Pages de manuel d'Image Packaging System

Référence : E25751
Mars 2012, E25751-03

ORACLE

120424@25097

Table des matières

Préface

Ce document fournit les pages de manuel des outils d'installation du système Oracle Solaris 11.

Présentation

La liste suivante donne une brève description de chaque section de page de manuel et les informations référencées :

- La section 1 décrit les commandes disponibles avec le système d'exploitation.
- La section 1M décrit les commandes qui sont principalement utilisées pour la maintenance et l'administration du système.
- La section 5 contient des informations diverses, telles que des tables de jeux de caractères par exemple.

Vous trouverez ci-dessous un format générique de page de manuel. Les pages de manuel de chaque section de manuel suivent généralement cet ordre, mais n'incluent que les en-têtes nécessaires. Par exemple, s'il n'y a aucun bogue à signaler, la section BOGUES est omise. Reportez-vous à la commande `man` pour plus d'informations générales sur les pages de manuel.

NOM	Cette section donne les noms des commandes ou fonctions documentées, suivis d'une brève description de leur action.
SYNOPSIS	Cette section présente la syntaxe des commandes ou des fonctions. Lorsqu'une commande ou un fichier n'existe pas dans le chemin standard, son nom de chemin d'accès complet s'affiche. Les options et les arguments sont classés par ordre alphabétique ; les arguments à une seule lettre apparaissent en début de liste et sont suivis des options avec leurs arguments, à moins qu'un ordre d'argument différent ne soit nécessaire. Les caractères spéciaux suivants sont utilisés dans cette section :

[ ]	Parenthèses. L'option ou l'argument qui se trouve entre parenthèses est facultatif. Si les parenthèses sont omises, l'argument doit être spécifié.
. . .	Points de suspension. Plusieurs valeurs peuvent être fournies pour l'argument précédent, ou l'argument précédent peut être spécifié plusieurs fois. Par exemple : *filename* . . .
|	Séparateur. Seul un des arguments séparés par ce caractère peut être spécifié à la fois.
{ }	Accolades. Les options et/ou arguments entre accolades sont interdépendants, de telle manière que tout ce qui se trouve entre accolades doit être traité comme une unité.

PROTOCOLE	Cette section est présente uniquement dans la section 3R pour indiquer le fichier de description de protocole.
DESCRIPTION	Cette section définit la fonctionnalité et le comportement du service. Il s'agit donc d'une description concise de ce que la commande fait. Elle ne traite pas des OPTIONS et ne donne pas d'EXEMPLES. Les commandes interactives, les sous-commandes, les requêtes, les macros et les fonctions sont décrites dans UTILISATION.
OPTIONS	Cette section répertorie les options de commande avec un résumé succinct de ce que fait chaque option. Les options sont répertoriées littéralement et dans l'ordre dans lequel elles apparaissent dans la section SYNOPSIS. Les arguments possibles pour les options sont présentés sous l'option et, le cas échéant, les valeurs par défaut sont fournies.
OPERANDES	Cette section répertorie les opérandes de la commande et décrit comment ils affectent les actions de la commande.

SORTIE	Cette section décrit la sortie générée par la commande : sortie standard, erreur standard ou fichiers de sortie.
VALEURS DE RETOUR	Si la page de manuel documente des fonctions qui renvoient des valeurs, cette section présente ces valeurs et décrit les conditions dans lesquelles elles sont renvoyées. Si une fonction peut renvoyer uniquement des valeurs constantes, telle que 0 ou –1, ces valeurs sont répertoriées dans des paragraphes balisés. Dans le cas contraire, un seul paragraphe décrit les valeurs de retour de chaque fonction. Les fonctions déclarées void ne renvoient pas de valeurs, elles ne sont donc pas traitées dans les VALEURS DE RETOUR.
ERREURS	En cas d'échec, la plupart des fonctions placent un code d'erreur dans la variable globale `errno` indiquant pourquoi elles ont échoué. Cette section répertorie par ordre alphabétique tous les codes d'erreur qu'une fonction peut générer et décrit les conditions qui causent chaque erreur. Lorsque plusieurs conditions peuvent entraîner la même erreur, chaque condition est décrite dans un paragraphe distinct dans le code d'erreur.
UTILISATION	Cette section dresse la liste des règles spéciales, des fonctionnalités et des commandes qui nécessitent des explications détaillées. Les sous-sections répertoriées ici sont utilisées pour expliquer la fonctionnalité intégrée : Commandes Modificateurs Variables Expressions Grammaire d'entrée
EXEMPLES	Cette section fournit des exemples d'utilisation d'une commande ou d'une fonction. Chaque fois que cela est possible, un exemple complet comprenant une entrée de ligne de commande et la réponse machine est donné. Chaque fois qu'un exemple est donné, l'invite est affichée comme `exemple%`, ou si l'utilisateur doit être connecté en tant que

	superutilisateur, comme `exemple#`. Les exemples sont suivis d'explications, de règles de substitution de variable ou de valeurs de retour. La plupart des exemples illustre des concepts des sections SYNOPSIS, DESCRIPTION, OPTIONS et UTILISATION.
VARIABLES D'ENVIRONNEMENT	Cette section répertorie toutes les variables d'environnement sur lesquelles la commande ou la fonction a un impact, suivies d'une brève description de l'effet.
ETAT DE SORTIE	Cette section répertorie les valeurs que la commande renvoie au programme ou shell appelant, et les conditions qui causent le retour de ces valeurs. En général, la valeur zéro est renvoyée pour confirmer l'exécution, et des valeurs différentes de zéro pour diverses conditions d'erreur.
FICHIERS	Cette section répertorie tous les noms de fichiers désignés dans la page de manuel : fichiers d'intérêt et fichiers créés ou requis par les commandes. Chacun d'entre eux est suivi d'un résumé descriptif ou d'une explication.
ATTRIBUTS	Cette section répertorie les caractéristiques des commandes, des utilitaires et des pilotes de périphérique en définissant le type d'attribut et la valeur correspondante. Reportez-vous à la page de manuel `attributes`(5) pour plus d'informations.
VOIR AUSSI	Cette section affiche la liste des références à d'autres pages de manuel, documentations internes et publications externes.
DIAGNOSTICS	Cette section répertorie les messages de diagnostic avec une brève explication de la condition à l'origine de l'erreur.
AVERTISSEMENTS	Cette section répertorie les avertissements concernant les conditions spéciales qui pourraient avoir de graves conséquences sur vos conditions de travail. Il ne s'agit pas d'une liste de diagnostics.
NOTES	Cette section présente des informations supplémentaires qui n'appartiennent à aucune autre partie de la page. Il s'agit d'un extra pour l'utilisateur,

	couvrant les points d'intérêt particuliers. Les informations critiques ne sont jamais abordées ici.
BOGUES	Cette section décrit les bogues connus et, dans la mesure du possible, suggère des solutions.

RÉFÉRENCE

Commandes utilisateur

Nom packagemanager – Interface graphique pour Image Packaging System

Synopsis

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

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

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

Description `packagemanager` est l'interface graphique pour `pkg`(5), le logiciel Image Packaging System.

Le Gestionnaire de packages vous permet d'effectuer les opérations suivantes :

- Rechercher, installer et supprimer des packages.
- Ajouter, supprimer et modifier des éditeurs.
- Créer, supprimer et gérer des environnements d'initialisation.

Si l'opérande *file* est spécifié et son suffixe est `.p5i`, `packagemanager` se lance en mode d'installation Web, ce qui ajoute un ou plusieurs éditeurs et un certain nombre de packages pour chaque éditeur.

Options Les options suivantes sont prises en charge :

`-h` ou `--help`
Affiche un message d'utilisation.

`-i` ou `--info-install` *file*
Permet d'indiquer un fichier `.p5i` pour exécuter `packagemanager` en mode d'installation Web. Le fichier spécifié doit avoir l'extension `.p5i`.

`-R` ou `--image-dir` *dir*
Fonctionne sur l'image résidant au niveau de *dir*, plutôt que sur celle détectée automatiquement.

`-U` ou `--update-all`
Met à jour tous les packages installés pour lesquels une mise à jour est disponible.

Remarque – Si les packages `package/pkg`, `package/pkg/package-manager` ou `package/pkg/update-manager` doivent être mis à jour, `packagemanager` met d'abord à jour ces packages, puis redémarre pour effectuer d'autres mises à jour.

Opérandes *file* Un fichier d'installation Web. Ce fichier doit avoir le suffixe `.p5i`. Reportez-vous à l'aide en ligne du Gestionnaire de packages pour plus d'informations sur l'installation Web.

Exemples **EXEMPLE 1** Opération sur l'image actuelle

Appelez `packagemanager` sur l'image actuelle.

```
$ packagemanager
```

EXEMPLE 2 Opération sur une image spécifiée

Appelez `packagemanager` sur l'image stockée sous `/aux0/example_root`.

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

EXEMPLE 3 Appel en mode d'installation Web

Appelez `packagemanager` en mode d'installation Web.

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

État de sortie

Les valeurs de sortie renvoyées sont les suivantes :

`0` Tout a fonctionné.

`1` Une erreur s'est produite.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

Fichiers

Les images `pkg`(5) peuvent être stockées arbitrairement dans un plus grand système de fichiers. De ce fait, le jeton `$IMAGE_ROOT` sert à distinguer les chemins d'accès relatifs. Pour une installation système standard, `$IMAGE_ROOT` équivaut à /.

`$IMAGE_ROOT/var/pkg`	Répertoire des métadonnées pour une image complète ou partielle.
`$IMAGE_ROOT/.org.opensolaris,pkg`	Répertoire des métadonnées pour une image d'utilisateur. Dans les métadonnées d'une image particulière, certains fichiers et répertoires contiennent des informations utiles lors de la réparation et de la récupération. Le jeton `$IMAGE_META` est utilisé pour désigner le répertoire de niveau supérieur qui contient les métadonnées. `$IMAGE_META` est en général l'un des deux chemins d'accès donnés ci-dessus.
`$IMAGE_META/gui-cache`	Emplacement des métadonnées en cache gérées par `packagemanager` pour accélérer le démarrage du programme et la commutation entre éditeurs.

D'autres chemins dans l'arborescence des répertoires `$IMAGE_META` sont privés et sujets à modification.

Attributs

Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg/package-manager`
Stabilité de l'interface	Non validé

Voir aussi `pm-updatemanager(1)`, `pkg(1)`, `pkg(5)`

Aide en ligne du Gestionnaire de packages

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

Remarques La commande `packagemanager` doit être appelée avec des privilèges suffisants pour fonctionner sur les fichiers et répertoires d'une image.

Nom pkg – Client de récupération d'Image Packaging System

Synopsis

```
/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
```

Description La commande `pkg()` correspond au client de récupération d'Image Packaging System. Avec une configuration correcte, `pkg` peut être appelé pour créer des emplacements de packages à installer, des images nommées et installer des packages dans ces images. Les packages sont publiés par les éditeurs, qui peuvent rendre leurs packages disponibles au niveau d'un ou de plusieurs référentiels, ou d'archives de package. La commande `pkg` extrait des packages à partir du référentiel d'un éditeur ou d'archives de package et installe les packages dans une image.

Un nom d'éditeur permet d'identifier une personne, un groupe de personnes, ou une organisation en tant que source d'un ou plusieurs packages. Afin d'éviter toute collision de nom d'éditeur et de faciliter l'identification de l'éditeur, une bonne pratique consiste à utiliser en tant que nom d'éditeur un nom de domaine représentant l'entité qui publie les packages.

Un référentiel est un emplacement dans lequel les clients peuvent publier et récupérer le contenu d'un package (fichiers contenus dans le package, tels que des documents et des programmes) et des métadonnées (informations sur le package, telles que son nom et sa description). A titre d'exemple, un éditeur nommé `example.org` peut avoir son référentiel situé à l'URI `http://example.org/repository`.

`pkg` peut également désinstaller les packages, actualiser les métadonnées de l'éditeur (la liste des packages disponibles, par exemple), valider l'installation d'un package sur une image, et interroger l'image par rapport à différents jetons. Ces requêtes peuvent également être effectuées sur des référentiels `pkg`(5).

Les images peuvent être de trois types : images complètes, capables de fournir un système complet, images partielles, liées à une image complète (image parent), mais ne fournissant pas un système complet, et images d'utilisateur.

Options Les options suivantes sont prises en charge :

`-R` *dir*
S'exécute sur l'image résidant au niveau de *dir*. Si aucun répertoire n'a été spécifié ou déterminé en fonction de l'environnement, la valeur par défaut est /. Reportez-vous à la section Variables d'environnement pour plus d'informations.

`--help` ou `-?`
Affiche un message d'utilisation.

Sous-commandes Les sous-commandes suivantes sont prises en charge :

`refresh [--full] [`*publisher* `...]`
Met à jour la liste du client des packages disponibles et des métadonnées d'éditeur pour chaque éditeur spécifié. Si aucun éditeur n'est spécifié, l'opération est effectuée pour tous les éditeurs.

Avec `--full`, force la récupération de toutes les métadonnées d'éditeur, au lieu de mettre à jour de manière incrémentielle, et demande à tous les serveurs proxy utilisés au cours de l'opération d'ignorer les données en cache. Cette option existe à des fins de dépannage et ne doit pas être utilisée de façon régulière.

`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* `...`
Installe et met à jour les packages vers la version la plus récente qui correspond au *pkg_fmri_pattern* autorisé par les packages installés dans l'image. Pour demander explicitement la dernière version d'un package, utilisez `latest` pour la portion de version de *pkg_fmri_pattern*. Par exemple, spécifiez `vim@latest`.

Certains fichiers de configuration peuvent être renommés ou remplacés pendant le processus d'installation. Pour plus d'informations sur la façon dont le système de packages

détermine les fichiers à conserver et sur la façon dont ils sont conservés pendant les opérations sur les packages, reportez-vous à la section “Actions sur les fichiers” dans la page de manuel `pkg`(5).

Si un package est sur la liste à éviter, son installation supprime ce package de cette liste.

Avec `-g`, ajoute temporairement le référentiel ou l'archive de packages spécifiés à la liste des sources stockées dans l'image à partir de laquelle les données de package peuvent être récupérées. Si des packages de sources spécifiées sont également disponibles depuis des éditeurs configurés dans l'image, le client récupère le contenu de ces packages à partir des sources spécifiées uniquement. Lorsque vous décidez de la version d'un package à utiliser, les éditeurs configurés dans l'image, mais introuvables dans les sources données sont prioritaires. Après l'installation ou la mise à jour, les packages fournis par les éditeurs introuvables dans l'image sont ajoutés à la configuration de l'image sans origine. Cette option peut être spécifiée plusieurs fois.

Avec `-n`, effectue un essai de l'opération sans aucune modification de package.

Avec `-q`, masque des messages de progression pendant l'opération demandée.

Avec `-v`, exécute des messages détaillés de progression pendant l'opération demandée et affiche des informations sur la planification détaillée (telles que la modification facettes, les médiateurs et les variantes). Cette option peut être spécifiée à plusieurs reprises pour afficher davantage d'informations de planification.

Avec `--accept`, indique que vous acceptez les conditions de licence des packages qui sont mis à jour ou installés. Si vous ne spécifiez pas cette option alors que les licences de package exigent l'acceptation, l'opération d'installation échoue.

Avec `--licenses`, affiche l'ensemble des licences des packages qui seront installés ou mis à jour lors de cette opération.

Avec `--no-backup-be`, ne crée pas d'environnement d'initialisation de sauvegarde.

Si un environnement d'initialisation est créé, l'option `--no-be-activate` permet de ne pas le définir en tant qu'environnement d'initialisation actif lors de la prochaine initialisation. Reportez-vous à `beadm`(1M) pour plus d'informations.

Avec `--no-index`, ne met pas à jour les index de recherche au terme de l'opération.

Avec `--no-refresh`, n'essaye pas de contacter les référentiels pour les éditeurs de l'image pour récupérer la toute dernière liste des packages disponibles et d'autres métadonnées.

Avec `--backup-be-name`, nomme l'environnement d'initialisation de sauvegarde créé à l'aide de l'argument donné. L'utilisation de `--backup-be-name` implique `--require-backup-be`. Reportez-vous également à `beadm`(1M).

Avec `--be-name`, renomme l'environnement d'initialisation créé à l'aide de l'argument donné. L'utilisation de `--be-name` implique `--require-new-be`. Reportez-vous également à `beadm`(1M).

Avec `--require-backup-be`, crée toujours un environnement d'initialisation de sauvegarde si un nouvel environnement d'initialisation ne sera pas créé. Sans cette option, un environnement d'initialisation de sauvegarde est créé en fonction de la stratégie d'image. Reportez-vous à `be-policy` dans la section Propriétés de l'image ci-dessous pour savoir quand les environnements d'initialisation de sauvegarde sont créés automatiquement.

Avec `--require-new-be`, crée toujours un nouvel environnement d'initialisation. Sans cette option, un environnement d'initialisation est créé en fonction de la stratégie d'image. Reportez-vous à `be-policy` dans la section Propriétés de l'image ci-dessous pour savoir quand les environnements d'initialisation sont créés automatiquement. Cette option ne peut pas être combinée avec `--require-backup-be`.

Avec `--deny-new-be`, ne crée pas de nouvel environnement d'initialisation. Cette opération n'est pas effectuée si un nouvel environnement d'initialisation est requis.

Avec `--reject`, empêche l'installation des packages dont le nom correspond au modèle donné. Si des packages de même nom sont déjà installés, ils sont supprimés dans le cadre de cette opération. Les packages rejetés qui sont la cible des dépendances de groupe sont placés sur la liste avoid.

`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* `...`
Supprime les packages installés qui correspondent à *pkg_fmri_pattern*.

Si un package est l'objet d'une dépendance du groupe, sa désinstallation le place dans la liste avoid. Reportez-vous à la sous-commande `avoid` ci-dessous.

Pour toutes les autres options, reportez-vous à la commande `install` ci-dessus pour connaître leur utilisation et leurs effets.

`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* `...]`
Sans argument, ou si l'astérisque (*) est l'un des modèles fournis, met à jour tous les packages installés sur l'image actuelle vers la version la plus récente autorisée par les contraintes imposées sur le système par les packages installés et la configuration de l'éditeur. Pour demander explicitement la dernière version d'un package, utilisez `latest` pour la portion de version de *pkg_fmri_pattern*. Par exemple, spécifiez `vim@latest`.

Si *pkg_fmri_pattern* est fourni, `update` remplace les packages installés qui correspondent à *pkg_fmri_pattern* par la version la plus récente autorisée par les modèles et les contraintes imposées sur le système par les packages installés et la configuration de l'éditeur. Des versions plus anciennes ou plus récentes que celle qui est déjà installée peuvent être spécifiées pour effectuer une mise à jour vers un package spécifique inférieur ou supérieur. La mise à jour de packages spécifiques par le biais du renommage des packages ou de limites obsolètes n'est pas prise en charge.

Tous les fichiers de configuration conservés qui font partie d'un package à mettre à jour vers un package inférieur à l'aide de `update` et qui ont été modifiés depuis que la version d'origine a été installée sont renommés avec l'extension `.update`. Pour plus d'informations sur la méthode utilisée par le système de packages pour déterminer quels fichiers conserver pendant les mises à niveau de package, consultez la section relative aux actions sur les fichiers dans la page de manuel `pkg`(5).

Avec `-f`, ne vérifie pas la mise à jour du client lors de la mise à jour de tous les packages installés.

Pour toutes les autres options, reportez-vous à la commande `install` ci-dessus pour connaître leur utilisation et leurs effets.

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

Sans arguments, affiche la liste des packages inclus dans l'image actuelle, ainsi que des informations comme la version et l'état installé. Avec des arguments, affiche des informations sur les packages spécifiés. Par défaut, les variantes de package pour une architecture ou un type de zone différents sont exclues. La sortie habituelle s'affiche sur trois colonnes :

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

La première colonne contient le nom du package. Si l'éditeur à partir duquel le package est installé (ou disponible, s'il n'est pas déjà installé) n'apparaît pas le premier dans l'ordre de recherche des éditeurs, le nom de l'éditeur est indiqué entre parenthèses après le nom du package. La deuxième colonne contient la version et la branche du package. Pour plus d'informations sur la version, la branche et les variantes, reportez-vous à la page de manuel `pkg`(5).

La dernière colonne contient un ensemble d'indicateurs affichant l'état du package :

- Un `i` dans la colonne `I` indique que le package est installé.
- Un `f` dans la colonne `F` indique que le paquet est gelé.
- Un `o` dans la colonne `O` indique que ce package est obsolète. Un `r` dans la colonne `O` indique que ce package a été renommé (autre état obsolète).

Avec `-H`, omet les en-têtes de la liste.

Avec `-a`, répertorie les packages installés et la dernière version des packages disponibles pour l'installation. Les packages sont considérés comme disponibles pour l'installation s'ils sont autorisés par les incorporations installées et par les variantes de l'image. Si un ou plusieurs modèles sont spécifiés, la version la plus récente correspondant au modèle spécifié et autorisée par n'importe quelle incorporation installée et par les variantes de l'image est répertoriée. Sans `-a`, seuls les packages installés sont répertoriés.

Avec `-f` et `-a`, répertorie toutes les versions de tous les packages pour toutes les variantes, indépendamment des contraintes liées aux incorporations ou de l'état d'installation. Pour répertorier explicitement la dernière version d'un package avec ces options, utilisez `latest` pour la partie concernant la version de *pkg_fmri_pattern*. Par exemple, spécifiez `vim@latest`.

Avec `-g`, utilise le référentiel ou l'archive de packages spécifiés en tant que source de données de package pour l'opération. Cette option peut être spécifiée plusieurs fois. L'utilisation de `-g` implique `-a` si `-n` n'est pas spécifié.

Avec `-n`, affiche les dernières versions de tous les packages connus, quel que soit l'état d'installation.

Avec `-s`, affiche un format court sur une seule ligne qui indique le nom du package et un résumé. Cette option peut être utilisée avec `-a`, `-n`, `-u` ou `-v`.

Avec `-u`, seuls les packages de versions plus récentes disponibles sont répertoriés. Cette option ne peut pas être utilisée avec `-g`.

Avec `-v`, affiche les FMRI complets des packages, y compris l'éditeur et la version complète, dans la première colonne (la colonne de version disparaît). Cette option peut être utilisée avec `-a`, `-n` ou `-u` .

Avec `--no-refresh`, n'essaye pas de contacter les référentiels pour les éditeurs de l'image pour récupérer la toute dernière liste des packages disponibles.

`info [-lr] [-g` *path_or_uri* `...] [--license] [`*pkg_fmri_pattern* `...]`
Afficher des informations sur les packages dans un format lisible par l'opérateur. Plusieurs modèles de FMRI peuvent être spécifiés. Sans modèles, affiche des informations sur tous les packages installés sur l'image.

Avec `-g`, utilise le référentiel ou l'archive de packages spécifiés en tant que source de données de package pour l'opération. Cette option peut être spécifiée plusieurs fois. L'utilisation de `-g` implique `-r`.

Avec `-l`, n'affiche que les informations sur les packages installés. Option spécifiée par défaut.

Avec `-r`, fait correspondre les packages en fonction des versions les plus récentes disponibles, en récupérant (si nécessaire) des informations sur les packages qui ne sont pas

installés à partir des référentiels des éditeurs configurés pour l'image. Au moins un package doit être spécifié lors de l'utilisation de cette option. Sans `-r`, seuls les packages installés sont affichés par défaut.

Avec `--licence`, affiche les textes de licence pour les packages. Cette option peut être combinée avec `-l` ou `-r`.

`contents [-Hmr] [-a` *attribute=pattern* `...] [-g` *path_or_uri* `...] [-o` *attribute*`,...] [-s` *sort_key*`] [-t` *action_type* `...] [`*pkg_fmri_pattern* `...]`

Affiche le contenu (attributs d'action) des packages. Sans options ni opérandes, affiche la valeur de l'attribut `path` pour les actions installées dans l'image actuelle, classées dans l'ordre alphabétique par valeur d'attribut. Pour plus d'informations sur les actions et leurs attributs, reportez-vous à la section relative aux actions dans la page de manuel `pkg`(5). Consultez également la liste des noms de pseudo-attributs ci-dessous.

Avec `-a`, limite la sortie aux actions qui ont un attribut nommé dans l'argument de l'option, avec une valeur qui correspond au modèle (glob) dans l'argument de l'option (dont le nom d'attribut est suivi d'un signe égal). Si plusieurs options `-a` sont données, les actions correspondantes s'affichent.

Avec `-g`, affiche des informations sur les packages qui peuvent être installés dans cette image à partir du référentiel ou de l'archive en question. Les packages pouvant être installés incluent les packages actuellement installés et ceux qui répondent aux critères d'installation sur cette image, comme les restrictions relatives aux variantes et aux facettes. Cette option peut être spécifiée plusieurs fois. L'utilisation de `-g` implique `-r`.

Avec `-m`, affiche les attributs de l'intégralité des actions incluses dans les packages spécifiés, notamment les actions qui n'ont pas pu être installées dans cette image.

Avec `-o`, affiche les attributs répertoriés, classés en fonction des valeurs du premier attribut de la liste. L'option `-o` peut être indiquée à plusieurs reprises, ou plusieurs attributs peuvent être spécifiés comme argument d'une option `-o` en séparant les noms d'attribut par une virgule. Seules les actions avec les attributs demandés sont affichées.

Avec `-r`, affiche des informations sur les dernières versions de packages disponibles qui peuvent être installées dans cette image à partir des référentiels des éditeurs configurés dans celle-ci. Les packages pouvant être installés incluent les packages actuellement installés et ceux qui répondent aux critères d'installation sur cette image, comme les restrictions relatives aux variantes et aux facettes. Au moins un package doit être spécifié lors de l'utilisation de cette option.

Avec `-s`, trie les actions en fonction de l'attribut d'action spécifié. S'il celui-ci n'est pas fourni, le tri s'effectue par défaut en fonction du chemin ou du premier attribut spécifié par l'option `-o`. L'option `-s` peut être indiquée à plusieurs reprises.

Avec `-t`, seules les actions du type spécifié sont répertoriées. Vous pouvez spécifier plusieurs types sous forme de liste séparée par des virgules. Cette option peut être spécifiée plusieurs fois.

Avec `-H`, omet les en-têtes de la liste.

Avec *pkg_fmri_pattern*, affiche des informations relatives aux packages nommés uniquement.

Plusieurs pseudo-noms d'attribut spéciaux sont disponibles pour plus de commodité :

`action.hash`	Valeur de hachage de l'action, si l'action porte une charge utile.
`action.key`	Valeur de l'attribut clé de l'action. Par exemple, pour une action de type `file`, l'attribut clé correspond au chemin d'accès au fichier. Certaines actions ne disposent pas d'attribut clé.
`action.name`	Nom de l'action. Par exemple, dans le cas d'une action sur un fichier, le nom est `file`.
`action.raw`	Tous les attributs des actions correspondantes.
`pkg.fmri`	FMRI complet du package contenant l'action, comme `pkg://solaris/web/amp@0.5.11,5.11-0.169:20110705T153434Z`.
`pkg.name`	Nom du package contenant l'action, comme `web/amp`.
`pkg.publisher`	Editeur du package contenant l'action, comme `solaris`.
`pkg.shortfmri`	Forme abrégée du FMRI du package contenant l'action, comme `pkg://solaris/web/amp@0.5.11,5.11-0.169`.

Les sous-commandes `contents` et `search` sont liées : chacune interroge le système quant au contenu des packages. La sous-commande `contents` affiche les actions incluses dans un ou plusieurs packages installés ou installables en filtrant la sortie en fonction des options spécifiées. La sous-commande `search` aborde la requête dans l'autre sens, et affiche le nom de tous les packages contenant un jeton fourni par l'utilisateur.

Chaque sous-commande est capable d'élaborer des requêtes dont est capable l'autre. Il faut bien choisir la sous-commande, car chacune possède une manière plus naturelle de formuler certaines requêtes.

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

Recherche des correspondances à la *requête*, et affiche les résultats. Celles dont les jetons sont indexés dépendent de l'action, mais peuvent inclure du hachage de contenu et des noms de chemin. Pour plus d'informations sur les actions et leurs attributs, reportez-vous à la section relative aux actions dans la page de manuel `pkg`(5). Consultez également la liste des noms de pseudo-attributs dans les sections `pkg contents` ci-dessus et `-o` ci-dessous.

Par défaut, les requêtes sont interprétées comme une série de termes à faire correspondre exactement. Les caractères `?` et `*` peuvent être utilisés comme caractères génériques de style `glob`(3C), pour une plus grande flexibilité dans les correspondances de requêtes.

Avec `-H`, omet les en-têtes.

Avec `-I`, utilise une recherche respectant la casse.

Par défaut, et avec `-a`, effectue la recherche et affiche les informations sur les actions correspondantes.

Par défaut, `search` élague les résultats à partir des packages plus anciens que la version installée et des versions de package exclus par incorporations actuelles. Utilisez `-f` pour afficher tous les résultats, quelle que soit la version du package.

Avec `-l`, recherche les packages installés de l'image.

Avec `-o`, les colonnes des résultats peuvent être contrôlées. L'option `-o` peut être indiquée à plusieurs reprises, ou plusieurs attributs peuvent être spécifiés comme argument d'une option `-o` en séparant les noms d'attribut par une virgule. Outre les pseudo-attributs décrits ci-dessus, les attributs suivants sont définis pour les résultats de la recherche :

`search.match`	Désigne la chaîne qui correspond à la requête de recherche.
`search.match_type`	Désigne l'attribut qui contient la chaîne qui correspond à la requête de recherche.

Avec `-p`, affiche les packages dont certaines actions correspondent à chaque terme de la requête. L'utilisation de cette option équivaut à placer les crochets angulaires (<>) autour de chaque terme dans la requête. Une description plus détaillée de l'opérateur <> est disponible ci-dessous.

Par défaut, et avec `-r`, recherche les référentiels correspondant aux éditeurs de l'image.

Avec `-s`, recherche le référentiel `pkg`(5) situé sur l'URI indiqué. Cette option peut être spécifiée plusieurs fois. Les archives de packages ne sont pas prises en charge.

Les deux options `-l` et `-r` (ou `-s`) peuvent être spécifiées ensemble, auquel cas des recherches locales et distantes sont effectuées.

En plus de la simple mise en correspondance de jetons et la recherche à l'aide de caractères génériques, un langage d'interrogation plus complexe est pris en charge. Il est possible d'effectuer des recherches de phrases en utilisant des guillemets simples ou doubles (`'` ou "). Assurez-vous de prendre votre shell en compte de telle sorte que `pkg` voie réellement `'` ou ".

La recherche booléenne à l'aide de ET et OU est prise en charge. Les requêtes par champ, ou structurées, sont prises en charge. La syntaxe doit être : *pkg_name*: *action_type*:*key*:*token*. Les champs manquants sont implicitement considérés comme des caractères génériques. Une recherche de `:basename:pkg` correspond à tous les types d'action dans tous les packages à l'aide d'une clé de `basename` et qui correspondent au jeton `pkg`. Des caractères génériques explicites peuvent être utilisés dans les champs `pkg_name` et `token`. Les champs `action_type` et `key` doivent correspondre exactement.

Pour convertir les actions en packages contenant ces actions, utilisez <>. Avec l'option -a, la recherche de token donne des informations sur les actions correspondant à token, alors que la recherche de <token> répertorie les packages contenant des actions qui correspondaient token.

verify [-Hqv] [*pkg_fmri_pattern* ...]
Valide l'installation des packages sur l'image actuelle. Si la stratégie de signature en cours pour les éditeurs concernés n'est pas ignore, les signatures de chaque package sont validées en fonction de cette stratégie. Reportez-vous à signature-policy dans Propriétés de l'image ci-dessous pour savoir comment les stratégies de signature sont appliquées.

Avec -H, omet les en-têtes de la sortie de vérification.

Avec -q, n'imprime rien, mais renvoie un échec s'il y a des erreurs fatales.

Avec -v, inclut les messages d'information concernant les packages.

fix [--accept] [--licenses] [*pkg_fmri_pattern* ...]
Corrige les éventuelles erreurs signalées par pkg verify. La vérification du contenu des packages installés repose sur une analyse de contenu personnalisée dont les résultats peuvent différer de ceux des autres programmes.

Avec --accept, indique que vous acceptez les conditions de licence des packages qui sont mis à jour ou installés. Si vous ne spécifiez pas cette option alors que les licences de package exigent l'acceptation, l'opération échoue.

Avec --licenses, affiche l'ensemble des licences des packages qui seront installés ou mis à jour lors de cette opération.

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* ...)
Rétablit les fichiers tels qu'ils étaient livrés. Tous les fichiers marqués avec une valeur particulière, ou les fichiers individuels peuvent être rétablis. La propriété des fichiers et les protections sont également restaurées.

Attention – Le fait de rétablir les valeurs par défaut de certains fichiers modifiables peut empêcher le système de s'initialiser, ou provoquer d'autres dysfonctionnements.

Pour toutes les autres options, reportez-vous à la commande install ci-dessus pour connaître leur utilisation et leurs effets.

mediator [-aH] [-F *format*] [*mediator* ...]
Affiche la version et/ou l'implémentation actuellement sélectionnée pour tous les médiateurs, ou, avec des arguments, uniquement pour les médiateurs spécifiés.

Avec -a, répertorie les médiateurs qu'il est possible de définir pour les packages actuellement installés.

Avec `-F`, spécifie un autre format de sortie. A l'heure actuelle, seul le format `tsv` (valeurs séparées par des tabulations) est valide.

Avec `-H`, omet les en-têtes de la liste.

`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* `...`
Définit la version et/ou l'implémentation des médiateurs spécifiés dans l'image actuelle.

Avec `-I`, définit l'implémentation de l'interface médiatrice à utiliser. Par défaut, si aucune version n'est spécifiée, toutes les versions d'implémentation sont autorisées. Pour spécifier une implémentation sans version, ajoutez un signe arobase (@).

Avec `-V`, définit la version de l'interface médiatrice à utiliser.

Si la version et/ou l'implémentation de médiateurs spécifiés n'est pas disponible actuellement, tous les liens utilisant ces médiateurs sont supprimés.

Pour toutes les autres options, reportez-vous à la commande `install` ci-dessus pour connaître leur utilisation et leurs effets.

`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* `...`
Rétablit la version et/ou l'implémentation des médiateurs spécifiés dans le système par défaut.

Avec `-I`, rétablit seulement l'implémentation de l'interface médiatrice.

Avec `-V`, rétablit seulement la version de l'interface médiatrice.

Pour toutes les autres options, reportez-vous à la commande `install` ci-dessus pour connaître leur utilisation et leurs effets.

`variant [-H] [variant.`*variant_name* `...]`
Sans arguments, affiche les valeurs actuelles de toutes les variantes définies dans cette image. Avec des arguments, affiche la valeur de chaque variante spécifiée (`variant.`*variant_name*) définie dans cette image.

Avec `-H`, omet les en-têtes de la liste.

Pour plus d'informations sur les variantes, reportez-vous à la section “Facettes et variantes” dans la page de manuel `pkg`(5).

`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* `...`
Modifie les valeurs des variantes spécifiées définies dans l'image actuelle.

Pour connaitre l'utilisation et les effets d'une option, reportez-vous à la commande `install` ci-dessus.

La modification de la valeur d'une variante peut entraîner la suppression, la mise à jour ou l'installation du contenu du package. La modification de la valeur d'une variante peut entraîner l'installation, la mise à jour ou la suppression de packages entiers en vue de se conformer à la nouvelle configuration de l'image. Pour plus d'informations sur les variantes, reportez-vous à la section “Facettes et variantes” dans la page de manuel `pkg`(5).

`facet [-H] [`*facet_name* `...]`
Sans arguments, affiche les valeurs actuelles de toutes les facettes explicitement définies dans cette image par le biais de la commande `pkg change-facet`. Avec des arguments, affiche la valeur de chaque facette spécifiée (*facet_name*) définie dans cette image.

Avec `-H`, omet les en-têtes de la liste.

Pour plus d'informations sur les facettes, reportez-vous à la section “Facettes et variantes” dans la page de manuel `pkg`(5).

`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] ...`
Modifie les valeurs des facettes spécifiées définies dans l'image actuelle.

Les facettes peuvent être définies sur `True` (Vrai) ou `False` (Faux). Si le paramètre `None` est défini pour une facette, la valeur par défaut `True` lui est appliquée. Ainsi, toutes les actions correspondantes seront installées. Pour plus d'informations sur les actions, reportez-vous à la section “Actions” dans la page de manuel `pkg`(5).

Pour connaitre l'utilisation et les effets d'une option, reportez-vous à la commande `install` ci-dessus.

La modification de la valeur d'une facette peut entraîner la suppression, la mise à jour ou l'installation du contenu des packages. La modification de la valeur d'une facette peut entraîner l'installation, la mise à jour ou la suppression de packages entiers en vue de se conformer à la nouvelle configuration de l'image. Pour plus d'informations sur les facettes, reportez-vous à la section “Facettes et variantes” dans la page de manuel `pkg`(5).

`avoid [`*pkg_fmri_pattern* `...]`
Permet d'éviter les packages spécifiés s'ils sont la cible d'une dépendance du groupe en plaçant les noms des packages qui correspondent actuellement aux modèles sur la liste avoid. Seuls les packages qui ne sont pas installés peuvent être évités. Si un package est actuellement la cible d'une dépendance du groupe, sa désinstallation le place dans la liste avoid.

Sans arguments, affiche chaque package évité accompagné des packages qui ont une dépendance de groupe sur ce package.

Les packages qui se trouvent sur la liste avoid sont installés si nécessaire pour satisfaire une dépendance obligatoire. Si cette dépendance est supprimée, le package est désinstallé.

`unavoid [`*pkg_fmri_pattern* `...]`
Supprime les packages spécifiés de la liste avoid. Les packages de la liste avoid qui correspondent à la dépendance de groupe d'un package installé ne peuvent pas être supprimés à l'aide de cette sous-commande. Pour supprimer de cette liste un package qui correspond à une dépendance de groupe, installez le package.

Sans arguments, affiche la liste des packages évités.

`freeze [-n] [-c` *reason*`] [`*pkg_fmri_pattern*`] ...`
Fige les packages spécifiés aux versions spécifiées. Si aucune version n'est donnée, le package doit être installé et figé à la version installée. Lorsqu'un package figé est installé ou mis à jour, sa version doit correspondre à celle à laquelle il a été figé. Par exemple, si un package a été figé à la version 1.2, il peut être mis à jour vers 1.2.1, 1.2.9, 1.2.0.0.1, et ainsi de suite. Ce package ne peut pas avoir un numéro de version correspondant à 1.3 ou 1.1. Un éditeur spécifié dans le paramètre *pkg_fmri_pattern* est utilisé pour trouver des packages correspondants. Cependant, les informations de l'éditeur ne sont pas enregistrées dans le cadre de l'opération de figement. Un package est figé par rapport à sa version uniquement, et non son éditeur. Le fait de figer un package qui est déjà figé remplace la version figée par la nouvelle version spécifiée.

Si aucun package n'est fourni, les informations relatives aux packages actuellement figés s'affichent : noms des packages, versions, date de figement des packages et raison associée.

Le fait de figer un package n'empêche pas la suppression du package. Aucun avertissement n'est affiché si le package est supprimé.

Avec `-c`, enregistre la valeur de *reason* avec les packages qui sont figés. La raison est affichée si un figement empêche l'installation ou la mise à jour de réussir.

Avec `-n`, effectue un essai de l'opération et affiche la liste des packages qui seraient figés, sans qu'aucun package ne soit effectivement figé.

`unfreeze [-n] [`*pkg_name_pattern*`] ...`
Supprime des packages spécifiés les contraintes imposées par le figement. Les versions éventuellement fournies sont ignorées.

Avec `-n`, effectue un essai de l'opération de libération et affiche la liste des packages qui seraient libérés, sans que les packages ne soient effectivement libérés.

`property [-H] [`*propname* `...]`
Affiche des informations de propriété d'image. Sans arguments, affiche les noms et les valeurs de toutes les propriétés de l'image. Si une liste spécifique des noms de propriété est requise, affiche les noms et les valeurs de ces propriétés. Pour une description des propriétés de l'image, reportez-vous à la section “Propriétés de l'image” ci-après.

Avec `-H`, omet les en-têtes de la liste.

`set-property` *propname propvalue*
Met à jour une propriété d'image existante ou en ajoute une nouvelle.

`add-property-value` *propname propvalue*
Ajoute une valeur à une propriété d'image ou ajoute une nouvelle propriété d'image.

`remove-property-value` *propname propvalue*
Supprime une valeur à partir d'une propriété d'image existante.

`unset-property` *propname* `...`
Supprime une ou plusieurs propriétés d'image existantes.

`publisher [-HPn] [`*publisher* `...]`
Affiche les informations concernant l'éditeur. Sans arguments, affiche la liste de tous les éditeurs, leur URI d'origine et les miroirs dans l'ordre de préférence de recherche. Si des éditeurs spécifiques sont requis, affiche la configuration détaillée de ces éditeurs.

Avec `-H`, omet les en-têtes de la liste.

Avec `-P`, affiche uniquement le premier éditeur dans l'ordre de recherche d'éditeurs.

Avec `-n`, afficher uniquement les éditeurs activés.

`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*
Met à jour un éditeur existant ou ajoute un éditeur de package. Si aucune option n'affecte l'ordre de recherche, les nouveaux éditeurs sont ajoutés à l'ordre de recherche et sont donc recherchés en dernier.

Avec `-P` ou `--search-first` définit l'éditeur spécifié en premier dans l'ordre de recherche. Lors de l'installation de nouveaux packages, cet éditeur est recherché en premier. Les mises à jour de packages déjà installés proviennent de l'éditeur qui a initialement fourni le package, tant que cet éditeur est l'éditeur résident ("sticky"). Quand `-P` ou `--search-first` est utilisée avec `-p`, seuls les éditeurs ajoutés sont placés en premier dans l'ordre de recherche.

Avec `--non-sticky`, indique que les packages initialement installés à partir de cet éditeur peuvent être mis à jour par des éditeurs de rang supérieur.

Avec `--sticky`, indique que les mises à jour de packages installés à partir de cet éditeur doivent également provenir de cet éditeur. Il s'agit du comportement par défaut.

Avec `-search-before`, modifie l'ordre de recherche des éditeurs de manière à ce que l'éditeur modifié soit recherché avant l'éditeur spécifié. Lorsque cette option est utilisée avec `-p`, `--search-before` s'applique uniquement aux éditeurs ajoutés.

Avec `-search-after`, modifie l'ordre de recherche des éditeurs de manière à ce que l'éditeur modifié soit recherché après l'éditeur spécifié. Lorsque cette option est utilisée avec `-p`, `--search-after` s'applique uniquement aux éditeurs ajoutés.

Avec `--approve-ca-cert`, ajoute le certificat fourni comme un certificat d'AC de confiance. Les hachages de la représentation PEM des certificats d'AC approuvés par l'utilisateur sont répertoriés dans la sortie détaillée de la commande `pkg publisher`.

Avec `--revoke-ca-cert`, traite le certificat avec le hachage donné de sa représentation PEM comme révoqué. Les hachages de la représentation PEM des certificats d'AC révoqués par l'utilisateur sont répertoriés dans la sortie détaillée de la commande `pkg publisher`.

Avec `--unset-ca-cert`, supprime le certificat avec le hachage donné de la liste des certificats approuvés et de la liste des certificats révoqués.

Avec `--set-property`, met à jour une propriété d'éditeur existante ou en ajoute une nouvelle.

Avec `--add-property-value`, ajoute une valeur à une propriété d'éditeur existante ou en ajoute une nouvelle.

Avec `--remove-property-value`, supprime une valeur d'une propriété d'éditeur existante.

Avec `--unset-property`, supprime une propriété d'éditeur existante.

Les options `-c` et `-k` spécifient respectivement le certificat SSL du client et la clé.

Avec `-g` (`--add-origin`), ajoute l'URI ou le chemin spécifiés en tant qu'origine de l'éditeur donné. Il doit s'agir de l'emplacement d'un référentiel ou d'une archive de packages.

Avec `-G` (`--remove-origin`), supprime l'URI ou le chemin de la liste des origines de l'éditeur donné. La valeur spéciale * peut être utilisée pour supprimer toutes les origines.

Avec `--no-refresh`, n'essaye pas de contacter les référentiels pour les éditeurs de l'image pour récupérer la toute dernière liste des packages disponibles et d'autres métadonnées.

Avec `--reset-uuid`, choisit un nouvel identificateur unique pour identifier cette image auprès de l'éditeur.

Avec `-m` (`--add-mirror`), ajoute l'URI en tant que miroir pour l'éditeur donné.

Avec `-M` (`--remove-mirror`), supprime l'URI de la liste des miroirs de l'éditeur donné. La valeur spéciale * peut être utilisée pour supprimer tous les miroirs.

Avec -p, récupère les informations de configuration relatives à l'éditeur à partir de l'URI de référentiel spécifiée. Si un éditeur est spécifié, seul l'éditeur correspondant est ajouté ou mis à jour. Si aucun éditeur n'est spécifié, tous les éditeurs sont ajoutés ou mis à jour de façon adéquate. Cette option ne peut pas être combinée avec les options -g, --add-origin, --G, - -remove-origin, -m, --add-mirror, - M, --remove-mirror, --disable, --enable , --no-refresh ou --reset-uuid.

Avec -e (--enable), active l'éditeur. Avec -d (--disable), désactive l'éditeur. Un éditeur désactivé n'est pas utilisé lorsque la liste de packages est complétée ou dans certaines opérations liées aux packages (installation, désinstallation, mise à jour). Cependant, il est toujours possible de définir et d'afficher les propriétés d'un éditeur désactivé. S'il n'existe qu'un seul éditeur, il ne peut pas être désactivé.

`/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*`]`

Avec -p, récupère les informations de configuration relatives à l'éditeur à partir de l'URI de référentiel spécifiée. Si un éditeur est spécifié, seul l'éditeur correspondant est ajouté ou mis à jour. Si aucun éditeur n'est spécifié, tous les éditeurs sont ajoutés ou mis à jour de façon adéquate. Reportez-vous à la section relative à la commande `pkg set-publisher` ci-dessus pour une description des autres options utilisables conjointement avec l'option -p. L'option -p ne peut pas être associée aux options -g, --add-origin, -G, --remove-origin, -m, --add-mirror, -M, --remove-mirror, --disable, --enable, --no-refresh ou --reset-uuid.

`unset-publisher` *publisher* `...`

Supprime la configuration associée à l'éditeur ou aux éditeurs donnés.

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

Affiche l'historique des commandes de l'image applicable.

Avec -H, omet les en-têtes de la liste.

Avec -t, affiche les enregistrements de journal, dans une liste d'horodatages séparés par des virgules, au format `%Y-%m-%dT%H:%M:%S` (voir `strftime`(3C)). Pour spécifier une plage de temps, utilisez un trait d'union (-) entre un horodatage de début et un horodatage de fin. Le mot-clé `now` peut être utilisé en tant qu'alias pour l'heure actuelle. Si les horodatages spécifiés contiennent des doublons ou des plages de dates qui se chevauchent, une seule instance de chaque événement historique en double est imprimée.

Avec -l, affiche les enregistrements de journal au format long, qui, en plus de du format standard, incluent le résultat de la commande, l'heure à laquelle l'exécution de la

commande s'est terminée, la version et le nom du client utilisé, le nom de l'utilisateur qui a effectué l'opération, ainsi que les erreurs survenues lors de l'exécution de la commande.

Avec `-n`, affiche uniquement le nombre spécifié des entrées les plus récentes.

Avec `-o`, affiche la sortie sous la forme d'une liste de noms de colonne séparés par des virgules. Les noms de colonne valides sont les suivants :

`be`	Nom de l'environnement d'initialisation sur lequel cette opération a été lancée.
`be_uuid`	`uuid` de l'environnement d'initialisation sur lequel cette opération a été lancée.
`client`	Nom du client.
`client_ver`	Version du client.
`command`	Ligne de commande utilisée pour cette opération.
`finish`	Heure à laquelle cette opération s'est terminée.
`ID`	ID de l'utilisateur qui a démarré cette opération.
`new_be`	Nouvel environnement d'initialisation créé par cette opération.
`new_be_uuid`	`uuid` du nouvel environnement d'initialisation créé par cette opération.
`operation`	Nom de l'opération.
`outcome`	Résumé du résultat de cette opération.
`reason`	Informations supplémentaires sur le résultat de cette opération.
`snapshot`	Instantané effectué au cours de cette opération. Il n'est enregistré que si l'instantané n'a pas été automatiquement supprimé après la réussite de l'opération.
`start`	Heure à laquelle cette opération a démarré.
`time`	Temps total nécessaire pour effectuer cette opération. Pour les opérations qui prennent moins d'une seconde, 0:00:00 est affiché.
`user`	Nom de l'utilisateur qui a démarré cette opération.

Si les colonnes `command` ou `reason` sont spécifiées, elles doivent être le dernier élément de la liste `-o`, afin de maintenir la séparation entre les champs de sortie. Ces deux colonnes ne peuvent pas être affichées dans la même commande `history` .

Un astérisque (*) s'affiche après les valeurs de `be` ou `new_be` si l'environnement d'initialisation n'est plus présent sur le système.

Les valeurs de be et new_be sont obtenues en recherchant le nom de l'environnement d'initialisation actuel, à l'aide des champs be_uuid ou new_be_uuid. Si un environnement d'initialisation a été renommé, puis supprimé, les valeurs affichées pour be et new_be sont les valeurs enregistrées au moment de l'opération pkg.

`purge-history`
Supprime toutes les informations d'historique existantes.

`rebuild-index`
Recrée l'index utilisé par `pkg search`. Il s'agit d'une opération de récupération non prévue pour une utilisation générale.

`update-format`
Met à jour le format de l'image sur la version en cours. Une fois cette opération terminée, l'image ne peut plus être utilisée avec des versions du système pkg(5) plus anciennes.

`version`
Affiche une chaîne unique qui identifie la version de pkg(1). Cette chaîne n'est pas garantie pour être comparable de n'importe quelle manière entre les versions.

`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*
A l'emplacement donné par *dir*, créez une image appropriée aux opérations sur les packages. L'image par défaut est de type utilisateur, tel qu'il est indiqué par l'option `-U` (`--user`). Le type d'image peut être défini sur une image complète (`--F` ou `--full`) ou sur une image partielle (`-P` ou `--partial`) liés à l'image complète englobant le chemin *dir* donné. Des origines supplémentaires peuvent être spécifiées à l'aide de `-g` ou `--origin`. Des miroirs supplémentaires peuvent être spécifiés à l'aide de `--m` ou `--mirror`.

Un référentiel de packages URI doit être fourni à l'aide de l'option `-p` ou `- -publisher`. Si un nom d'éditeur est également fourni, seul cet éditeur est ajouté lorsque l'image est créée. Si un nom de l'éditeur n'est pas fourni, tous les éditeurs connus par le référentiel spécifié sont ajoutés à l'image. Une tentative de récupération du catalogue associé à cet éditeur est effectuée après les opérations de création initiales.

Pour les éditeurs qui se servent de l'authentification SSL client, une clé client et un certificat client peuvent être enregistrés via les options `-c` et `-k`. Ce certificat et cette clé sont utilisés pour tous les éditeurs ajoutés pendant la création de l'image.

Si l'image doit être exécutée dans un contexte de zone non globale, l'option `- z` (`--zone`) peut être utilisée pour définir une variante adéquate.

Avec `-f` (`--force`), contraint la commande à créer une image sur une image existante. Cette option doit être utilisée avec précaution.

Avec `--no-refresh`, n'essaye pas de contacter les référentiels pour les éditeurs de l'image pour récupérer la toute dernière liste des packages disponibles et d'autres métadonnées.

Avec `--variant`, définit la variante spécifiée sur la valeur indiquée Pour plus d'informations sur les variantes, reportez-vous à la section “Facettes et variantes” dans la page de manuel `pkg`(5).

Avec `--facet`, définit la facette spécifiée sur la valeur indiquée. Pour plus d'informations sur les facettes, reportez-vous à la section “Facettes et variantes” dans la page de manuel `pkg`(5).

Avec `--set-property`, définit la propriété de l'image spécifiée sur la valeur indiquée. Pour une description des propriétés de l'image, reportez-vous à la section “Propriétés de l'image” ci-après.

Propriétés de l'image

Les propriétés suivantes définissent les caractéristiques de l'image. Ces propriétés stockent des informations sur l'objectif, le contenu et le comportement de l'image. Pour afficher les valeurs actuelles des propriétés de l'image, exécutez la commande `pkg property`. Pour modifier les valeurs de ces propriétés, exécutez les commandes `pkg set-property` et `pkg unset-property`.

`be-policy`
(string) Permet d'indiquer si un environnement d'initialisation est créé au cours d'opérations relatives aux packages. Les valeurs suivantes sont autorisées :

`default`	Applique la valeur de stratégie de création d'un environnement d'initialisation par défaut, soit `create-backup`.
`always-new`	Nécessite une réinitialisation de l'ordinateur pour toutes les opérations sur les packages et leur exécution dans un nouvel environnement d'initialisation défini comme actif à la prochaine initialisation. Aucun environnement d'initialisation de sauvegarde n'est créé sauf explicitement demandé. Cette stratégie est la plus sûre, mais elle est plus stricte que ce qui est demandé dans la plupart des sites, dans la mesure où aucun package ne peut être ajouté sans qu'il faille réinitialiser le système.

`create-backup`
Pour les opérations sur les packages qui nécessitent une réinitialisation, un nouvel environnement d'initialisation est créé et défini comme actif à la prochaine initialisation. Si les packages sont modifiés ou si du contenu pouvant avoir des répercussions sur le noyau est installé, et que l'opération a une incidence sur l'environnement d'initialisation actif, un environnement de sauvegarde est créé, mais il n'est pas défini comme actif. Un environnement d'initialisation de sauvegarde peut également être explicitement demandé.

Cette stratégie est potentiellement dangereuse si des logiciels récemment installés entraînent une instabilité du système, ce qui est assez rare.

`when-required`
Pour les opérations sur les packages qui nécessitent une réinitialisation, un nouvel environnement d'initialisation est créé et défini comme actif à la prochaine initialisation. Aucun environnement d'initialisation de sauvegarde n'est créé sauf explicitement demandé.

Cette stratégie est la plus dangereuse car, si un changement dans les packages de l'environnement d'initialisation actif rend impossible toute modification ultérieure, il peut arriver qu'il n'y ait pas d'environnement d'initialisation de secours récent.

`ca-path`
(string) Nom de chemin d'accès qui pointe vers un répertoire dans lequel les certificats d'AC sont conservés pour les opérations SSL. Le format de ce répertoire est spécifique à l'implémentation SSL sous-jacente. Pour utiliser un autre emplacement pour les certificats de CA de confiance, modifiez cette valeur de manière à ce qu'elle pointe vers un autre répertoire. Reportez-vous aux paragraphes `CApath` de `SSL_CTX_load_verify_locations`(3openssl) pour connaître les exigences concernant le répertoire des certificats d'AC.

Valeur par défaut : `/etc/openssl/certs`

`check-certificate-revocation`
(boolean) Si cette propriété est définie sur `True`, le client de package tente de contacter les points de distribution CRL dans les certificats utilisés pour la vérification des signatures afin de déterminer si le certificat a été révoqué depuis son émission.

Valeur par défaut : `False`

`flush-content-cache-on-success`
(boolean) Si la valeur est définie sur `True`, le client du package supprime les fichiers de son cache de contenu au terme de l'installation ou de la mise à jour. Pour les opérations de mise à jour, le contenu n'est supprimé que dans l'environnement d'initialisation source. Lorsqu'une opération sur les packages se produit ensuite sur l'environnement d'initialisation de destination, le client du package vide son cache de contenu si cette option n'a pas été modifiée.

Cette propriété peut être utilisée pour limiter la taille du cache de contenu sur les systèmes dotés d'un espace disque réduit. Cette propriété peut ralentir les opérations.

Valeur par défaut : `True`

`mirror-discovery`
(boolean) Cette propriété demande au client de découvrir les miroirs de contenu link-local à l'aide de mDNS et DNS-SD. Si cette propriété est définie sur `True`, le client tente de télécharger le contenu du package à partir de miroirs qu'il détecte de manière dynamique. Pour exécuter un miroir qui publie son contenu via mDNS, reportez-vous à la page de manuel `pkg.depotd`(1M).

Valeur par défaut : `False`

`send-uuid`
(boolean) Envoie l'identificateur unique universel (UUID) de l'image pendant les opérations sur le réseau. Bien que les utilisateurs puissent désactiver cette option, certains référentiels de réseau peuvent refuser de communiquer avec des clients qui ne fournissent pas un UUID.

Valeur par défaut : `True`

`signature-policy`
(string) Détermine les vérifications à effectuer sur les manifestes lors de l'installation, la mise à jour, la modification ou la vérification des packages dans l'image. La stratégie finale appliquée à un package dépend de la combinaison des stratégies de l'image et de l'éditeur. La combinaison sera au moins aussi stricte que la plus stricte des deux stratégies prises individuellement. Par défaut, le client du package ne vérifie pas si les certificats ont été révoqués. Pour activer ces vérifications, qui peuvent nécessiter que le client contacte des sites web externes, définissez la propriété d'image `check-certificate-revocation` sur `True`. Les valeurs suivantes sont autorisées :

`ignore`	Ignore les signatures pour tous les manifestes.
`verify`	Vérifie que tous les fichiers manifestes avec signatures sont valablement signés, mais ne nécessite pas que tous les packages installés soient signés. Il s'agit de la valeur par défaut.
`require-signatures`	Demande à ce que tous les nouveaux packages installés disposent au moins d'une signature valide. Les commandes `pkg fix` et `pkg verify` avertissent également lorsqu'un package installé ne possède pas de signature valide.
`require-names`	Suit les mêmes exigences que `require-signatures` mais nécessite aussi que les chaînes répertoriées dans la propriété `signature-required-names` s'affichent en tant que nom commun des certificats utilisés pour vérifier les chaînes de confiance des signatures.

`signature-required-names`
(list of strings) Liste de noms qui doivent être considérés comme des noms communs de certificats lors de la validation des signatures d'un package.

`trust-anchor-directory`
(string) Nom de chemin d'accès au répertoire qui contient les ancres de confiance pour l'image. Ce chemin est relatif à l'image. La valeur par défaut est `ignore`.

`use-system-repo`
(boolean) Cette propriété indique si l'image doit utiliser le référentiel du système comme une source pour l'image et la configuration de l'éditeur et comme un serveur proxy pour la communication avec les éditeurs fournis. La valeur par défaut est `False`. Reportez-vous à `pkg.sysrepo`(1M) pour plus d'informations sur les référentiels du système.

Propriétés de l'éditeur

Les propriétés suivantes définissent la stratégie de signature d'un éditeur particulier. Les propriétés d'image portant le même nom définissent la stratégie de signature de l'image. Pour afficher les valeurs actuelles des propriétés d'un éditeur particulier, exécutez la commande `pkg publisher` *publisher_name*. Pour modifier les valeurs des propriétés de signature de ces éditeurs, utilisez les options `--set-property` et `--unset-property` de la commande `pkg set-publisher`.

`signature-policy`
(string) Cette propriété fonctionne exactement comme pour la propriété d'image du même nom, à l'exception près qu'elle ne s'applique qu'aux packages d'un éditeur particulier.

`signature-required-names`
(list of strings) Cette propriété fonctionne exactement comme pour la propriété d'image du même nom, à l'exception près qu'elle ne s'applique qu'aux packages d'un éditeur particulier.

Exemples

EXEMPLE 1 Création d'une image avec un éditeur configuré

Créez une image complète, avec l'éditeur `example.com`, stocké dans `/aux0/example_root`.

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

EXEMPLE 2 Création d'une image, spécification de plusieurs origines et d'un miroir supplémentaires

Créez une image complète, avec l'éditeur `example.com`, qui dispose également d'un miroir supplémentaire, de deux autres origines, et qui est stocké dans `/aux0/example_root`.

```
$ 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
```

EXEMPLE 3 Création d'une image avec aucun éditeur configuré

Créez une image complète avec aucun éditeur configuré dans `/aux0/example_root` .

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

EXEMPLE 4 Installation d'un package

Installez la dernière version du package `widget` dans l'image actuelle.

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

EXEMPLE 5 Création de la liste du contenu spécifié d'un package

Répertoriez le contenu du package `system/file-system/zfs`. Affichez le nom de l'action, le mode de fichier (s'il est défini), la taille (si elle est définie), le chemin d'accès, et la cible (s'il existe un lien). Limitez l'action aux types `dir`, `file`, `link`, et `hardlink`, étant donné qu'en

EXEMPLE 5 Création de la liste du contenu spécifié d'un package *(Suite)*

spécifiant l'attribut `action.name`, disponible pour toutes les actions, vous affichez une ligne pour toutes les actions, ce qui n'est pas souhaité ici.

```
$ 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
...
```

EXEMPLE 6 Création de la liste du contenu spécifié de deux packages

Répertoriez le contenu des packages `web/browser/firefox` et `mail/thunderbird`, en limitant l'affichage au nom du package et aux attributs de chemin d'accès des actions dont l'attribut `path` se termine par `.desktop` ou `.png`.

```
$ 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
...
```

EXEMPLE 7 Recherche d'un package

Recherchez le jeton `bge` dans la base de données des packages.

```
$ 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
```

EXEMPLE 7 Recherche d'un package *(Suite)*

Le jeton est dans le package `driver/network/bge`, chacun étant le nom de base de l'action de fichier représentant `/kernel/drv/` *arch*`/bge` et comme un nom de pilote.

EXEMPLE 8 Recherche des packages qui dépendent du package spécifié

Recherchez les packages installés qui dépendent de `package/pkg`.

```
$ 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
```

EXEMPLE 9 Recherche des dépendances

Recherchez toutes les dépendances `incorporate` dans les packages installés.

```
$ 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
incorporate depend pkg:/BRCMbnxe@0.5.11,5.11-0.133 pkg:/consolidation/osnet/osnet-incorporation@0.5
...
```

EXEMPLE 10 Ajout d'un éditeur

Ajoutez un nouvel éditeur `example.com`, avec un référentiel situé dans `http://www.example.com/repo`.

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

EXEMPLE 11 Ajout d'un éditeur avec une clé et un certificat

Ajoutez un nouvel éditeur `example.com`, avec un référentiel sécurisé situé dans `https://secure.example.com/repo`, et une clé et un certificat stockés dans le répertoire `/root/creds`.

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

EXEMPLE 12 Ajout et configuration automatique d'un éditeur

Ajoutez un nouvel éditeur à l'aide d'un référentiel situé dans `/export/repo` utilisant la configuration automatique.

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

EXEMPLE 13 Ajout et configuration manuelle d'un éditeur

Ajoutez un nouvel éditeur `example.com` avec un référentiel situé dans `/export/repo/example.com` utilisant une configuration manuelle.

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

EXEMPLE 14 Vérification de tous les packages signés

Configurez une image pour vérifier tous les packages signés.

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

EXEMPLE 15 Demande de signature de tous les packages

Configurez une image pour exiger que tous les packages soient signés, et que la chaîne `example.com` soit vue comme un nom commun pour l'un des certificats de la chaîne d'approbation.

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

EXEMPLE 16 Demande de signature de tous packages d'un éditeur spécifié

Configurez une image pour exiger que tous les packages installés à partir de l'éditeur `example.com` soient signés.

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

EXEMPLE 17 Demande d'une chaîne spécifiée dans la chaîne de confiance

Ajoutez la chaîne `foo` à la liste de noms communs de l'image qui doivent être vus dans la chaîne de confiance d'une signature pour qu'elle soit considérée valide.

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

EXEMPLE 18 Suppression d'une chaîne à partir de la chaîne de confiance pour un éditeur spécifié

Supprimez la chaîne `foo` dans la liste de noms communs qui doivent être vus pour valider une signature pour l'éditeur `example.com`.

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

EXEMPLE 19 Ajout d'un certificat d'AC de confiance

Ajoutez le certificat stocké dans `/tmp/example_file.pem` en tant que certificat d'AC de confiance pour l'éditeur `example.com`.

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

EXEMPLE 20 Révocation d'un certificat

Révoquez le certificat avec le hachage `a12345` pour l'éditeur `example.com`, afin d'empêcher le certificat de valider des signatures pour les packages dans `example.com`.

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

EXEMPLE 21 Omission de données d'un certificat

Faites que `pkg` omette que le certificat `a12345` n'a jamais été ajouté ou révoqué par l'utilisateur.

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

EXEMPLE 22 Mise à niveau vers une version antérieure d'un package

Mettez à niveau le package installé `foo@1.1` vers une version plus ancienne.

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

EXEMPLE 23 Changement d'installation d'un package conflictuel

Dans le cas de deux packages entrant en conflit, changez de package installé. Supposons que le package A dépend du package B ou C, et que B et C sont incompatibles. Si A et B sont installés, exécutez la commande suivante pour utiliser C au lieu de B sans désinstaller A :

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

EXEMPLE 24 Création de la liste des packages inclus dans une archive de packages

Répertoriez toutes les versions de tous les packages d'une archive de packages.

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

EXEMPLE 25 Création de la liste des packages inclus dans un référentiel de packages

Répertoriez toutes les versions de tous les packages d'un référentiel.

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

EXEMPLE 26 Affichage des informations sur un package d'une archive de packages

Affichez les informations concernant la dernière version d'un package dans une archive de packages. Ce package peut ou non être actuellement installé.

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

EXEMPLE 27 Affichage du contenu d'un package d'une archive de packages

Affichez le contenu d'un package dans une archive de packages. Ce package n'est pas actuellement installé.

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

EXEMPLE 28 Suppression de tous les miroirs et origines d'un éditeur

Supprimez tous les miroirs et origines d'un éditeur et ajoutez une nouvelle origine.

EXEMPLE 28 Suppression de tous les miroirs et origines d'un éditeur *(Suite)*

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

Variables d'environnement

`PKG_IMAGE`
Répertoire contenant l'image à utiliser pour les opérations sur les packages. Ignoré si `-R` est spécifié.

`PKG_CLIENT_CONNECT_TIMEOUT`
Délai d'attente (en secondes) lors de la tentative de connexion pendant les opérations de transport (pour chaque tentative) avant que le client n'abandonne l'opération. La valeur zéro (0) représente un délai d'attente illimité.

Valeur par défaut : 60

`PKG_CLIENT_LOWSPEED_TIMEOUT`
Durée (en secondes) en dessous de la vitesse limite `lowspeed` (1024 octets/s) pendant les opérations de transport au-delà de laquelle le client abandonne l'opération. La valeur zéro (0) indique que le client ne doit pas abandonner l'opération.

Valeur par défaut : 30

`PKG_CLIENT_MAX_CONSECUTIVE_ERROR`
Nombre maximum d'erreurs de transport transitoire avant que le client n'abandonne l'opération. La valeur zéro (0) indique que le client ne doit pas abandonner l'opération.

Valeur par défaut : 4

`PKG_CLIENT_MAX_REDIRECT`
Nombre maximum de redirections HTTP ou HTTPS autorisées pendant les opérations de transport avant qu'une connexion ne soit abandonnée. La valeur zéro (0) indique que le client ne doit pas abandonner l'opération.

Valeur par défaut : 5

`PKG_CLIENT_MAX_TIMEOUT`
Nombre maximum de tentatives de transport par hôte avant que le client abandonne l'opération. La valeur zéro (0) indique que le client ne doit pas abandonner l'opération.

Valeur par défaut : 4

`http_proxy`, `https_proxy`
Serveur proxy HTTP ou HTTPS.

État de sortie

Les valeurs de sortie renvoyées sont les suivantes :

`0` La commande a réussi.

`1` Une erreur s'est produite.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

3 Plusieurs opérations ont été demandées, mais seules certaines d'entre elles ont réussi.

4 Aucune modification n'a été faite (rien à faire).

5 L'opération demandée ne peut pas être effectuée sur une image active.

6 L'opération demandée ne peut pas être effectuée, car les licences pour les packages installés ou mis à jour n'ont pas été acceptées.

7 L'image est en cours d'utilisation par un autre processus. Elle ne peut pas être modifiée.

99 Une exception imprévue est survenue.

Fichiers Une image pkg(5) peut être située arbitrairement dans un plus grand système de fichiers. Dans les descriptions de fichier suivantes, le jeton `$IMAGE_ROOT` est utilisé pour distinguer les chemins d'accès relatifs. Pour une installation système standard, `$IMAGE_ROOT` équivaut à `/`.

`$IMAGE_ROOT/var/pkg` Répertoire des métadonnées pour une image complète ou partielle.

`$IMAGE_ROOT/.org.opensolaris,pkg` Répertoire des métadonnées pour une image d'utilisateur.

Dans les métadonnées d'une image particulière, certains fichiers et répertoires contiennent des informations utiles lors de la réparation et de la récupération. Le jeton `$IMAGE_META` fait référence au répertoire de niveau supérieur contenant les métadonnées. `$IMAGE_META` est en général l'un des deux chemins d'accès donnés ci-dessus.

`$IMAGE_META/lost+found` Emplacement des répertoires entrant en conflit et fichiers déplacés au cours d'une opération sur les packages. Emplacement du contenu décompressé d'un répertoire supprimé.

`$IMAGE_META/publisher` Contient un répertoire pour chaque éditeur. Chaque répertoire stocke des métadonnées spécifiques à l'éditeur.

D'autres chemins dans l'arborescence des répertoires `$IMAGE_META` sont privés et sujets à modification.

Attributs Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg`
Stabilité de l'interface	Non validé

Voir aussi pkgsend(1), pkg.depotd(1m), glob(3C), pkg(5), beadm(1M)

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

Nom pkgdepend – Analyseur de dépendances Image Packaging System

Synopsis

```
/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 ...
```

Description `pkgdepend` est utilisé pour générer et résoudre les dépendances des packages. Un package peut dépendre de fichiers d'autres packages. `pkgdepend` est généralement utilisé en deux passes : génération de dépendance de fichier et résolution fichier à package.

La sous-commande `generate` examine le contenu d'un package et détermine les fichiers externes dont le package a besoin.

La sous-commande `resolve` permet d'accéder à la liste des fichiers à partir de l'étape `generate` et effectue des recherches dans un ensemble de packages de référence afin de déterminer les noms des packages contenant les fichiers dépendants. L'ensemble de packages de référence qui font l'objet de recherche pour les fichiers dépendants sont les packages actuellement installés sur le système de l'éditeur.

Plusieurs composants des fichiers livrés sont utilisés comme sources d'informations de dépendance :

ELF	Les en-têtes ELF dans les fichiers livrés sont analysés pour identifier les informations de dépendance, avec les options `-k` et `-D` modifiant les informations obtenues. Pour en savoir plus sur les dépendances ELF, reportez-vous à la page de manuel `ldd`(1) et au document *Linker and Libraries Guide* .
Scripts	Les scripts shell qui contiennent des lignes `#!` référençant un interpréteur engendrent une dépendance sur le package qui fournit cet interpréteur.
Python	Les scripts python sont tout d'abord analysés comme des scripts. En outre, toute importation déclarée par le script peut également servir de source d'informations de dépendance.
Liens physiques	Les liens physiques dans les fichiers manifestes engendrent une dépendance sur le package qui fournit la cible du lien.
SMF	Les manifestes de service SMF fournis qui incluent des dépendances `require_all` entraînent des dépendances sur les packages qui fournissent les manifestes SMF qui fournissent ces FMRI.

Options Les options suivantes sont prises en charge :

`-R` *dir* S'exécute sur l'image résidant au niveau de *dir*. Si aucun répertoire n'a été spécifié ou déterminé en fonction de l'environnement, la valeur par défaut est /. Reportez-vous à la section Variables d'environnement pour plus d'informations.

`--help` ou `-?` Affiche un message d'utilisation.

Sous-commandes Les sous-commandes suivantes sont prises en charge :

`generate [-IMm] -d` *dir* `[-d` *dir*`] [-D` *name*`=`*value*`] [-k` *path*`]` *manifest_file*
Produit les dépendances sur les fichiers du manifeste spécifié par *manifest_file*.

Avec `-I`, affiche les dépendances qui sont satisfaites dans *manifest_file*. N'utilisez pas le résultat de `-I` avec `pkgdepend resolve`.

Avec `-M`, affiche une liste des types de fichiers qui n'ont pas pu être analysés.

Avec `-m`, répète le manifeste d'origine avec les dépendances ajoutées postérieurement qui sont détectées.

Avec `-d`, ajoute *dir* à une liste de répertoires dans lesquels rechercher les fichiers manifestes.

Pour chaque `-D`, ajoute la *value* comme moyen de développer le jeton *name* dans les chemins d'exécution des dépendances de fichier ELF.

Pour chaque `-k`, ajoute le *path* à la liste de chemins d'exécution pour rechercher les modules de noyau. L'utilisation de l'argument `-k` supprime les chemins par défaut `/kernel` et `/usr/kernel`.

Les chemins d'exécution comme ceux spécifiés par l'option `-k` peuvent également être spécifiés par action ou par manifeste en utilisant l'attribut action ou manifeste `pkg.depend.runpath`. La valeur de l'attribut `pkg.depend.runpath` est une chaîne de valeurs séparées par des deux-points (:) des chemins à utiliser.

L'utilisation de `-k` est remplacée par n'importe quel attribut `pkg.depend.runpath` défini dans le manifeste ou l'action.

Le jeton spécial `$PKGDEPEND_RUNPATH` peut être utilisé comme composant de l'attribut `pkg.depend.runpath` afin d'inclure le chemin d'exécution système standard pour le fichier en cours d'analyse.

Dans certains cas, vous pouvez être amené à empêcher la génération automatique des dépendances. Si, par exemple, un package fournit un exemple de script Python qui importe un ensemble de modules, ces modules importés par l'exemple de script ne sont pas des dépendances du package qui fournit l'exemple de script. Utilisez l'attribut action ou manifest `pkg.depend.bypass-generate` pour empêcher la génération de dépendances contre les fichiers spécifiés.

Les valeurs `pkg.depend.bypass-generate` sont des expressions régulières Python correspondant aux noms de fichier. Les expressions régulières sont implicitement ancrées au début et à la fin du chemin du fichier. La valeur donnée dans l'exemple suivant correspond à `this/that` mais ne correspond pas à `something/this/that/the/other`.

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

Pour plus d'informations sur la syntaxe d'expression régulière Python, utilisez la commande `pydoc re` ou reportez-vous à une documentation complète à l'adresse suivante : `http://docs.python.org/dev/howto/regex.html`.

Lorsque les manifestes d'entrée `pkgdepend generate` contiennent des fichiers manifestes SMF, les services ou instances SMF déclarés par ces derniers sont inclus dans la sortie de `pkgdepend`. Ces services ou instances SMF se présentent sous la forme d'une action `set` portant le nom `org.opensolaris.smf.fmri`.

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

Transforme les dépendances des fichiers ayant des dépendances au niveau des packages qui distribuent ces fichiers. Les dépendances sont d'abord résolues par rapport aux manifestes fournis sur la ligne de commande, puis par rapport aux packages installés sur le système. Par défaut, les dépendances de chaque manifeste sont placées dans un fichier nommé *manifest_file*`.res`.

Avec `-m`, répète le manifeste duquel sont supprimées les dépendances générées à l'étape `generate`, avant d'ajouter les dépendances résolues.

Avec `-o`, génère les résultats sur la sortie standard. Cette option est destinée à l'humain. L'ajout de cette sortie à un fichier peut engendrer un manifeste non valide. Les options `-d` et `-s` sont fortement recommandées à la place de l'option `-o` pour une utilisation dans un pipeline pour le traitement d'un manifeste.

Avec `-d`, enregistre les dépendances résolues de chaque manifeste dans un fichier distinct de *output_dir*. Par défaut, chaque fichier porte le même nom de base que le manifeste à la source des dépendances qu'il contient.

Avec `-s`, pour chaque fichier de sortie, ajoute *suffix* au nom de base du fichier qui était la source des dépendances résolues. Un "." est ajouté avant *suffix* s'il n'est pas fourni.

Avec `-S`, résout uniquement par rapport aux manifestes fournis sur la ligne de commande et non ceux installés sur le système.

Avec `-v`, inclut d'autres métadonnées de débogage de dépendance de package.

Exemples

EXEMPLE 1 Génération de dépendances

Générez les dépendances pour le manifeste écrit en `foo`, dont le répertoire de contenu se situe dans `./bar/baz`, et stockez les résultats dans `foo.fdeps`.

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

EXEMPLE 2 Résolution de dépendances

Résolvez les dépendances de fichiers dans `foo.fdeps` et `bar.fdeps` les unes par rapport aux autres et par rapport aux packages installés sur le système.

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

EXEMPLE 3 Génération et résolution de dépendances pour deux manifestes

Générez les dépendances de fichier pour deux manifestes (`foo` et `bar`) et conservez toutes les informations dans les manifestes d'origine. Ensuite, résolvez les dépendances de fichier et placez les manifestes obtenus dans `./res`. Ces manifestes obtenus peuvent être utilisés avec `pkgsend publish`.

```
$ 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
```

EXEMPLE 4 Ajout de valeurs à des jetons pour des dépendances de fichier ELF

Remplacez tous les jetons `PLATFORM` dans les chemins d'exécution des fichiers ELF par `sun4v` et `sun4u` lors de la génération des dépendances pour le manifeste écrit en `foo` dont le répertoire de contenu est dans /.

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

EXEMPLE 5 Spécification d'un répertoire de module de noyau

Spécifiez `/kmod` comme répertoire dans lequel vous souhaitez rechercher les modules de noyau lors de la génération des dépendances pour le manifeste écrit en `foo` dont le répertoire de contenu est /.

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

EXEMPLE 6 Contournement de la génération de dépendances

Ajoutez `opt/python` au chemin d'exécution Python standard pour un script Python donné et contournez la génération de dépendances par rapport à tous les modules Python appelés `test` pour un fichier fourni en tant que `opt/python/foo/file.py` .

Evitez de générer les dépendances par rapport aux fichiers fournis dans `usr/lib/python2.6/vendor-packages/xdg`.

```
$ 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
```

EXEMPLE 6 Contournement de la génération de dépendances *(Suite)*

```
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
```

Variables d'environnement

`PKG_IMAGE` Spécifie le répertoire qui contient l'image à utiliser pour les opérations de package. Cette valeur est ignorée si `-R` est spécifié.

État de sortie

Les valeurs de sortie renvoyées sont les suivantes :

`0` Tout a fonctionné.

`1` Une erreur s'est produite.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

`99` Une exception imprévue est survenue.

Attributs

Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg`
Stabilité de l'interface	Non validé

Voir aussi

pkg(5)

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

Nom pkgdiff – Compare les manifestes de package

Synopsis

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

Description `pkgdiff` compare deux manifestes de package et signale les différences. Avant la comparaison, `pkgdiff` trie les manifestes et les actions selon un ordre cohérent.

La sortie se présente sous la forme suivante :

+ *complete_action* Cette action est dans *file2*, mais pas dans *file1*.

- *complete_action* Cette action est dans *file1*, mais pas dans *file2*.

actionname keyvalue [*variant values, if any*]

- *attribute1=value1* Cet *attribut* et cette *valeur* résident dans *file1*, mais pas dans *file2*.

+ *attribute2=value2* Cet *attribut* et cette *valeur* résident dans *file2*, mais pas dans *file1*.

Les actions avec différentes variantes mais les mêmes type et valeur d'attribut clé sont traitées comme des actions distinctes à des fins de comparaison. Par conséquent, les actions qui modifient les attributs sont affichées dans leur intégralité plutôt que comme des modifications d'attribut.

Options Les options suivantes sont prises en charge :

`-i` *attribute* Ignore *attribute* s'il est présent lors des comparaisons. Les valeurs de hachage de fichier peuvent être ignorées avec `-i` *hash*. Cette option ne peut pas être utilisée avec l'option `-o`. Cette option peut être répétée.

`-o` *attribute* Signale uniquement les différences dans *attribute*. Cette option ne peut pas être utilisée avec l'option `-i`. Cette option supprime toute modification d'action qui n'affecte pas *attribut* sur une action.

`-v` *name= value* Calcule uniquement les différences pour cette valeur de variante. Par exemple, calcule uniquement les différences pour `arch=sparc`. Ce repère de variante est supprimé de toutes les actions avant la comparaison. Seule une valeur peut être spécifiée par variante. Cette option peut être répétée pour différentes variantes.

État de sortie Les valeurs de sortie renvoyées sont les suivantes :

`0` Aucune différence trouvée.

`1` Des différences ont été trouvées.

`>1` Une erreur s'est produite.

`99` Une exception imprévue est survenue.

Attributs Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg`
Stabilité de l'interface	Non validé

Voir aussi pkg(5)

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

Nom pkgfmt – Formatage d'un manifeste de package

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

Description `pkgfmt` sans les options `-c` ou `-d` formate un manifeste de package de manière cohérente, avec des retours à la ligne à 80 caractères, le tri des actions par type et le tri des attributs. Les lignes qui ne sont pas analysées comme des actions, telles que les macros, les commentaires ou les transformations, n'apparaissent pas dans l'ordre de tri.

Si aucun argument n'est fourni, `pkgfmt` lit `stdin` jusqu'à l'EOF, puis écrit le manifeste formaté dans `stdout`. Tout manifeste spécifié sur la ligne de commande est formaté sur place.

`pkgfmt` avec l'option `-c` vérifie si les manifestes sont mis en forme dans le style `pkgfmt`. L'option `- d` affiche les différences si le fichier n'est pas correctement formaté.

Options Les options suivantes sont prises en charge :

`-c` Vérifie que le manifeste est formaté selon le style `pkgfmt`.

`-d` Affiche les différences de manifeste par rapport à la version au format unifié.

`-u` Ne justifie pas les lignes à 80 caractères. Cette option est utile pour appliquer des outils de traitement de texte traditionnels à des manifestes de package.

État de sortie Les valeurs de sortie renvoyées sont les suivantes :

`0` La commande a réussi.

`1` Les options `-c` ou `-d` ont été spécifiées, et un ou plusieurs manifestes ne sont pas au format normal `pkgfmt` ou une erreur s'est produite.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

`99` Une exception imprévue est survenue.

Attributs Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg`
Stabilité de l'interface	Non validé

Voir aussi `pkg`(5)

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

Nom pkglint – Lint de package Image Packaging System

Synopsis

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

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

Description `pkglint` exécute une série de vérifications sur un ou plusieurs manifestes de package, éventuellement référençant un autre référentiel.

`pkglint` doit être utilisé au cours du processus de création du package, avant sa publication. `pkglint` effectue des tests exhaustifs sur les manifestes qui peuvent être trop coûteux à effectuer pendant les opérations normales de `pkgsend`(1) ou `pkg.depotd`(1M). Les vérifications `pkglint` comprennent des tests pour les actions en double, les attributs manquants et autorisations de fichier inhabituelles.

Les fichiers manifestes pour Lint peuvent être transmis en tant que liste séparée par des espaces de fichiers locaux sur la ligne de commande, ou des manifestes peuvent être récupérés à partir d'un référentiel.

Au cours de la récupération des manifestes à partir de référentiels, lors de sa première exécution, `pkglint` crée et remplit les images d'utilisateur de `pkg`(5) dans le répertoire de cache indiqué. Si l'option `-r` est fournie, une image d'utilisateur nommée *cache_dir*`/ref_image` est créée pour le référentiel de référence. Si l'option `-l` est fournie, une image d'utilisateur nommée *cache_dir*`/lint_image` est créée pour le référentiel Lint. Aucun contenu n'est installé dans ces images. Ces images sont utilisées uniquement par `pkglint` pour récupérer des manifestes dans les référentiels.

Les appels suivants de `pkglint` peuvent réutiliser le répertoire de cache et peuvent omettre tous les arguments `-r` ou `-l`.

`pkglint` fournit une prise en charge limitée pour la configuration des éditeurs dans le répertoire de cache. Utilisez `pkg`(1) pour effectuer une configuration d'éditeur plus complexe sur ces images.

`pkglint` permet aux créateurs de package de contourner les vérifications pour un manifeste ou une action donné. Un manifeste ou une action qui contient l'attribut `pkg.limited` défini sur `True` ne génère pas de sortie Lint pour ce manifeste ou cette action.

Des paramètres `pkg.linted` granulaires supplémentaires peuvent être créés à l'aide de sous-chaînes de noms de vérification `pkglint`. Par exemple, `pkg.linted.` *check*`.`*id* défini sur `True` contourne toutes les vérifications avec le nom *check*`.`*id* pour le manifeste ou l'action donné.

Le comportement de `pkglint` peut être configuré en spécifiant un fichier `pkglintrc`. Par défaut, `pkglint` recherche dans `/usr/share/lib/pkg/pkglintrc` et `$HOME/.pkglintrc` pour les options de configuration. Utilisez l'option `-f` pour spécifier un autre fichier de configuration.

Au cours de l'exécution de Lint, les erreurs ou les avertissements sont imprimés dans `stderr`.

Options Les options suivantes sont prises en charge :

`-b` *build_no* Indique un numéro de version utilisé pour limiter la liste des packages utilisés au cours de l'opération de Lint à partir des référentiels Lint et de référence. Si aucune option `-b` n'est spécifiée, les dernières versions des packages sont utilisées. Reportez-vous à la propriété de configuration `version.pattern`.

`-c` *cache_dir* Spécifie un répertoire local utilisé pour la mise en cache des métadonnées de package à partir des référentiels Lint et de référence.

`-l` *lint_uri* Indique un URI représentant l'emplacement du référentiel Lint. Les publications basées sur HTTP et le système de fichiers sont prises en charge. Si vous spécifiez `-l`, vous devez également spécifier `-c`.

`-L` Répertorie les vérifications Lint connues et exclues, puis quitte. Affiche le nom abrégé et la description de chaque vérification. Associée à l'indicateur `-v`, affiche la méthode qui implémente la vérification au lieu de la description.

`-f` *config_file* Configure les sessions `pkglint` à l'aide du fichier de configuration *config_file*.

`-p` *regexp* Spécifie une expression régulière utilisée pour restreindre la liste des packages à consulter à partir du référentiel Lint. Tous les manifestes du référentiel de référence sont chargés (à condition qu'ils correspondent à la valeur de `-b`, si fourni), en ignorant ce motif.

`-r` *repo_uri* Indique un URI représentant l'emplacement du référentiel de référence. Si vous spécifiez `-r`, vous devez également spécifier `-c`.

`-v` Exécute `pkglint` en mode détaillé, remplaçant les paramètres `log_level` dans le fichier de configuration.

`--help` ou `-?` Affiche un message d'utilisation.

Fichiers Le fichier de configuration `pkglintrc` prend les arguments clés et de valeurs suivants :

`log_level` Niveau minimum auquel transmettre les messages Lint. Les messages Lint inférieurs à ce niveau sont ignorés. La valeur par défaut est `INFO`.

Les niveaux de journal dans l'ordre du moins grave au plus grave sont `DEBUG`, `INFO`, `WARNING`, `ERROR` et `CRITICAL`.

`do_pub_checks` Si `True`, effectue des vérifications qui pourraient n'avoir de sens que pour les packages publiés. La valeur par défaut est `True`.

`pkglint.ext.*`	Le mécanisme de plug-in de `pkglint` permet l'ajout d'autres modules Lint au moment de l'exécution. Toute clé démarrant avec `pkglint.ext.` prend une valeur qui doit être un module Python entièrement spécifié. Reportez-vous à la section Développeurs pour de plus amples informations.
`pkglint.exclude`	Liste séparée par des espaces de modules Python entièrement spécifiés, de classes ou des noms de fonctions à omettre dans l'ensemble des vérifications effectuées.
`use_progress_tracker`	Si `True`, utilise un outil de suivi de progression lors de l'itération sur des manifestes pendant les exécutions Lint. La valeur par défaut est `True`.
`version.pattern`	Motif de version utilisé lors de la spécification d'un numéro de version par rapport à laquelle effectuer une opération Lint (`-b`). S'il n'est pas spécifié dans le fichier de configuration, l'option `-b` utilise le motif `*,5.11-0.`, correspondant à tous les composants de la version 5.11, avec un préfixe de branche de 0.

Développeurs Pour étendre l'ensemble des vérifications effectuées par `pkglint`, la sous-classe `pkg.lint.base.Checker` et ses sous-classes, `ManifestChecker`, `ActionChecker` et `ContentChecker`. Ajoutez le nom de module Python qui contient ces classes pour une nouvelle clé `pkglint.ext.` dans le fichier de configuration.

Des instances de ces nouvelles sous-classes sont créées par `pkglint` au démarrage. Des méthodes à l'intérieur de chaque sous-classe avec l'argument de mot-clé spécial `pkglint_id` sont appelées au cours de la session Lint. Ces méthodes doivent avoir la même signature que la méthode `check()` correspondante dans la super-classe. Les méthodes doivent également se voir affecter un attribut `pkglint_desc`, qui est utilisé comme la description imprimée par `pkglint -L`.

Les paramètres sont disponibles aux sous-classes `Checker`, ce qui leur permet de régler leur comportement. La convention de nommage de paramètre recommandée est *pkglint_id*.*name*. Les valeurs de paramètre peuvent être stockées dans le fichier de configuration, ou peuvent être accédées depuis les manifestes ou actions récupérés à l'aide de la méthode `LintEngine.get_param()`. Lors de l'accès aux paramètres à partir du manifeste, le préfixe `pkg.lint` est ajouté au début du nom de la clé pour garantir que les paramètres `pkglint` ne se superposent pas à n'importe quelle valeur d'action ou de manifeste existante.

Exemples **EXEMPLE 1** Première exécution sur un référentiel particulier

Exécution d'une `pkglint` pour la première fois sur un référentiel donné.

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

EXEMPLE 2 Exécution suivante sur le même référentiel

Une exécution suivante avec le même référentiel utilisé dans l'exemple 1.

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

EXEMPLE 3 Utilisation d'un référentiel Lint avec un ensemble de manifestes limité

Exécution d'une session `pkglint` avec un référentiel Lint et spécification d'un sous-ensemble de manifestes à vérifier.

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

EXEMPLE 4 Spécification d'une version

Exécution d'une session `pkglint` par rapport à une version donnée en mode détaillé.

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

EXEMPLE 5 Modification d'un fichier de configuration

Un fichier de configuration avec un nouveau module Lint, excluant certaines vérifications.

```
$ 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
```

État de sortie

Les valeurs de sortie renvoyées sont les suivantes :

`0` La commande a réussi.

`1` Une ou plusieurs sorties émises par des vérifications Lint.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

`99` Une exception imprévue est survenue.

Attributs

Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	package/pkg
Stabilité de l'interface	Non validé

Voir aussi pkg(1), pkg.depotd(1m), pkgsend(1), pkg(5)

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

Nom pkgmerge – Utilitaire de fusion de package Image Packaging System

Synopsis

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

Description `pkgmerge` est un outil de publication de package permettant la création de packages à variantes multiples. Il effectue cette opération en fusionnant les packages dont les noms et les versions sont identiques (à l'exclusion de l'horodatage), en balisant les actions qui sont uniques dans les versions en cours de fusion avec le nom et la valeur de la variante spécifiée pour la source donnée, puis en publiant les nouveaux packages dans le référentiel cible. Seule la version la plus récente de chaque package de chaque source est utilisée.

Si une action a l'attribut `pkg.merge.blend` défini sur le nom de la variante en cours de fusion, cette action est copiée dans l'autre manifeste avant la fusion afin que l'action s'affiche sans balise de variante ajouté dans la sortie finale. Notez que l'attribut `pkg.merge.blend` lui-même est supprimé de toutes les actions dans le manifeste de sortie. Cet attribut peut être répété avec des valeurs différentes pour des fusions à plusieurs passes.

Les actions non identiques qui fournissent au même chemin dans un résultat de manifeste d'entrée entraîne la sortie de `pkgmerge` avec une erreur.

Options Les options suivantes sont prises en charge :

`-d` *dest_repo*
: Chemin de système de fichiers ou URI du référentiel cible vers lequel publier les packages fusionnés. Le référentiel cible doit déjà exister. Les nouveaux référentiels peuvent être créés à l'aide de `pkgrepo`(1).

`-n`
: Effectue une série de tests sans apporter de modifications au référentiel cible.

`-s` *variant= value*[,...],*src_repo*
: Nom et la valeur de la variante à utiliser pour les packages de cette source, suivis par le chemin de système de fichiers ou l'URI du référentiel source ou l'archive de package à partir desquels récupérer les packages. Plusieurs variantes peuvent être spécifiées en les séparant par des virgules. Les mêmes variantes doit être nommées pour toutes les sources. Cette option peut être spécifiée plusieurs fois.

`--help` ou `-?`
: Affiche un message d'utilisation.

Variables d'environnement La variable d'environnement suivante est prise en charge :

`TMPDIR` Chemin absolu au répertoire dans lequel les données temporaires doivent être stockées lors de l'exécution du programme. Si elle n'est pas définie, la valeur par défaut est de stocker des données temporaires dans `/var/tmp`.

Exemples

EXEMPLE 1 Spécification du nom et de la valeur de la variante

Marque chaque package trouvé dans la source indiquée avec le nom et la valeur de la variante donnée pour la source à partir de laquelle il a été récupéré.

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

Exemple de package :

```
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
```

Exemple de package après l'opération :

```
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
```

EXEMPLE 2 Fusion et publication de packages

Fusionnez la version la plus récente de chaque package des sources données et publiez les nouveaux packages dans le référentiel cible :

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

Exemple de package depuis la source 1 (SPARC) :

```
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
```

Exemple de package depuis la source 2 (i386) :

```
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
```

Package fusionné :

```
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
```

EXEMPLE 3 Fusion de packages avec débogage et sans débogage pour les systèmes i386 et SPARC

Fusion de la version la plus récente de chaque package dans un ensemble de référentiels avec débogage et sans débogage pour les systèmes i386 et SPARC :

EXEMPLE 3 Fusion de packages avec débogage et sans débogage pour les systèmes i386 et SPARC *(Suite)*

```
$ 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
```

Exemple de package depuis la source 1 (SPARC sans débogage) :

```
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
```

Exemple de package depuis la source 2 (SPARC avec débogage) :

```
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
```

Exemple de package depuis la source 3 (i386 sans débogage) :

```
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
```

Exemple de package depuis la source 4 (i386 avec débogage) :

```
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
```

Package fusionné :

```
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
```

EXEMPLE 4 Fusion à l'aide de pkg.merge.blend

Fusion de packages pour deux architectures qui ne se heurtent pas, à l'aide de l'attribut `pkg.merge.blend`.

EXEMPLE 4 Fusion à l'aide de pkg.merge.blend *(Suite)*

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

Exemple de package depuis la source 1 (SPARC) :

```
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
```

Exemple de package depuis la source 2 (i386) :

```
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
```

Package fusionné :

```
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
```

État de sortie Les valeurs de sortie renvoyées sont les suivantes :

`0` La commande a réussi.

`1` Une erreur s'est produite.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

`99` Une exception imprévue est survenue.

Attributs Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	package/pkg
Stabilité de l'interface	Non validé

Voir aussi pkgrepo(1), pkg(5)

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

Nom pkgmogrify – Transmogrifeur de manifeste Image Packaging System

Synopsis

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

Description Lors de l'édition de manifestes de package, `pkgmogrify` simplifie les transformations normales nécessaires à l'automatisation de la génération de logiciel et de la publication de package.

`pkgmogrify` fournit les éléments suivants :

- Remplacement de macro facilitant le partage d'un seul manifeste d'une architecture ou d'une plate-forme à l'autre
- Inclusion d'autres manifestes ou fragments de manifeste comme composants et transformations standard.
- Transformation d'actions de package, incluant la modification, la suppression ou l'ajout d'attributs d'action.

Options Les options suivantes sont prises en charge :

`-D` *name= value*
Définit *name* comme une macro, avec la valeur *value*. Les macros s'affichent dans le fichier d'entrée au format `$(macro)`. La substitution est répétée jusqu'à ce qu'il n'y ait plus de traductions trouvées. Voici des idiomes courants :

- Elimination de lignes d'un manifeste sur d'autres architectures à l'aide d'une balise spécifique à une architecture au début de la ligne :

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

  Lors du traitement de l'architecture SPARC, cette macro est une chaîne vide. Lors du traitement d'autres architectures, cette macro doit être définie sur # sur la ligne de commande, ce qui élimine cette action dans le manifeste sur l'architecture actuelle.
- Indication spécifique à la plate-forme des parties des noms de chemins, telles que le nom du répertoire de l'architecture 64 bits pour des exécutables et des bibliothèques :

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

  Ces macros doivent être définies sur la valeur souhaitée dans la ligne de commande. Il n'y a aucune valeur de macro prédéfinie.

`-I` *include_directory*
Ajoute le répertoire spécifié au chemin de recherche pour les deux fichiers spécifiés sur la ligne de commande et les directives include intégrées.

`-O` *outputfile*
Ecrit la sortie de manifeste dans le fichier spécifié. Le fichier n'est pas écrit si une erreur se produit ou si une directive de transformation entraîne une opération d'abandon. Par défaut, la sortie de manifeste est écrite dans `stdout`.

`-P` *printfile*
Consigne les messages de sortie résultant des opérations d'impression de directive de transformation dans le fichier spécifié. Le fichier n'est pas écrit si une erreur se produit ou si une directive de transformation entraîne une opération d'abandon. Par défaut, la sortie d'impression est écrite dans `stdout`.

`-i`
Ignore les directives include dans les fichiers. Seuls les fichiers spécifiés sur la ligne de commande (ou `stdin`) sont traités.

`-v`
Rédige des commentaires dans le manifeste de sortie montrant l'effet des transformations. Ces informations sont utiles à des fins de débogage.

`--help` ou `-?`
Affiche un message d'utilisation.

Directives intégrées

Deux types de directives sont pris en charge dans les fichiers manifestes : les directives include et les directives transform.

Les directives include se présentent sous la forme suivante :

`<include` *file*`>`

Cette directive pousse `pkgmogrify` à rechercher un fichier nommé `file` tout d'abord dans le répertoire en cours, puis dans les répertoires indiqués avec l'option `-I`. Si le fichier est trouvé, le contenu du fichier est inséré dans le manifeste à l'endroit où la directive est détectée. S'il n'est pas trouvé, `pkgmogrify` s'arrête et un message d'erreur s'affiche.

Les directives transform se présentent sous la forme suivante :

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

Ces directives sont cumulées jusqu'à ce que toutes les entrées soient lues en mémoire, puis appliquées aux actions dans l'ordre dans lequel elles ont été détectées.

Les critères de correspondance se présentent sous la forme suivante :

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

Un des *types d'action* spécifié doit correspondre. Tous les *attributs* spécifiés doivent correspondre. La syntaxe d'expression régulière utilisée est celle de Python. Pour plus d'informations sur la syntaxe d'expression régulière Python, utilisez la commande `pydoc re` ou reportez-vous à une documentation complète à l'adresse suivante : `http://docs.python.org/dev/howto/regex.html`. L'expression régulière est ancrée au début, mais pas à la fin. Par conséquent, une expression régulière correspondant à des fichiers par leur extension doit inclure `.*` au début et `$` à la fin.

Plusieurs critères peuvent être spécifiés en les séparant par des espaces.

Les opérations suivantes sont disponibles :

`add` Ajoute une valeur à un attribut. Cette opération prend deux arguments. Le premier argument est le nom de l'attribut et le second est la valeur.

`default` Définit la valeur d'un attribut s'il n'existe pas encore. Cette opération prend les deux mêmes arguments que l'opération `add`.

`delete` Supprime des valeurs d'attribut. Cette opération prend deux arguments. Le premier argument est le nom de l'attribut. Le second argument est une expression régulière correspondant aux valeurs d'attribut supprimées. Contrairement à l'expression régulière utilisée pour correspondre à une action, cette expression n'est pas ancrée.

`drop` Annule cette action.

`edit` Modifie un attribut de l'action. Cette opération prend trois arguments. Le premier argument est le nom de l'attribut et le deuxième est une expression régulière correspondant à la valeur de l'attribut. Le troisième argument est la chaîne de remplacement qui se substitue à la partie de la valeur correspondant à l'expression régulière. Contrairement à l'expression régulière utilisée pour correspondre à une action, cette expression n'est pas ancrée. Des références arrière d'expressions régulières normales, sous la forme `\1`, `\2`, et ainsi de suite, sont disponibles dans la chaîne de remplacement si des groupes sont définis dans l'expression régulière.

`emit` Emet une ligne dans le flux de sortie du manifeste. Il doit s'agir d'une chaîne d'action valide, vide (ce qui se traduit par une ligne vierge) ou un commentaire (un # suivi d'un texte arbitraire).

`exit` Arrête le traitement du manifeste. Aucun manifeste n'est produit et aucune opération `print` n'est appliquée. Si l'un des arguments est fourni, il doit s'agir d'un entier, et est utilisé en tant que code de sortie. La valeur par défaut est 0. Si deux arguments sont fournis, le premier est le code de sortie et le second est un message à imprimer dans `stderr`.

`print` Imprime un message dans le fichier de sortie spécifié avec `-P`.

`set` Définit la valeur d'un attribut. Cette opération prend les deux mêmes arguments que l'opération `add`.

Toutes les opérations, à l'exception de `delete` et `drop` prennent des arguments, pouvant être facultatifs, dont le contenu va dans le flux de sortie. Ces chaînes peuvent contenir trois types différents de jetons spéciaux qui permettent à la sortie de contenir des informations qui ne sont pas basées sur une transformation fixe de chaque action.

Le premier type de substitution permet à l'opération de faire référence aux valeurs d'attributs de l'action en cours en plaçant le nom de l'attribut à l'intérieur des parenthèses suivant un signe de pourcentage. Par exemple, `%(alias)` fait référence à la valeur de l'action de l'attribut `alias`.

Il existent plusieurs attributs synthétiques. Deux sont uniques à `pkgmogrify` :

- `pkg.manifest.filename` fait référence au nom du fichier dans lequel l'action a été trouvée.
- `pkg.manifest.lineno` fait référence à la ligne sur laquelle l'action a été trouvée.

Trois attributs synthétiques sont similaires à ceux utilisés dans `pkg`(1) :

- `action.hash` fait référence à la valeur de hachage de l'action si l'action dispose d'une charge utile. Dans le cadre d'actions avec des charges utiles, les opérations `set` peuvent modifier le hachage de l'action en agissant sur l'attribut `action.hash`.
- `action.key` fait référence à la valeur de l'attribut clé.
- `action.name` fait référence au nom du type d'action.

Si l'attribut dont la valeur est demandée n'existe pas, `pkgmogrify` s'arrête et un message d'erreur s'affiche. Pour empêcher que sortie avec erreur, faites suivre le nom de l'attribut par `;notfound=` et une valeur de remplacement à la place de la valeur de l'attribut. Par exemple, `%(alias;notfound='no alias')` imprime la valeur de l'attribut `alias` si elle existe, et imprime `no alias` dans le cas contraire.

Si l'attribut dont la valeur est demandée est multivalué, chaque valeur est imprimée, séparée par des espaces. Tout comme le jeton `notfound`, les jetons `prefix`, `suffix` et `sep` peuvent être utilisés pour modifier ce comportement. La chaîne indiquée par `prefix` est ajoutée au début de chaque valeur, la chaîne indiquée par `suffix` est ajoutée à chaque valeur, et `sep` est placé entre le suffixe d'une valeur et le préfixe de la suivante.

Tout comme les attributs d'action, les directives `pkgmogrify` peuvent référencer des attributs de package à l'aide d'accolades au lieu de parenthèses : `%{pkg.fmri}`. A l'endroit où la directive transform est appliquée, l'attribut doit avoir été défini dans une action `set` ou il est considéré comme `notfound`, comme décrit plus haut. Lorsque le traitement a atteint la fin du fichier manifeste décrivant le package, les attributs sont effacés pour le package suivant.

Cela est utile non seulement pour référencer les attributs de package comme s'ils étaient des attributs d'action, mais également pour faire les correspondre, et même temporairement les modifier. C'est pour cela qu'un nom d'action synthétique, `pkg`, est disponible (uniquement dans le contexte de `pkgmogrify`) à l'utilisation dans ces situations.

Quand `pkgmogrify` a terminé la lecture d'un manifeste spécifié sur la ligne de commande et que ce manifeste a défini un attribut de package `pkg.fmri`, `pkgmogrify` crée cette action `pkg` synthétique, dont les attributs sont les attributs du package. Une directive `<transform>` peut ensuite correspondre sur cette action, tout comme n'importe quel autre type d'action.

Les opérations effectuées sur une action `pkg` sont spéciales car elles ont lieu uniquement dans la mémoire et n'affectent pas directement le manifeste émis. Par exemple, essayer de définir un attribut sur une action `pkg` par l'intermédiaire des opérations `add`, `default` ou `set` opérations n'entraîne pas l'ajout d'une action `set` au manifeste, bien qu'elle soit disponible pour d'autres

directives <transform> auxquelles correspondre. Toute tentative d'opération emit sur une action pkg provoque une erreur. Pour ajouter un attribut de package, utilisez plutôt emit sur une action set.

Le troisième type de substitution est une référence arrière. Cette substitution n'est pas comme celles utilisables dans les opérations edit, mais est une référence à des groupes répertoriés dans la correspondance de transformation dans la partie de gauche de la ->. Ceux-ci sont indiqués par %<1>, %<2>, et ainsi de suite, dans l'ordre indiqué dans la zone de critères de correspondance.

L'ordre de traitement est le suivant :

1. Les lignes sont lues à partir des fichiers d'entrée.
2. Les macros sont appliquées.
3. Les directives <include ...> et <transform> sont traitées, ce qui entraîne la découverte et la lecture de fichiers supplémentaires.
4. Une fois que toutes les entrées ont été accumulées, chaque ligne dans l'entrée est convertie en actions et toutes les transformations sont appliquées.
5. Une fois le traitement terminé et réussi, la sortie est écrite.

Exemples

EXEMPLE 1 Ajout de balises aux manifestes SMF

Ajoutez des balises aux manifestes SMF afin qu'ils soient importés lorsque le package est installé sur un système actif.

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

EXEMPLE 2 Déplacement de fichiers

Déplacez des fichiers de usr/sfw/bin vers usr/bin.

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

EXEMPLE 3 Spécification d'une initialisation nécessaire

Ajoutez les balises reboot-needed aux fichiers sous /kernel qui ne sont pas des fichiers .conf. Notez que cet exemple s'appuie sur l'application des transformations à chaque action dans l'ordre indiqué dans les fichiers d'entrée.

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

Il est également possible de le faire avec une unique règle de transformation avec des expressions régulières.

EXEMPLE 4 Conversion d'attribut FMRI en action depend

Convertissez l'attribut de package `pkg.fmri` en action `depend` pour faire partie d'une incorporation.

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

EXEMPLE 5 Impression d'une liste de numéros de bogue

Imprimez une liste séparée par des virgules de numéros de bogue cités et préfixés.

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

EXEMPLE 6 Autorisation d'attributs manquants

Imprimez un message en toute sécurité même lorsqu'un attribut est manquant.

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

EXEMPLE 7 Définition de valeurs par défaut

Définissez les valeurs par défaut de propriétaire, de groupe et d'autorisation.

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

EXEMPLE 8 Ajout de dépendances à des packages qui ne sont pas marqués comme obsolètes

Pour n'importe quel package qui n'est pas marqué comme obsolète, ajoutez une dépendance sur l'incorporation pour la consolidation qui fournit le package. Cet ensemble de transformations doit se produire après la lecture du manifeste ou la dépendance sera toujours émise. Etant donné que la modification d'une action `pkg` n'a aucun effet permanent, il n'est pas nécessaire de nettoyer les attributs correspondant à `pkg.obsolete=false`.

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

EXEMPLE 9 Sortie et impression d'un message lorsqu'une erreur est détectée

Sortie en erreur avec un message lorsqu'un attribut obsolète est découvert dans un manifeste.

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

EXEMPLE 10 Définition de la facette de paramètres régionaux appropriée

Définissez la facette de paramètres régionaux appropriée pour le nom de chemin considéré.

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

État de sortie Les valeurs de sortie renvoyées sont les suivantes :

`0` Tout a fonctionné.

`1` Quelque chose de mauvais mais prévu s'est passé.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

`99` Erreur de traitement inattendue.

Fichiers

`/usr/share/pkg/transforms`	Ce répertoire contient des fichiers avec des transformations utiles pour définir des facettes, des actionneurs et autres attributs.

Attributs Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg`
Stabilité de l'interface	Non validé

Voir aussi pkg(1), pkg(5)

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

Nom pkgrecv – Utilitaire de récupération de contenu Image Packaging System

Synopsis

```
/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
```

Description `pkgrecv` permet à l'utilisateur de récupérer des packages à partir d'un référentiel `pkg`(5) ou d'une archive de packages. `pkgrecv` peut également publier de nouveau les packages récupérés vers un autre référentiel de packages ou les archiver. Par défaut, les packages sont récupérés au format de référentiel de package approprié pour une utilisation avec `pkg`(1), `pkg.depotd`(1M) et des outils de publication de packages.

Après une opération `pkgrecv`, exécutez `pkgrepo refresh` ou `pkgrepo rebuild` sur le référentiel pour construire des index de recherche.

Options Les options suivantes sont prises en charge :

`-a` Stocke les données du package récupéré dans une archive `pkg`(5) à l'emplacement spécifié par `-d`. Le fichier ne doit pas déjà exister. Cette option peut être utilisée uniquement avec des destinations de système de fichiers. Bien que cela ne soit pas nécessaire, l'utilisation d'une extension de fichiers `.p5p` (par exemple, `archive.p5p`) est vivement recommandée. Cette option ne peut pas être combinée à `--raw`.

`-c` *cache_dir* Chemin d'un répertoire qui sera utilisé pour la mise en cache du contenu téléchargé. Si ce répertoire n'est pas fourni, le client sélectionne automatiquement un répertoire de cache. Dans le cas où un téléchargement est interrompu et qu'un répertoire de cache a été automatiquement sélectionné, utilisez cette option pour reprendre le téléchargement. Reportez-vous à la section Variables d'environnement ci-dessous pour plus de détails sur la manière de définir l'emplacement utilisé pour le stockage temporaire des données.

`-d` *path_or_uri* Chemin de système de fichiers ou URI de la cible vers lesquels republier les packages. Si `-a` est spécifié, la cible est une nouvelle archive de package (qui n'existe pas encore). Autrement, la cible doit être un référentiel de packages qui existe déjà. Les nouveaux référentiels peuvent être créés à l'aide de `pkgrepo`(1).

`-h` Affiche un message d'utilisation.

`-k` Conserve le contenu du package récupéré compressé. Cette option est ignorée en cas de nouvelle publication. Le contenu du package compressé ne doit pas être utilisé avec `pkgsend`(1).

`-m` *match* Contrôle le comportement de correspondance à l'aide des valeurs suivantes :

`all-timestamps`	Inclut tous les horodatages correspondants et pas seulement le dernier (implique `all-versions`).
`all-versions`	Inclut toutes les versions correspondantes, et pas seulement la dernière.

`-n` Effectue une série de tests sans apporter de modifications.

`-r` Récupère de manière récursive toutes les dépendances de la liste des packages fournie.

`-s` *src_repo_uri* URI représentant l'emplacement d'un référentiel `pkg`(5) ou d'une archive de packages à partir desquels recevoir des données de package.

`--cert` *file* Spécifie un fichier de certificat SSL client à utiliser pour la récupération de package à partir d'un référentiel HTTPS.

`--key` *file* Spécifie un fichier clé SSL client à utiliser pour la récupération de package à partir d'un référentiel HTTPS.

`--newest` Répertorie les dernières versions des packages disponibles depuis le référentiel, puis quitte. (Toutes les autres options, à l'exception de `-s`, sont ignorées.)

`--raw` Récupère et stocke les données brutes de package dans un ensemble de structures de répertoire par racine et version à l'emplacement spécifié par `-d`. Cette option peut être utilisée uniquement avec des destinations de système de fichiers. Ces données de package peuvent être utilisées pour modifier et publier à nouveau des packages de manière commode, par exemple en corrigeant des contenus de fichiers ou en fournissant des métadonnées de package supplémentaires. Cette option ne peut pas être combinée à `-a`.

Exemples

EXEMPLE 1 Liste des packages les plus récents

Répertorie les packages les plus récents disponibles à partir du référentiel sur le système nommé `test`.

```
$ 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
```

EXEMPLE 2 Récupération des données de package brutes

Recevez le package `c++-runtime` de l'exemple 1 dans un format adapté pour une utilisation avec `pkgsend publish`.

```
$ pkgrecv -s http://test \
-d /local/repo --raw \
```

EXEMPLE 2 Récupération des données de package brutes *(Suite)*

```
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
```

EXEMPLE 3 Récupération des dépendances d'un système

Recevez le package `editor/vim` et toutes ses dépendances à partir du système nommé `test`.

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

EXEMPLE 4 Récupération de toutes les versions

Recevez toutes les versions du package `editor/vim` depuis le système nommé `test`.

```
$ 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
```

EXEMPLE 5 Récupération de toutes les versions et republication à distance

Recevez toutes les versions du package `library/zlib` depuis le système nommé `test` et publiez-les de nouveau vers un référentiel distant nommé `remote`.

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

EXEMPLE 6 Récupération des dépendances d'un référentiel

Recevez le package `editor/gnu-emacs` et toutes ses dépendances à partir du référentiel situé dans `/export/repo`.

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

EXEMPLE 7 Récupération de packages supplémentaires

Recevez tous les packages qui n'existent pas déjà dans le référentiel à l'emplacement `http://example.com:10000`.

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

EXEMPLE 8 Création d'une archive de packages

Créez une archive de packages contenant le package `editor/gnu-emacs` et toutes ses dépendances à partir du référentiel situé dans `http://example.com:10000`.

EXEMPLE 8 Création d'une archive de packages *(Suite)*

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

EXEMPLE 9 Copie de packages d'une archive vers un référentiel

Copiez tous les packages dans une archive de packages vers un référentiel existant situé dans `/export/repo`.

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

Variables d'environnement

Les variables d'environnement suivantes sont prises en charge :

`PKG_DEST` Chemin d'un répertoire pour enregistrer le package récupéré, ou le chemin système fichiers ou l'URI d'un référentiel ou d'une archive de packages où les packages seront copiés.

`PKG_SRC` URI ou chemin système fichiers représentant l'emplacement d'un référentiel `pkg`(5) ou d'une archive de packages à partir desquels récupérer des packages.

`TMPDIR` Chemin absolu au répertoire dans lequel les données temporaires doivent être stockées lors de l'exécution du programme. Si elle n'est pas définie, la valeur par défaut est de stocker des données temporaires dans `/var/tmp`.

État de sortie

Les valeurs de sortie renvoyées sont les suivantes :

`0` La commande a réussi.

`1` Une erreur s'est produite.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

`3` Plusieurs opérations ont été demandées, mais seules certaines d'entre elles ont réussi.

`99` Une exception imprévue est survenue.

Attributs

Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg`
Stabilité de l'interface	Non validé

Voir aussi

`pkgrepo`(1), `pkgsend`(1), `pkg`(5)

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

Nom pkgrepo – Utilitaire de gestion des référentiels Image Packaging System

Synopsis

```
/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
```

Description `pkgrepo` offre la possibilité de créer et de gérer des référentiels de packages `pkg`(5). Un référentiel de packages un ensemble prédéfini de répertoires et de fichiers qui autorise le stockage et la récupération des données de package par `pkg`(1) et des clients de publication comme `pkgsend`(1) ou `pkgrecv`(1). De plus, lorsqu'un accès réseau à un référentiel de packages est nécessaire, `pkg.depotd`(1m) peut fournir aux clients un accès au référentiel pour stocker et/ou récupérer les données de package.

Options Les options suivantes sont prises en charge :

`--help` ou `-?` Affiche un message d'utilisation.

Sous-commandes Les sous-commandes suivantes sont prises en charge :

`create [--version` *ver*`]` *uri_or_path*
Crée un référentiel `pkg`(5) à l'emplacement spécifié.

Cette sous-commande peut être utilisée uniquement avec des référentiels sur le système de fichiers.

Avec `--version`, créez un référentiel dans un format compatible avec la version spécifiée. Par défaut, des référentiels de version 4 sont créés. Les versions prises en charge sont :

3 Prend en charge le stockage des packages pour un unique éditeur, catalogue version 1 et recherche version 1.

4 Prend en charge le stockage des packages pour plusieurs éditeurs, catalogue version 1 et recherche version 1.

`add-publisher -s` *repo_uri_or_path publisher* `...`
Ajoute les éditeurs spécifiés au référentiel. Les nouveaux éditeurs n'ont aucun package ou contenu.

Cette sous-commande peut être utilisée uniquement avec des référentiels de version 4 sur le système de fichiers.

`get [-F` *format*`] [-p` *publisher* `...] -s` *repo_uri_or_path* `[`*section/property* `...]`
Affiche les informations de propriété pour le référentiel ou ses éditeurs.

Par défaut, chaque propriété et sa valeur sont imprimées sur des lignes distinctes. Les valeurs de chaîne ASCII vides sont représentées par une paire de guillemets doubles (""). Les métacaractères, nouvelle ligne, espace et onglet de shell Bourne suivants dans les valeurs de chaîne ASCII doivent être neutralisés par des barres obliques inverses (\) :

```
; & ( ) | ^ < > \ " ’ ‘
```

Reportez-vous à la section Exemples.

Pour obtenir la liste des propriétés possibles et l'objectif et la valeur de chaque propriété, reportez-vous à la sous-commande `set` ci-dessous.

Avec `-F`, spécifie un autre format de sortie. La valeur de *format* peut être `tsv` (valeurs séparées par des virgules), `json` (notation objet JavaScript sous la forme d'une seule ligne) ou `json-formatted` (notation objet JavaScript, formatée pour une meilleure lisibilité).

Avec `-H`, omet les en-têtes de la liste.

Avec `-p`, affiche les informations de propriété pour l'éditeur donné. La valeur spéciale `all` affiche les propriétés de tous les éditeurs. Cette option peut être spécifiée plusieurs fois.

Avec `-s`, fonctionne sur le référentiel situé à l'URI ou le chemin système de fichiers indiqué.

`info [-F` *format*`] [-H] [-p` *publisher* `...] - s` *repo_uri_or_path*
Affiche une liste des éditeurs package connus par le référentiel. La liste comprend le nombre de packages pour chaque éditeur, quand les données de package de l'éditeur a été mises à jour pour la dernière fois, et l'état des données de package de l'éditeur (par exemple, si elle sont en cours de traitement).

Avec `-F`, spécifie un autre format de sortie. La valeur de *format* peut être `tsv` (valeurs séparées par des virgules), `json` (notation objet JavaScript sous la forme d'une seule ligne) ou `json-formatted` (notation objet JavaScript, formatée pour une meilleure lisibilité).

Avec `-H`, omet les en-têtes de la liste.

Avec `-p`, n'affiche que les données pour l'éditeur donné. Sans p, les données de tous les éditeurs sont affichées. Cette option peut être spécifiée plusieurs fois.

Avec `-s`, fonctionne sur le référentiel situé à l'URI ou le chemin système de fichiers indiqué.

`list [-F` *format*`] [-H] [-p` *publisher* `...] - s` *repo_uri_or_path* `[`*pkg_fmri_pattern* `...]`
Répertorie les packages dans le référentiel *repo_uri_or_path* qui correspond aux motifs *pkg_fmri_pattern* spécifiés. Si aucun motif n'a été spécifié, tous les packages du référentiel sont répertoriés.

Dans la sortie par défaut, la première colonne contient le nom de l'éditeur du package. La deuxième colonne contient le nom du package. La troisième colonne est un indicateur de l'état d'un package. Une valeur de `o` dans la colonne d'état indique que le package est obsolète. Une valeur de `r` dans la colonne d'état indique que le package a été renommé, ce qui est une forme d'obsolescence. La quatrième colonne contient la version et la branche du package. Reportez-vous à `pkg`(5) pour plus d'informations concernant la version et la branche.

Avec `-F`, spécifie un autre format de sortie. La valeur de *format* peut être `tsv` (valeurs séparées par des virgules), `json` (notation objet JavaScript sous la forme d'une seule ligne) ou `json-formatted` (notation objet JavaScript, formatée pour une meilleure lisibilité).

Avec `-H`, omet les en-têtes de la liste.

Avec `-p`, n'affiche que les packages pour l'éditeur donné. Sans p, les packages de tous les éditeurs sont affichés. Cette option peut être spécifiée plusieurs fois.

Avec `-s`, fonctionne sur le référentiel situé à l'URI ou le chemin système de fichiers indiqué.

`rebuild [-p` *publisher* `...] - s` *repo_uri_or_path* `[--no-catalog] [--no-index]`
Rejette tout catalogue, recherche et autres informations en mémoire cache trouvés dans le référentiel, puis les recrée sur la base du contenu actuel du référentiel.

Avec `-p`, effectue l'opération uniquement pour l'éditeur donné. Sans p, ou si la valeur spéciale `all` est spécifiée, l'opération est effectuée pour tous les éditeurs. Cette option peut être spécifiée plusieurs fois.

Avec `-s`, fonctionne sur le référentiel situé à l'URI ou le chemin système de fichiers indiqué.

Avec `--no-catalogue`, ne reconstruit pas les données de package.

Avec `--no-index`, ne reconstruit pas les index de recherche.

`refresh [-p` *publisher* `...] - s` *repo_uri_or_path* `[--no-catalog] [--no-index]`
Catalogue les nouveaux packages trouvés dans le référentiel et met à jour tous les index de recherche. Elle est destinée à être utilisée avec une publication différée (options `--no-catalog` ou `--no-index` de `pkgsend`).

Avec `-p`, effectue l'opération uniquement pour l'éditeur donné. Sans p, ou si la valeur spéciale `all` est spécifiée, l'opération est effectuée pour tous les éditeurs. Cette option peut être spécifiée plusieurs fois.

Avec `-s`, fonctionne sur le référentiel situé à l'URI ou le chemin système de fichiers indiqué.

Avec `--no-catalog`, n'ajoute pas de nouveaux packages.

Avec `--no-index`, ne met pas à jour les index de recherche.

`remove [-n] [-p` *publisher* `...] -s` *repo_uri_or_path pkg_fmri_pattern* `...`
Supprime les packages correspondant aux motifs indiqués du référentiel, y compris les fichiers qu'ils référencent qui ne sont pas en cours d'utilisation par d'autres packages.

Remarque – Toutes les données d'index de recherche des éditeurs relatifs sont supprimées.

Cette sous-commande peut être utilisée uniquement avec des référentiels sur le système de fichiers.

Attention – Cette opération n'est pas réversible et ne doivent pas être utilisés pendant que d'autres clients accèdent au référentiel car elle peut entraîner leur échec pendant les opérations de récupération.

Avec `-n`, effectue un essai de l'opération sans aucune modification de package. Une liste des packages à supprimer s'affiche avant la fermeture de l'application.

Avec `-p`, supprime uniquement les packages correspondants à l'éditeur donné. Sans p, les packages correspondants sont supprimés pour tous les éditeurs. Cette option peut être spécifiée plusieurs fois.

Avec `-s`, fonctionne sur le référentiel situé à l'URI ou le chemin système de fichiers indiqué.

`set [-p` *publisher*`] - s` *repo_uri_or_path section/property* `=[`*value*`] ... or` *section/property* `=([`*value*`]) ...`
Définit la valeur des propriétés spécifiées pour le référentiel ou l'éditeur.

Cette sous-commande peut être utilisée uniquement avec des référentiels sur le système de fichiers.

Avec `-p`, définit uniquement les données de propriété pour l'éditeur donné. Si l'éditeur n'existe pas déjà, il est ajouté. La valeur spéciale `all` peut être utilisée pour définir la propriété pour tous les éditeurs.

Avec `-s`, fonctionne sur le référentiel situé à l'URI ou le chemin système de fichiers indiqué.

Les propriétés et les valeurs peuvent être spécifiées à l'aide d'une des formes suivantes :

section/*property*=	Efface la valeur de propriété.
section/*property*= *value*	Remplace la valeur de propriété par la valeur donnée.

section/*property*=(*value1 value2 valueN*) Remplace la valeur de propriété par la liste de valeurs.

Pour les référentiels de versions 3 et 4, les propriétés suivantes peuvent être définies pour le référentiel :

`publisher/prefix` Chaîne qui représente le nom de l'éditeur par défaut. Le premier caractère doit être compris entre a-z, A-Z ou 0-9. Le reste de la chaîne ne peut contenir que des caractères compris entre 0-9, -, ., a-z ou A-Z. Cette valeur indique l'éditeur à utiliser en présence de plusieurs packages de l'éditeur ou lorsque les packages sont publiés dans le référentiel et qu'un éditeur n'est pas spécifié.

Pour les référentiels de versions 3 et 4, les propriétés suivantes peuvent être définies pour des éditeurs individuels dans le référentiel :

`publisher/alias` Chaîne qui représente l'alias par défaut que les clients doivent utiliser lors de l'ajout d'un éditeur à l'aide des données de configuration du référentiel. Le premier caractère doit être compris entre a-z, A-Z ou 0-9. Le reste de la chaîne ne peut contenir que des caractères compris entre 0-9, -, ., a-z ou A-Z.

`repository/collection_type` Peut avoir la valeur `core` ou `supplemental`, indiquant le type de packages proposé dans ce référentiel.

Le type `core` indique que le référentiel contient toutes les dépendances déclarées par des packages dans le référentiel. Le type `core` est principalement utilisé pour les référentiels de système d'exploitation.

Le type `supplemental` indique que le référentiel contient des packages qui s'appuient sur ou sont destinés à être utilisés avec des packages se trouvant dans un autre référentiel.

`repository/description` Paragraphe de texte brut qui décrit l'objectif et le contenu du référentiel.

`repository/detailed_url` URI qui représente l'emplacement d'un document (tel qu'une page web) qui fournit des informations supplémentaires sur le référentiel.

`repository/legal_uris` Liste d'emplacements (URI) pour des documents qui fournissent des informations légales supplémentaire sur le référentiel.

`repository/mirrors`	Liste d'emplacements (URI) de référentiels contenant une copie du contenu du package du référentiel mais pas les métadonnées du package.
`repository/name`	Chaîne de texte brut qui contient le nom du référentiel.
`repository/origins`	Liste d'emplacements (URI) de référentiels contenant une copie complète du contenu et des métadonnées de package du référentiel.
`repository/refresh_seconds`	Nombre entier représentant le nombre de secondes que les clients doivent attendre avant de vérifier le référentiel pour les données de package mis à jour après chaque vérification de mise à jour.
`repository/registration_uri`	URI représentant l'emplacement d'une ressource qui doit être utilisée pour obtenir des informations d'identification pour accéder au référentiel. Une page Web d'enregistrement est un exemple.
repository/related_uris	Liste d'emplacements (URI) de référentiels contenant les packages qui peuvent intéresser les utilisateurs.

Les propriétés non documentée ici, mais répertoriés dans la sortie de la sous-commande `get`, sont réservées à un usage interne et ne doivent pas être définies.

`version`
Affiche une chaîne unique qui identifie la version du système `pkg`(5). Les valeurs générées par l'opération `version` ne peuvent pas être triées et ne sont pas sûres pour la comparaison au-delà de l'égalité.

Exemples

EXEMPLE 1 Création d'un référentiel de packages

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

EXEMPLE 2 Affichage d'informations

Affichez un récapitulatif des éditeurs et le nombre de packages dans un référentiel.

```
$ 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
```

EXEMPLE 3 Reconstruction de catalogues et de données de recherche

Reconstruction des catalogues et des données de recherche du référentiel

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

EXEMPLE 4 Actualisation des catalogues et des données de recherche

Actualisez les catalogues et les données de recherche du référentiel.

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

EXEMPLE 5 Affichage de toutes les propriétés de référentiel

```
$ 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
```

EXEMPLE 6 Affichage de toutes les propriétés d'éditeur

```
$ 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     ()
```

EXEMPLE 7 Configuration de l'éditeur par défaut

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

EXEMPLE 8 Configuration d'une propriété d'éditeur

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

EXEMPLE 9 Ajout d'un nouvel éditeur au référentiel

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

État de sortie

Les valeurs de sortie renvoyées sont les suivantes :

`0` La commande a réussi.

`1` Une erreur s'est produite.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

`3` Plusieurs opérations ont été demandées, mais seules certaines d'entre elles ont réussi.

`4` Aucune modification n'a été faite (rien à faire).

`99` Une exception imprévue est survenue.

Attributs

Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg`
Stabilité de l'interface	Non validé

Voir aussi

`pkg`(1), `pkgrecv`(1), `pkgsend`(1), `pkg.depotd`(1m), `pkg`(5)

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

Nom pkgsend – Client de publication d'Image Packaging System

Synopsis

```
/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 ...]
```

Description `pkgsend` permet de publier de nouveaux packages et nouvelles versions de package vers un référentiel IPS à l'aide de manifestes de package. Pour créer ou gérer des référentiels, reportez-vous à `pkgrepo`(1). Pour créer des archives de packages à partir de packages dans un référentiel existant, reportez-vous à `pkgrecv`(1). Pour plus d'informations sur les manifestes de package, reportez-vous à `pkg`(5).

Après une opération `pkgsend`, exécutez `pkgrepo refresh` ou `pkgrepo rebuild` sur le référentiel pour construire des index de recherche.

Options Les options suivantes sont prises en charge :

`--help` ou `-?` Affiche un message d'utilisation.

Sous-commandes Les sous-commandes suivantes sont prises en charge :

`generate [-T` *pattern*`] [--target` *file*`]` *source* `...`
Lit chaque *source* (telle qu'un package SVR4, un répertoire ou un fichier `tar`) et émet le manifeste qui décrit la *source* à `stdout`. Dans le manifeste de sortie, le propriétaire des actions `file` et `dir` est défini sur `root` et leur groupe sur `bin`.

Vous pouvez ensuite annoter le manifeste de sortie, ajouter ou analyser ses dépendances à l'aide de `pkgdepend`(1) et vérifier son exactitude à l'aide de `pkglint`(1) avant de le transmettre à la sous-commande `publish`.

Les sources suivantes sont prises en charge :

- Packages SVR4 au format système de fichiers
- Packages SVR4 au format de flux de données
- Fichiers `tar`
- Répertoires

Si le nom de base des fichiers dans la source correspond aux motifs spécifiés avec `-T`, l'horodatage du fichier est ajouté à l'action de ce fichier. Le *pattern* utilise les règles de correspondance de shell :

*	Correspond à tous les caractères.
?	Correspond à un caractère unique.
[*seq*]	Correspond à n'importe quel caractère dans *seq*.

`![`*seq*`]` Correspond à n'importe quel caractère ne figurant pas dans *seq*.

Lorsque l'origine indiquée est un répertoire, il n'existe aucun moyen simple de distinguer une action `file` d'une action `hardlink` lorsqu'il existe plusieurs noms de chemins pour un seul inode. Normalement, le premier trouvé dans le parcours du système de fichiers est considéré comme un fichier et le reste comme des liens physiques. Cela peut être arbitraire, en fonction de l'implémentation du système de fichiers. Pour spécifier les noms de chemin qui doivent être traités comme des fichiers, transmettre chaque nom de chemin en tant qu'argument de l'option `--target`. Cette option n'a aucun effet sur d'autres types de sources, car ils sont capables d'exprimer quels noms de chemin sont des fichiers ou des liens physiques.

Quand des packages SVR4 sont fournis en tant que source, `pkgsend` vérifie qu'aucun fichier avec scripts d'action de classe n'est présent et qu'aucun script de pré-installation, de post-installation, de pré-suppression ou de post-suppression n'est présent. Une exception est faite pour les manifestes SMF installés avec la classe `manifeste`. `BASEDIR` est supprimé de tous les chemins réadressables.

Le paramètre `DESC` SVR4 est converti en valeur `pkg.description`. Le paramètre `NAME` SVR4 est converti en valeur `pkg.summary`.

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

Publie un package à l'aide des manifestes de package spécifiés vers le référentiel de packages cible, récupérant des fichiers pour le package de la source fournie. Si plusieurs manifestes sont spécifiés, ils sont joints dans l'ordre indiqué. Si un manifeste n'est pas spécifié, il est lu à partir de `stdin`.

Avec `-b`, ajoute le bundle spécifié à la liste de sources à parcourir lors de la recherche de fichiers dans le manifeste. Les bundles sont des sources telles que les fichiers tar et les packages SVR4. Si cette option est spécifiée plusieurs fois, les sources sont recherchées dans l'ordre dans lequel elles apparaissent sur la ligne de commande. Si `-b` et `-d` sont spécifiés, les sources de `-d` sont recherchées en premier. Pour obtenir une description des bundles pris en charge et de la façon dont ils sont utilisés, reportez-vous à la sous-commande `generate` ci-dessus.

Avec `-d`, ajoute le répertoire spécifié à la liste de sources à parcourir lors de la recherche de fichiers dans le manifeste. Si cette option est spécifiée plusieurs fois, les sources sont recherchées dans l'ordre dans lequel elles apparaissent sur la ligne de commande. Pour obtenir une description des sources prises en charge et de la façon dont elles sont utilisées, reportez-vous à la sous-commande `generate` ci-dessus.

Avec `-s`, publie le package vers le référentiel situé à l'URI ou le chemin système de fichiers indiqué. Reportez-vous à la section Notes ci-dessous pour plus d'informations sur les restrictions et les suggestions de publication. Reportez-vous également à la section Variables d'environnement.

Avec `--no-catalog`, n'ajoute pas le package au catalogue de l'éditeur. Cette option est recommandée lorsque plusieurs packages sont en cours de publication en même temps alors que des mises à jour de catalogues d'éditeur doivent être effectuées en série. Une fois la publication terminée, la sous-commande `refresh` de `pkgrepo` (1) peut être utilisée pour ajouter les nouveaux packages aux catalogues d'éditeurs respectifs.

Pour toutes les autres options, reportez-vous à la sous-commande `generate` ci-dessus pour connaître leur utilisation et leurs effets.

Variables d'environnement

`PKG_REPO` Chemin ou URI du référentiel de destination.

Exemples

EXEMPLE 1 Création et publication d'un package

Créez un package à l'aide de `pkgsend generate` et publiez-le.

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

Ajoutez le package FMRI pour l'éditeur `example.com` au début de `foo.p5m`.

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

Le manifeste qui en résulte doit ressembler à ceci :

```
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
```

EXEMPLE 2 Création et publication d'un package Trivial

Créez un manifeste pour l'éditeur `example.com` contenant les lignes suivantes :

```
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
```

Publiez le package :

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

EXEMPLE 3 Utilisation d'un manifeste existant

Publiez un package à l'aide d'une publication basée sur le système de fichiers et d'un manifeste existant.

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

État de sortie Les valeurs de sortie renvoyées sont les suivantes :

`0` La commande a réussi.

`1` Une erreur s'est produite.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

`99` Une exception imprévue est survenue.

Attributs Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg`
Stabilité de l'interface	Non validé

Voir aussi `pkgdepend`(1), `pkgrepo`(1), `pkg.depotd`(1m), `pkg`(5)

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

Remarques En raison de restrictions de protocole de publication, les publications basées sur le système de fichiers doivent être utilisées lors de la publication de fichiers de package individuels dont la taille est supérieure à 128 Mo. Une publication basée sur le système de fichiers est également recommandée lorsqu'un contrôle d'accès à un référentiel est nécessaire.

Lors de l'utilisation d'une publication basée sur le système de fichiers, tous les processus `pkg.depotd` qui servent le référentiel cible doivent être redémarrés une fois la publication terminée pour que les modifications soient prises en compte dans son interface Web ou les réponses à des recherches. Reportez-vous à `pkg.depotd`(1M) pour plus d'informations.

Nom pkgsign – Utilitaire de signature d'Image Packaging System

Synopsis

```
/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) ...
```

Description `pkgsign` met à jour le manifeste pour le FMRI donné en place dans le référentiel par l'ajout d'une action de signature à l'aide de la clé et des certificats fournis. Le package modifié conserve l'horodatage d'origine.

Options Les options suivantes sont prises en charge :

Avec `-a`, utilise l'algorithme de signature *hash_algorithm* au lieu de la valeur par défaut. L'algorithme de signature par défaut est `rsa-sha256`. Les algorithmes de signature pris en charge sont `rsa-sha256`, `rsa-sha384`, `rsa-sha512`, `sha256`, `sha384`, et `sha512`. Un algorithme de signature qui spécifie uniquement un algorithme de hachage fait que la valeur de signature est le hachage du manifeste de package. Un algorithme de signature qui spécifie `rsa` et d'un algorithme de hachage fait que la valeur de signature est le hachage du manifeste signé avec la clé privée fournie (reportez-vous aux options `-c` et `-k`).

Avec `-c`, ajoute le certificat *path_to_signing_certificate* comme certificat à utiliser lors de la vérification de la valeur de signature dans l'action. L'option `-c` ne peut être utilisée qu'avec l'option `-k`.

Avec `-i`, ajoute le certificat *path_to_intermediate_cert* comme certificat à utiliser lors de la validation du certificat *path_to_signing_certificate* donné sous forme d'argument à `-c`. Plusieurs certificats peuvent être fournis en spécifiant `-i` plusieurs fois.

Avec `-k`, utilise la clé privée stockée dans *path_to_private_key* pour signer le manifeste. L'option `-k` ne peut être utilisée qu'avec l'option `-c`. Si `-k` n'est pas définie, la valeur de signature est le hachage du manifeste.

Avec `-n`, effectue une série de test qui ne modifie en rien le référentiel.

Avec `-s`, signe les packages dans le référentiel à *path_or_uri*.

Avec `--help`, affiche un message d'utilisation.

Avec `--no-index`, ne met pas à jour les index de recherche de référentiel une fois que le manifeste signé a été republié.

Avec `--no-catalog`, ne met pas à jour le catalogue de référentiel une fois que le manifeste signé a été republié.

Exemples

EXEMPLE 1 Signature à l'aide de la valeur de hachage du manifeste

Signez un paquet publié à `http://localhost:10000` à l'aide de la valeur de hachage du manifeste. Souvent utile pour les tests.

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

EXEMPLE 2 Signature à l'aide d'une clé et d'un certificat

Signez un package publié dans le référentiel de fichiers dans `/foo/bar` avec `rsa-sha384` hacher et signer le manifeste. La clé de signature est dans `/key/usr2.key`, son certificat associé est dans `/key/usr2.cert` , et un certificat nécessaire pour valider le certificat est dans `/icerts/usr1.cert` .

```
$ 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
```

État de sortie

Les valeurs de sortie renvoyées sont les suivantes :

`0` La commande a réussi.

`1` Une erreur s'est produite.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

`3` Plusieurs opérations ont été demandées, mais seules certaines d'entre elles ont réussi.

`99` Une exception imprévue est survenue.

Attributs

Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg`
Stabilité de l'interface	Non validé

Voir aussi

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

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

Nom pm-updatemanager – Application pour mettre à jour des packages

Synopsis

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

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

Description `pm-updatemanager` vérifie et installe les mises à jour disponibles pour les packages installés sur le système.

Remarque – Si les packages `package/pkg`, `package/pkg/package-manager` ou `package/pkg/update-manager` doivent être mis à jour, `pm-updatemanager` met d'abord à jour ces packages, puis redémarre pour effectuer d'autres mises à jour.

Options Les options suivantes sont prises en charge :

`-h` ou `--help`	Affiche un message d'utilisation.
`-d` ou `--debug`	Exécute `pm-updatemanager` en mode de débogage.
`-R` ou `--image-dir` *dir*	Fonctionne sur l'image résidant au niveau de *dir*, plutôt que sur celle détectée automatiquement.

Exemples

EXEMPLE 1 Mise à jour de l'image actuelle

Appelez `pm-updatemanager` sur l'image actuelle. Vérifie et installe toutes les mises à jour disponibles pour les packages installés dans l'image actuelle.

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

Il s'agit de la même commande que celle invoquée par l'option du menu du bureau : Système>Administration>Gestionnaire de mises à jour s'ouvre.

EXEMPLE 2 Mise à jour d'une image spécifiée

Appelez `packagemanager` sur l'image stockée sous `/aux0/example_root`.

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

État de sortie Les valeurs de sortie renvoyées sont les suivantes :

`0` Tout a fonctionné.

`1` Une erreur s'est produite.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

Attributs Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg/update-manager`

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Stabilité de l'interface	Non validé

Voir aussi packagemanager(1), pkg(1), pkg(5)

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

Remarques Lorsque vous travaillez sur une image que vous ne possédez pas, `pm-updatemanager` doit être appelé avec suffisamment de privilège. Vous devrez normalement appeler `pm-updatemanager` avec `/usr/lib/pm-launch` dans ces circonstances.

RÉFÉRENCE

Commandes d'administration système

Nom pkg.depotd – Serveur de dépôt Image Packaging System

Synopsis

```
/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]
```

Description `pkg.depotd` correspond au serveur de dépôt du système d'empaquetage d'image. Il offre un accès réseau aux données stockées dans un référentiel de packages. Les clients qui ne prennent pas en charge l'accès direct au référentiel par le biais du système de fichiers, ou pour lesquels l'accès réseau est la seule méthode de transport disponible ou la méthode préférée, utilisent généralement le dépôt de packages.

Les clients tels que `pkg`(1), le client d'extraction, peuvent extraire une liste de packages et de métadonnées de package à partir d'un référentiel, directement ou par l'intermédiaire du serveur de dépôt. `pkgsend`(1), le client de publication, peut envoyer de nouvelles versions des packages à un référentiel directement ou par l'intermédiaire du serveur de dépôt. `pkgrepo`(1) peut servir à créer des référentiels à utiliser avec le serveur de dépôt, ou pour les gérer directement et par le biais le serveur de dépôt.

`pkg.depotd` est généralement exécuté en tant que service sur le système. Les développeurs de package et de logiciels peuvent exécuter des copies privées à fins de test.

Le dépôt n'offre pas de méthodes de contrôle d'accès. Par défaut, tous les clients en mesure de se connecter sont à même de lire toutes les données de package et de publier de nouvelles versions de package. Il existe une exception lors d'une exécution sous l'utilitaire de gestion des services (SMF) : la valeur par défaut est d'exécuter en mode lecture seule. Les remarques qui suivent décrivent certaines des meilleures pratiques pour le maintien d'un serveur de dépôt public en constante évolution.

Propriétés Smf Le serveur `pkg.depot` est généralement configuré à l'aide des propriétés `smf`(5) associées à son service. Les propriétés suivantes sont reconnues :

`pkg/address` (`net_address`) Adresse IP sur laquelle écouter les connexions. La valeur par défaut est 0.0.0.0 (`INADDR_ANY`), qui est à l'écoute sur toutes les interfaces actives. Pour écouter sur toutes les interfaces IPv6 actives, utilisez `::`. Seule la première valeur est utilisée.

`pkg/content_root` (`astring`) Chemin d'accès au système de fichiers au niveau duquel l'instance doit trouver son contenu statique et tout autre contenu Web. La valeur par défaut est `/usr/share/lib/pkg`.

`pkg/debug` (`astring`) Liste séparée par des virgules de fonctions de débogage à activer. Les valeurs possibles sont les suivantes :

`headers` Consigne les en-têtes de chaque demande dans le journal des erreurs.

`pkg/disable_ops` (`astring`) Liste séparée par des virgules d'opérations qui doivent être désactivées pour le serveur de dépôt. Les opérations sont données comme *operation*[/*version*] (`catalog` ou `search_1`, par exemple).

`pkg/image_root` (`astring`) Chemin d'accès à l'image dont les informations de fichier seront utilisées comme cache pour les données de fichiers.

`pkg/inst_root` (`astring`) Chemin d'accès au système de fichiers au niveau duquel l'instance doit trouver ses données de référentiel. Obligatoire sauf si `file_root` ou `PKG_REPO` est fourni. La valeur par défaut est `/var/pkgrepo`.

`pkg/log_access` (`astring`) Destination pour toute information d'accès consignée par le processus de dépôt. Les valeurs possibles sont `stderr`, `stdout`, `none` ou un nom de chemin absolu. La valeur par défaut est `stdout` si `stdout` est un `tty`. Si `stdout` n'est pas un `tty`, la valeur par défaut est `none`.

`pkg/log_errors` (`astring`) Destination de toute erreur ou autre information consignée par le processus de dépôt. Les valeurs possibles sont `stderr`, `stdout`, `none` ou un nom de chemin absolu. La valeur par défaut est `stderr`.

`pkg/mirror` (`boolean`) Indique si le mode de mise en miroir de package est utilisé. Lorsque la valeur de cette option est True (Vrai), les opérations de publication et de métadonnées sont désactivées et l'interface utilisateur de navigateur fournie est limitée. Cette propriété ne peut pas être définie sur true lorsque la propriété `pkg/readonly` est true. La valeur par défaut est `false`.

`pkg/port` (`count`) Numéro de port d'écoute des requêtes de package entrantes pour cette instance. Si un certificat

SSL et des informations de clés n'ont pas été fournis, la valeur par défaut est 80. Dans le cas contraire, la valeur par défaut est 443.

`pkg/proxy_base` (`uri`) Permet de modifier l'URL de base pour le serveur de dépôt et s'avère particulièrement utile lorsqu'il s'exécute derrière Apache ou un autre serveur Web, dans une configuration de proxy inversé.

`pkg/readonly` (`boolean`) Définit si des opérations de modification, comme celles lancées par `pkgsend`(1), sont désactivées. Les opérations de récupération sont toujours disponibles. Cette propriété ne peut pas être définie sur true lorsque la propriété `pkg/mirror` est true. La valeur par défaut est `true`.

`pkg/socket_timeout` (`count`) Nombre maximal de secondes pendant lesquelles le serveur doit attendre une réponse d'un client avant la fermeture d'une connexion. La valeur par défaut est 60.

`pkg/sort_file_max_size` (`count`) Taille maximale du fichier de tri de l'indexeur. Permet de limiter la quantité de RAM utilisée par le dépôt pour l'indexation ou de l'augmenter pour la vitesse.

`pkg/ssl_cert_file` (`astring`) Nom de chemin absolu d'un fichier de certificat chiffré en PEM. La valeur par défaut est `none`. Cette propriété doit être utilisée avec `ssl_key_file`. Le dépôt répond aux demandes SSL uniquement si `ssl_cert_file` et `/ssl_key_file` sont fournis.

`pkg/ssl_dialog` (`astring`) Spécifie la méthode à utiliser pour obtenir la phrase de passe nécessaire pour déchiffrer le fichier `ssl_key_file`. Les valeurs possibles sont les suivantes :

`builtin`	Invite pour la phrase de passe. Il s'agit de la valeur par défaut.
`exec:`*/path/to/program*	Exécutez le programme externe spécifié pour obtenir la phrase de passe. Le premier argument pour le programme est `''`

et est réservé. Le deuxième argument pour le programme est le numéro de port du serveur. La phrase de passe est imprimée sur `stdout`.

`smf:fmri` — Tente de récupérer la valeur de la propriété `pkg_secure/ssl_key_passphras` à partir de l'instance de service associée au FMRI.

`pkg/ssl_key_file` — (`astring`) Nom de chemin absolu d'un fichier de clés privées chiffré en PEM. Cette propriété doit être utilisée avec `ssl_cert_file`. Le dépôt répond aux demandes SSL uniquement si `/ssl_key_file` et `ssl_cert_file` sont fournis.

`pkg/threads` — (`count`) Nombre de threads démarrés pour traiter les demandes. La valeur par défaut est 60. Convient uniquement aux petits déploiements. Cette valeur doit être égale à environ 20 fois le nombre de clients simultanés. La valeur maximale de `threads` est fixée à 5000.

`pkg/writable_root` — (`astring`) Chemin d'accès au système de fichiers sur un répertoire auquel le programme a accès en écriture. Avec l'option `-readonly`, elle permet au serveur de dépôt de créer des fichiers, notamment les index de recherche, sans avoir accès en écriture aux informations sur le package.

`pkg_secure/ssl_key_passphrase` — (`astring`) Mot de passe à utiliser pour déchiffrer le `pkg/ssl_key_file`. Cette valeur est protégée par une autorisation en lecture à l'aide de l'attribut `solaris.smf.read.pkg-server`.

La présentation et le comportement de l'interface utilisateur du navigateur du serveur de dépôt sont contrôlés à l'aide des propriétés suivantes :

`pkg_bui/feed_description` — (`astring`) Paragraphe descriptif pour le flux RSS/Atom.

`pkg_bui/feed_icon` — (`astring`) Nom du chemin d'une petite image utilisée pour représenter le flux RSS/Atom. Le nom du chemin doit être

relatif à `content_root`. La valeur par défaut est `web/_themes/pkg-block-icon.png`.

`pkg_bui/feed_logo` (`astring`) Nom du chemin d'une image de grande taille qui sera utilisée pour marquer ou identifier visuellement le flux RSS/Atom. Cette valeur doit être relative à `content_root`. La valeur par défaut est `web/_themes/pkg-block-icon.png`.

`pkg_bui/feed_name` (`astring`) Nom court et descriptif des flux RSS/Atom généré par le dépôt desservant le référentiel. La valeur par défaut est “package repository feed”.

`pkg_bui/feed_window` (`count`) Nombre d'heures avant la génération du dernier flux pour le référentiel, à inclure lors de la génération du flux.

Le dépôt de packages est également capable de servir de serveur miroir pour les images de client local depuis `pkg`(5). Cela permet à des clients partageant un sous-réseau sur un réseau local de refléter leurs caches de fichiers. Les clients peuvent télécharger des fichiers l'un de l'autre, réduisant ainsi la charge sur le serveur de dépôt de packages. Cette fonctionnalité est disponible sous la forme d'un autre service de dépôt configuré par `smf`(5). Elle utilise mDNS et `dns-sd` pour la détection des services.

Le miroir mDNS est généralement configuré à l'aide des propriétés `smf`(5) associées à son service. Les propriétés suivantes sont reconnues :

`pkg/image_root` (`astring`) Chemin d'accès à l'image dont les informations de fichier seront utilisées comme cache pour les données de fichiers. La valeur par défaut est `/`.

`pkg/port` (`count`) Numéro de port d'écoute des requêtes de package entrantes pour cette instance. La valeur par défaut est 80.

Options `pkg.depotd` peut lire ses informations de configuration de base à partir d'un fichier ou des données de propriété d'une instance de service `smf`(5) existante.

`--cfg` *source* Nom de chemin du fichier à utiliser pour la lecture et l'écriture des données de configuration ou bien chaîne au format `smf:`*fmri*, où *fmri* est l'identificateur de ressource de gestion des pannes (FMRI) de l'instance depuis laquelle il faut lire les données de configuration. Pour plus de détails sur le format du fichier spécifié, reportez-vous à la section “Configuration du dépôt” ci-dessous.

En l'absence de source de configuration préexistante ou pour remplacer des valeurs lues à partir d'un fichier de configuration fourni à l'aide de `--cfg`, vous pouvez utiliser les options suivantes afin de modifier le comportement par défaut du serveur de dépôt :

`-a address`	Reportez-vous à la section `pkg/address` ci-dessus.
`--content-root` *root_dir*	Reportez-vous à la section `pkg/content_root` ci-dessus.
`-d` *inst_root*	Reportez-vous à la section `pkg/inst_root` ci-dessus.
`--debug` *features*	Reportez-vous à la section `pkg/debug` ci-dessus.
`--disable-ops` *op_list*	Reportez-vous à la section `pkg/disable_ops` ci-dessus.
`--image-root` *path*	Reportez-vous à la section `pkg/image_root` ci-dessus.
`--log-access` *dest*	Reportez-vous à la section `pkg/log_access` ci-dessus.
`--log-errors` *dest*	Reportez-vous à la section `pkg/log_errors` ci-dessus.
`--mirror`	Reportez-vous à la section `pkg/mirror` ci-dessus.
`-p` *port*	Reportez-vous à la section `pkg/port` ci-dessus.
`--proxy-base` *url*	Reportez-vous à la section `pkg/proxy_base` ci-dessus. Cette option est ignorée si une valeur vide est fournie.
`--readonly`	Reportez-vous à la section `pkg/readonly` ci-dessus.
`-s` *threads*	Reportez-vous à la section `pkg/threads` ci-dessus.
`-s-ort-file-max-size` *bytes*	Reportez-vous à la section `pkg/sort_file_max_size` ci-dessus.
`--ssl-cert-file` *source*	Reportez-vous à la section `pkg/ssl_cert_file` ci-dessus.
`--ssl-dialog` *type*	Reportez-vous à la section `pkg/ssl_dialog` ci-dessus.
`--ssl-key-file` *source*	Reportez-vous à la section `pkg/ssl_key_file` ci-dessus.
`-t` *socket_timeout*	Reportez-vous à la section `pkg/socket_timeout` ci-dessus.
`--writable-root` *path*	Reportez-vous à la section `pkg/writable_root` ci-dessus.

D'autres fonctionnalités d'administration et de gestion pour les référentiels de packages sont fournies par `pkgrepo`(1).

Configuration du dépôt

Lorsqu'un fichier de configuration est fourni (au lieu d'un FMRI `smf`(5)) à l'aide de l'option `--cfg`, le serveur de dépôt lit et écrit toutes les données de configuration dans un format de texte simple. Les données de configuration sont décrites dans la section “Propriétés SMF” ci-dessus. Les données de configuration se composent de sections, qui commencent par un en-tête `[`*section*`]`, suivi d'entrées `name =` *value*. Les suites sont dans le style de RFC 822. Les valeurs peuvent être réparties sur plusieurs lignes en insérant un espace au début des lignes de suite.

Les valeurs requises non fournies dans le fichier de configuration doivent être fournies à l'aide de l'option figurant dans la section Options ci-dessus. Un exemple de fichier de configuration peut ressembler à ceci :

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

[pub_example_com]
feed_description = example.com’s software
  update log
```

Exemples

EXEMPLE 1 Activation du serveur de dépôt

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

EXEMPLE 2 Modification du port d'écoute du serveur

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

EXEMPLE 3 Activation du miroir

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

Variables d'environnement

`PKG_REPO` Spécifie le répertoire qui contient le référentiel à servir. Cette valeur est ignorée si `-d` est spécifié.

`PKG_DEPOT_CONTENT` Spécifie le répertoire de stockage du contenu statique servi par le dépôt. Les fichiers répertoriés ci-dessous sous Fichiers doivent être présents dans ce répertoire, bien que leur contenu puisse être différent de celui du contenu fourni par défaut.

État de sortie

Les valeurs de sortie renvoyées sont les suivantes :

`0` Opération réussie.

`1` Une erreur s'est produite.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

`99` Une exception imprévue est survenue.

Fichiers

`/usr/share/lib/pkg` Emplacement du contenu de la présentation par défaut. Modifiez `pkg/content_root` pour sélectionner un autre emplacement.

Attributs

Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	package/pkg
Stabilité de l'interface	Non validé

Voir aussi dns-sd(1M), mdnsd(1M), pkg(1), pkgrepo(1), pkgsend(1), syslogd(1M), smf(5)

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

Remarques Le service pkd.depotd est géré par SMF sous l'identifiant de service svc:/application/pkg/server.

Le service de miroir mDNS est géré par SMF sous l'identifiant de service svc:/application/pkg/dynamic-mirror.

Pour contrôler l'accès en lecture au dépôt, vous pouvez utiliser un proxy inverse HTTP en combinaison avec des méthodes d'authentification, telles que l'accès client en fonction du certificat SSL, que pkg(1) prend en charge en natif.

Les modifications apportées à la configuration ou aux données du package à l'aide d'opérations sur le système de fichiers exigent un redémarrage du processus de serveur de dépôt afin que les modifications soient répercutées dans les opérations et la sortie. Suivez l'une des méthodes ci-après pour redémarrer le processus de serveur de dépôt :

- Utilisez svcadm(1M) pour redémarrer l'instance application/pkg/server.
- Envoyez un signal SIGUSR1 vers le processus de serveur de dépôt en utilisant kill(1). Cette opération exécute un “redémarrage progressif”, qui laisse le processus intact mais recharge toutes les données de configuration, de package et de recherche :

  # kill -USR1 *pid*

Nom pkg.sysrepo – Configuration de référentiel système Image Packaging System

Synopsis

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

Description `pkg.sysrepo` est utilisé pour générer les fichiers de configuration pour le référentiel système Image Packaging System (IPS). `pkg.sysrepo` est appelé par le service SMF `svc:/application/pkg/system-repository`. Les modifications de configuration doivent être apportées aux propriétés dans le service SMF.

Le référentiel système est responsable de la fourniture d'un accès aux référentiels de packages configurés dans une image de référence par le biais d'un proxy centralisé. Les modifications de configuration d'éditeur apportées à cette image de référence sont immédiatement visibles par tous les clients configurés pour utiliser le référentiel système.

Le référentiel système est principalement utilisé dans la zone globale pour octroyer aux zones non globales l'accès aux référentiels configurés dans la zone globale. Les services SMF `svc:/application/pkg/zones-proxyd` et `svc:/application/pkg/zones-proxy-client` sont chargés de fournir le transport entre les zones non globales et la zone globale. Ce transport n'est utilisé que par `pkg`(5).

Notez que seuls les référentiels de fichiers http, https et v4 sont pris en charge. Les référentiels de fichiers p5p ou de formats plus anciens ne sont pas pris en charge. Pour plus d'informations sur les versions de référentiel, reportez-vous à la section `pkgrepo`(1).

Options Les options suivantes sont prises en charge :

`-c` *cache_dir* Chemin d'accès absolu au répertoire que doit utiliser le référentiel système pour la mise en cache des réponses de l'éditeur configuré.

Par défaut, un cache de fichier est utilisé. Cependant, la valeur spéciale `memory` permet d'indiquer que le cache en mémoire doit être utilisé. La valeur spéciale `None` peut être utilisée pour indiquer que le référentiel système ne doit pas effectuer de mise en cache. Ce paramètre doit être configuré à l'aide de la propriété SMF `config/cache_dir`.

`-p` *port* Port que le référentiel système doit utiliser pour écouter les demandes. Ce paramètre doit être configuré à l'aide de la propriété SMF `config/port`.

`-s` *cache_size* Valeur d'entier en méga-octets qui définit la taille maximale du cache du référentiel système. Ce paramètre doit être configuré à l'aide de la propriété SMF `config/cache_max`.

`-w` *http_proxy* Chaîne au format *scheme*`://` *hostname*`[:`*port*`]` qui définit un proxy Web que le référentiel système peut utiliser pour accéder aux référentiels de packages basés sur http. Ce paramètre peut être configuré à l'aide de la propriété SMF `config/http_proxy`.

`-W` *https_proxy* Chaîne au format *scheme*`://` *hostname*`[:`*port*`]` qui définit un proxy Web que le référentiel système peut utiliser pour accéder aux référentiels de packages basés sur https. Ce paramètre peut être configuré à l'aide de la propriété SMF `config/https_proxy`.

Exemples

EXEMPLE 1 Activation du référentiel système

```
$ svcadm enable svc:/application/pkg/system-repository
```

État de sortie

Les valeurs de sortie renvoyées sont les suivantes :

`0` La commande a réussi.

`1` La commande n'a pas pu écrire une configuration valide.

`2` Des options de ligne de commande incorrectes ont été spécifiées.

`99` Une exception imprévue est survenue.

Attributs

Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg`
Stabilité de l'interface	Non validé

Voir aussi

`pkg`(1), `pkg.depotd`(1m), `pkg`(5)

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

RÉFÉRENCE

Normes, environnements et macros

Nom pkg – Image Packaging System

Description La structure IPS (Image Packaging System, système d'empaquetage d'image) `pkg(5)` permet de gérer les logiciels tout au long de leur cycle de vie (installation, mise à jour et suppression). L'empaquetage d'image gère les logiciels en unités de packages, c'est-à-dire des collections d'actions, définies par un ensemble de paires clé/valeur données et, souvent, d'une charge utile. Dans de nombreux cas, les actions sont des fichiers trouvés dans un système de fichiers, mais elles peuvent également représenter d'autres objets pouvant être installés, tels que des pilotes, des services et des utilisateurs.

Fmri et versions de packages Chaque package est représenté par un identificateur de ressource de gestion des pannes (FMRI) avec le schéma `pkg:`. Le FMRI complet pour un package se compose d'un schéma, d'un éditeur, le nom du package, et d'une chaîne de version au format suivant :

```
pkg://solaris/system/library/c++-runtime@0.5.11,5.11-0.174.0.0.0.0.0:20110921T190358Z
```

`solaris` est l'éditeur. `system/library/c++-runtime` est le nom du package. Bien que l'espace de noms soit hiérarchique et de profondeur quelconque, il n'y a aucun confinement imposé ; le nom est arbitraire. Les informations sur l'éditeur sont facultatives, mais doivent être précédées de `pkg://` s'il est présent. Un FMRI qui inclut l'éditeur est souvent désigné comme étant "entièrement qualifié". Si les informations sur l'éditeur ne sont pas présentes, le nom de package doit généralement être précédé par `pkg:/`.

Souvent, les clients de package permettent d'omettre le schéma d'un FMRI s'il ne contient pas d'informations sur l'éditeur. Par exemple, `pkg:/system/library/c++-runtime` correspond à `system/library/c++-runtime`. Si le schéma est omis, les clients permettent également l'omission de tous les composants d'un nom de package excepté le dernier pour la mise en correspondance. Par exemple, vous pouvez écrire indifféremment `system/library/c++-runtime`, `library/c++-runtime` ou `c++-runtime`, ce qui correspond aux packages nommés `c++-runtime` ou dont le nom se termine par `/c++-runtime`.

Un nom d'éditeur permet d'identifier une personne, un groupe de personnes, ou une organisation en tant que source d'un ou plusieurs packages. Afin d'éviter toute collision de nom d'éditeur et de faciliter l'identification de l'éditeur, une bonne pratique consiste à utiliser en tant que nom d'éditeur un nom de domaine représentant l'entité qui publie les packages.

La version suit le nom du package, séparés par un signe arobase (@). La version se compose de quatre séquences de nombres, séparées par des signes de ponctuation. Les éléments des trois premières séquences sont séparés par des points, et les séquences sont arbitrairement longues. Les zéros non significatifs dans les composants de version (par exemple, 01.1 ou 1.01) ne sont pas autorisés. Les zéros finaux (par exemple, 1.10) sont autorisés.

La première partie de la version désigne la version du composant. Pour les composants étroitement liés au système d'exploitation, il s'agit généralement de la valeur de `uname -r` pour cette version du système d'exploitation. Pour un composant avec son propre cycle de développement, cette séquence est un numéro de version à points, tels que 2.4.10.

La seconde partie de la version qui, si elle est présente, doit suivre une virgule (,), est la version de compilation. La version de compilation indique la version du système d'exploitation sous laquelle le contenu du package a été créé, et fournit une version du système d'exploitation minimale sous laquelle le contenu est prévu pour fonctionner correctement.

La troisième partie de la version, si elle est présente, doit suivre un trait d'union (-), et désigne la version de la branche. La version de la branche est un composant de version qui donne des informations spécifiques au fournisseur. La version de la branche peut être incrémentée lorsque les métadonnées de package sont modifiées, indépendamment de la version du composant. La version de la branche peut contenir un numéro de version de compilation ou d'autres informations.

La quatrième partie de la version, si elle est présente, doit suivre un signe deux-points (:) et désigne un horodatage. L'horodatage indique la date à laquelle le package a été publié.

Lorsque vous effectuez des comparaisons entre les versions, aucun des composants de la version complète n'est pris en compte, à moins que les composants à sa gauche soient les mêmes. Par conséquent, une version 4.3-1 est supérieure à 4.2-7 car 4.3 est supérieur à 4.2, et 4.3-3 est supérieur à 4.3-1 car 3 est supérieur à 1.

De nombreuses parties du système, n'affichent pas les FMRI complets, et acceptent les entrées plus courtes afin de réduire le volume d'informations affichées ou requises. En règle générale, le schéma, l'éditeur, la version de compilation et l'horodatage peuvent être omis. Parfois, toutes les informations sur la version peuvent être omises.

Actions Les actions représentent les objets d'un système pouvant être installés. Les actions sont décrites dans le fichier manifeste d'un package. Chaque action est essentiellement constituée de son nom et d'un attribut clé. Ensemble, elles se rapportent à un objet unique suivant un historique des versions. Les actions peuvent comprendre d'autres attributs. Certains attributs sont interprétés directement par le système d'empaquetage. D'autres attributs ne peuvent être utiles qu'à l'administrateur système ou à l'utilisateur final.

Les actions ont une représentation en texte simple :

action_name attribute1=value1 attribute2=value2 . . .

Les noms d'attributs ne peuvent pas contenir de blancs, de guillemets ou de signes égal (=). Tous les caractères après le premier signe égal appartiennent à la valeur. Les valeurs peuvent contenir tous ces caractères, bien qu'il faille placer les espaces entre guillemets simples ou doubles. Les guillemets simples n'ont pas besoin d'être remplacés par des caractères d'échappement à l'intérieur d'une chaîne qui est entre guillemets doubles, et les guillemets doubles n'ont pas besoin d'être remplacés par des caractères d'échappement à l'intérieur d'une chaîne qui est entre guillemets simples. Un guillemet peut être précédé d'une barre oblique inverse (\) afin d'éviter l'interruption de la chaîne entre guillemets. Une barre oblique inverse peut être placée dans une séquence d'échappement avec une barre oblique inverse.

Les attributs peuvent être nommés plus d'une fois avec plusieurs valeurs. Celles-ci sont traitées en tant que listes à puces.

Les actions avec de nombreux attributs peuvent créer des lignes longues dans un fichier manifeste. Ces lignes peuvent être renvoyées à la ligne en terminant chaque ligne incomplète par une barre oblique inverse. Notez que ce caractère de suite doit apparaître entre les paires attribut/valeur. Ni les attributs, ni leurs valeurs, ni la combinaison des deux, ne peuvent être fractionnés.

La liste des attributs ci-dessous n'est pas exhaustive. En fait, les attributs qui peuvent être reliés à une action sont arbitraires, et les ensembles d'attributs standard sont faciles à augmenter pour incorporer des développements futurs.

Certains attributs d'action déclenchent l'exécution d'opérations supplémentaires en dehors du contexte du package. Ces attributs sont présentés dans la section Actionneurs ci-dessous.

Actions sur les fichiers

L'action `file` représente un fichier ordinaire. L'action `file` fait référence à une charge utile et possède quatre attributs standard :

`path` Chemin du système de fichiers où le fichier est installé. C'est un attribut clé d'action `file`.

`mode` Les autorisations d'accès (sous forme numérique) au fichier. Il s'agit d'autorisations simples, et non d'ACL.

`owner` Nom de l'utilisateur propriétaire du fichier.

`group` Nom du groupe auquel appartient le fichier.

La charge utile est un attribut de position dans la mesure où il n'est pas nommé. Il s'agit du premier mot après le nom de l'action. Dans un manifeste publié, il correspond au hachage `SHA-1` du contenu du fichier. S'il est présent dans un manifeste qui n'a pas encore été publié, il représente le chemin d'accès à la charge utile. Reportez-vous à `pkgsend`(1). L'attribut de hachage peut être utilisé à la place de l'attribut de position, mais sa valeur doit inclure un signe égal. Les deux peuvent être utilisés dans la même action. Cependant, les hachages doivent être identiques.

Il existe d'autres attributs :

`preserve` Indique que le contenu du fichier ne doit pas être remplacé pendant la mise à niveau s'il est déterminé que le contenu a été modifié depuis l'installation ou la dernière mise à niveau du fichier. Lors des installations initiales, si un fichier existant est trouvé, il est récupéré (et stocké dans `/var/pkg/lost+found`).

Si la valeur de `preserve` est `renameold`, le fichier existant est renommé avec l'extension `.old`, puis remplacé par le nouveau fichier.

Si la valeur de `preserve` est `renamenew`, le fichier existant est isolé, et le nouveau fichier est installé avec l'extension `.new`.

Si la valeur de `preserve` est `legacy`, ce fichier n'est pas installé lors d'installations de package initiales. Pendant les mises à niveau, tout fichier existant est renommé avec l'extension `.legacy`, puis remplacé par le nouveau fichier.

Si la valeur de `preserve` est `true` (ou une valeur non répertoriée ci-dessus, comme `strawberry`), le fichier existant est isolé, et le nouveau fichier n'est pas installé.

`overlay` Indique si l'action permet à d'autres packages de fournir un fichier au même emplacement ou si elle fournit un fichier destiné à en recouvrir un autre. Cette fonctionnalité est destinée à être utilisée avec les fichiers de configuration qui ne participent à aucun auto-assemblage (par exemple, `/etc/motd`) et qui peuvent être remplacés en toute sécurité.

Si `overlay` n'est pas spécifié, plusieurs packages ne peuvent pas fournir des fichiers au même emplacement.

Si la valeur de `overlay` est `allow`, un autre package est autorisé à fournir un fichier au même emplacement. Cette valeur n'a aucun effet, sauf si l'attribut `preserve` est également défini.

Si la valeur de `overlay` est `true`, le fichier fourni par l'action remplace toute autre action qui a spécifié `allow`. Les modifications apportées au fichier installé sont conservées dans la valeur de l'attribut `preserve` du fichier de recouvrement. Lors de la suppression, le contenu du fichier est conservé si l'action en cours de recouvrement est toujours installée, que l'attribut `preserve` soit ou non spécifié. Seule une action peut en recouvrir une autre, et les attributs `mode`, `owner` et `group` doivent correspondre.

Les fichiers peuvent également être "goûtés" et, en fonction de leur saveur, posséder des attributs supplémentaires. Pour des fichiers ELF, les attributs suivants sont reconnus :

`elfarch` Architecture du fichier ELF. Il s'agit de la sortie de `uname -p` sur l'architecture pour laquelle le fichier est créé.

`elfbits` `32` ou `64`.

`elfhash` Hachage des sections ELF "intéressantes" dans le fichier. Ce sont les sections qui sont mises en correspondance dans la mémoire lorsque le fichier binaire est chargé. Il s'agit des seules sections à prendre en compte lorsqu'il s'agit de déterminer si le comportement exécutable de deux fichiers binaires diffère.

`original_name` Cet attribut est utilisé pour traiter les fichiers modifiables passant d'un package à un autre ou d'un endroit à un autre, ou les deux. Il prend la forme du nom du package d'origine, suivi de deux-points et du chemin

d'accès d'origine au fichier. Tout fichier supprimé est enregistré avec son package et son chemin, ou avec la valeur de l'attribut `original_name` s'il est spécifié. Tout fichier modifiable en cours d'installation, ayant l'attribut `original_name` configuré, utilise le fichier de ce nom s'il est supprimé dans le cadre de la même opération d'empaquetage.

`revert-tag` Cet attribut est utilisé pour marquer les fichiers modifiables qui doivent être rétablis sous la forme d'un jeu. Vous pouvez spécifier plusieurs valeurs `revert-tag`. Le fichier redevient un manifeste lorsque `pkg revert` est appelé et que l'une de ces balises est spécifiée. Reportez-vous à `pkg`(1).

Actions sur les répertoires

L'action `dir` est similaire à l'action `file` dans la mesure où elle représente un objet système de fichiers. L'action `dir` représente un répertoire au lieu d'un fichier ordinaire. L'action `dir` possède les mêmes quatre attributs standard que l'action `file`, et `path` constitue l'attribut clé.

Les répertoires sont des références comptées dans IPS. Lorsque le dernier package qui faisait explicitement ou implicitement référence à un répertoire ne le fait plus, ce répertoire est supprimé. Si ce répertoire contient des objets système de fichiers non empaquetés, ces éléments sont déplacés vers `$IMAGE_META/lost+found`. Reportez-vous à la section Fichiers pour plus d'informations sur `$IMAGE_META`.

Pour déplacer du contenu non empaqueté dans un nouveau répertoire, l'attribut suivant peut s'avérer utile :

`salvage-from` Nomme un répertoire d'éléments récupérés. Un répertoire doté de cet attribut hérite lors de sa création du contenu du répertoire récupéré s'il existe.

Actions sur les liens

L'action `link` représente un lien symbolique. L'action `link` possède les attributs standard suivants :

`path`
Chemin du système de fichiers où le lien symbolique est installé. C'est un attribut clé d'action `link`.

`target`
La cible du lien symbolique. L'objet système de fichiers associé à la résolution du lien.

`mediator`
Spécifie l'entrée dans l'espace de noms de médiation partagé par tous les noms de chemin faisant partie d'un groupe de médiation donné (par exemple, `python`). La médiation de liens peut être effectuée en fonction de `mediator-version` et/ou `mediator-implementation`. Tous les liens de médiation d'un chemin donné doivent indiquer le même médiateur. Toutefois, toutes les versions et implémentations de médiateur ont besoin de fournir un lien à un chemin donné. Si une médiation ne fournit pas de lien, le lien est supprimé lorsque que la médiation est sélectionnée. Un attribut

`mediator`, en combinaison avec une version spécifique et/ou une implémentation représente une médiation qui peut être sélectionnée en vue d'une utilisation par le système d'empaquetage.

`mediator-version`
Spécifie la version (exprimée sous la forme d'une séquence d'entiers non négatifs séparés par des points) de l'interface décrite par l'attribut `mediator`. Cet attribut est requis si `mediator` est spécifié et `mediator-implementation` ne l'est pas. Un administrateur système local peut définir la version à utiliser explicitement. La valeur indiquée doit généralement correspondre à la version d'un package fournissant le lien (par exemple, `runtime/python-26` doit utiliser `mediator-version=2.6`), bien que ce ne soit pas nécessaire.

`mediator-implementation`
Spécifie l'implémentation du médiateur à utiliser en plus ou à la place de `mediator-version`. Les chaînes d'implémentation ne sont pas considérées comme étant ordonnées, et `pkg` (5) sélectionne une chaîne arbitrairement si celle-ci n'est pas explicitement spécifiée par un administrateur système.

La valeur peut être une chaîne de caractères de longueur arbitraire composée de caractères alphanumériques et d'espaces. Si l'implémentation peut contenir une version ou en possède une, la version doit être spécifiée à la fin de la chaîne, après un @ (exprimée sous la forme d'une séquence d'entiers non négatifs séparés par des points). Si plusieurs versions d'une implémentation existent, le comportement par défaut consiste à sélectionner l'implémentation avec la plus grande version.

Si une seule instance d'un lien de médiation d'implémentation est installée sur un système, dans un chemin particulier, alors c'est celle-ci qui est automatiquement sélectionnée. Si, plus tard, des liens sont installés dans ce chemin d'accès, le lien reste inchangé à moins qu'un fournisseur, un site, ou un remplacement local ne s'applique, ou que l'un des liens possède un médiateur de version.

`mediator-priority`
Lors de la résolution des conflits dans les liens de médiation, `pkg`(5) choisit généralement le lien avec la plus grande valeur `mediator-version` ou basé sur `mediator-implementation` si ce n'est pas possible. Cet attribut est utilisé pour spécifier une valeur de remplacement pour le processus de résolution des conflits normal.

Si cet attribut n'est pas spécifié, la logique de sélection mediator par défaut est appliquée.

Si la valeur est `vendor`, ce lien est préféré à ceux qui ne disposent pas d'un attribut `mediator-priority` spécifié.

Si la valeur est `site`, le lien est préféré à ceux qui ont une valeur `vendor` ou qui ne disposent pas d'un attribut `mediator-priority` spécifié.

Un administrateur système local peut remplacer la logique de sélection décrite ci-dessus.

Actions sur les liens physiques

L'action `hard link` représente un lien physique. Elle possède les mêmes attributs que l'action `link`, et `path` est également son attribut clé.

Actions sur les pilotes

L'action `driver` représente un pilote de périphérique. L'action `driver` ne fait pas référence à une charge utile. Les fichiers driver eux-mêmes doivent être installés comme les actions `file`. Les attributs suivants sont reconnus (reportez-vous à `add_drv`(1M) pour plus d'informations) :

`name` Nom du pilote. Il s'agit généralement, mais pas toujours, du nom de fichier binaire de pilote. C'est un attribut clé d'action `driver`.

`alias` Alias pour le pilote. Un pilote donné peut avoir plusieurs attributs `alias`. Aucune règle de citation particulière n'est requise.

`class` Représente une classe de pilote. Un pilote donné peut avoir plusieurs attributs `class`.

`perms` Autorisations du système de fichiers pour accéder aux noeuds de périphérique du pilote.

`clone_perms` Autorisations du système de fichiers pour accéder aux noeuds mineurs du pilote clone de ce pilote.

`policy` Stratégie de sécurité supplémentaire pour le périphérique. Un pilote donné peut avoir plusieurs attributs `policy`, mais aucune spécification du périphérique mineur ne peut être présente dans plus d'un attribut.

`privs` Privilèges utilisés par le pilote. Un pilote donné peut avoir plusieurs attributs `privs`.

`devlink` Entrée dans `/etc/devlink.tab`. La valeur correspond à la ligne exacte pour aller dans le fichier, avec des onglets représentés par `\t`. Reportez-vous à `devlinks`(1M) pour plus d'informations. Un pilote donné peut avoir plusieurs attributs `devlink`.

Actions sur les dépendances

L'action `depend` représente une dépendance entre packages. Un package peut dépendre d'un autre package, en raison d'une fonctionnalité qu'il requiert pour son fonctionnement, voire son installation, et dont l'autre package dispose. Les dépendances peuvent être facultatives. Si une dépendance n'est pas satisfaite au moment de l'installation, le système d'empaquetage tente d'installer ou de mettre à jour le package dépendant vers une version suffisamment récente, en fonction d'autres contraintes.

Les attributs suivants sont reconnus :

`fmri` FMRI du package dépendant. C'est un attribut clé d'action `dependency`. La valeur `fmri` ne doit pas inclure l'éditeur. Le nom du package est supposé être complet. Les dépendances de type `require-any` peuvent avoir plusieurs attributs `fmri`. La version est facultative sur la valeur du `fmri`, bien que pour certains types de dépendances, un `fmri` sans version n'a aucune signification.

`type` Type de la dépendance.

Si la valeur est `require`, la dépendance est requise et doit avoir une version égale ou supérieure à la version spécifiée dans l'attribut `fmri`. Si la version n'est pas spécifiée, n'importe quelle version satisfait cette dépendance. Un package ne peut pas être installé si aucune des dépendances obligatoires ne peut être satisfaite.

Si la valeur est `optional`, la dépendance, le cas échéant, doit être au niveau de version spécifié ou supérieur.

Si la valeur est `exclude`, le package qui la contient ne peut pas être installé si la dépendance est présente au niveau de version spécifié ou supérieur. Si aucune version n'est spécifiée, le package dépendant ne peut pas être installé simultanément avec le package spécifiant la dépendance.

Si la valeur est `incorporate`, la dépendance est facultative, mais la version du package dépendant est limitée. Reportez-vous à la section Contraintes et figement ci-dessous.

Si la valeur est `require-any`, n'importe quel package dépendant spécifié par plusieurs attributs `fmri` peut satisfaire la dépendance, suivant les mêmes règles que pour le type de dépendance `require`.

Si la valeur est `conditional`, la dépendance est requise uniquement si le package défini par l'attribut `predicate` est présent sur le système.

Si la valeur est `origin`, la dépendance doit, le cas échéant, être sur la valeur spécifiée ou, au mieux, sur l'image à modifier avant l'installation. Si la valeur de l'attribut `root-image` est `true`, la dépendance doit être présente sur l'image root afin d'installer ce package.

Si la valeur est `group`, la dépendance est requise, sauf si le package est sur la liste avoid de l'image. Notez que les packages obsolètes satisfont silencieusement la dépendance du groupe. Reportez-vous à la sous-commande `avoid` dans `pkg`(1).

Si la valeur est `parent`, la dépendance est ignorée si l'image n'est pas une image enfant. Si l'image est une image enfant, la dépendance doit être présente dans l'image parent. La version du package correspondant à une dépendance `parent` est la même que celle utilisée pour les dépendances `incorporate`.

`predicate` FMRI indiquant le prédicat pour les dépendances `conditional`.

`root-image` N'a d'effet que pour les dépendances `origin`, comme mentionné ci-dessus.

Actions sur les licences

L'action `license` représente un fichier licence ou d'autres fichiers d'informations associés au contenu du package. Un package peut fournir des licences, des exclusions de responsabilité ou d'autres conseils au programme d'installation via l'utilisation d'une action `license`.

La charge utile de l'action `license` est fournie dans le répertoire des métadonnées d'image associées au package, et doit contenir uniquement des données texte lisible par l'homme. Il ne doit pas contenir de balises HTML ou toute autre forme de balisage. Par le biais d'attributs, les actions `license` peuvent indiquer aux clients que la charge utile associée doit être affichée et/ou soumise à acceptation. La méthode d'affichage et/ou d'acceptation est à la discrétion des clients.

Les attributs suivants sont reconnus :

`license` C'est un attribut clé d'action `license`. Cet attribut fournit une description précise de la licence qui aide les utilisateurs à déterminer le contenu sans lire le texte de la licence lui-même. Voici quelques exemples de valeurs :

- 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

La valeur `license` doit être unique au sein d'un package. Il est recommandé d'ajouter la version de la licence dans la description, comme illustré dans les exemples ci-dessus. Si un package comprend du code sous plusieurs licences, utilisez plusieurs actions `license`. La longueur maximale de la valeur d'attribut license est fixée à 64 caractères.

`must-accept` Si la valeur est `true`, cette licence doit être acceptée par un utilisateur pour que le package associé puisse être installé ou mis à jour. L'omission de cet attribut équivaut à une valeur `false`. La méthode d'acceptation (interactive ou basée sur la configuration, par exemple) est à la discrétion des clients.

`must-display` Si la valeur est `true`, la charge utile de l'action doit être affichée par les clients pendant les opérations d'empaquetage. L'omission de cette valeur équivaut à une valeur `false`. Cet attribut ne doit pas être utilisé pour les mentions de copyright. Il ne peut servir qu'aux véritables licences ou à d'autres contenus qui doivent être affichés au cours des opérations. La méthode d'affichage est à la discrétion des clients.

Actions sur le contenu hérité

L'action `legacy` représente les données de package utilisées par un système d'empaquetage hérité. Les attributs associés à cette action sont ajoutés aux bases de données du système hérité de sorte que les outils effectuant des requêtes dans ces bases de données peuvent fonctionner

comme si le package hérité était installé. En particulier, elle doit suffire à convaincre le système hérité que le package nommé par l'attribut `pkg` est installé sur le système, de sorte que ce package peut être utilisé pour satisfaire les dépendances.

Les attributs suivants, nommés conformément aux paramètres de `pkginfo`(4), sont reconnus :

`category` Valeur pour le paramètre `CATEGORY`. La valeur par défaut est `system`.

`desc` Valeur pour le paramètre `DESC`.

`hotline` Valeur pour le paramètre `HOTLINE`.

`name` Valeur pour le paramètre `NAME`. La valeur par défaut est `none provided`.

`pkg` Abréviation du package en cours d'installation. La valeur par défaut est le nom du FMRI du package. C'est un attribut clé d'action `legacy`.

`vendor` Valeur pour le paramètre `VENDOR`.

`version` Valeur pour le paramètre VERSION. La valeur par défaut est la version issue du FMRI du package.

Actions sur les ensembles

L'action `set` représente un attribut de niveau de package, ou des métadonnées, tel que la description du package.

Les attributs suivants sont reconnus :

`name` Nom de l'attribut.

`valeur` Valeur de l'attribut.

L'action `set` peut fournir n'importe quelles métadonnées choisies par l'auteur du package. Toutefois, il y a un certain nombre de noms d'attribut bien définis qui ont une signification particulière auprès du système d'empaquetage.

`pkg.fmri` Reportez-vous à la section “FMRI et versions de packages” sous “Description”.

`info.classification` Un ou plusieurs jetons qu'un client `pkg`(5) peut utiliser pour classifier le package. La valeur doit avoir comporter schéma (tel que "org.opensolaris.category.2008" ou "org.acm.class.1998") et la classification réelle, par exemple "Applications/Games", séparés par deux-points (:).

`pkg.description` Description détaillée du contenu et des fonctionnalités de ce package, en général de la longueur d'un paragraphe.

`pkg.obsolete` Si la valeur est `true`, le package est marqué comme obsolète. Un package obsolète peut n'avoir aucune autre action que des actions sur les ensembles, et ne doit pas être marqué comme renommé.

`pkg.renamed`	Si la valeur est `true`, le package a été renommé. Il doit exister dans le package une ou plusieurs actions `depend` qui pointent également vers les versions du package sur lesquelles ce package a été renommé. Un package ne peut pas être marqué à la fois comme renommé et obsolète, mais peut avoir n'importe quel nombre d'actions set.
`pkg.summary`	Brève description du package sur une seule ligne.

Actions sur les groupes

L'action `group` définit un groupe UNIX tel que défini dans`group`(4). Les mots de passe de groupe ne sont pas pris en charge. Les groupes définis avec cette action ne présentent généralement pas de liste d'utilisateurs. Vous pouvez ajouter des utilisateurs avec l'action `user`. Les attributs suivants sont reconnus :

`groupname`	Valeur pour le nom du groupe.
`gid`	ID numérique unique du groupe. La valeur par défaut est le premier groupe disponible sous 100.

Actions sur les utilisateurs

L'action `user` définit un utilisateur UNIX tel que défini dans les fichiers `/etc/passwd`, `/etc/shadow`, `/etc/group` et `/etc/ftpd/ftpusers`. Pour les utilisateurs définis avec cet attribut, des entrées sont ajoutées aux fichiers appropriés.

Les attributs suivants sont reconnus :

`username`	Nom unique de l'utilisateur
`password`	Mot de passe chiffré de l'utilisateur. La valeur par défaut est `*LK*`. Reportez-vous à `shadow`(4).
`uid`	UID unique de l'utilisateur. La valeur par défaut est la première valeur disponible sous 100.
`group`	Nom du groupe principal de l'utilisateur. Se trouve dans `/etc/group`.
`gcos-field`	La valeur du champ `gcos` dans `/etc/passwd`. La valeur par défaut est `username`.
`home-dir`	Répertoire personnel de l'utilisateur. La valeur par défaut est /.
`login-shell`	Le shell par défaut de l'utilisateur. La valeur par défaut est vide.
`group-list`	Groupes secondaires auxquels l'utilisateur appartient. Reportez-vous à `group`(4).
`ftpuser`	Peut être défini sur `true` ou `false`. La valeur par défaut de `true` indique que l'utilisateur est autorisé à se connecter via FTP. Reportez-vous à `ftpusers`(4).

`lastchg`	Le nombre de jours entre le 1er janvier 1970 et la date à laquelle le mot de passe a été modifié pour la dernière fois. La valeur par défaut est vide. Reportez-vous à `shadow`(4).
`min`	Nombre minimum de jours requis entre les modifications de mot de passe. Ce champ doit être défini sur 0 ou plus pour activer le vieillissement du mot de passe. La valeur par défaut est vide. Reportez-vous à `shadow`(4).
`max`	Nombre maximal de jours pendant lesquels le mot de passe est valide. La valeur par défaut est vide. Reportez-vous à `shadow`(4).
`warn`	Nombre de jours avant l'expiration du mot de passe où un avertissement signalant l'expiration est envoyé à l'utilisateur. Reportez-vous à `shadow`(4).
`inactive`	Nombre de jours d'inactivité autorisés pour cet utilisateur. Ce nombre est calculé par ordinateur. Les informations sur l'heure de la dernière connexion sont issues du fichier `lastlog` de l'ordinateur. Reportez-vous à `shadow`(4).
`expire`	Date absolue exprimée sous la forme d'un nombre de jours écoulés depuis l'époque UNIX (1er janvier 1970). Lorsque cette valeur est atteinte, la connexion ne peut plus être utilisée. Par exemple, une valeur d'expiration de 13514 spécifie un délai d'expiration de connexion au 1er janvier 2007. Reportez-vous à `shadow`(4).
`flag`	Défini sur empty. Reportez-vous à `shadow`(4).

Actionneurs Dans certains contextes, l'exécution d'autres opérations peut être appropriée en préparation à ou suite à l'introduction d'une action donnée. En général, ces opérations supplémentaires sont nécessaires uniquement sur une image du système actif et sont propres au système d'exploitation. Lorsque plusieurs actions impliquées dans l'installation ou le retrait d'un package ont les mêmes actionneurs, l'opération correspondant à la présence d'un actionneur est exécutée une fois pour cette installation ou ce retrait.

Les actionneurs non spécifiés correctement peuvent entraîner un échec de l'installation du package, notamment si l'actionneur ne peut pas déterminer un moyen de rendre sûre la progression de l'installation.

Les actionneurs suivants peuvent être définis :

`reboot-needed`
: Peut être défini sur `true` ou `false`. Pendant l'installation d'un package, si une action est installée ou mise à jour avec cet actionneur défini sur `true`, la transaction relative au package peut requérir une réinitialisation. Certaines implémentations client peuvent nécessiter des étapes supplémentaires, telles que l'exécution de toute opération d'empaquetage à l'aide d'un clone de l'image, dans le cas où l'image est une image du système actif.

`disable_fmri`, `refresh_fmri`, `restart_fmri`, `suspend_fmri`
Chacun de ces actionneurs prend la valeur d'un FMRI d'une instance de service sur laquelle fonctionner pendant l'installation ou le retrait du package. `disable_fmri` désactive le FMRI donné avant la suppression d'une action, par le biais de la sous-commande `disable` dans `svcadm`(1M). `refresh_fmri` et `restart_fmri` actualisent ou redémarrent le FMRI donné après l'installation, la mise à jour ou la suppression d'une action à l'aide des sous-commandes respectives de `svcadm`(1M). Enfin, `suspend_fmri` désactive temporairement le FMRI avant la phase d'installation de l'action, puis l'active au terme de cette phase.

La valeur peut contenir un modèle qui correspond à plusieurs instances de service. Toutefois, elle doit le faire explicitement à l'aide d'un glob accepté par `svcs`(1), au lieu de procéder implicitement en n'indiquant pas toutes les instances.

Contraintes et figement

Les questions de savoir quand un package doit passer à une nouvelle version, quand il doit être ajouté ou supprimé du système, quelle version doit être sélectionnée et si la suppression est autorisée dépendent d'un grand nombre de contraintes imposées au package. Ces contraintes peuvent être définies par d'autres packages sous forme de dépendances, ou par l'administrateur sous forme de figements.

La forme de contrainte la plus commune est fournie par la dépendance `require`, comme décrit à la section Actions sur les dépendances ci-dessus. Ce type de restriction empêche le package d'être déclassé ou supprimé.

La plupart des parties du système d'exploitation sont encapsulées par packages appelés *incorporations*. Ces packages fournissent principalement des contraintes représentées par la dépendance `incorporate`.

Comme décrit ci-dessus, un package incorporé n'a pas besoin d'être présent sur le système, mais s'il s'y trouve, il spécifie à la fois une version minimum inclusive et une version maximum exclusive. Par exemple, si le FMRI dépendant porte la version 1.4.3, aucune version inférieure à 1.4.3 ne répond à la dépendance, pas plus qu'une version égale ou supérieure à 1.4.4. Cependant, les versions qui ne font qu'étendre la séquence (comme 1.4.3.7) sont autorisées.

Les incorporations sont utilisées pour forcer la mise à niveau simultanée de certaines parties du système. Pour certains composants, tels que la bibliothèque C et le noyau, il s'agit d'une condition de base. Pour d'autres, tels qu'un simple composant userland n'ayant aucune dépendance, la mise à niveau synchrone sert uniquement à fournir un ensemble de versions connu et testé d'un package auquel peut faire référence une version particulière de l'incorporation.

Puisqu'une incorporation constitue simplement un package, elle peut être supprimée, et toutes les contraintes qu'elle offre sont donc relâchées. Cependant, un grand nombre d'incorporations fournies par Oracle Solaris sont requises par les packages qu'elles intègrent, car le relâchement ne serait pas sûr.

Toute tentative de mise à niveau d'un package vers une version qui n'est pas autorisée par une incorporation installée échouera, et ne tentera pas de trouver une version plus récente de l'incorporation pour satisfaire la demande. Si la contrainte elle-même doit être déplacée, et si l'incorporation qui la spécifie ne peut pas être supprimée, l'incorporation doit être mise à niveau vers une version qui spécifie une version souhaitée de la contrainte. La mise à niveau d'une incorporation entraîne celle de tous les packages incorporés qui ne satisfont pas les contraintes fournies par la nouvelle version.

Un administrateur système peut contraindre un package à l'aide de la commande `pkg freeze`. Le package nommé est contraint à la version installée sur le système si aucune version n'est fournie. Si un package avec version est fourni, cette contrainte administrative, ou figement, agit comme si une dépendance incorporate était installée à l'endroit où l'attribut `fmri` possède la valeur de la version du package fournie.

Un gel n'est jamais annulé automatiquement par le système d'empaquetage. Pour relâcher une contrainte, utilisez la commande `pkg unfreeze`.

Editeurs et référentiels

Comme expliqué ci-dessus, un éditeur est un simple nom que les clients du package utilisent pour identifier le fournisseur de packages. Les éditeurs peuvent distribuer les packages en utilisant les référentiels de packages et/ou archives de packages. Il y a deux types de référentiels actuellement pris en charge par le système de packages : référentiels d'origine et référentiels miroirs.

L'*origine* est un référentiel de packages qui contient toutes les métadonnées (catalogues, manifestes, et les index de recherche) et du contenu (fichiers) pour un ou plusieurs packages. Si plusieurs origines sont configurées pour un éditeur donné dans une image, l'API du client du package tente de choisir la meilleure origine à partir de laquelle extraire les données de package. Il s'agit du type de référentiel le plus courant, implicitement créé chaque fois que la commande `pkgsend` ou `pkgrecv` est utilisée sur un référentiel de packages.

Un *miroir* est un référentiel de package incluant uniquement le contenu d'un package (fichiers). Si un ou plusieurs miroirs sont configurés pour un éditeur dans une image, l'API du client préfère les miroirs pour l'extraction du contenu d'un package et tente de choisir le meilleur miroir à partir duquel extraire les données de package. Si le miroir est inaccessible, n'a pas le contenu requis, ou est plus lent, l'API du client extrait le contenu de n'importe quel référentiel d'origine configuré. Les miroirs sont conçus pour le partage de contenu dans un ensemble de clients de confiance en miroir avec la fonctionnalité de mise en miroir dynamique `pkg.depotd`(1M). Les miroirs sont également destinés à être utilisés pour authentifier l'accès aux métadonnées du package, mais ils distribuent le contenu du package sans authentification. Par exemple, un client peut être configuré avec une origine `https` requérant une paire clé/certificat SSL pour accéder aux ressources et un miroir `http` fournissant le contenu du package. De cette manière, seuls les clients autorisés peuvent installer ou mettre à jour les packages, et il est possible de faire l'économie du temps système nécessaire à l'authentification lors de l'extraction du contenu d'un package. Un miroir peut être créé par la suppression de tous les sous-répertoires d'un référentiel, à l'exception de ceux

nommés `file` et à leurs parents. Un référentiel d'origine peut être également alloué en tant que miroir à l'aide du mode miroir de `pkg.depotd`(1M).

Facettes et variantes

Les logiciels peuvent avoir des composants optionnels et des composants incompatibles. Les environnements linguistiques et la documentation sont des exemples de composants optionnels. Les binaires SPARC ou x86 et les binaires de débogage et de non-débogage sont des exemples de composants incompatibles.

Dans IPS, les composants optionnels sont appelés des *facettes* et les composants incompatibles sont appelés des *variantes*. Les facettes et les variantes sont spécifiées sous forme de repères sur les actions de package. Chaque repère de facette et de variante possède un nom et une valeur. Une seule action peut avoir plusieurs repères de facette et de variante. Parmi les exemples de composants incluant plusieurs repères de facette et de variante, il faut citer un fichier d'en-tête propre à l'architecture utilisé par les développeurs ou un composant réservé à une zone globale SPARC.

`variant.arch=sparc` est un exemple de repère de variante. Voici un exemple de repère de facette : `facet.devel=true`. Il est souvent fait référence aux facettes et variantes sans les faire précéder de `facet.` et `variant.`.

Les facettes et les variantes sont des propriétés spéciales de l'image et ne peuvent pas être définies sur des packages individuels. Pour afficher les valeurs actuelles des facettes et variantes définies dans l'image, exécutez les commandes `pkg facet` et `pkg variant`, comme décrit à la page de manuel `pkg`(1). Pour modifier les valeurs des facettes et variantes définies dans l'image, exécutez les commandes `pkg change-facet` et `pkg change-variant`.

Les facettes sont booléennes : elles peuvent uniquement être définies sur `true` (activé) ou `false` (désactivé). Par défaut, toutes les facettes sont considérées comme définies sur `true` dans une image. Un repère de facette sur une action doit uniquement avoir la valeur `true` ; les autres valeurs ont des comportements indéterminés. Une facette définie sur l'image peut être une facette complète telle que `doc.man` ou un modèle tel que `locale.*`. L'intérêt de cette caractéristique est qu'elle vous permet de désactiver une partie de l'espace de noms d'une facette et de n'activer que des facettes individuelles en son sein. Par exemple, vous pouvez désactiver tous les environnements linguistiques, puis n'activer qu'un ou deux environnements linguistiques particuliers, comme illustré dans l'exemple suivant :

```
# pkg change-facet locale.*=false
[output about packages being updated]
# pkg change-facet locale.en_US=true
[output about packages being updated]
```

La plupart des variantes peuvent avoir un nombre de valeurs quelconque. Par exemple, il est possible de définir la variante `arch` sur `i386`, `sparc`, `ppc`, `arm` ou une autre architecture prise en charge par la distribution. (Seules les valeurs `i386` et `sparc` sont utilisées dans Oracle Solaris.) Les variantes `debug` constituent une exception. Il est possible de définir les variantes `debug` uniquement sur `true` ou `false`. Aucune autre valeur ne correspond à un comportement

défini. Si une action sur un fichier présente à la fois des versions avec et sans débogage, il faut définir explicitement la variante `debug` pour les deux versions, comme illustré dans l'exemple suivant :

```
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
```

Pour qu'un package utilisant une variante donnée puisse être installé, il faut que cette variante soit définie sur l'image. Les variantes `arch` et `zone` sont définies par le programme qui crée l'image et installe son contenu initial. Les variantes `debug.*` sont définies sur `false` dans l'image par défaut.

Les facettes et les variantes définies sur l'image déterminent si une action particulière est installée ou non.

- Les actions sans repère de facette ou de variante sont toujours installées.
- Les actions avec repères de facettes sont installées, sauf si toutes les facettes ou tous les modèles de facettes correspondant aux repères sont définis sur `false` dans l'image. Si une seule facette est définie sur `true` ou n'est pas explicitement définie (`true` est la valeur par défaut), l'action est installée.
- Les actions avec repères de variantes sont uniquement installées si les valeurs des repères de variantes sont les mêmes que celles définies dans l'image.
- Les actions comportant à la fois des repères de facettes et des repères de variantes sont installées si les facettes et les variantes autorisent l'installation des actions concernées.

Vous pouvez créer vos propres repères de facettes et de variantes. Les repères suivants sont couramment utilisés dans Oracle Solaris.

Nom de la variante	Valeurs possibles
`variant.arch`	`sparc`, `i386`
`variant.opensolaris.zone`	`global`, `nonglobal`
`variant.debug.*`	`true`, `false`

La liste suivante présente quelques exemples de repères de facettes utilisés dans Oracle Solaris :

```
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
```

Stratégies des images

Les stratégies d'image sont définies par des propriétés de l'image avec des valeurs booléennes. Reportez-vous à la section “Propriétés de l'image” dans la page de manuel pkg(1) pour une descriptions des propriétés `flush-content-cache-on-success` et `send-uuid` ainsi que des informations sur l'affichage et la modification de leurs valeurs.

Fichiers

Les images pkg(5) peuvent être stockées arbitrairement dans un plus grand système de fichiers. De ce fait, le jeton `$IMAGE_ROOT` sert à distinguer les chemins d'accès relatifs. Pour une installation système standard, `$IMAGE_ROOT` équivaut à /.

`$IMAGE_ROOT/var/pkg`	Répertoire des métadonnées pour une image complète ou partielle.
`$IMAGE_ROOT/.org.opensolaris,pkg`	Répertoire des métadonnées pour une image d'utilisateur.

Dans les métadonnées d'une image particulière, certains fichiers et répertoires contiennent des informations utiles lors de la réparation et de la récupération. Le jeton `$IMAGE_META` est utilisé pour désigner le répertoire de niveau supérieur qui contient les métadonnées. `$IMAGE_META` est en général l'un des deux chemins d'accès donnés ci-dessus.

`$IMAGE_META/lost+found`	Emplacement des répertoires entrant en conflit et fichiers déplacés au cours d'une opération sur les packages.
`$IMAGE_META/publisher`	Contient un répertoire pour chaque éditeur. Chaque répertoire stocke des métadonnées spécifiques à l'éditeur.

D'autres chemins dans l'arborescence des répertoires `$IMAGE_META` sont privés et sujets à modification.

Attributs

Reportez-vous à `attributes`(5) pour obtenir la description des attributs suivants :

TYPE D'ATTRIBUT	VALEUR DE L'ATTRIBUT
Disponibilité	`package/pkg`
Stabilité de l'interface	Non validé

Voir aussi

pkg(1), pkgsend(1), pkg.depotd(1m), pkg.sysrepo(1m), svcs(1), svcadm(1M)

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