend(1)、pkg(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

名前　pkgmerge – Image Packaging System パッケージマージユーティリティー

形式

```
/usr/bin/pkgmerge [-n] -d dest_repo
    -s variant=value[,...],src_repo ...
    [pkg_fmri_pattern ...]
```

機能説明　pkgmerge は、複数バリアントのパッケージを作成するためのパッケージ公開ツールです。このツールはそのために、名前とバージョン (タイムスタンプは除外) が同じパッケージをマージし、特定のソースの指定されたバリアント名および値とマージされるバージョン内で一意であるアクションをタグ付けしたあと、新しいパッケージをターゲットリポジトリに公開します。各ソースのすべてのパッケージの最新バージョンのみが使用されます。

アクションの属性 pkg.merge.blend がマージされるバリアントの名前に設定されている場合は、アクションの最終的な出力に追加されたバリアントタグが含まれないようにするために、そのアクションはマージの前にほかのマニフェストにコピーされます。属性 pkg.merge.blend 自体は、出力マニフェストですべてのアクションから削除されることに注意してください。この属性は、複数のパスマージに異なる値を使用して繰り返すことができます。

入力マニフェスト内の同じパスに入る同一でないアクションがあると、pkgmerge はエラーで終了します。

オプション　サポートしているオプションは、次のとおりです。

-d *dest_repo*
　マージされたパッケージを公開する先のターゲットリポジトリのファイルシステムのパスまたは URI。このターゲットリポジトリはすでに存在している必要があります。新しいリポジトリは、pkgrepo(1) を使用して作成できます。

-n
　ターゲットリポジトリを変更することなく、試験実行を行います。

-s *variant= value*[,...],*src_repo*
　このソースのパッケージに使用するバリアント名および値。そのあとに、パッケージを取得する元のソースリポジトリまたはパッケージアーカイブのファイルシステムのパスまたは URI が続きます。複数のバリアントをコンマで区切って指定できます。すべてのソースに対して同じバリアントを指定する必要があります。このオプションは複数回指定できます。

--help or -?
　使用方法に関するメッセージを表示します。

環境変数　次の環境変数がサポートされています。

TMPDIR　プログラム実行中に一時データが格納されるディレクトリの絶対パス。設定されていない場合、一時データはデフォルトで /var/tmp に格納されます。

使用例

例 1 バリアント名および値を指定する

指定されたソース内に見つかった各パッケージを、取得元のソースに指定された特定のバリアント名および値でタグ付けします。

```
$ pkgmerge -s arch=sparc,http://src.example.com \
-d http://dest.example.com
```

サンプルパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T163427Z
dir group=sys mode=0755 owner=root path=usr
```

操作のあとのサンプルパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T163427Z
set name=variant.arch value=sparc
dir group=sys mode=0755 owner=root path=usr
```

例 2 パッケージをマージおよび公開する

特定のソースの最新バージョンの各パッケージをマージし、新しいパッケージをターゲットリポジトリに公開します。

```
$ pkgmerge -s arch=sparc,http://src1.example.com \
-s arch=i386,http://src2.example.com \
-d /path/to/target/repository
```

ソース 1 (SPARC) のサンプルパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T121410Z
file id mode=0555 owner=root group=bin path=usr/bin/foo
dir group=sys mode=0755 owner=root path=usr
```

ソース 2 (i386) のサンプルパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T163427Z
file id mode=0555 owner=root group=bin path=usr/bin/foo
dir group=sys mode=0755 owner=root path=usr
```

マージされたパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T163427Z
set name=variant.arch value=sparc value=i386
file id mode=0555 owner=root group=bin path=usr/bin/foo variant.arch=sparc
file id mode=0555 owner=root group=bin path=usr/bin/foo variant.arch=i386
dir group=sys mode=0755 owner=root path=usr
```

例 3　i386 システムと SPARC システムのデバッグおよびデバッグ以外のパッケージをマージする

i386 システムと SPARC システムの一連のデバッグおよびデバッグ以外のリポジトリ内の最新バージョンの各パッケージをマージします。

```
$ pkgmerge -s arch=sparc,debug=false,/repo/sparc-nondebug \
-s arch=sparc,debug=true,/repo/sparc-debug \
-s arch=i386,debug=false,/repo/i386-nondebug \
-s arch=i386,debug=true,/repo/i386-debug \
-d /path/to/target/repository
```

ソース 1 (SPARC デバッグ以外) のサンプルパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T121410Z
file id mode=0555 owner=root group=bin path=usr/bin/foo
dir group=sys mode=0755 owner=root path=usr
```

ソース 2 (SPARC デバッグ) のサンプルパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T121411Z
file id mode=0555 owner=root group=bin path=usr/bin/foo
dir group=sys mode=0755 owner=root path=usr
```

ソース 3 (i386 デバッグ以外) のサンプルパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T163427Z
file id mode=0555 owner=root group=bin path=usr/bin/foo
dir group=sys mode=0755 owner=root path=usr
```

ソース 4 (i386 デバッグ) のサンプルパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T163428Z
file id mode=0555 owner=root group=bin path=usr/bin/foo
dir group=sys mode=0755 owner=root path=usr
```

マージされたパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T163428Z
set name=variant.arch value=sparc value=i386
set name=variant.debug value=false value=true
file id mode=0555 owner=root group=bin path=usr/bin/foo variant.arch=sparc
     variant.debug=false
file id mode=0555 owner=root group=bin path=usr/bin/foo variant.arch=sparc
     variant.debug=true
file id mode=0555 owner=root group=bin path=usr/bin/foo variant.arch=i386
     variant.debug=false
file id mode=0555 owner=root group=bin path=usr/bin/foo variant.arch=i386
     variant.debug=true
dir group=sys mode=0755 owner=root path=usr
```

例 4　pkg.merge.blend を使用してマージする

pkg.merge.blend 属性を使用して、競合しない 2 つのアーキテクチャーのパッケージをマージします。

```
$ pkgmerge -s arch=sparc,http://src1/example.com \
-s arch=i386,http://src2.example.com \
-d /path/to/target/repository
```

ソース 1 (SPARC) のサンプルパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T121410Z
file 1d5eac1aab628317f9c088d21e4afda9c754bb76 mode=0555 owner=root \
    group=bin path=usr/bin/sparc/foo pkg.merge.blend=arch
file d285ada5f3cae14ea00e97a8d99bd3e357caadc0 mode=0555 owner=root \
    group=bin path=usr/bin/foo
dir group=sys mode=0755 owner=root path=usr
```

ソース 2 (i386) のサンプルパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T163427Z
file a285ada5f3cae14ea00e97a8d99bd3e357cb0dca mode=0555 owner=root \
    group=bin path=usr/bin/i386/foo pkg.merge.blend=arch
file d285ada5f3cae14ea00e97a8d99bd3e357caadc0 mode=0555 owner=root \
    group=bin path=usr/bin/foo
dir group=sys mode=0755 owner=root path=usr
```

マージされたパッケージ:

```
set name=pkg.fmri value=pkg://example.com/foo@5.11,5.11-0.200:20381001T163427Z
set name=variant.arch value=sparc value=i386
file d285ada5f3cae14ea00e97a8d99bd3e357caadc0 mode=0555 owner=root \
    group=bin path=usr/bin/foo
file a285ada5f3cae14ea00e97a8d99bd3e357cb0dca mode=0555 owner=root \
    group=bin path=usr/bin/i386/foo
file 1d5eac1aab628317f9c088d21e4afda9c754bb76 mode=0555 owner=root \
    group=bin path=usr/bin/sparc/foo
dir group=sys mode=0755 owner=root path=usr
```

終了ステータス

次の終了ステータスが返されます。

0　コマンドが成功しました。

1　エラーが発生した。

2　無効なコマンド行オプションが指定された。

99　予期しない例外が発生しました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目

pkgrepo(1)、pkg(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

名前　pkgmogrify – Image Packaging System マニフェスト変換ツール

形式

```
/usr/bin/pkgmogrify [-vi] [-I includedir ...]
    [-D macro=value ...] [-O outputfile]
    [-P printfile] [inputfile ...]
```

機能説明　`pkgmogrify` は、パッケージマニフェストのプログラムによる編集を可能にして、ソフトウェアの構築やパッケージの公開を自動化するときに必要な標準的な変換を簡略化します。

`pkgmogrify` は、次の機能を提供します。

- マクロ置換。単一のマニフェストを各種アーキテクチャーやプラットフォーム間で簡単に共有できます。
- ほかのマニフェスト、または標準のコンポーネントや変換などのマニフェストフラグメントの取り込み。
- パッケージアクションの変換。アクション属性の変更、削除、または追加が含まれます。

オプション　サポートしているオプションは、次のとおりです。

`-D` *name=value*
name を値 *value* とともにマクロとして定義します。マクロは、入力ファイル内で `$(macro)` として指定されます。置換は、それ以上の変換が見つからなくなるまで繰り返されます。一般的な語法には次のものがあります。

- 行の先頭にあるアーキテクチャー固有のタグを使用した、ほかのアーキテクチャー上のマニフェスト内の行の除外。

  ```
  $(sparc_ONLY)file ...
  ```

  SPARC アーキテクチャーを処理する場合、このマクロは空の文字列に設定されます。ほかのアーキテクチャーを処理する場合、このマクロはコマンド行で `#` に設定されるため、このアクションが現在のアーキテクチャー上のマニフェストから除外されます。
- パス名のプラットフォーム固有の部分の指定。実行可能ファイルおよびライブラリのための 64 ビットアーキテクチャーディレクトリの名前などがあります。

  ```
  file NOHASH path=usr/bin/$(ARCH64)/cputrack ...
  ```

  これらのマクロは、コマンド行で目的の値に設定するようにしてください。定義済みのマクロ値は存在しません。

`-I` *include_directory*
指定されたディレクトリを、コマンド行で指定されたファイルと、埋め込みのインクルード指令の両方の検索パスに追加します。

-O *outputfile*
マニフェスト出力を指定されたファイルに書き込みます。このファイルは、エラーが発生した場合や、変換指令によって強制的な中止操作が実行された場合は書き込まれません。デフォルトでは、マニフェスト出力は標準出力に書き込まれます。

-P *printfile*
変換指令の出力操作から得られた出力を指定されたファイルに書き込みます。このファイルは、エラーが発生した場合や、変換指令によって強制的な中止操作が実行された場合は書き込まれません。デフォルトでは、この出力は標準出力に書き込まれます。

-i
ファイル内のインクルード指令を無視します。コマンド行 (または標準入力) で指定されたファイルのみが処理されます。

-v
出力マニフェストに変換の効果を示すコメントを書き込みます。この情報がデバッグに役立つことがあります。

--help or -?
使用方法に関するメッセージを表示します。

埋め込みの指令

マニフェストファイルでは、インクルード指令と変換指令という 2 つのタイプの指令がサポートされています。

インクルード指令の形式は次のとおりです。

```
<include file>
```

この指令により、pkgmogrify は file という名前のファイルを、最初に現在のディレクトリで、次に -I オプションで指定されたディレクトリで検索します。見つかった場合は、ファイルの内容がマニフェストの、この指令が置かれている場所に挿入されます。見つからなかった場合、pkgmogrify はエラーで終了します。

変換指令の形式は次のとおりです。

```
<transform matching-criteria -> operation>
```

これらの指令は、すべての入力がメモリー内に読み取られるまで累積されたあと、各指令が検出された順序でアクションに適用されます。

一致条件の形式は次のとおりです。

```
[action-type ... ] [attribute=<value-regexp> ...]
```

指定された *action-types* のいずれかが一致する必要があります。指定された *attributes* のすべてが一致する必要があります。使用されている正規表現の構文は Python の構文です。Python 正規表現の構文については、コマンド pydoc re を使用するか、または http://docs.python.org/dev/howto/regex.html にあるより完全なド

キュメントを参照してください。正規表現は先頭に固定され、末尾には固定されません。そのため、ファイルを拡張子で照合する正規表現には先頭に .* を含める必要があり、さらに末尾には $ を含めるようにしてください。

複数の条件をスペースで区切って指定できます。

次の操作が使用できます。

add　属性に値を追加します。この操作は 2 つの引数を取ります。最初の引数は属性の名前であり、2 番目の引数は値です。

default　属性の値がまだ存在しない場合は、その値を設定します。この操作は add 操作と同じく 2 つの引数を取ります。

delete　属性値を削除します。この操作は 2 つの引数を取ります。最初の引数は属性の名前です。2 番目の引数は、削除される属性値を照合するための正規表現です。アクションを照合するために使用される正規表現とは異なり、この表現は固定されません。

drop　このアクションを破棄します。

edit　アクションの属性を変更します。この操作には 3 つの引数を指定します。最初の引数は属性の名前であり、2 番目の引数は属性値を照合するための正規表現です。3 番目の引数は、正規表現で一致した値の部分に置き換えられる置換文字列です。アクションを照合するために使用される正規表現とは異なり、この表現は固定されません。正規表現でグループが定義されている場合は、置換文字列内で通常の正規表現の後方参照 (形式は \1、\2 など) を使用できます。

emit　マニフェスト出力ストリームに行を出力します。これは、有効なアクション文字列、空 (空行が生成されます)、またはコメント (# のあとに任意のテキストが続きます) である必要があります。

exit　マニフェスト処理を終了します。マニフェストは出力されず、print 操作は適用されません。引数が 1 つ指定されている場合、その引数は整数である必要があり、終了コードとして使用されます。デフォルト値は 0 です。引数が 2 つ指定されている場合、最初の引数は終了コードであり、2 番目の引数は標準エラー出力に出力されるメッセージです。

print　-P で指定された出力ファイルにメッセージを出力します。

set　属性の値を設定します。この操作は add 操作と同じく 2 つの引数を取ります。

delete と drop を除くすべての操作は、出力ストリームにその内容が出力される (省略可能である可能性がある) 引数を取ります。これらの文字列には、3 種類の特殊なトークンを含めることができます。これらのトークンにより、各アクションの固定された変換に基づかない情報を出力に含めることが可能になります。

最初の種類の置換を使用すると、パーセント記号に続けて属性の名前を丸括弧に入れて指定することによって、操作で現在のアクションの属性の値を参照できます。たとえば、%(alias) は、アクションの alias 属性の値を参照します。

いくつかの合成属性が存在します。次の 2 つは pkgmogrify に固有です

- pkg.manifest.filename は、アクションが見つかったファイルの名前を参照します。
- pkg.manifest.lineno は、アクションが見つかった行を参照します。

次の 3 つの合成属性は、pkg(1) で使用されるものに似ています。

- action.hash は、アクションにペイロードが含まれている場合、そのアクションのハッシュ値を参照します。ペイロードを含むアクションの場合、set 操作は、action.hash 属性を操作することによってアクションのハッシュを変更できます。
- action.key は、キー属性の値を参照します。
- action.name は、アクションタイプの名前を参照します。

値を要求された属性が存在しない場合、pkgmogrify はエラーで終了します。エラーで終了することを回避するには、属性名のあとに ;notfound= と、属性値の代わりに置換する値を指定します。たとえば、%(alias;notfound='no alias') は、属性 alias の値が存在する場合はその値を出力し、それ以外の場合は no alias を出力します。

値を要求された属性が複数値の場合は、各値がスペースで区切られて出力されます。notfound トークンと同様に、トークン prefix、suffix、および sep を使用してこの動作を変更できます。prefix で指定された文字列は各値の先頭に追加され、suffix で指定された文字列は各値の末尾に追加され、sep はある値の接尾辞とその次の値の接頭辞の間に配置されます。

アクション属性と同様に、pkgmogrify 指令では %{pkg.fmri} のように、丸括弧の代わりに中括弧を使用してパッケージ属性を参照できます。変換指令が適用される時点で、この属性は set アクションで定義されている必要があります。そうしないと、この属性は上で説明した notfound として扱われます。処理が、パッケージを記述しているマニフェストファイルの最後に達すると、属性は次のパッケージのためにクリアーされます。

これは、パッケージ属性をアクション属性であるかのように参照するためだけでなく、これらの属性を照合したり、一時的に変更したりするためにも有効です。したがって、これらの状況では、合成アクション名 pkg を (pkgmogrify のコンテキストでのみ) 使用できます。

pkgmogrify がコマンド行で指定されたマニフェストの読み取りを完了し、そのマニフェストで pkg.fmri パッケージ属性が定義されている場合、pkgmogrify はこの合成

アクション pkg を作成します。属性はパッケージの属性になります。その後、<transform> 指令は、ほかのアクションタイプと同様に、このアクションに照合できます。

pkg アクションに対する操作は、メモリー内でのみ実行されるために、出力されたマニフェストに直接影響を与えないという点で特殊です。たとえば、add、default、または set 操作を使用して pkg アクションに属性を設定しようとしても、照合するほかの <transform> 指令では使用可能であるにもかかわらず、set アクションがマニフェストに追加されません。pkg アクションに対して emit を実行しようとすると、エラーが発生します。パッケージ属性を追加するには、代わりに set アクションに対して emit を実行します。

3 番目の種類の置換は後方参照です。この置換は edit 操作で使用可能なものとは異なり、 -> の左側にある変換一致に列挙されているグループへの参照です。これらは、一致条件にある順序で、%<1>、%<2> などによって指定されます。

処理の順序は次のとおりです。

1. 入力ファイルから行が読み取られます。
2. マクロが適用されます。
3. <include ...> および <transform> 指令が処理されることにより、追加のファイルが検出され、読み取られます。
4. すべての入力が累積されたあと、入力内の各行がアクションに変換され、すべての変換が適用されます。
5. 処理が正常に完了すると、出力が書き込まれます。

使用例

例 1 SMF マニフェストにタグを追加する

サービス管理機能 (SMF) マニフェストにタグを追加して、パッケージがライブシステムにインストールされたときにこれらのマニフェストがインポートされるようにします。

```
<transform file path=(var|lib)/svc/manifest/.*\.xml -> \
    add restart_fmri svc:/system/manifest-import:default>
```

例 2 ファイルを移動する

ファイルを usr/sfw/bin から usr/bin に移動します。

```
<transform file -> edit path usr/sfw/bin usr/bin>
```

例 3 リブートの必要性を指定する

reboot-needed タグを、/kernel の下にある、.conf ファイル以外のファイルに追加します。この例では、入力ファイルに存在する順序で各アクションに変換が適用されることを利用しています。

例 3 リブートの必要性を指定する (続き)

```
<transform file path=kernel/.* -> set reboot-needed true>
<transform file path=kernel/.*\.conf -> delete reboot-needed .*>
```

これはまた、正規表現を使用した 1 つの変換規則でも実行できます。

例 4 FMRI 属性を依存アクションに変換する

パッケージ属性 pkg.fmri を incorporation の一部になるように depend アクションに変換します。

```
<transform set name=pkg.fmri -> \
    emit depend type=incorporate fmri=%(value)>
<transform set name=pkg.fmri -> drop>
```

例 5 バグ番号のリストを出力する

引用符で囲まれた、接頭辞付きのバグ番号のコンマ区切りリストを出力します。

```
set name=bugs value=12345 value=54321 value=13579 value=97531
<transform set name=bugs -> \
    print %(value;sep=",";prefix="bug='";suffix="'")>
```

例 6 欠落した属性を許可する

属性がない場合でも、安全にメッセージを出力します。

```
<transform driver -> print Found aliases: %(alias;notfound=<none>)>
```

例 7 デフォルト値を設定する

所有者、グループ、およびアクセス権のデフォルト値を設定します。

```
<transform file dir -> default owner root>
<transform file dir -> default group bin>
<transform file -> default mode 0444>
<transform dir -> default mode 0755>
```

例 8 廃止としてマークされていないパッケージに依存関係を追加する

廃止としてマークされていないすべてのパッケージについて、そのパッケージを提供する統合のための incorporation への依存関係を追加します。この一連の変換は、マニフェストが読み取られたあとに実行される必要があります。そうしないと、依存関係が常に出力されます。pkg アクションの変更には永続的な効果がないため、pkg.obsolete=false に一致する属性をクリーンアップする必要はありません。

```
<transform pkg -> default pkg.obsolete false>
<transform pkg pkg.obsolete=false -> emit depend \
    fmri=consolidation/$(CONS)/$(CONS)-incorporation type=require>
```

例 9 エラーが検出された場合は終了してメッセージを出力する

廃止された属性がマニフェスト内に検出された場合は、エラーメッセージを出力します。

```
<transform file dir link hardlink opensolaris.zone=.* -> \
    exit 1 The opensolaris.zone attribute is obsolete.>
```

例 10 適切なロケールファセットを設定する

対象のパス名に適したロケールファセットを設定します。

```
<transform dir file link hardlink path=.*/locale/([^/]+).* -> \
    default facet.locale.%<1> true>
```

終了ステータス

次の終了ステータスが返されます。

0 すべてが動作しました。

1 予期しない不具合が発生しました。

2 無効なコマンド行オプションが指定された。

99 予期しない処理エラー。

ファイル

`/usr/share/pkg/transforms` このディレクトリには、ファセット、アクチュエータ、およびその他の属性を設定するために役立つ変換を含むファイルが入っています。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目

pkg(1)、pkg(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

名前

pkgrecv – Image Packaging System 内容取得ユーティリティー

形式

```
/usr/bin/pkgrecv [-s src_uri] [-a] [-d (path|dest_uri)]
    [-c cache_dir] [-kr] [-m match] [-n] [--raw]
    [--key keyfile --cert certfile] (fmri|pattern) ...

/usr/bin/pkgrecv [-s src_uri] --newest
```

機能説明

pkgrecv を使用すると、ユーザーは pkg(5) リポジトリまたはパッケージアーカイブからパッケージを取得できます。また、pkgrecv はオプションで、取得されたパッケージを別のパッケージリポジトリに再公開したり、アーカイブしたりすることもできます。デフォルトでは、パッケージは pkg(1)、pkg.depotd(1M)、およびパッケージ公開ツールでの使用に適したパッケージリポジトリ形式で取得されます。

pkgrecv 操作後に、リポジトリに対して pkgrepo refresh または pkgrepo rebuild を実行し、検索インデックスを構築します。

オプション

サポートしているオプションは、次のとおりです。

-a
: 取得されたパッケージデータを、-d で指定された場所にある pkg(5) アーカイブ内に格納します。このファイルがすでに存在していてはいけません。このオプションは、ファイルシステムベースの出力先の場合にのみ使用できます。必須ではありませんが、.p5p のファイル拡張子 (たとえば、archive.p5p) を使用することを強くお勧めします。このオプションを --raw と組み合わせることはできません。

-c *cache_dir*
: ダウンロードされた内容をキャッシュするために使用されるディレクトリのパス。このディレクトリが指定されていない場合は、クライアントによってキャッシュディレクトリが自動的に選択されます。ダウンロードが中断されたときに、キャッシュディレクトリが自動的に選択されていた場合は、このオプションを使用してダウンロードを再開します。一時的なデータストレージに使用される場所を設定する方法についての詳細は、下の「環境変数」のセクションを参照してください。

-d *path_or_uri*
: パッケージを再公開する先のターゲットのファイルシステムのパスまたは URI。-a が指定されている場合、ターゲットはまだ存在しない新しいパッケージアーカイブです。指定されていない場合、ターゲットはすでに存在するパッケージリポジトリである必要があります。新しいリポジトリは、pkgrepo(1) を使用して作成できます。

-h
: 使用方法に関するメッセージを表示します。

-k
: 取得されたパッケージの内容を圧縮されたままにします。このオプションは、再公開時には無視されます。圧縮されたパッケージの内容を pkgsend(1) で使用しないようにしてください。

`-m` *match*
: 次の値を使用して、マッチング動作を制御します。

  `all-timestamps`
  : 最新のタイムスタンプだけでなく、一致するすべてのタイムスタンプを含めます (`all-versions` が暗黙的に指定されます)。

  `all-versions`
  : 最新のバージョンだけでなく、一致するすべてのバージョンを含めます。

`-n`
: 変更を行うことなく、試験実行を行います。

`-r`
: 提供されたパッケージリストのすべての依存関係を再帰的に取得します。

`-s` *src_repo_uri*
: パッケージデータ受信先の pkg(5) リポジトリまたはパッケージアーカイブの場所を表す URI。

`--cert` *file*
: HTTPS リポジトリからのパッケージ取得に使用するクライアント SSL 証明書ファイルを指定します。

`--key` *file*
: HTTPS リポジトリからのパッケージ取得に使用するクライアント SSL キーファイルを指定します。

`--newest`
: 指定されたリポジトリから使用可能な最新バージョンのパッケージを一覧表示し、終了します (`-s` を除くその他のオプションはすべて無視されます)。

`--raw`
: raw パッケージデータを取得し、`-d` で指定された場所に、幹およびバージョンごとの一連のディレクトリ構造内に格納します。このオプションは、ファイルシステムベースの出力先の場合にのみ使用できます。このパッケージデータを使用すると、一般にはファイルの内容を修正するか、または追加のパッケージメタデータを提供することによって、パッケージを便利に変更して再公開することができます。このオプションを `-a` と組み合わせることはできません。

使用例

例 1　最新のパッケージを一覧表示する

`test` という名前のシステム上のリポジトリから使用可能な最新のパッケージを一覧表示します。

```
$ pkgrecv -s http://test --newest
pkg://solaris/system/library/c++-runtime@0.5.11,5.11-0.174.0.0.0.0.0:20110921T190358Z
pkg://solaris/system/library/freetype-2@2.4.8,5.11-0.175.1.0.0.7.1234:20120109T215840Z
pkg://solaris/system/library/math@0.5.11,5.11-0.174.0.0.0.0.0:20110921T190432Z
```

例 2　raw パッケージデータを取得する

例 1 から `c++-runtime` パッケージを `pkgsend publish` で使用するために適した形式で受け取ります。

例2 rawパッケージデータを取得する (続き)

```
$ pkgrecv -s http://test \
-d /local/repo --raw \
c++-runtime@0.5.11,5.11-0.174.0.0.0.0.0:20110921T190358Z
Processing packages for publisher solaris ...
Retrieving and evaluating 1 package(s)...
PROCESS                                          ITEMS     GET (MB)    SEND (MB)
Completed                                          1/1      3.5/3.5      0.0/0.0
$ ls /local/repo
pkg5.repository  publisher  system%2Flibrary%2Fc%2B%2B-runtime
```

例3 システムから依存関係を取得する

test という名前のシステムから、パッケージ editor/vim とそのすべての依存関係を受け取ります。

```
$ pkgrecv -s http://test -d /local/repo -r editor/vim
```

例4 すべてのバージョンを取得する

test という名前のシステムから、パッケージ editor/vim のすべてのバージョンを受け取ります。

```
$ pkgrecv -s http://test -d /local/repo -m all-versions editor/vim
Processing packages for publisher solaris ...
Retrieving and evaluating 2 package(s)...
PROCESS                                        ITEMS     GET (MB)    SEND(MB)
Completed                                        2/2    16.7/16.7    44.9/44.9
```

例5 すべてのバージョンを取得し、リモートから再公開する

test という名前のシステムから、パッケージ library/zlib のすべてのバージョンを受け取り、それを remote という名前のシステム上のリモートリポジトリに再公開します。

```
$ pkgrecv -s http://test -d http://remote:10000 -m all-versions library/zlib
```

例6 リポジトリから依存関係を取得する

/export/repo にあるリポジトリから、パッケージ editor/gnu-emacs とそのすべての依存関係を受け取ります。

```
$ pkgrecv -s /export/repo -d /local/repo -r editor/gnu-emacs
```

例7 追加のパッケージを取得する

http://example.com:10000 にあるリポジトリから、まだ存在しないすべてのパッケージを受信します。

```
$ pkgrecv -s http://example.com:10000 -d /my/pkg/repo '*'
```

例8 パッケージアーカイブを作成する

http://example.com:10000 にあるリポジトリから、パッケージ editor/gnu-emacs とそのすべての依存関係を含むパッケージアーカイブを作成します。

```
$ pkgrecv -s http://example.com:10000 -d /my/emacs.p5p -a -r editor/gnu-emacs
```

例9 パッケージをアーカイブからリポジトリにコピーする

パッケージアーカイブ内のすべてのパッケージを /export/repo にある既存のリポジトリにコピーします。

```
$ pkgrecv -s /my/archive.p5p -d /export/repo '*'
```

環境変数

次の環境変数がサポートされています。

PKG_DEST　取得されたパッケージを保存する先のディレクトリのパス、またはパッケージがコピーされるリポジトリまたはパッケージアーカイブのファイルシステムのパスまたは URI。

PKG_SRC　パッケージ取得先の pkg(5) リポジトリまたはパッケージアーカイブの場所を表す URI またはファイルシステムのパス。

TMPDIR　プログラム実行中に一時データが格納されるディレクトリの絶対パス。設定されていない場合、一時データはデフォルトで /var/tmp に格納されます。

終了ステータス

次の終了ステータスが返されます。

0　コマンドが成功しました。

1　エラーが発生した。

2　無効なコマンド行オプションが指定された。

3　複数の操作が要求されましたが、それらの一部のみが成功しました。

99　予期しない例外が発生しました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目

pkgrepo(1)、pkgsend(1)、pkg(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

名前　pkgrepo – Image Packaging System リポジトリ管理ユーティリティー

形式

```
/usr/bin/pkgrepo create [--version ver] uri_or_path

/usr/bin/pkgrepo add-publisher -s repo_uri_or_path publisher ...

/usr/bin/pkgrepo get [-F format] [-p publisher ...]
    -s repo_uri_or_path [section/property ...]

/usr/bin/pkgrepo info [-F format] [-H] [-p publisher ...]
    -s repo_uri_or_path

/usr/bin/pkgrepo list [-F format] [-H] [-p publisher ...]
    -s repo_uri_or_path [pkg_fmri_pattern ...]

/usr/bin/pkgrepo rebuild [-p publisher ...]
    -s repo_uri_or_path [--no-catalog] [--no-index]

/usr/bin/pkgrepo refresh [-p publisher ...]
    -s repo_uri_or_path [--no-catalog] [--no-index]

/usr/bin/pkgrepo remove [-n] [-p publisher ...]
    -s repo_uri_or_path pkg_fmri_pattern ...

/usr/bin/pkgrepo set [-p publisher] -s repo_uri_or_path
    section/property=[value] ... or
    section/property=([value]) ...

/usr/bin/pkgrepo help

/usr/bin/pkgrepo version
```

機能説明　pkgrepo を使用すると、pkg(5) パッケージリポジトリの作成および管理を行うことができます。パッケージリポジトリは、pkg(1) や、pkgsend(1) または pkgrecv(1) などの公開クライアントがパッケージデータを格納したり取得したりできるようにするための、定義済みの一連のディレクトリおよびファイルです。さらに、パッケージリポジトリへのネットワークベースのアクセスが必要な場合、pkg.depotd(1m) は、クライアントにパッケージデータを格納したり取得したりするためのリポジトリへのアクセスを提供できます。

オプション　サポートしているオプションは、次のとおりです。

--help or -?　使用方法に関するメッセージを表示します。

サブコマンド　サポートされているサブコマンドは次のとおりです。

create [--version *ver*] *uri_or_path*
指定された場所に pkg(5) リポジトリを作成します。

このサブコマンドは、ファイルシステムベースのリポジトリでのみ使用できます。

`--version` が指定されている場合は、指定されたバージョンと互換性がある形式でリポジトリを作成します。デフォルトでは、バージョン 4 のリポジトリが作成されます。サポートされるバージョンは次のとおりです。

3　　1 つのパブリッシャー、カタログバージョン 1、および検索バージョン 1 でのパッケージの格納をサポートします。

4　　複数のパブリッシャー、カタログバージョン 1、および検索バージョン 1 でのパッケージの格納をサポートします。

`add-publisher -s` *repo_uri_or_path publisher* `...`
指定されたパブリッシャーをリポジトリに追加します。新しいパブリッシャーにパッケージや内容は含まれていません。

このサブコマンドは、バージョン 4 のファイルシステムベースのリポジトリでのみ使用できます。

`get [-F` *format*`] [-p` *publisher* `...] -s` *repo_uri_or_path* `[`*section/property* `...]`
リポジトリまたはそのパブリッシャーのプロパティー情報を表示します。

デフォルトでは、各プロパティーとその値が個別の行に出力されます。空の ASCII 文字列値は、二重引用符 ("") のペアで表されます。ASCII 文字列値内の次の Bourne シェルのメタキャラクタと、改行、スペース、およびタブは、バックスラッシュ文字 (\) でエスケープする必要があります。

```
; & ( ) | ^ < > \ " ’ ‘
```

「使用例」のセクションを参照してください。

指定できるプロパティーのリスト、および各プロパティーの目的と値については、下の `set` サブコマンドを参照してください。

`-F` が指定されている場合は、代替の出力形式を指定します。*format* の値は、`tsv` (Tab Separated Values)、`json` (単一行としての JavaScript Object Notation)、または `json-formatted` (読みやすい形式にされた JavaScript Object Notation) にできます。

`-H` が指定されている場合は、リストのヘッダーを省略します。

`-p` が指定されている場合は、指定されたパブリッシャーのプロパティー情報を表示します。特殊な値 `all` により、すべてのパブリッシャーのプロパティーが表示されます。このオプションは複数回指定できます。

`-s` が指定されている場合は、指定された URI またはファイルシステムのパスにあるリポジトリを操作します。

`info [-F` *format*`] [-H] [-p` *publisher* `...] - s` *repo_uri_or_path*
リポジトリで認識されているパッケージパブリッシャーのリストを表示します。このリストには、パブリッシャーごとのパッケージの数、パブリッシャーのパッケージデータが最後に更新された日時、およびパブリッシャーのパッケージデータのステータス (現在処理されているかどうかなど) が含まれます。

-F が指定されている場合は、代替の出力形式を指定します。*format* の値は、tsv (Tab Separated Values)、json (単一行としての JavaScript Object Notation)、または json-formatted (読みやすい形式にされた JavaScript Object Notation) にできます。

-H が指定されている場合は、リストのヘッダーを省略します。

-p が指定されている場合は、指定されたパブリッシャーのデータのみが表示されます。指定されていない場合は、すべてのパブリッシャーのデータが表示されます。このオプションは複数回指定できます。

-s が指定されている場合は、指定された URI またはファイルシステムのパスにあるリポジトリを操作します。

`list [-F` *format*`] [-H] [-p` *publisher* `...] - s` *repo_uri_or_path* `[`*pkg_fmri_pattern* `...]`
指定された *pkg_fmri_pattern* にマッチする *repo_uri_or_path* リポジトリのパッケージを一覧表示します。パターンが指定されない場合、リポジトリのすべてのパッケージが一覧表示されます。

デフォルトの出力では、最初の列にパッケージのパブリッシャーの名前が含まれます。2 番目の列にはパッケージの名前が含まれます。3 番目の列には、パッケージのステータスを示すフラグが含まれます。ステータス列の o の値は、パッケージが廃止されていることを示します。ステータス列の r の値は、パッケージの名前が変更されたことを示します (廃止の形態の 1 つです)。4 番目の列には、パッケージのリリースおよびブランチのバージョンが含まれます。リリースバージョンとブランチバージョンについては、pkg(5) を参照してください。

-F が指定されている場合は、代替の出力形式を指定します。*format* の値は、tsv (Tab Separated Values)、json (単一行としての JavaScript Object Notation)、または json-formatted (読みやすい形式にされた JavaScript Object Notation) にできます。

-H が指定されている場合は、リストのヘッダーを省略します。

-p が指定されている場合は、指定されたパブリッシャーのパッケージのみが表示されます。指定されていない場合は、すべてのパブリッシャーのパッケージが表示されます。このオプションは複数回指定できます。

-s が指定されている場合は、指定された URI またはファイルシステムのパスにあるリポジトリを操作します。

`rebuild [-p` *publisher* `...] - s` *repo_uri_or_path* `[--no-catalog] [--no-index]`
リポジトリ内に見つかったすべてのカタログ、検索、およびその他のキャッシュされた情報を破棄し、それをリポジトリの現在の内容に基づいて再作成します。

-p が指定されている場合は、指定されたパブリッシャーについてのみ操作を実行します。指定されていない場合や、特殊な値 all が指定されている場合は、すべてのパブリッシャーについて操作が実行されます。このオプションは複数回指定できます。

-s が指定されている場合は、指定された URI またはファイルシステムのパスにあるリポジトリを操作します。

--no-catalog が指定されている場合は、パッケージデータを再構築しません。

--no-index が指定されている場合は、検索インデックスを再構築しません。

refresh [-p *publisher* ...] - s *repo_uri_or_path* [--no-catalog] [--no-index]
リポジトリ内に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。これは、遅延公開 (pkgsend の --no-catalog または --no-index オプション) で使用されることを目的にしています。

-p が指定されている場合は、指定されたパブリッシャーについてのみ操作を実行します。指定されていない場合や、特殊な値 all が指定されている場合は、すべてのパブリッシャーについて操作が実行されます。このオプションは複数回指定できます。

-s が指定されている場合は、指定された URI またはファイルシステムのパスにあるリポジトリを操作します。

--no-catalog が指定されている場合は、新しいパッケージを追加しません。

--no-index が指定されている場合は、検索インデックスを更新しません。

remove [-n] [-p *publisher* ...] -s *repo_uri_or_path pkg_fmri_pattern* ...
リポジトリから、指定されたパターンに一致するパッケージを削除します。これらのパッケージが参照している、ほかのどのパッケージでも使用されていないすべてのファイルも削除されます。

注 – 関連するパブリッシャーのすべての検索インデックスデータが削除されます。

このサブコマンドは、ファイルシステムベースのリポジトリでのみ使用できます。

注意 – この操作は元に戻せません。また、ほかのクライアントがそのリポジトリにアクセスしている間に使用すべきではありません。使用すると、それらのクライアントが取得操作中に失敗する可能性があります。

-n を指定すると、パッケージの変更は行わずに試しに操作を実行します。終了する前に、削除されるパッケージのリストが表示されます。

-p が指定されている場合は、指定されたパブリッシャーの一致するパッケージのみを削除します。指定されていない場合は、すべてのパブリッシャーの一致するパッケージがすべて削除されます。このオプションは複数回指定できます。

-s が指定されている場合は、指定された URI またはファイルシステムのパスにあるリポジトリを操作します。

`set [-p` *publisher*`] - s` *repo_uri_or_path section/property* `=[`*value*`] ... or` *section/property* `=([`*value*`]) ...`

リポジトリまたはパブリッシャーの指定されたプロパティーの値を設定します。

このサブコマンドは、ファイルシステムベースのリポジトリでのみ使用できます。

`-p` が指定されている場合は、指定されたパブリッシャーのプロパティーデータのみを設定します。パブリッシャーがまだ存在しない場合は、追加されます。特殊な値 `all` を使用すると、すべてのパブリッシャーのプロパティーを設定できます。

`-s` が指定されている場合は、指定された URI またはファイルシステムのパスにあるリポジトリを操作します。

プロパティーと値は、次のいずれかの形式を使用して指定できます。

section/*property*=	プロパティー値をクリアーします。
section/*property*= *value*	プロパティー値を指定された値に置き換えます。
section/*property*=(*value1 value2 valueN*)	プロパティー値を値のリストに置き換えます。

リポジトリバージョン 3 および 4 の場合は、リポジトリの次のプロパティーを設定できます。

`publisher/prefix`	デフォルトのパブリッシャーの名前を表す文字列。最初の文字は a-z、A-Z、または 0-9 である必要があります。文字列の残りの部分には、文字 0-9、-、.、a-z、および A-Z のみを含めることができます。この値は、複数の発行元のパッケージが存在するか、パッケージがリポジトリに公開され、発行元が指定されていない場合に、使用するべき発行元を示します。

リポジトリバージョン 3 および 4 の場合は、リポジトリ内の個々のパブリッシャーの次のプロパティーを設定できます。

`publisher/alias`	リポジトリの構成データを使用してパブリッシャーを追加するときにクライアントが使用すべきデフォルトの別名を表す文字列。最初の文字は a-z、A-Z、または 0-9 である必要があります。文字列の残りの部分には、文字 0-9、-、.、a-z、および A-Z のみを含めることができます。

repository/collection_type	このリポジトリで提供されるパッケージのタイプを示す値 core または supplemental を持つことができます。 core 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。core タイプは主に、オペレーティングシステムリポジトリに使用されます。 supplemental タイプは、このリポジトリには、別のリポジトリ内に配置されているパッケージに依存するか、またはそれらのパッケージとともに使用されるためのパッケージが含まれていることを示します。
repository/description	リポジトリの目的と内容を説明した標準テキストの段落。
repository/detailed_url	リポジトリに関する追加情報を提供するドキュメント (Web ページなど) の場所を表す URI。
repository/legal_uris	リポジトリに関する追加の使用条件情報を提供するドキュメントの場所 (URI) のリスト。
repository/mirrors	リポジトリのパッケージ内容のコピーを含むが、パッケージのメタデータは含まないリポジトリの場所 (URI) のリスト。
repository/name	リポジトリの名前を含む標準テキスト文字列。
repository/origins	リポジトリのパッケージのメタデータと内容の完全なコピーを含むリポジトリの場所 (URI) のリスト。
repository/refresh_seconds	クライアントが、更新されたパッケージデータがあるかどうかを調べるためにリポジトリをチェックする際に、各更新チェックのあとに待つべき秒数を表す整数値。
repository/registration_uri	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を表す URI。この 1 つの例に、登録 Web ページがあります。
repository/related_uris	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 (URI) のリ

スト。

ここでは文書化されていないが、get サブコマンドの出力に一覧表示されるプロパティーは内部使用のために予約されているため、設定しないようにしてください。

version
pkg(5) システムのバージョンを識別する一意の文字列を表示します。version 操作によって生成される値はソート可能ではないため、等しいかどうかを超えて安全に比較することはできません。

使用例

例1 パッケージリポジトリを作成する

```
$ pkgrepo create /my/repository
```

例2 情報を表示する

パブリッシャーのサマリーと、リポジトリ内のパッケージの数を表示します。

```
$ pkgrepo info -s /my/repository
PUBLISHER   PACKAGES STATUS UPDATED
example.com 5        online 2011-07-22T18:09:09.769106Z
$ pkgrepo info -s http://pkg.oracle.com/solaris/release/
PUBLISHER PACKAGES STATUS UPDATED
solaris   3941     online 2010-11-12T19:24:25.967246Z
```

例3 カタログと検索データを再構築する

リポジトリのカタログと検索データを再構築します。

```
$ pkgrepo rebuild -s /my/repository
```

例4 カタログと検索データを再表示する

リポジトリのカタログと検索データを再表示します。

```
$ pkgrepo refresh -s /my/repository
$ pkgrepo refresh -s http://example.com/repository
```

例5 すべてのリポジトリプロパティーを表示する

```
$ pkgrepo get -s /my/repository
SECTION    PROPERTY VALUE
publisher  prefix   ""
repository version  4
$ pkgrepo get -s http://pkg.oracle.com/solaris/release/
SECTION    PROPERTY VALUE
publisher  prefix   solaris
repository version  4
```

例 6 すべてのパブリッシャープロパティーを表示する

```
$ pkgrepo get -s http://pkg.oracle.com/solaris/release/ -p all
PUBLISHER SECTION    PROPERTY         VALUE
solaris   publisher  alias
solaris   publisher  prefix           solaris
solaris   repository collection-type  core
solaris   repository description      This\ repository\ serves\ the\ Oracle\
Solaris\ 11\ Package\ repository.
solaris   repository legal-uris       ()
solaris   repository mirrors          (http://pkg-cdn1.oracle.com/solaris.release/)
solaris   repository name             Oracle\ Solaris\ 11\ Package\ Repository
solaris   repository origins          ()
solaris   repository refresh-seconds
solaris   repository registration-uri ""
solaris   repository related-uris     ()
```

例 7 デフォルトのパブリッシャーを設定する

```
$ pkgrepo set -s /my/repository publisher/prefix=example.com
```

例 8 パブリッシャープロパティーを設定する

```
$ pkgrepo set -s /my/repository -p example.com \
repository/origins=http://example.com/repository
```

例 9 新しいパブリッシャーをリポジトリに追加する

```
$ pkgrepo add-publisher -s /my/repository example.com
```

終了ステータス

次の終了ステータスが返されます。

0 コマンドが成功しました。

1 エラーが発生した。

2 無効なコマンド行オプションが指定された。

3 複数の操作が要求されましたが、それらの一部のみが成功しました。

4 変更は行われませんでした。処理がありません。

99 予期しない例外が発生しました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目 pkg(1)、pkgrecv(1)、pkgsend(1)、pkg.depotd(1m)、pkg(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

名前　pkgsend – Image Packaging System の公開クライアント

形式

```
/usr/bin/pkgsend [options] command [cmd_options] [operands]

/usr/bin/pkgsend generate [-T pattern] [--target file]
    source ...

/usr/bin/pkgsend publish [-b bundle ...] [-d source ...]
    [-s repo_uri_or_path] [-T pattern] [--no-catalog]
    [manifest ...]
```

機能説明　pkgsend では、パッケージのマニフェストを使用して、新しいパッケージと新しいパッケージのバージョンをイメージパッケージングリポジトリに公開できます。リポジトリを作成または管理するには、pkgrepo(1) を参照してください。既存のリポジトリ内でパッケージからパッケージアーカイブを作成するには、pkgrecv(1) を参照してください。パッケージのマニフェストの詳細は、pkg(5) を参照してください。

pkgsend 操作後に、リポジトリに対して pkgrepo refresh または pkgrepo rebuild を実行し、検索インデックスを構築します。

オプション　サポートしているオプションは、次のとおりです。

--help or -?　使用方法に関するメッセージを表示します。

サブコマンド　サポートされているサブコマンドは次のとおりです。

generate [-T *pattern*] [--target *file*] *source* ...

各 *source* (SVR4 パッケージ、ディレクトリ、tar ファイルなど) を読み取り、その *source* を標準出力に表示するマニフェストを生成します。出力マニフェストで、file および dir アクションの所有者は root に設定され、グループは bin に設定されています。

これで、出力されたマニフェストに注釈を加え、pkgdepend(1) を使用して依存関係を追加または分析し、 pkglint(1) を使用してその正当性を検証してから publish サブコマンドに渡すことができるようになります。

サポートされているソースを次に示します。

- ファイルシステム形式の SVR4 パッケージ
- データストリーム形式の SVR4 パッケージ
- tar ファイル
- ディレクトリ

ソース内のファイルのベース名が -T で指定されたパターンに一致している場合、ファイルのタイムスタンプがそのファイルのアクションに追加されます。*pattern* は、次のシェルマッチング規則を使用します。

*　すべてと一致します。

?　任意の単一文字と一致します。

[*seq*] *seq* 内にある任意の文字と一致します。

![*seq*] *seq* 内にない文字と一致します。

指定されたソースがディレクトリ内にある場合、単一の i ノードに対して複数のパス名があると、ファイルのアクションをハードリンクのアクションと明確に区別できません。通常、ファイルシステム調査で最初に見つかったものがファイルとして扱われ、残りのものがハードリンクとして扱われます。この処理は、ファイルシステムの実装に応じて自由に設定できます。ファイルとして扱うパス名を指定するには、各パス名を引数として `--target` オプションに渡します。このオプションはほかの種類のソースには影響しません。これは、ソースにはパス名がファイルであるかハードリンクであるかを示す機能があるためです。

SVR4 パッケージがソースとして提供されている場合、`pkgsend` は、クラスアクションスクリプトを持つファイルが存在せず、プリインストールスクリプト、ポストインストールスクリプト、削除前スクリプト、または削除後スクリプトが存在することを確認します。`manifest` クラスを使用してインストールされた SMF マニフェストがある場合、例外が作成されます。`BASEDIR` は、すべての再配置可能パスから削除されます。

SVR4 `DESC` パラメータは、`pkg.description` 値に変換されます。SVR4 `NAME` パラメータは、`pkg.summary` 値に変換されます。

`publish [-b` *bundle* `...] [-d` *source* `...] [-s` *repo_uri_or_path*`] [-T` *pattern*`] [--no-catalog] [`*manifest* `...]`

指定されたパッケージマニフェストを使用するパッケージをターゲットパッケージリポジトリに公開します。これにより、指定されたソースからそのパッケージ用のファイルが取得されます。複数のマニフェストが指定されている場合、それらのマニフェストは指定された順序で追加されます。マニフェストが指定されていない場合、`stdin` からマニフェストが読み取られます。

`-b` を指定すると、マニフェスト内でファイルを検索するときに、指定されたバンドルが、検索するソースの一覧に追加されます。バンドルは、tar ファイルや SVR4 パッケージなどのソースです。このオプションが複数回指定されている場合、ソースはコマンド行に表示される順序で検索されます。`-b` と `-d` の両方が指定されている場合、`-d` のソースが最初に検索されます。サポートされているバンドルとその使用方法の詳細は、上記の `generate` サブコマンドを参照してください。

`-b` を指定すると、マニフェスト内でファイルを検索するときに、指定されたディレクトリが、検索するソースの一覧に追加されます。このオプションが複数回指定されている場合、ソースはコマンド行に表示される順序で検索されます。サポートされているソースとその使用方法の詳細は、上記の `generate` サブコマンドを参照してください。

-s を指定すると、特定の URI またはファイルシステムパスに存在しているリポジトリにパッケージが公開されます。公開についての制限事項と推奨事項の詳細は、次の「注意事項」のセクションを参照してください。また、「環境変数」のセクションも参照してください。

--no-catalog を指定すると、パブリッシャーのカタログにパッケージが追加されなくなります。パブリッシャーのカタログの更新は連続で実行されるため、複数のパッケージを一度に公開する場合には常にこのオプションを使用することを推奨します。公開が完了すると、pkgrepo(1) の refresh サブコマンドを使用して、新しいパッケージを各パブリッシャーのカタログに追加できます。

その他のすべてのオプションの使用法と効果については、上記の generate サブコマンドを参照してください。

環境変数

PKG_REPO　公開先リポジトリのパスまたは URI です。

使用例

例 1　パッケージの生成と公開

pkgsend generate を使用してパッケージを作成し、そのパッケージを公開します。

```
$ pkgsend generate /path/to/proto > /path/to/manifests/foo.p5m
```

example.com パブリッシャーのパッケージ FMRI を、foo.p5m の先頭に追加します。

```
set name=pkg.fmri value=pkg://example.com/foo@1.0
```

結果として生成されるマニフェストは、次のようになります。

```
set name=pkg.fmri value=pkg://example.com/foo@1.0
dir group=sys mode=0755 owner=root path=usr
dir group=bin mode=0755 owner=root path=usr/bin
file usr/bin/foo group=bin mode=0555 owner=root path=usr/bin/foo
```

```
$ pkgsend publish -s http://example.com:10000 -d /path/to/proto \
/path/to/manifests/foo.p5m
```

例 2　簡易パッケージの作成と公開

次の行を含むパブリッシャー example.com に対してマニフェストを作成します。

```
set name=pkg.fmri value=pkg://example.com/foo@1.0-1
file /exdir/foo mode=0555 owner=root group=bin path=/usr/bin/foo
```

パッケージを公開します。

```
$ pkgsend publish -s http://example.com:10000 -d /exdir
```

例 3　既存のマニフェストの使用

ファイルシステムベースの公開と既存のマニフェストを使用してパッケージを公開します。

例 3 既存のマニフェストの使用 *(続き)*

```
$ pkgsend publish -s /tmp/example_repo -d /tmp/pkg_files \
/tmp/pkg_manifest
```

終了ステータス

次の終了ステータスが返されます。

0 コマンドが成功しました。

1 エラーが発生した。

2 無効なコマンド行オプションが指定された。

99 予期しない例外が発生しました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目

pkgdepend(1)、pkgrepo(1)、pkg.depotd(1m)、pkg(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

注意事項

公開プロトコルの制限事項により、サイズが 128MB を超えるパッケージファイルを個別に公開する場合には、ファイルシステムベースの公開を使用する必要があります。ファイルシステムベースの公開は、リポジトリのアクセス制御が必要な場合にも推奨されます。

ファイルシステムベースの公開を使用する場合、公開が完了して Web インタフェースまたは検索応答で変更が反映されたあとに、公開先リポジトリを提供している pkg.depotd プロセスを再開する必要があります。詳細は、pkg.depotd(1M) を参照してください。

名前　pkgsign – Image Packaging System の署名ユーティリティー

形式

```
/usr/bin/pkgsign [-a hash_algorithm]
    [-c path_to_signing_certificate]
    [-i path_to_intermediate_cert] ...
    [-k path_to_private_key] [-n] -s path_or_uri
    [--help] [--no-index] [--no-catalog]
    (fmri|pattern) ...
```

機能説明　pkgsign は、指定されたキーと証明書を使用して署名アクションを追加することにより、リポジトリ内で所定の FMRI のマニフェストを直接更新します。変更されたパッケージは、元のタイムスタンプを保持します。

オプション　サポートしているオプションは、次のとおりです。

-a を指定すると、デフォルトの署名アルゴリズムではなく、*hash_algorithm* 署名アルゴリズムが使用されます。デフォルトの署名アルゴリズムは rsa-sha256 です。サポートされている署名アルゴリズムは、rsa-sha256、rsa-sha384、rsa-sha512、sha256、sha384、sha512 です。ハッシュアルゴリズムを指定するだけの署名アルゴリズムを使用すると、署名の値は、パッケージのマニフェストのハッシュになります。rsa とハッシュアルゴリズムを指定する署名アルゴリズムを使用すると、署名の値は、提供された非公開鍵で署名が追加されたマニフェストのハッシュになります (-c オプションと -k オプションを参照)。

-c を指定すると、証明書 *path_to_signing_certificate* が、このアクションで署名の値を検証するときに使用する証明書として追加されます。-c オプションは、-k オプションでのみ使用されます。

-i を指定すると、証明書 *path_to_intermediate_cert* が、-c への引数として指定された証明書 *path_to_signing_certificate* を検証するときに使用する証明書として追加されます。複数の証明書を追加するには、-i を複数回指定します。

-k を指定すると、*path_to_private_key* に格納された非公開鍵を使用してマニフェストに署名が追加されます。-k オプションは、-c オプションでのみ使用されます。-k が設定されていない場合、署名の値はマニフェストのハッシュになります。

-n を指定すると、評価実行が行われ、リポジトリは一切変更されません。

-s を指定すると、*path_or_uri* にあるリポジトリ内のパッケージに署名が追加されます。

--help を指定すると、使用法のメッセージが表示されます。

--no-index を指定すると、署名付きマニフェストが再公開されたあとに、リポジトリの検索インデックスが更新されません。

--no-catalog を指定すると、署名付きマニフェストが再公開されたあとに、リポジトリのカタログが更新されません。

使用例

例1 マニフェストのハッシュ値を使用した署名の追加

マニフェストのハッシュ値を使用して、http://localhost:10000 に公開されたパッケージに署名を追加します。多くの場合、これはテストに役立ちます。

```
$ pkgsign -s http://localhost:10000 -a sha256 \
example_pkg@1.0,5.11-0:20100626T030108Z
```

例2 キーと証明書を使用した署名の追加

rsa-sha384 を使用して、/foo/bar 内のファイルリポジトリに公開されたパッケージに署名を追加し、マニフェストをハッシュ化して署名を追加します。署名キーは /key/usr2.key に、関連の証明書は /key/usr2.cert に、その証明書の検証に必要な証明書は /icerts/usr1.cert にあります。

```
$ pkgsign -s file:///foo/bar/ -a rsa-sha384 \
-k /key/usr2.key -c /key/usr2.cert -i /icerts/usr1.cert \
example_pkg@1.0,5.11-0:20100626T031341Z
```

終了ステータス

次の終了ステータスが返されます。

0 コマンドが成功しました。

1 エラーが発生した。

2 無効なコマンド行オプションが指定された。

3 複数の操作が要求されましたが、それらの一部のみが成功しました。

99 予期しない例外が発生しました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目

pkg(1), pkgrecv(1)、 pkgsend(1)、 pkgrepo(1)、 pkg(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

名前

pm-updatemanager – パッケージを更新するためのアプリケーション

形式

```
/usr/bin/pm-updatemanager [options]

/usr/bin/pm-updatemanager [-hdR] [--help] [--debug]
    [--image-dir dir]
```

機能説明

pm-updatemanager は、システムにインストールされたパッケージで利用可能な更新の有無を確認し、インストールします。

注 – package/pkg、package/pkg/package-manager、または package/pkg/update-manager パッケージを更新する必要がある場合、pm-updatemanager は、最初にこれらのパッケージを更新してから、残りの更新の実行を再開します。

オプション

サポートしているオプションは、次のとおりです。

-h or --help	使用方法に関するメッセージを表示します。
-d or --debug	デバッグモードで pm-updatemanager を実行します。
-R or --image-dir *dir*	自動的に検出されたイメージではなく、*dir* をルートとするイメージに対して処理を行います。

使用例

例 1 現在のイメージの更新

現在のイメージ上で pm-updatemanager を呼び出します。現在のイメージにインストールされたパッケージで利用可能なすべての更新の有無を確認し、インストールします。

```
$ /usr/lib/pm-launch pm-updatemanager
```

これは、デスクトップのメニューオプションで「システム」>「管理」>「Update Manager」を選択したときに呼び出されるコマンドと同じです。

例 2 指定されたイメージの更新

/aux0/example_root に格納されている pm-updatemanager を呼び出します。

```
$ /usr/lib/pm-launch pm-updatemanager -R /aux0/example_root
```

終了ステータス

次の終了ステータスが返されます。

0 すべてが動作しました。

1 エラーが発生した。

2 無効なコマンド行オプションが指定された。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg/update-manager
インタフェースの安定性	不確実

関連項目

packagemanager(1)、pkg(1)、pkg(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

注意事項

所有権がないイメージに対して処理を行う場合、必要な権限を使用して pm-updatemanager を呼び出す必要があります。通常、このような状況では、/usr/lib/pm-launch を使用して pm-updatemanager を呼び出します。

参 照

システム管理コマンド

名前

pkg.depotd – Image Packaging System 集積サーバー

形式

```
/usr/lib/pkg.depotd [-a address] [-d inst_root] [-p port]
    [-s threads] [-t socket_timeout] [--add-content]
    [--cfg] [--content-root] [--debug feature_list]
    [--disable-ops=op[/1][,...]]
    [--log-access] [--log-errors] [--mirror]
    [--proxy-base url] [--readonly] [--rebuild]
    [--ssl-cert-file] [--ssl-dialog] [--ssl-key-file]
    [--writable-root]
```

機能説明

pkg.depotd は Image Packaging System 用の集積サーバーです。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。ファイルシステムを通してのリポジトリへの直接アクセスをサポートしないクライアント、またはネットワークアクセスが、トランスポートの唯一使用可能な方法であるか、または優先される方法である場合、一般にパッケージ集積を使用します。

pkg(1) などのクライアント (取得するクライアント) はリポジトリから直接または集積サーバー経由で、パッケージおよびパッケージメタデータのリストを取得できます。pkgsend(1) (発行元クライアント) は、リポジトリに直接または集積サーバー経由で、新しいバージョンのパッケージを送信できます。pkgrepo(1) を使用して、集積サーバーで使用するリポジトリを作成したり、直接または集積サーバー経由で、それらを管理したりすることができます。

pkg.depotd はシステムで一般にサービスとして実行されます。パッケージおよびソフトウェア開発者は、テストのためにプライベートコピーを実行する場合があります。

集積庫はそれ自体のアクセス制御方法を提供していません。デフォルトで、接続可能なすべてのクライアントがすべてのパッケージデータを読み込み、新しいパッケージバージョンを公開できます。例外は、SMF (Service Management Facility) のもとで実行した場合であり、デフォルトで読み取り専用モードで実行します。下の注意事項のセクションに、発展するコンテンツを格納するパブリック集積サーバーの保守に関するベストプラクティスを説明します。

Smf プロパティー

pkg.depot サーバーは、一般にそのサービスに関連付けられた smf(5) プロパティーによって構成されます。次のプロパティーが認識されます。

pkg/address

(net_address) 接続を待機する IP アドレス。デフォルト値は 0.0.0.0 (INADDR_ANY) で、すべてのアクティブなインタフェースで待機します。すべてのアクティブな IPv6 インタフェースで待機するには、:: を使用します。最初の値のみが使用されます。

pkg/content_root

(astring) インスタンスがその静的コンテンツまたはその他の Web コンテンツを検索するファイ

ルシステムパス。デフォルト値は /usr/share/lib/pkg です。

pkg/debug
(astring) 有効にするデバッグ機能のカンマ区切りのリスト。取り得る値:

headers　すべての要求のヘッダーをエラーログに記録します。

pkg/disable_ops
(astring) 集積サーバーに対して無効にする操作のカンマ区切りのリスト。操作は *operation*[/*version*] と指定します (たとえば、catalog または search_1)。

pkg/image_root
(astring) ファイル情報がファイルデータのキャッシュとして使用されるイメージのパス。

pkg/inst_root
(astring) インスタンスがそのリポジトリデータを検索するファイルシステムパス。file_root または PKG_REPO が指定されていない場合、必須です。デフォルト値は /var/pkgrepo です。

pkg/log_access
(astring) 集積プロセスによって記録されるアクセス関連情報の保存先。取り得る値は、stderr、stdout、none、または絶対パス名です。stdout が tty の場合、デフォルト値は stdout です。stdout が tty でない場合、デフォルト値は none です。

pkg/log_errors
(astring) 集積プロセスによって記録されるエラーやその他の情報の保存先。 取り得る値は、stderr、stdout、none、または絶対パス名です。 デフォルト値は stderr です。

pkg/mirror
(boolean) パッケージミラーモードを使用するかどうかを設定します。true の場合、公開およびメタデータ操作は無効にされ、限定されたブラウザユーザーインタフェースのみが提供されます。pkg/readonly プロパティーが true の場合、このプロパティーは true にできません。デフォルト値は false です。

pkg/port
(count) 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。SSL 証明書およびキー情報が提供されていない場合、デフォルト値は80 です。提供されている場合、デフォルト値は 443 です。

`pkg/proxy_base`
: (`uri`) これは、集積サーバーのベース URL を変更し、リバースプロキシ構成の Apache またはその他の Web サーバーの背後で実行する場合にもっとも役立ちます。

`pkg/readonly`
: (`boolean`) `pkgsend`(1) によって起動された操作などの変更操作を無効にするかどうかを設定します。取得操作は引き続き使用できます。`pkg/mirror` プロパティーが true の場合、このプロパティーは true にできません。デフォルト値は `true` です。

`pkg/socket_timeout`
: (`count`) サーバーが接続を閉じるまで、クライアントからの応答を待機する最大秒数。デフォルト値は 60 です。

`pkg/sort_file_max_size`
: (`count`) インデクサソートファイルの最大サイズ。集積庫がインデックスの作成に使用する RAM の量を制限したり、速度向上のために RAM の量を増やしたりする場合に使用されます。

`pkg/ssl_cert_file`
: (`astring`) PEM エンコードされた証明書ファイルの絶対パス名。デフォルト値は `none` です。このプロパティーは `ssl_key_file` とともに使用する必要があります。`ssl_cert_file` と `/ssl_key_file` の両方が指定されている場合、集積庫は SSL 要求にのみ応答します。

`pkg/ssl_dialog`
: (`astring`) `ssl_key_file` の暗号化の解除に使用するパスフレーズを取得するために使用する方法を指定します。取り得る値:

  `builtin`
  : パスフレーズの入力を要求します。これがデフォルト値です。

  `exec:`*/path/to/program*
  : 指定された外部プログラムを実行して、パスフレーズを取得します。プログラムの最初の引数は `''` であり、予約されています。プログラムの 2 番目の引数はサーバーのポート番

号です。パスフレーズは stdout に出力されます。

smf:fmri — FMRI に関連するサービスインスタンスからプロパティー pkg_secure/ssl_key_passphrase の値を取得しようとします。

pkg/ssl_key_file — (astring) PEM エンコードされたプライベートキーファイルの絶対パス名。このプロパティーはプロパティー ssl_cert_file とともに使用する必要があります。/ssl_key_file と ssl_cert_file の両方が指定されている場合、集積庫は SSL 要求にのみ応答します。

pkg/threads — (count) 要求を処理するために起動されるスレッド数。デフォルト値は 60 です。小規模の展開にのみ適しています。この値は同時実行クライアント数の約 20 倍にしてください。threads の最大値は 5000 です。

pkg/writable_root — (astring) プログラムが書き込みアクセスできるディレクトリのファイルシステムパス。これは -readonly オプションとともに使用して、集積サーバーがパッケージ情報への書き込みアクセスを必要とせずに、検索インデックスなどのファイルを作成できるようにします。

pkg_secure/ssl_key_passphrase — (astring) pkg/ssl_key_file の暗号化を解除するために使用するパスワード。この値は、属性 solaris.smf.read.pkg-server を使用して読み取り権限保護されます。

集積サーバーのブラウザユーザーインタフェース (BUI) の表示および動作は次のプロパティーを使用して制御されます。

pkg_bui/feed_description — (astring) RSS/Atom フィードの説明の段落。

pkg_bui/feed_icon — (astring) RSS/Atom フィードを視覚的に表現するために使用される小さなイメージのパス名。パス名は content_root の相対パス名にしてください。デフォルト値は web/_themes/pkg-block-icon.png です。

`pkg_bui/feed_logo`	(`astring`) RSS/Atom フィードを視覚的にブランディングまたは識別するために使用される大きなイメージのパス名。この値は `content_root` の相対パス名にしてください。デフォルト値は `web/_themes/pkg-block-icon.png` です。
`pkg_bui/feed_name`	(`astring`) リポジトリにサービスを提供する集積庫によって生成される、RSS/Atom フィードの短い説明的な名前。デフォルト値は「package repository feed」です。
`pkg_bui/feed_window`	(`count`) フィードの生成時に含める、リポジトリのフィードが最後に生成されるまでの時間数。

パッケージ集積庫は、`pkg`(5) からのローカルクライアントイメージのミラーサーバーとして機能することもできます。これにより、LAN 上のサブネットを共有するクライアントはそれらのファイルキャッシュをミラー化します。クライアント間で互いにファイルをダウンロードできるため、パッケージ集積サーバーの負荷が軽減されます。この機能は、`smf`(5) によって構成される代替集積サービスとして利用可能です。サービス検出に mDNS および `dns-sd` を使用します。

mDNS ミラーは一般にそのサービスに関連付けられた `smf`(5) プロパティーによって構成されます。次のプロパティーが認識されます。

`pkg/image_root`	(`astring`) ファイル情報がファイルデータのキャッシュとして使用されるイメージのパス。デフォルト値は `/` です。
`pkg/port`	(`count`) 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。デフォルト値は 80 です。

オプション

`pkg.depotd` はその基本構成情報をファイルから、または既存の `smf`(5) サービスインスタンスのプロパティーデータから読み取ることができます。

`--cfg` *source*	構成データの読み込みおよび書き込み時に使用するファイルのパス名または `smf:`*fmri* の形式の文字列であり、*fmri* は構成データを読み込むインスタンスのサービス障害管理リソース識別子 (FMRI) です。指定されるファイルの形式については、後述する「集積庫の構成」を参照してください。

利用可能な既存の構成ソースがない場合や、`--cfg` を使用して指定された構成ファイルから読み取られた値をオーバーライドする場合は、次のオプションを使用して、集積サーバーのデフォルト動作を変更できます。

`-a address`	上記の `pkg/address` を参照してください。
`--content-root` *root_dir*	上記の `pkg/content_root` を参照してください。
`-d` *inst_root*	上記の `pkg/inst_root` を参照してください。

`--debug` *features*	上記の `pkg/debug` を参照してください。
`--disable-ops` *op_list*	上記の `pkg/disable_ops` を参照してください。
`--image-root` *path*	上記の `pkg/image_root` を参照してください。
`--log-access` *dest*	上記の `pkg/log_access` を参照してください。
`--log-errors` *dest*	上記の `pkg/log_errors` を参照してください。
`--mirror`	上記の `pkg/mirror` を参照してください。
`-p` *port*	上記の `pkg/port` を参照してください。
`--proxy-base` *url*	上記の `pkg/proxy_base` を参照してください。空の値が指定されている場合はこのオプションが無視されます。
`--readonly`	上記の `pkg/readonly` を参照してください。
`-s` *threads*	上記の `pkg/threads` を参照してください。
`-s-ort-file-max-size` *bytes*	上記の `pkg/sort_file_max_size` を参照してください。
`--ssl-cert-file` *source*	上記の `pkg/ssl_cert_file` を参照してください。
`--ssl-dialog` *type*	上記の `pkg/ssl_dialog` を参照してください。
`--ssl-key-file` *source*	上記の `pkg/ssl_key_file` を参照してください。
`-t` *socket_timeout*	上記の `pkg/socket_timeout` を参照してください。
`--writable-root` *path*	上記の `pkg/writable_root` を参照してください。

パッケージリポジトリの追加の管理機能は、`pkgrepo`(1) によって提供されます。

集積庫の構成

(`smf`(5) FMRI の代わりに) `--cfg` オプションを使用して構成ファイルが指定されている場合、集積サーバーは単純なテキスト形式ですべての構成データを読み取りおよび書き込みします。構成データについては、上記の「SMF プロパティー」で説明しています。構成ファイルは [*section*] ヘッダーが先頭に付き、`name` = *value* エントリが後に続くセクションから構成されます。継続のスタイルは RFC 822 です。継続行を空白で始めることによって、値を複数行に分割できます。

構成ファイルで指定されない必須の値は、上記の「オプション」で示すオプションを使用して指定する必要があります。構成ファイルの例は次のようになります。

```
[pkg]
port = 80
inst_root = /export/repo

[pub_example_com]
```

```
feed_description = example.com’s software
  update log
```

使用例

例 1 集積サーバーの有効化

```
# svcadm enable application/pkg/server
```

例 2 サーバーの待機ポートの変更。

```
# svccfg -s application/pkg/server setprop pkg/port = 10000
# svcadm refresh application/pkg/server
# svcadm restart application/pkg/server
```

例 3 ミラーの有効化

```
# svcadm enable application/pkg/dynamic-mirror
```

環境変数

PKG_REPO	サービスを提供するリポジトリを格納したディレクトリを指定します。-d が指定されている場合は、この値は無視されます。
PKG_DEPOT_CONTENT	集積庫によってサービスが提供される静的コンテンツを格納するディレクトリを指定します。下記の「ファイル」のファイルは、このディレクトリに存在するべきですが、それらの内容が提供されるデフォルトの内容と異なっていてもかまいません。

終了ステータス

次の終了ステータスが返されます。

0	操作が成功しました。
1	エラーが発生した。
2	無効なコマンド行オプションが指定された。
99	予期しない例外が発生しました。

ファイル

/usr/share/lib/pkg	デフォルトのプレゼンテーションコンテンツの場所。別の場所を選択するには、pkg/content_root を変更します。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目

dns-sd(1M)、mdnsd(1M)、pkg(1)、pkgrepo(1)、pkgsend(1)、syslogd(1M)、smf(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

注意事項

pkd.depotd サービスは SMF によって、サービス識別子 svc:/application/pkg/server のもとに管理されます。

mDNS ミラーサービスは SMF によって、サービス識別子 svc:/application/pkg/dynamic-mirror のもとに管理されます。

集積庫への読み取りアクセスを制御するには、pkg(1) がネィティブでサポートするクライアントベースの SSL 証明書アクセスなどの認証方法と組み合わせて、HTTP リバースプロキシを使用できます。

ファイルシステムベースの操作を使用した構成の変更やパッケージデータの変更では、変更が操作や出力に反映されるように、集積サーバープロセスの再起動が必要です。集積サーバープロセスを再起動するには、次のいずれかの方法を使用します。

- svcadm(1M) を使用して、application/pkg/server インスタンスを再起動します。
- kill(1) を使用して、集積サーバープロセスに SIGUSR1 シグナルを送ります。これは、プロセスをそのままにして、すべての構成、パッケージ、および検索データを再ロードするする「正常な再起動」を実行します。

  # kill -USR1 *pid*

名前　pkg.sysrepo – Image Packaging System システムリポジトリ構成

形式

```
pkg.sysrepo -p port [-c cache_dir] [-s cache_size]
    [-w http_proxy] [-W https_proxy]
```

機能説明

pkg.sysrepo は Image Packaging System (IPS) システムリポジトリの構成ファイルを生成するために使用します。pkg.sysrepo は svc:/application/pkg/system-repository SMF (Service Management Facility) サービスによって呼び出されます。構成の変更は、SMF サービスのプロパティーに対して行うようにしてください。

システムリポジトリは、集中管理プロキシ経由で参照イメージに構成されているパッケージリポジトリへのアクセスを提供する責任があります。その参照イメージへの発行元の構成の変更は、システムリポジトリを使用するように構成されているすべてのクライアントにただちに認識されます。

システムリポジトリは、非大域ゾーンから、大域ゾーンに構成されているリポジトリにアクセスできるようにするために、主に大域ゾーンで使用されます。SMF サービス svc:/application/pkg/zones-proxyd および svc:/application/pkg/zones-proxy-client は非大域ゾーンと大域ゾーン間のトランスポートを提供する責任があります。このトランスポートは pkg(5) によってのみ使用されます。

http、https、および v4 ファイルリポジトリのみがサポートされます。p5p ベースのファイルリポジトリまたはそれより古いファイルリポジトリの形式はサポートされません。リポジトリバージョンの詳細については、pkgrepo(1) を参照してください。

オプション　サポートしているオプションは、次のとおりです。

-c *cache_dir*　構成されている発行元からの応答をキャッシュするために、システムリポジトリによって使用されるディレクトリの絶対パス。

デフォルトで、ファイルキャッシュが使用されます。ただし、特殊な値 memory を使用して、メモリー内キャッシュを使用するように指示できます。特殊な値 None を使用して、システムリポジトリでキャッシュを実行しないように指示できます。この設定は、config/cache_dir SMF プロパティーを使用して構成するようにしてください。

-p *port*　システムリポジトリが要求を待機するために使用するポート。この設定は、config/port SMF プロパティーを使用して構成するようにしてください。

-s *cache_size*　システムリポジトリの最大キャッシュサイズを定義するメガバイト単位の整数値。この設定は、config/cache_max SMF プロパティーを使用して構成するようにしてください。

-w *http_proxy*　システムリポジトリが http ベースのパッケージリポジトリにアクセスするために使用できる Web プロキシを定義する形式 *scheme*:// *hostname*[:*port*] の文字列。この設定は、config/http_proxy SMF プロパティーを使用して構成できます。

-W *https_proxy*　システムリポジトリが https ベースのパッケージリポジトリにアクセスするために使用できる Web プロキシを定義する形式 *scheme*:// *hostname*[:*port*] の文字列。この設定は、config/https_proxy SMF プロパティーを使用して構成できます。

使用例

例 1　システムリポジトリの有効化

```
$ svcadm enable svc:/application/pkg/system-repository
```

終了ステータス

次の終了ステータスが返されます。

0　コマンドが成功しました。

1　コマンドは有効な構成の書き込みに失敗しました。

2　無効なコマンド行オプションが指定された。

99　予期しない例外が発生しました。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目

pkg(1)、pkg.depotd(1m)、pkg(5)

http://hub.opensolaris.org/bin/view/Project+pkg/

参 照

標準、環境、マクロ

名前

pkg – Image Packaging System

機能説明

Image Packaging System、pkg(5) は、ソフトウェアのライフサイクル管理 (インストール、アップグレード、および削除) のために提供されるフレームワークです。Image Packaging System は、一連のキーと値のペアおよび (場合によっては) データペイロードで定義されたアクションのコレクションである、パッケージの単位でソフトウェアを管理します。多くの場合、アクションはファイルシステム内に存在するファイルですが、ドライバ、サービス、ユーザーなどのその他のインストール可能オブジェクトも表します。

パッケージの **Fmri** とバージョン

各パッケージは、スキーム pkg: を使用して、障害管理リソース識別子 (FMRI) によって表されます。パッケージの完全な FMRI は、次の形式のスキーム、パブリッシャー、パッケージ名、およびバージョン文字列で構成されます。

```
pkg://solaris/system/library/c++-runtime@0.5.11,5.11-0.174.0.0.0.0.0:20110921T190358Z
```

solaris は発行元です。system/library/c++-runtime はパッケージ名です。名前空間は階層的であり、その深さは任意ですが、適用される包含関係は存在しません。名前は基本的に任意です。パブリッシャー情報は省略可能ですが、存在する場合は、前に pkg:// を付ける必要があります。パブリッシャーを含む FMRI は多くの場合、「完全指定」と呼ばれます。パブリッシャー情報が存在しない場合は、一般に、パッケージ名の前に pkg:/ が付けられます。

クライアントをパッケージ化すると、通常、FMRI にパブリッシャー情報が含まれていない場合に FMRI のスキームを省略できます。たとえば、pkg:/system/library/c++-runtime は system/library/c++-runtime と書くことができます。スキームが省略される場合、クライアントは照合の目的で、パッケージ名の最後のコンポーネントを除くすべてのコンポーネントを省略することもできます。たとえば、system/library/c++-runtime は library/c++-runtime または c++-runtime と書くことができ、これらは c++-runtime というパッケージまたは /c++-runtime で終わるパッケージ名に一致します。

パブリッシャーの名前によって、人、人のグループ、または組織が 1 つ以上のパッケージのソースとして識別されます。パブリッシャーの名前の競合を避け、パブリッシャーを識別しやすくするために、パッケージを公開するエンティティーを表すドメイン名をパブリッシャーの名前として使用することがベストプラクティスです。

パッケージ名のあとに、アットマーク記号 (@) で区切られたバージョンが続きます。バージョンは、句読文字で区切られた 4 つの数字の並びで構成されます。最初の 3 つの並びに含まれる要素はドットで区切られ、これらの並びの長さは任意です。バージョンコンポーネント内の先頭のゼロ (たとえば、01.1 または 1.01) は許可されません。末尾のゼロ (たとえば、1.10) は許可されます。

バージョンの最初の部分はコンポーネントバージョンです。オペレーティングシステムに緊密に結合されたコンポーネントの場合、これは通常、そのバージョンのオ

ペレーティングシステムでの uname -r の値です。独自の開発ライフサイクルを持つコンポーネントの場合、この並びはドットで区切られたリリース番号 (2.4.10 など) です。

バージョンの 2 番目の部分 (存在する場合はコンマ (,) に続いている必要があります) はビルドバージョンです。ビルドバージョンは、パッケージの内容が構築されたオペレーティングシステムのバージョンを指定し、その内容が正常に実行されることを予測できるオペレーティングシステムのバージョンに関する最小の境界が提供されます。

バージョンの 3 番目の部分 (存在する場合はハイフン (-) に続いている必要があります) はブランチバージョンです。ブランチバージョンは、ベンダー固有の情報を提供するバージョン管理コンポーネントです。ブランチバージョンは、コンポーネントバージョンとは独立に、パッケージ化のメタデータが変更されたときに 1 増やすことができます。ブランチバージョンには、ビルド番号やその他の情報が含まれていることがあります。

バージョンの 4 番目の部分 (存在する場合はコロン (:) に続いている必要があります) はタイムスタンプです。タイムスタンプは、そのパッケージがいつ公開されたかを表します。

バージョン間の比較を実行する場合、その左側にあるコンポーネントが同じでないかぎり、バージョン全体のコンポーネントは考慮されません。したがって、「4.3」が「4.2」より大きいため「4.3-1」は「4.2-7」より大きく、「3」が「1」より大きいため「4.3-3」は「4.3-1」より大きくなります。

システムの多くの部分では、表示される情報量や必要な情報量を減らすために、必要に応じて FMRI を表示するときには短縮し、入力も短い形式を受け入れます。通常、スキーム、パブリッシャー、ビルドバージョン、およびタイムスタンプは省略できます。すべてのバージョン管理情報を省略できることもあります。

アクション

アクションはシステム上のインストール可能オブジェクトを表します。アクションはパッケージのマニフェストに記述されます。すべてのアクションは、最初に名前とキー属性を含みます。また、これらはバージョン履歴をたどる過程で一意のオブジェクトを参照します。アクションには、このほかの属性も含まれます。一部の属性は、パッケージシステムによって直接解釈されます。その他の属性は、システム管理者またはエンドユーザーにのみ役立つ可能性があります。

アクションには、次の単純なテキスト表現があります。

action_name attribute1=value1 attribute2=value2 ...

属性の名前に、空白、引用符、または等号 (=) を含めることはできません。最初の等号のあとの文字はすべて、値に属します。値にはすべての文字を含めることができますが、スペースは単一引用符または二重引用符で囲む必要があります。二重引用符で囲まれている文字列の内部で単一引用符をエスケープする必要はなく、単一引用符で囲まれている文字列の内部で二重引用符をエスケープする必要はありませ

ん。引用符の前にバックスラッシュ (\) 文字を付けて、引用文字列が終了しないようにすることができます。バックスラッシュは、バックスラッシュでエスケープできます。

複数の値を使用して、属性に複数回名前を付けることができます。これらは、順序なしリストとして扱われます。

多くの属性を持つアクションによって、マニフェストファイル内に長い行が作成されることがあります。このような行は、不完全な各行をバックスラッシュで終了することにより折り返すことができます。この継続文字は、属性と値のペアの間に置く必要があることに注意してください。属性も、その値も、さらにその組み合わせも分割することはできません。

下に示されている属性セットは、これがすべてではありません。実際、アクションに付加できる属性は任意であるため、標準の属性セットは、将来の開発を実装するために容易に拡張できます。

特定のアクション属性によって、パッケージ化コンテキストの外部で追加の操作が実行されます。これらの属性は、下の「アクチュエータ」のセクションで説明されています。

ファイルアクション

`file` アクションは、通常ファイルを表します。`file` アクションはペイロードを参照し、次の 4 つの標準属性があります。

`path` ファイルがインストールされているファイルシステムのパス。これは `file` アクションのキー属性です。

`mode` ファイルのアクセス権 (数値形式)。これらは ACL ではなく、単純なアクセス権のみです。

`owner` ファイルを所有するユーザーの名前。

`group` ファイルを所有するグループの名前。

ペイロードは、名前が付けられないという点で位置属性です。これは、アクション名のあとの最初の単語です。公開されたマニフェストでは、これはファイルの内容の `SHA-1` ハッシュです。これから公開する必要のあるマニフェスト内に存在する場合は、ペイロードを見つけることのできるパスを表します。`pkgsend(1)` を参照してください。値に等号が含まれている場合は、位置属性の代わりにハッシュ属性を使用できます。この両方を同じアクションで使用できます。ただし、ハッシュは同じである必要があります。

その他の属性には次のものがあります。

`preserve` これは、ファイルの内容がそのファイルのインストールまたは最後のアップグレード以降に変更されたと判定された場合、アップグレード時にその内容を上書きすべきではないことを指定します。初期インストール時に既存のファイルが見つかった場合、そのファイルは回収されます (`/var/pkg/lost+found` 内に格納されます)。

preserve の値が renameold である場合は、既存のファイルの名前が拡張子 .old を使用して変更され、新しいファイルがその場所に置かれます。

preserve の値が renamenew である場合は、既存のファイルがそのままになり、新しいファイルが拡張子 .new を使用してインストールされます。

preserve の値が legacy である場合は、初期のパッケージインストールでこのファイルはインストールされません。アップグレード時、既存のファイルはすべて拡張子 .legacy を使用して名前が変更され、新しいファイルがその場所に置かれます。

preserve の値が true (または、strawberry などの上に示されていない値) である場合は、既存のファイルがそのままになり、新しいファイルはインストールされません。

overlay　これは、このアクションによってほかのパッケージが同じ場所にファイルを提供できるか、またはこのアクションによって別のファイルをオーバーレイすることを目的にしたファイルが提供されるかを指定します。この機能は、どの自己アセンブリにも参加しておらず (たとえば、/etc/motd)、かつ安全に上書きできる構成ファイルで使用されることを目的にしています。

overlay が指定されていない場合は、複数のパッケージが同じ場所にファイルを提供することはできません。

overlay の値が allow である場合は、ほかの 1 つパッケージが同じ場所にファイルを提供することを許可されます。preserve 属性も同時に設定されていないかぎり、この値は意味を持ちません。

overlay の値が true である場合は、このアクションによって提供されたファイルにより、allow を指定したほかのアクションがすべて上書きされます。インストールされたファイルへの変更は、オーバーレイしているファイルの preserve 属性の値に基づいて保持されます。削除時、ファイルの内容は、preserve 属性が指定されたかどうかには関係なく、オーバーレイされているアクションがまだインストールされている場合は保持されます。別のファイルをオーバーレイできるのは 1 つのアクションだけであり、mode、owner、および group 属性が一致している必要があります。

ファイルは「味見」してみることもできます。その「風味」に応じて、追加で属性を付けることができます。ELF ファイルの場合は、次の属性が認識されます。

elfarch　ELF ファイルのアーキテクチャー。これは、そのファイルが構築されたアーキテクチャー上での uname -p の出力です。

elfbits　これは 32 または 64 です。

elfhash　これは、そのファイル内の「興味深い」ELF セクションのハッシュです。これらは、バイナリがロードされるときにメモリーにマップされるセクションです。これらは、2 つのバイナリの実行可能ファイルの動作が異なるかどうかを判定するときに考慮する必要のある唯一のセクションです。

original_name　この属性は、パッケージからパッケージに、または場所から場所に、あるいはその両方で移動している編集可能なファイルを処理するために使用されます。この形式は、元のパッケージの名前のあとに、コロンとこのファイルの元のパスが続きます。削除されているファイルはすべて、そのパッケージとパス、または original_name 属性の値 (指定されている場合) のどちらかを使用して記録されます。original_name 属性が設定された、インストールされている編集可能なファイルはすべて、それが同じパッケージ化の操作の一部として削除されている場合は、その名前のファイルを使用します。

revert-tag　この属性は、セットとして元に戻すべき編集可能なファイルをタグ付けするために使用されます。複数の revert-tag 値を指定できます。これらのいずれかのタグを指定して pkg revert が呼び出されると、ファイルはマニフェストで定義された状態に戻ります。pkg(1) を参照してください。

ディレクトリアクション

dir アクションは、ファイルシステムオブジェクトを表すという点で file アクションに似ています。dir アクションは、通常ファイルの代わりにディレクトリを表します。dir アクションには、file アクションと同じ 4 つの標準属性があり、path がキー属性です。

ディレクトリは、IPS でカウントされる参照です。あるディレクトリを明示的または暗黙的に参照している最後のパッケージが参照を行わなくなると、そのディレクトリは削除されます。そのディレクトリにパッケージ解除されたファイルシステムオブジェクトが含まれている場合、それらの項目は $IMAGE_META/lost+found に移動されます。$IMAGE_META についての詳細は、「ファイル」のセクションを参照してください。

パッケージ解除された内容を新しいディレクトリに移動するために、次の属性が役立つことがあります。

salvage-from　これは、回収された項目のディレクトリを指定します。このような属性を持つディレクトリは、作成時、回収されたディレクトリの内容を継承します (その内容が存在する場合)。

リンクアクション

link アクションはシンボリックリンクを表します。link アクションには、次の標準属性があります。

path
シンボリックリンクがインストールされるファイルシステムのパス。これは link アクションのキー属性です。

target
シンボリックリンクのターゲット。リンクの解決先のファイルシステムオブジェクト。

mediator
特定のメディエーショングループ (たとえば、python) に参加しているすべてのパス名によって共有されているメディエーション名前空間内のエントリを指定します。mediator-version または mediator-implementation、あるいはその両方に基づいてリンクメディエーションを実行できます。特定のパス名に対してメディエートされたリンクはすべて、同じメディエータを指定する必要があります。ただし、すべてのメディエータバージョンおよび実装が、特定のパスでリンクを提供する必要はありません。メディエーションがリンクを提供していない場合は、そのメディエーションが選択されたときにリンクが削除されます。特定のバージョンまたは実装、あるいはその両方と組み合わせた mediator は、パッケージシステムで使用するために選択できるメディエーションを表します。

mediator-version
mediator 属性で記述されたインタフェースの (負にならない整数のドットで区切られた並びとして表された) バージョンを指定します。この属性は、mediator が指定され、mediator-implementation は指定されていない場合に必要です。ローカルシステム管理者は、使用するバージョンを明示的に設定できます。指定された値は一般に、リンクを提供しているパッケージのバージョンに一致するようにしてください (たとえば、runtime/python-26 は mediator-version=2.6 を使用します)。ただし、これは必須ではありません。

mediator-implementation
mediator-version に加えて、またはこの代わりに使用するメディエータの実装を指定します。実装の文字列は順序付けられているとは見なされず、システム管理者によって明示的に指定されていない場合は、pkg (5) によって文字列が任意に選択されます。

この値は、英数字とスペースで構成された任意の長さの文字列にすることができます。実装自体をバージョン管理できるか、または実際にバージョン管理されている場合は、文字列の最後の @ のあとに (負にならない整数のドットで区切られた並びとして表された) バージョンを指定します。実装の複数のバージョンが存在する場合、デフォルトの動作では、最大のバージョンを持つ実装が選択されます。

特定のパスにある実装メディエーションリンクの 1 つのインスタンスだけがシステムにインストールされている場合は、その 1 つのインスタンスが自動的に選択されます。このパスにある将来のリンクがインストールされても、ベンダー、サ

イト、またはローカルのオーバーライドが適用されないかぎり、またはいずれかのリンクのバージョンがメディエートされた場合、このリンクは切り替えられません。

`mediator-priority`
メディエートされたリンクの競合を解決する場合、pkg(5) は通常、`mediator-version` の最大の値を持つリンクを選択するか、またはそれが不可能な場合は `mediator-implementation` に基づいて選択します。この属性は、正常な競合解決処理のためのオーバーライドを指定するために使用されます。

この属性が指定されていない場合は、デフォルトのメディエータ選択ロジックが適用されます。

この値が `vendor` である場合は、このリンクが、`mediator-priority` が指定されていないリンクより優先されます。

この値が `site` である場合は、このリンクが、`vendor` の値を持つリンクや、`mediator-priority` が指定されていないリンクより優先されます。

ローカルシステム管理者は、上で説明した選択ロジックをオーバーライドできます。

ハードリンクアクション

`hardlink` アクションは、ハードリンクを表します。このアクションは `link` アクションと同じ属性を持ち、そのキー属性も同じく `path` です。

ドライバアクション

`driver` アクションはデバイスドライバを表します。`driver` アクションはペイロードを参照しません。ドライバファイル自体を `file` アクションとしてインストールする必要があります。次の属性が認識されます (詳細は add_drv(1M) を参照)。

`name` ドライバの名前。多くの場合はドライババイナリのファイル名ですが、必ずしもそうとは限りません。これは `driver` アクションのキー属性です。

`alias` これはドライバの別名を表します。特定のドライバが複数の `alias` 属性を持つことができます。特殊な引用符の規則は必要ありません。

`class` これはドライバクラスを表します。特定のドライバが複数の `class` 属性を持つことができます。

`perms` これは、ドライバのデバイスノードのファイルシステムアクセス権を表します。

`clone_perms` これは、このドライバに対する複製ドライバのマイナーノードのファイルシステムアクセス権を表します。

`policy` これは、デバイスのための追加のセキュリティーポリシーを指定します。特定のドライバが複数の `policy` 属性を持つことができますが、マイナーデバイスの指定が複数の属性に存在することはできません。

privs　これは、ドライバで使用される特権を指定します。特定のドライバが複数の privs 属性を持つことができます。

devlink　これは /etc/devlink.tab 内のエントリを指定します。この値はファイルに書き込まれる行そのものであり、タブは \t で示されます。詳細は、devlinks(1M) を参照してください。特定のドライバが複数の devlink 属性を持つことができます。

依存アクション

depend アクションは、パッケージ間の依存関係を表します。あるパッケージが別のパッケージに依存することがあります。たとえば、最初のパッケージの機能を有効にする (場合によってはインストールする) ために、2 番目のパッケージの機能が必要な場合があります。依存関係は省略可能です。インストール時に依存関係が満たされない場合、パッケージシステムはほかの制約に応じて、依存パッケージをインストールするか、または十分に新しいバージョンに更新しようとします。

次の属性が認識されます。

fmri　依存パッケージを表す FMRI。これは dependency アクションのキー属性です。fmri 値にパブリッシャーを含めることはできません。パッケージ名は完全であると見なされます。タイプ require-any の依存関係は、複数の fmri 属性を持つことができます。fmri 値でバージョンは省略可能ですが、依存関係のタイプによっては、バージョンのない fmri は無意味である場合があります。

type　依存関係のタイプ。

この値が require である場合は、依存関係が必要であり、fmri 属性で指定されたバージョン以上のバージョンを持っている必要があります。バージョンが指定されていない場合は、任意のバージョンが依存関係を満たします。パッケージの必要な依存関係のいずれかを満たすことができない場合、そのパッケージはインストールできません。

この値が optional である場合は、依存関係 (存在する場合) が、指定されたバージョンレベル以上に存在する必要があります。

この値が exclude である場合は、指定されたバージョンレベル以上に依存関係が存在すると、包含するパッケージをインストールできません。バージョンが指定されていない場合、依存パッケージは、依存関係を指定しているパッケージと同時にインストールできません。

この値が incorporate である場合、依存関係は省略可能ですが、依存パッケージのバージョンが制約されます。下の「制約と凍結」を参照してください。

この値が require-any である場合は、複数の fmri 属性で指定された複数の依存パッケージのうちのいずれか 1 つが、依存関係のタイプ require と同じ規則に従って依存関係を満たすことができます。

この値が conditional である場合は、predicate 属性で定義されたパッケージがシステム上に存在する場合にのみ依存関係が必要です。

この値が origin である場合は、依存関係 (存在する場合) が、インストールの前に変更されるイメージ上の指定された値以上に存在する必要があります。root-image 属性の値が true である場合、このパッケージをインストールするには、/ をルートとするイメージ上に依存関係が存在する必要があります。

この値が group である場合は、パッケージがイメージ回避リスト上にないかぎり、依存関係が必要です。廃止されたパッケージは、暗黙のうちにグループの依存関係を満たすことに注意してください。pkg(1) の avoid サブコマンドを参照してください。

この値が parent である場合は、このイメージが子イメージでなければ、依存関係は無視されます。このイメージが子イメージである場合は、親イメージ内に依存関係が存在する必要があります。parent 依存関係でのパッケージバージョンの照合は、incorporate 依存関係で使用されるものと同じです。

predicate　conditional 依存関係の述語を表す FMRI。

root-image　先に説明した origin 依存関係に対してのみ有効です。

ライセンスアクション

license アクションは、パッケージの内容に関連したライセンスやその他の情報ファイルを表します。パッケージは license アクションの使用を通して、ライセンス、免責条項、またはその他のガイダンスをパッケージインストーラに提供できます。

license アクションのペイロードは、パッケージに関連したイメージメタデータディレクトリに提供され、人間が読める形式のテキストデータのみが含まれているべきです。HTML やその他の形式のマークアップが含まれていてはいけません。license アクションは、属性を通して、関連するペイロードが表示または同意、あるいはその両方を必要としていることをクライアントに示すことができます。表示または同意、あるいはその両方の方法は、クライアントに任されています。

次の属性が認識されます。

license　これは license アクションのキー属性です。この属性は、ユーザーがライセンスのテキスト自体を読まなくてもその内

容を判断できるように支援するための、ライセンスの意味のある説明を提供します。この値のいくつかの例を次に示します。

- ABC Co. Copyright Notice
- ABC Co. Custom License
- Common Development and Distribution License 1.0 (CDDL)
- GNU General Public License 2.0 (GPL)
- GNU General Public License 2.0 (GPL) Only
- MIT License
- Mozilla Public License 1.1 (MPL)
- Simplified BSD License

`license` 値は、パッケージ内で一意である必要があります。上のいくつかの例に示すように、説明にライセンスのバージョンを含めることをお勧めします。パッケージに複数のライセンスに基づくコードが含まれる場合、複数の `license` アクションを使用します。ライセンス属性値の長さは、64 文字以下にしてください。

`must-accept` `true` の場合は、関連するパッケージをインストールまたは更新するには、ユーザーがこのライセンスに同意する必要があります。この属性を省略すると、`false` と同等になります。同意の方法 (たとえば、対話型または構成ベース) は、クライアントに任されています。

`must-display` `true` の場合は、パッケージ化の操作中に、クライアントがこのアクションのペイロードを表示する必要があります。この値を省略すると、`false` と同等になります。この属性は、コピーライト表示に使用してはいけません。操作中に表示する必要のある実際のライセンスまたはその他の素材にのみ使用してください。表示の方法は、クライアントに任されています。

レガシーアクション

`legacy` アクションは、従来のパッケージシステムで使用されるパッケージデータを表します。このアクションに関連付けられた属性は、従来のシステムのデータベースに追加されます。そのため、これらのデータベースに問い合わせを行なっているツールは、従来のパッケージが実際にインストールされているかのように動作できます。特にこれにより、`pkg` 属性で指定されたパッケージがシステムにインストールされていることを従来のシステムに十分に確信させることができるため、このパッケージを、依存関係を満たすために使用できるようになります。

`pkginfo`(4) のパラメータに従って指定された次の属性が認識されます。

`category` `CATEGORY` パラメータの値。デフォルト値は `system` です。

`desc` `DESC` パラメータの値。

`hotline` `HOTLINE` パラメータの値。

`name` `NAME` パラメータの値。デフォルト値は `none provided` です。

`pkg` インストールされるパッケージの略語。デフォルト値は、パッケージの FMRI の名前です。これは `legacy` アクションのキー属性です。

`vendor` `VENDOR` パラメータの値。

`version` VERSION パラメータの値。デフォルト値は、パッケージの FMRI のバージョンです。

設定アクション

`set` アクションは、パッケージの説明などの、パッケージレベルの属性 (またはメタデータ) を表します。

次の属性が認識されます。

`name` 属性の名前。

`value` 属性に与えられた値。

`set` アクションは、パッケージ作成者が選択した任意のメタデータを提供できます。ただし、パッケージシステムにとって特別な意味を持つ、適切に定義された属性名がいくつか存在します。

`pkg.fmri` 「説明」セクションの「パッケージの FMRI とバージョン」を参照してください。

`info.classification` `pkg`(5) クライアントがパッケージを分類するために使用できる 1 つ以上のトークン。この値には、スキーム (「org.opensolaris.category.2008」や「org.acm.class.1998」など) と実際の分類 (「アプリケーション/ゲーム」など) がコロン (:) で区切られて含まれています。

`pkg.description` パッケージの内容と機能の詳細な説明。通常は、1 つの段落程度の長さです。

`pkg.obsolete` `true` の場合、パッケージは廃止としてマークされています。廃止されたパッケージには、ほかの設定アクション以外のアクションは存在せず、また名前変更としてマークすることはできません。

`pkg.renamed` `true` の場合は、パッケージの名前が変更されました。このパッケージには、このパッケージの名前が変更された先のパッケージバージョンを指す 1 つ以上の `depend` アクションも存在する必要があります。パッケージを名前変更と廃止の両方としてマークすることはできませんが、それ以外は任意の数の設定アクションが存在できます。

`pkg.summary` パッケージの短い、1 行の説明。

グループアクション

`group` アクションは、`group`(4) で定義されるのと同様の UNIX グループを定義します。グループパスワードのサポートはありません。このアクションで定義されたグ

ループには最初、ユーザーリストがありません。ユーザーは、user アクションを使用して追加できます。次の属性が認識されます。

groupname　グループの名前の値。

gid　グループの一意の数値 ID。デフォルト値は、100 未満の最初に空いているグループです。

ユーザーアクション

user アクションは、/etc/passwd、/etc/shadow、/etc/group、および /etc/ftpd/ftpusers ファイルで定義されるのと同様の UNIX ユーザーを定義します。このアクションでユーザーを定義すると、しかるべきファイルにエントリが追加されます。

次の属性が認識されます。

username　ユーザーの一意の名前。

password　ユーザーの暗号化パスワード。デフォルト値は *LK* です。shadow(4) を参照してください。

uid　ユーザーの一意の UID。デフォルト値は、100 未満の最初に空いている値です。

group　ユーザーのプライマリグループの名前。/etc/group に存在する必要があります。

gcos-field　/etc/passwd 内の gcos フィールドの値。デフォルト値は username です。

home-dir　ユーザーのホームディレクトリ。デフォルト値は / です。

login-shell　ユーザーのデフォルトのシェル。デフォルト値は空です。

group-list　ユーザーが属しているセカンダリグループ。group(4) を参照してください。

ftpuser　true または false に設定できます。true のデフォルト値は、ユーザーが FTP 経由のログインを許可されていることを示します。ftpusers(4) を参照してください。

lastchg　1970 年 1 月 1 日と、パスワードが最後に変更された日付の間の日数。デフォルト値は空です。shadow(4) を参照してください。

min　パスワード変更の間の必要な最小日数。パスワードの有効期限を有効にするには、このフィールドを 0 以上に設定する必要があります。デフォルト値は空です。shadow(4) を参照してください。

max　パスワードが有効な最大日数。デフォルト値は空です。shadow(4) を参照してください。

`warn` パスワードの期限が切れる前にユーザーに警告が表示される日数。shadow(4) を参照してください。

`inactive` そのユーザーに許可されている非活動の日数。これはマシンごとにカウントされます。最終ログインに関する情報は、そのマシンの lastlog ファイルから取得されます。shadow(4) を参照してください。

`expire` UNIX エポック (1970 年 1 月 1 日) 以降の日数として表される絶対的な日付。この日数に達すると、ログインを使用できなくなります。たとえば、13514 の期限切れの値は、ログインの有効期限が 2007 年 1 月 1 日であることを指定します。shadow(4) を参照してください。

`flag` 空に設定されています。shadow(4) を参照してください。

アクチュエータ コンテキストによっては、特定のアクションの準備として、またはその導入のあとに、追加の操作を実行することが適している場合があります。これらの追加の操作は一般に、ライブシステムイメージ上でのみ必要であり、またオペレーティングシステムに固有のものです。パッケージのインストールまたは削除に関与する複数のアクションのアクチュエータが同一である場合、アクチュエータの存在に対応する操作は、そのインストールまたは削除に対して 1 回実行されます。

アクチュエータを誤って指定すると、そのアクチュエータがインストールを安全に進めるための手段を決定できない場合、パッケージのインストールが失敗することがあります。

次のアクチュエータが定義されています。

`reboot-needed`
: true または false に設定できます。パッケージのインストール中に、このアクチュエータが true に設定されたアクションがインストールまたは更新された場合は、パッケージ化のトランザクションをリブートが必要であるとして通知できます。特定のクライアント実装では、イメージがライブシステムイメージである場合、イメージの複製を使用してパッケージ操作全体を実行するなどの、追加の手順が実行される可能性があります。

`disable_fmri`、`refresh_fmri`、`restart_fmri`、`suspend_fmri`
: これらの各アクチュエータは、パッケージのインストールまたは削除中に操作するサービスインスタンスの FMRI の値を取ります。disable_fmri を指定すると、svcadm(1M) の disable サブコマンドに従って、アクションの削除の前に特定の FMRI が無効になります。refresh_fmri および restart_fmri を指定すると、svcadm(1M) の対応するサブコマンドに従って、アクションのインストール、更新、または削除のあとに特定の FMRI が更新または再起動されます。最後に、suspend_fmri を指定すると、アクションのインストールフェーズの前に特定の FMRI が一時的に無効になり、そのフェーズが完了したあとに有効になります。

この値には、複数のサービスインスタンスに一致するパターンを含めることができます。ただし、それをインスタンスを示さずに暗黙的に行うのではなく、svcs(1) によって受け入れられた glob を使用して明示的に行う必要があります。

制約と凍結

パッケージが新しいバージョンに移行される場合、またはシステムに追加されたり、システムから削除されたりする場合、選択されるバージョンや、削除が許可されるかどうかは、そのパッケージに対するさまざまな制約によって決定されます。これらの制約は、依存関係の形式でほかのパッケージが定義するか、または凍結の形式で管理者が定義できます。

制約のもっとも一般的な形式は、上の「依存アクション」で説明したように、require 依存関係によって提供されます。このような制約によって、パッケージがダウングレードまたは削除されることが回避されます。

オペレーティングシステムのほとんどの部分は、*incorporation* と呼ばれるパッケージによってカプセル化されています。これらのパッケージは主に、incorporate 依存関係によって表される制約を提供します。

上で説明したように、組み込まれたパッケージがシステム上に存在する必要はありませんが、存在する場合は、包括的な最小バージョンと排他的な最大バージョンの両方を指定します。たとえば、依存する FMRI のバージョンが 1.4.3 の場合、1.4.3 未満のバージョンは依存関係を満たさず、1.4.4 以上のバージョンでも依存関係を満たしません。ただし、1.4.3.7 など、単にドット形式の数列を拡張したバージョンはインストールできます。

incorporation は、システムの各部の同期的なアップグレードを強制的に行うために使用されます。C ライブラリやカーネルなどの一部のコンポーネントでは、これは基本的な要件です。ほかには依存関係が存在しない単純なユーザーランドコンポーネントなどのその他のコンポーネントでは、incorporation の特定のバージョンから参照できるテストされた、既知の一連のパッケージバージョンを提供するためだけに、同期アップグレードが使用されます。

incorporation は単なるパッケージであるため、削除することができ、それが提供しているすべての制約がそれによって緩和されます。ただし、その緩和は安全ではないため、Oracle Solaris によって提供される多くの incorporation が、それらの incorporation によって組み込まれているパッケージには必要です。

パッケージを、インストール済みの incorporation によって許可されていないバージョンにアップグレードしようとしても、要求を満たすためにその incorporation の新しいバージョンを見つけようとする試みは行われず、そのアップグレードは失敗します。制約自体を移動する必要があるが、それを指定している incorporation を削除できない場合は、その incorporation を、制約の目的のバージョンを指定するバージョンにアップグレードする必要があります。incorporation をアップグレードすると、その新しいバージョンによって提供される制約を満たさない組み込まれたパッケージもすべてアップグレードされます。

システム管理者は、pkg freeze コマンドを使用してパッケージを制約できます。バージョンが指定されていない場合、指定されたパッケージは、システムにインストールされているバージョンに制約されます。バージョン管理されたパッケージが指定された場合、この管理上の制約 (つまり、凍結) は、fmri 属性が指定されたパッケージバージョンの値を持った状態で incorporate 依存関係がインストールされているかのように機能します。

凍結がパッケージシステムによって自動的に解除されることはありません。制約を緩和するには、pkg unfreeze コマンドを使用します。

発行元とリポジトリ

上で詳細に説明したように、パブリッシャーとは単に、パッケージクライアントがパッケージのプロバイダを識別するために使用する名前です。パブリッシャーは、パッケージリポジトリまたはパッケージアーカイブ、あるいはその両方を使用してパッケージを配布できます。現在パッケージシステムでサポートされているリポジトリのタイプには、起点リポジトリとミラーリポジトリの 2 つがあります。

起点は、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) と内容 (ファイル) を含むパッケージリポジトリです。イメージ内の特定のパブリッシャーに対して複数の起点が構成されている場合、パッケージクライアント API は、パッケージデータの取得元として最適な起点を選択しようとします。これはもっとも一般的なタイプのリポジトリであり、パッケージリポジトリ上で pkgsend または pkgrecv が使用された場合は常に、暗黙的に作成されます。

ミラーは、パッケージの内容 (ファイル) のみを含むパッケージリポジトリです。イメージ内の特定のパブリッシャーに対して 1 つ以上のミラーが構成されている場合、クライアント API はパッケージ内容の取得のためのミラーを優先し、パッケージの内容の取得元として最適なミラーを選択しようとします。ミラーが到達不可能か、必要な内容が含まれていないか、またはより低速な場合、クライアント API は、構成されているいずれかの起点リポジトリから内容を取得します。ミラーは、pkg.depotd(1M) の動的ミラー機能を使用している、信頼できる一連のクライアント間の内容共有に使用されることを目的にしています。ミラーはまた、パッケージのメタデータへのアクセスを認証するために使用されることも目的にしています。ただし、パッケージの内容は認証なしで配布します。たとえば、あるクライアントが、アクセスするには SSL キーと証明書のペアが必要な https 起点と、パッケージの内容を提供する http ミラーを使用して構成される可能性があります。このようにして、認可クライアントだけがパッケージをインストールまたは更新できるようにしながら、パッケージ内容の取得のための認証のオーバーヘッドが回避されます。ミラーは、file という名前のサブディレクトリとその親を除く、リポジトリのすべてのサブディレクトリを削除することによって作成できます。また、pkg.depotd(1M) のミラーモードを使用すると、起点リポジトリもミラーとしてプロビジョニングできます。

ファセットとバリアント

ソフトウェアには、省略可能なコンポーネントや、相互に排他的なコンポーネントが含まれることがあります。省略可能なコンポーネントの例には、ロケールやドキュメントがあります。相互に排他的なコンポーネントの例には、SPARC バイナリと x86 バイナリや、デバッグバイナリと非デバッグバイナリなどがあります。

IPS では、省略可能なコンポーネントをファセット、相互に排他的なコンポーネントをバリアントと呼びます。ファセットとバリアントはパッケージアクションのタグとして指定します。各ファセットタグおよびバリアントタグには名前と値があります。1 つのアクションに複数のファセットタグおよびバリアントタグを付けることができます。複数のファセットおよびバリアントタグのあるコンポーネントの例には、開発者によって使われるアーキテクチャー固有のヘッダーファイルや SPARC 大域ゾーン専用のコンポーネントなどがあります。

バリアントタグの例は `variant.arch=sparc` です。ファセットタグの例は `facet.devel=true` です。ファセットとバリアントは、`facet.` や `variant.` を先頭に付けずに参照されることがよくあります。

ファセットとバリアントはイメージの特殊なプロパティーであり、個々のパッケージには設定できません。イメージに設定されたファセットおよびバリアントの現在の値を表示するには、pkg(1) のマニュアルページに示すように、`pkg facet` コマンドと `pkg variant` コマンドを使用します。イメージに設定されたファセットおよびバリアントの値を変更するには、`pkg change-facet` コマンドと `pkg change-variant` コマンドを使用します。

ファセットはブール型です。それらには `true` (有効) または `false` (無効) のみ設定できます。デフォルトで、イメージ内のすべてのファセットは、`true` に設定されていると見なされます。アクションのファセットタグの値には `true` のみを指定すべきであり、それ以外の値では動作が不確定になります。イメージに設定されるファセットは、`doc.man` などの完全なファセットか、`locale.*` などのパターンになります。これは、ファセット名前空間の一部を無効にし、その中の個々のファセットのみを有効にする場合に役立ちます。たとえば、次の例に示すように、すべてのロケールを無効にしてから、1 つか 2 つの特定のロケールのみを有効にすることができます。

```
# pkg change-facet locale.*=false
[output about packages being updated]
# pkg change-facet locale.en_US=true
[output about packages being updated]
```

ほとんどのバリアントは任意の数の値を設定できます。たとえば、`arch` バリアントには、`i386`、`sparc`、`ppc`、`arm`、またはディストリビューションがサポートしているどのようなアーキテクチャーでも設定できます。(Oracle Solaris では `i386` と `sparc` のみが使用されます。)例外は `debug` バリアントです。`debug` バリアントは、`true` または `false` のみ設定でき、ほかの値では動作が不定になります。ファイルアクションに非デバッグバージョンとデバッグバージョンの両方がある場合、次の例に示すように、両方のバージョンに該当する `debug` バリアントが明示的に設定されている必要があります。

```
file group=sys mode=0644 overlay=allow owner=root \
  path=etc/motd pkg.csize=115 pkg.size=103 preserve=true \
  variant.debug.osnet=true

file group=sys mode=0644 overlay=allow owner=root \
  path=etc/motd pkg.csize=68 pkg.size=48 preserve=true \
  variant.debug.osnet=false
```

バリアントを使用するパッケージをインストールするために、バリアント値をイメージに設定する必要があります。arch および zone バリアントは、イメージを作成し、その初期コンテンツをインストールするプログラムによって設定されます。イメージ内の debug.* バリアントはデフォルトで false です。

イメージに設定されたファセットとバリアントは、特定のアクションがインストールされるかどうかに影響します。

- ファセットまたはバリアントタグのないアクションは常にインストールされます。
- ファセットタグのあるアクションは、イメージ上のタグに一致するすべてのファセットまたはファセットパターンが false に設定されていないかぎり、インストールされます。いずれかのファセットが true に設定されているか、明示的に設定されていない (true はデフォルト) 場合、アクションがインストールされます。
- バリアントタグのあるアクションは、すべてのバリアントタグの値がイメージに設定されている値と同じ場合にのみインストールされます。
- ファセットタグとバリアントタグの両方があるアクションは、ファセットとバリアントの両方でアクションのインストールが許可されている場合にインストールされます。

独自のファセットおよびバリアントタグを作成できます。Oracle Solaris では、次のタグが一般に使用されます。

バリアント名	取り得る値
variant.arch	sparc、i386
variant.opensolaris.zone	global、nonglobal
variant.debug.*	true、false

次のリストに、Oracle Solaris で使用される小さなファセットタグの例を示します。

```
facet.devel             facet.doc
facet.doc.html          facet.doc.info
facet.doc.man           facet.doc.pdf
facet.locale.de         facet.locale.en_GB
```

```
facet.locale.en_US      facet.locale.fr
facet.locale.ja_JP      facet.locale.zh_CN
```

イメージポリシー

イメージポリシーはブール値を持つイメージプロパティーによって定義されます。flush-content-cache-on-success および send-uuid プロパティーについてとそれらの値の表示および変更方法については、pkg(1) のマニュアルページのイメージプロパティーに関する項目を参照してください。

ファイル

pkg(5) イメージはより大きなファイルシステム内に任意に配置できるため、トークン $IMAGE_ROOT を使用して相対パスが区別されます。標準的なシステムインストールでは、$IMAGE_ROOT は / と同等です。

$IMAGE_ROOT/var/pkg	完全または部分的なイメージのメタデータディレクトリ。
$IMAGE_ROOT/.org.opensolaris,pkg	ユーザーイメージのメタデータディレクトリ。

特定のイメージのメタデータ内の特定のファイルおよびディレクトリに、修復や復旧中に役立つ情報を含めることができます。トークン $IMAGE_META は、メタデータが含まれる最上位ディレクトリを参照するために使用されます。通常、$IMAGE_META は前述の 2 つのパスのいずれかです。

$IMAGE_META/lost+found	パッケージ操作中に移動された、競合するディレクトリおよびファイルの場所。
$IMAGE_META/publisher	パブリッシャーごとに 1 つのディレクトリが含まれます。各ディレクトリにはパブリッシャー固有のメタデータが格納されます。

$IMAGE_META ディレクトリ階層内のその他のパスは非公開であり、変更される可能性があります。

属性

次の属性については、attributes(5) を参照してください。

属性タイプ	属性値
使用条件	package/pkg
インタフェースの安定性	不確実

関連項目

pkg(1)、pkgsend(1)、pkg.depotd(1m)、pkg.sysrepo(1m)、svcs(1)、svcadm(1M)

http://hub.opensolaris.org/bin/view/Project+pkg/