이미지 패키징 시스템 매뉴얼 페이지

부품 번호: E25754
2012년 3월, E25754-03

ORACLE

120410@25097

목차

머리말

이 문서는 Oracle Solaris 11 시스템 설치 도구에 대한 매뉴얼 페이지를 제공합니다.

개요

다음은 각 매뉴얼 페이지 절 및 참조된 정보를 간략히 설명한 것입니다.

- 1절에서는 운영 체제에서 사용 가능한 명령에 대해 설명합니다.
- 1M절에서는 시스템 유지 관리 및 관리 용도로 사용되는 명령에 대해 설명합니다.
- 5절에서는 문자 세트 테이블과 같은 기타 정보에 대해 설명합니다.

매뉴얼 페이지에 대한 일반 형식은 다음과 같습니다. 각 매뉴얼 절의 매뉴얼 페이지는 일반적으로 이 순서를 따르지만 필요한 제목만 포함합니다. 예를 들어, 보고할 버그가 없을 경우 버그 절이 없습니다. 일반적인 매뉴얼 페이지에 대한 자세한 내용은 `man` 명령을 참조하십시오.

NAME

이 절에서는 문서화된 명령 또는 함수의 이름, 용도에 대한 간략한 설명을 차례로 제공합니다.

SYNOPSIS

이 절에서는 명령 또는 함수의 구문을 보여 줍니다. 명령 또는 파일이 표준 경로에 존재하지 않는 경우 전체 경로 이름이 표시됩니다. 다른 인수 순서가 필요하지 않은 경우 옵션 및 인수는 단일 문자 인수가 먼저 오고 인수가 있는 옵션이 뒤에 오는 사전순으로 정렬됩니다.

이 절에서 사용되는 특수 문자는 다음과 같습니다.

[ ] 대괄호입니다. 이러한 대괄호로 묶인 옵션 또는 인수는 선택적입니다. 괄호를 생략할 경우 인수를 지정해야 합니다.

. . . 생략 부호입니다. 이전 인수에 대한 값을 여러 개 제공하거나 이전 인수를 여러 번 지정할 수 있습니다. (예: *filename* . . .)

| 구분자입니다. 이 문자로 구분되는 인수 중 하나만 한 번에 지정할 수 있습니다.

	{ } 중괄호입니다. 중괄호로 묶인 옵션 및/또는 인수는 상호 독립적이므로 이 안에 묶인 모든 항목은 하나의 단위로 처리되어야 합니다.
프로토콜	이 절은 프로토콜 설명 파일을 지정하기 위한 세부절 3R에서만 표시됩니다.
DESCRIPTION	이 절에서는 서비스의 기능 및 동작을 정의합니다. 따라서 명령의 용도에 대해 간략히 설명합니다. 옵션 또는 예에 대해서는 설명하지 않습니다. 대화식 명령, 하위 명령, 요청, 매크로 및 함수는 사용법에서 설명됩니다.
OPTIONS	이 절에서는 각 옵션의 용도에 대한 간략한 요약과 함께 명령 옵션을 나열합니다. 옵션은 문자 순서대로, 그리고 개요 절에 표시되는 순서대로 나열됩니다. 옵션에 대해 사용 가능한 인수는 옵션에서 설명되며 적합한 경우 기본값이 제공됩니다.
OPERANDS	이 절에서는 명령 피연산자를 나열하며 명령 피연산자가 명령 작업에 어떤 방식으로 영향을 끼치는지 설명합니다.
OUTPUT	이 절에서는 명령으로 생성된 출력(표준 출력, 표준 오류 또는 출력 파일)에 대해 설명합니다.
반환 값	매뉴얼 페이지에서 값을 반환하는 함수가 설명되는 경우 이 절에서는 이러한 값을 나열하고 값이 반환되는 조건에 대해 설명합니다. 함수가 상수 값(예: 0 또는 –1)을 반환할 수 있을 경우 태그가 지정된 단락에 이러한 값이 나열됩니다. 그렇지 않을 경우 단일 단락에서 각 함수에 대한 반환 값을 설명합니다. void로 선언된 함수는 값을 반환하지 않으므로 반환 값에서 설명되지 않습니다.
오류	실패 시 거의 모든 함수는 실패 원인을 나타내는 오류 코드를 전역 변수 `errno`에 배치합니다. 이 절에서는 함수가 생성할 수 있는 모든 오류를 사전순으로 나열하고 각 오류를 발생시킬 수 있는 조건에 대해 설명합니다. 동일한 오류를 발생할 수 있는 조건이 두 개 이상인 경우 각 조건이 오류 코드 아래의 별도 단락에서 설명됩니다.
USAGE	이 절에서는 자세한 설명이 필요한 특수한 규칙, 기능 및 명령을 나열합니다. 여기서 나열되는 세부절에서는 내장 기능이 설명됩니다.

	명령 수정자 변수 표현식 입력 문법
EXAMPLES	이 절에서는 사용법 또는 명령이나 함수 사용 방법에 대한 예를 제공합니다. 가능한 경우 명령줄 항목 및 시스템 응답을 포함하는 전체 예가 표시됩니다. 예가 제공되는 경우 프롬프트는 `example%` 또는 `example#`(사용자가 수퍼 유저여야 하는 경우)로 표시됩니다. 예 뒤에는 설명, 변수 대체 규칙 또는 반환 값이 표시됩니다. 대부분의 예에서는 개요, 설명, 옵션 및 사용법 절의 개념을 보여 줍니다.
ENVIRONMENT VARIABLES	이 절에서는 명령 또는 함수로 영향을 받는 환경 변수와 결과에 대한 간략한 설명을 차례로 나열합니다.
EXIT STATUS	이 절에서는 명령이 호출 프로그램 또는 셸에 반환하는 값과 해당 값을 반환한 조건을 나열합니다. 일반적으로 성공적인 완료의 경우 0이 반환되며 다양한 오류 상태의 경우 0 이외의 값이 반환됩니다.
FILES	이 절에서는 매뉴얼 페이지가 참조하는 모든 파일, 관심 있는 파일 및 명령으로 만들어지거나 요구되는 파일을 나열합니다. 뒤에는 설명이 포함된 요약 또는 설명이 표시됩니다.
ATTRIBUTES	이 절에서는 속성 유형 및 해당 값을 정의하여 명령, 유틸리티 및 장치 드라이버의 특성을 나열합니다. 자세한 내용은 `attributes(5)` 매뉴얼 페이지를 참조하십시오.
SEE ALSO	이 절에서는 다른 매뉴얼 페이지에 대한 참고 자료, 자체 개발 설명서 및 외부 발행물을 나열합니다.
진단	이 절에서는 오류를 발생시킨 조건에 대한 간략한 설명과 함께 진단 메시지를 나열합니다.
경고	이 절에서는 작업 환경에 심각한 영향을 끼칠 수 있는 특수한 상태에 대한 경고를 나열합니다. 진단 목록은 아닙니다.
NOTES	이 절에서는 페이지의 어떤 부분에도 속하지 않는 추가 정보를 나열합니다. 이 절은 본론에서 벗어난 형식을 사용하여 특별히 관련이 있는 사항을 다룹니다. 여기서는 중요한 정보를 다루지 않습니다.

버그	이 절에서는 알려진 버그에 대해 설명하고 가능한 경우 해결 방법을 제안합니다.

참 고

사용자 명령

이름

packagemanager – 이미지 패키징 시스템 GUI

개요

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

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

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

설명

packagemanager는 이미지 패키징 시스템 소프트웨어 pkg(5)의 그래픽 사용자 인터페이스입니다.

패키지 관리자를 사용하면 다음 작업을 수행할 수 있습니다.

- 패키지를 검색, 설치 및 제거합니다.
- 게시자를 추가, 제거 및 수정합니다.
- 부트 환경을 만들고, 제거 및 관리합니다.

file 피연산자가 지정되고 해당 접미어가 .p5i인 경우 packagemanager는 하나 이상의 게시자와 각 게시자의 여러 패키지를 추가하는 웹 설치 모드로 실행됩니다.

옵션

다음 옵션이 지원됩니다.

-h 또는 --help
사용법 메시지를 표시합니다.

-i 또는 --info-install *file*
.p5i 파일을 지정하여 웹 설치 모드로 packagemanager를 실행할 수 있도록 허용합니다. *file*의 접미어는 .p5i여야 합니다.

-R 또는 --image-dir *dir*
자동으로 검색되는 이미지 대신 *dir*에 루트 지정된 이미지에서 작동합니다.

-U 또는 --update-all
사용 가능한 업데이트가 있는 설치된 패키지를 모두 업데이트합니다.

주 – package/pkg, package/pkg/package-manager 또는 package/pkg/update-manager 패키지를 업데이트해야 할 경우 packagemanager는 먼저 해당 패키지를 업데이트한 후 다시 시작하여 나머지 업데이트를 수행합니다.

피연산자

file 웹 설치 파일입니다. 이 파일의 접미어는 .p5i여야 합니다. 웹 설치에 대한 자세한 내용은 패키지 관리자 온라인 도움말을 참조하십시오.

예

예 1 현재 이미지에서 작동

현재 이미지에서 packagemanager를 호출합니다.

```
$ packagemanager
```

예 2 지정된 이미지에서 작동

/aux0/example_root에 저장된 이미지에서 packagemanager를 호출합니다.

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

예 3 웹 설치 모드로 호출

웹 설치 모드로 packagemanager를 호출합니다.

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

종료 상태

다음 종료 값이 반환됩니다.

0 모든 요소가 작동되었습니다.

1 오류가 발생했습니다.

2 잘못된 명령줄 옵션이 지정되었습니다.

파일

보다 큰 파일 시스템 내에서는 pkg(5) 이미지가 임의적으로 배치될 수 있으므로 $IMAGE_ROOT 토큰이 상대 경로를 구별하는 데 사용됩니다. 일반적인 시스템 설치에서는 $IMAGE_ROOT가 /와 동일합니다.

$IMAGE_ROOT/var/pkg	전체 또는 부분 이미지에 대한 메타 데이터 디렉토리입니다.
$IMAGE_ROOT/.org.opensolaris,pkg	사용자 이미지에 대한 메타 데이터 디렉토리입니다. 특정 이미지의 메타 데이터 내에서 특정 파일 및 디렉토리에는 손상 복구 및 복구 중 유용한 정보가 포함됩니다. $IMAGE_META 토큰은 메타 데이터가 포함된 최상위 레벨 디렉토리를 가리키는 데 사용됩니다. 일반적으로 $IMAGE_META는 위에 지정된 두 개 경로 중 하나입니다.
$IMAGE_META/gui-cache	프로그램 시작 및 게시자 간 전환 속도를 높이기 위해 packagemanager가 유지 관리하는 캐시된 메타 데이터의 위치입니다.

$IMAGE_META 디렉토리 계층의 기타 경로는 개인용이므로 변경할 수 있습니다.

속성

다음 속성에 대한 설명은 attributes(5)를 참조하십시오.

속성 유형	속성 값
Availability	package/pkg/package-manager

속성 유형	속성 값
Interface Stability	커밋되지 않음

참조

pm-updatemanager(1), pkg(1), pkg(5)

패키지 관리자 온라인 도움말

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

주

packagemanager는 이미지의 파일 및 디렉토리에 대한 충분한 작동 권한이 있는 상태에서 호출해야 합니다.

이름

pkg – 이미지 패키징 시스템 검색 클라이언트

개요

```
/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
```

설명

pkg는 이미지 패키징 시스템용 검색 클라이언트입니다. 유효한 구성을 사용할 경우 pkg를 호출하여 패키지 설치 및 이미지 호출 위치를 만들고 해당 이미지에 패키지를 설치할 수 있습니다. 하나 이상의 저장소 또는 패키지 아카이브에 있는 패키지를 사용 가능하도록 설정할 수 있는 게시자가 패키지를 게시합니다. pkg는 게시자의 저장소 또는 패키지 아카이브에서 패키지를 검색하여 이미지에 설치합니다.

게시자 이름은 개인, 개인 그룹 또는 조직을 하나 이상의 패키지에 대한 소스로 식별합니다. 게시자 이름이 충돌하지 않도록 하고 게시자를 쉽게 식별할 수 있도록 하려면 패키지 게시 엔티티를 게시자 이름으로 나타내는 도메인 이름을 사용하는 것이 가장 좋습니다.

저장소는 클라이언트가 패키지 내용(프로그램, 문서 등 패키지 내에 포함된 파일) 및 메타 데이터(패키지 이름, 설명 등 패키지에 대한 정보)를 게시하고 검색할 수 있는

위치입니다. 예를 들어, 이름이 example.org인 게시자의 저장소는 URI http://example.org/repository에 있을 수 있습니다.

또한 pkg는 패키지를 제거하고, 게시자 메타 데이터(예: 사용 가능한 패키지 목록)를 새로 고치고, 이미지에 설치된 패키지를 검증하고, 이미지에서 다양한 토큰을 질의할 수 있습니다. 해당 질의는 pkg(5) 저장소로 구성될 수도 있습니다.

이미지 유형에는 세 가지가 있습니다. 전체 시스템을 제공할 수 있는 전체 이미지, 전체 이미지(상위 이미지)에 링크되지만 고유의 전체 시스템을 제공하지 않는 부분 이미지 및 사용자 이미지입니다.

옵션

다음 옵션이 지원됩니다.

-R *dir*
*dir*에 루트 지정된 이미지에 대해 작업을 수행합니다. 디렉토리를 지정하지 않았거나 디렉토리가 환경에 따라 달라지지 않는 경우 기본값은 /입니다. 자세한 내용은 "환경 변수" 절을 참조하십시오.

--help 또는 -?
사용법 메시지를 표시합니다.

하위 명령

지원되는 하위 명령은 다음과 같습니다.

refresh [--full] [*publisher* ...]
지정된 각 게시자에 대해 클라이언트가 사용 가능한 패키지 및 게시자 메타 데이터 목록을 업데이트합니다. 게시자를 지정하지 않을 경우 모든 게시자에 대해 작업이 수행됩니다.

--full을 사용할 경우 증분 업데이트를 시도하는 대신 강제로 모든 게시자 메타 데이터를 전체적으로 검색하고 작업 중 사용된 프록시가 캐시된 데이터를 무시하도록 요청합니다. 이 옵션은 문제 해결 용도로 제공되는 것이므로 일반적인 용도로 사용하지 않아야 합니다.

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* ...
패키지를 설치하고, 이미지에 설치된 패키지가 허용하는 *pkg_fmri_pattern*과 일치하는 최신 버전으로 패키지를 업데이트합니다. 패키지의 최신 버전을 명시적으로 요청하려면 *pkg_fmri_pattern*의 버전 부분에 latest를 사용합니다. 예를 들어, vim@latest를 지정하십시오.

설치 프로세스 중 일부 구성 파일은 이름이 바뀌거나 대체될 수 있습니다. 패키지 시스템에서 보존할 파일을 확인하는 방법 및 패키지 작업 중 이러한 파일이 보존되는 방법에 대한 자세한 내용은 pkg(5) 매뉴얼 페이지의 “File 작업”을 참조하십시오.

패키지가 무시 목록에 있을 경우 패키지를 설치하면 해당 목록에서 패키지가 제거됩니다.

-g를 사용할 경우 패키지 데이터를 검색할 이미지의 소스 목록에 지정된 패키지 저장소 또는 아카이브를 일시적으로 추가합니다. 지정된 소스의 패키지를 이미지의 구성된 게시자에서도 사용할 수 있을 경우 클라이언트는 지정된 소스에서만 해당 패키지에 대한 내용을 검색합니다. 사용할 패키지의 버전을 결정할 때는 이미지에서 구성되었지만 지정된 소스에서 발견되지 않은 게시자가 우선합니다. 설치 또는 업데이트 후에는 게시자가 제공했지만 이미지에서 발견되지 않은 모든 패키지가 원본 없이 이미지 구성에 추가됩니다. 이 옵션은 여러 번 지정할 수 있습니다.

-n을 사용할 경우 패키지를 변경하지 않은 상태로 테스트 작업 실행을 수행합니다.

-q를 사용할 경우 요청된 작업 중 진행률 메시지를 숨깁니다.

-v를 사용할 경우 요청된 작업 중 자세한 진행률 메시지를 표시하고 상세 계획 정보(예: 페이싯, 조정자 및 변형 변경)를 표시합니다. 이 옵션을 여러 번 지정하여 표시되는 계획 정보의 양을 늘릴 수 있습니다.

--accept를 사용할 경우 업데이트 또는 설치된 패키지의 라이센스 계약 조건에 동의하는 것입니다. 이 옵션을 제공하지 않고 동의해야 할 패키지 라이센스가 있을 경우 설치 작업이 실패합니다.

--licenses를 사용할 경우 이 작업의 일부로 설치 또는 업데이트된 패키지에 대한 모든 라이센스를 표시합니다.

--no-backup-be를 사용할 경우 백업 부트 환경을 만들지 않습니다.

--no-be-activate를 사용할 경우 부트 환경이 만들어지면 다음 번 부트 시 이 부트 환경을 활성 BE로 설정하지 않습니다. 자세한 내용은 beadm(1M)을 참조하십시오.

--no-index를 사용할 경우 작업이 성공적으로 완료된 후 검색 색인을 업데이트하지 않습니다.

--no-refresh를 사용할 경우 이미지 게시자가 사용 가능한 패키지 및 기타 메타 데이터의 최신 목록을 검색하기 위해 저장소에 연결하려고 시도하지 않습니다.

--backup-be-name을 사용할 경우 지정된 인수를 사용하여 만들어진 백업 부트 환경의 이름을 지정합니다. --backup-be-name을 사용하면 -require-backup-be가 지정된 것을 의미합니다. beadm(1M)을 참조하십시오.

--be-name을 사용할 경우 새로 만들어진 부트 환경의 이름이 지정된 인수로 바뀝니다. --be-name을 사용하면 --require-new-be가 지정된 것을 의미합니다. beadm(1M)을 참조하십시오.

--require-backup-be를 사용할 경우 새 부트 환경이 만들어지지 않으면 항상 백업 부트 환경을 만듭니다. 이 옵션을 사용하지 않을 경우 백업 부트 환경이 이미지 정책을 기반으로 만들어집니다. 백업 부트 환경이 자동으로 만들어지는 경우에 대한 설명은 아래의 "이미지 등록 정보"에 나오는 be-policy를 참조하십시오.

`--require-new-be`를 사용할 경우 항상 새 부트 환경을 만듭니다. 이 옵션을 사용하지 않을 경우 부트 환경이 이미지 정책을 기반으로 만들어집니다. 부트 환경이 자동으로 만들어지는 경우에 대한 설명은 아래의 "이미지 등록 정보"에 나오는 `be-policy`를 참조하십시오. 이 옵션은 `--require-backup-be`와 함께 사용할 수 없습니다.

`--deny-new-be`를 사용할 경우 새 부트 환경을 만들지 않습니다. 새 부트 환경이 필요한 경우 이 작업이 수행되지 않습니다.

`--reject`를 사용할 경우 지정된 패턴과 이름이 일치하는 패키지가 설치되지 않도록 합니다. 일치하는 패키지가 이미 설치되어 있으면 이 작업의 일부로 제거됩니다. 그룹 종속성의 대상인 거부된 패키지는 무시 목록에 배치됩니다.

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

*pkg_fmri_pattern*과 일치하는 설치된 패키지를 제거합니다.

패키지가 그룹 종속성의 주체인 경우 해당 패키지를 제거하면 무시 목록에 배치됩니다. 아래의 `avoid` 하위 명령을 참조하십시오.

기타 모든 옵션의 경우 사용법 및 결과는 위의 `install` 명령을 참조하십시오.

`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* `...]`

인수를 사용하지 않거나 별표(*)가 제공된 패턴 중 하나인 경우 현재 이미지에 설치된 모든 패키지를, 설치된 패키지 및 게시자 구성에 따라 시스템에 적용되는 제약 조건에서 허용하는 최신 버전으로 업데이트합니다. 패키지의 최신 버전을 명시적으로 요청하려면 *pkg_fmri_pattern*의 버전 부분에 `latest`를 사용합니다. 예를 들어, `vim@latest`를 지정하십시오.

*pkg_fmri_pattern*을 제공할 경우 `update`는 설치되어 있으며 *pkg_fmri_pattern*과 일치하는 패키지를, 설치된 패키지 및 게시자 구성에 따라 시스템에 적용되는 패턴 및 제약 조건에서 허용하는 최신 버전으로 바꿉니다. 이미 설치된 버전보다 오래된 버전 또는 최신 버전을 지정하여 해당 위치에서 특정 패키지를 다운그레이드 또는 업그레이드할 수 있습니다. 특정 패키지를 업데이트할 때는 패키지 이름 바꾸기 또는 폐기 이외의 작업이 지원되지 않습니다.

`update`를 통해 다운그레이드할 패키지에 포함되며 원래 버전 설치 이후 변경된 적이 있는 보존된 모든 구성 파일은 `.update` 확장자를 사용하여 이름이 바뀝니다. 패키지 시스템에서 보존할 파일을 확인하는 방법 및 패키지 업그레이드 중 이러한 파일이 보존되는 방법에 대한 자세한 내용은 `pkg(5)` 매뉴얼 페이지의 “File 작업”을 참조하십시오.

`-f` 옵션을 사용할 경우 설치된 모든 패키지를 업데이트할 때 클라이언트 최신 검사를 실행하지 않습니다.

기타 모든 옵션의 경우 사용법 및 결과는 위의 install 명령을 참조하십시오.

list [-Hafnsuv] [-g *path_or_uri* ...] [--no-refresh] [*pkg_fmri_pattern* ...]
인수를 사용하지 않을 경우 버전 및 설치 상태와 같은 정보를 비롯한 현재 이미지의 패키지 목록을 표시합니다. 인수를 사용할 경우 지정된 패키지에 대한 정보를 표시합니다. 기본적으로 다른 구조 또는 영역 유형에 대한 패키지 변형은 제외됩니다. 일반적으로 출력은 다음과 같은 세 가지 열에 제공됩니다.

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

첫번째 열에는 패키지 이름이 포함됩니다. 설치된 패키지를 제공한 게시자(패키지가 설치되지 않은 경우 사용 가능한 패키지를 제공한 게시자)가 게시자 검색 순서의 첫번째가 아닌 경우 게시자 이름은 패키지 이름 뒤의 괄호 안에 나열됩니다. 두번째 열에는 패키지의 릴리스 및 분기 버전이 포함됩니다. 릴리스 및 분기 버전과 변형에 대한 내용은 pkg(5) 매뉴얼 페이지를 참조하십시오.

마지막 열에는 패키지 상태를 보여 주는 일련의 플래그가 포함됩니다.

- I 열의 i는 패키지가 설치되어 있음을 나타냅니다.
- F 열의 f는 패키지가 고정되어 있음을 나타냅니다.
- O 열의 o는 패키지가 오래되었음을 나타냅니다. O 열의 r은 폐기 형식으로 패키지의 이름이 바뀌었음을 나타냅니다.

-H를 사용할 경우 목록에서 헤더를 생략합니다.

-a를 사용할 경우 설치된 패키지 및 설치할 수 있는 최신 버전의 패키지를 나열합니다. 패키지는 설치된 통합 및 이미지 변형에 의해 허용되는 경우 설치할 수 있는 것으로 간주됩니다. 하나 이상의 패턴을 지정할 경우 지정된 패턴과 일치하며 설치된 통합 및 이미지 변형에 의해 허용되는 최신 버전이 나열됩니다. -a를 사용하지 않을 경우 설치된 패키지만 나열합니다.

-f 및 -a를 사용할 경우 통합 제약 조건 또는 설치 상태에 관계없이 모든 변형에 대한 전체 패키지의 모든 버전을 나열합니다. 해당 옵션을 사용할 때 명시적으로 패키지의 최신 버전을 나열하려면 *pkg_fmri_pattern*의 버전 부분에 latest를 사용하십시오. 예를 들어, vim@latest를 지정하십시오.

-g를 사용할 경우 지정된 패키지 저장소 또는 아카이브를 작업에 대한 패키지 데이터의 소스로 사용합니다. 이 옵션은 여러 번 지정할 수 있습니다. -n을 지정하지 않을 경우 -g를 사용하려면 -a가 필요합니다.

-n을 사용할 경우 설치 상태에 관계없이 알려진 모든 패키지의 최신 버전을 표시합니다.

-s를 사용할 경우 패키지 이름 및 요약을 제공하는 약식의 한 행을 표시합니다. 이 옵션은 -a, -n, -u 또는 -v와 함께 사용할 수 있습니다.

`-u`를 사용할 경우 사용 가능한 최신 버전이 있는 패키지만 나열합니다. 이 옵션은 `-g`와 함께 사용할 수 없습니다.

`-v`를 사용할 경우 게시자 및 전체 버전을 비롯하여 전체 패키지 FMRI를 모두 첫번째 열에 표시합니다(VERSION 열이 사라짐). 이 옵션은 `-a`, `-n` 또는 `-u`와 함께 사용할 수 있습니다.

`--no-refresh`를 사용할 경우 이미지 게시자가 사용 가능한 패키지의 최신 목록을 검색하기 위해 저장소에 연결하려고 시도하지 않습니다.

`info [-lr] [-g` *path_or_uri* `...] [--license] [`*pkg_fmri_pattern* `...]`
사용자가 읽을 수 있는 형식으로 패키지 정보를 표시합니다. FMRI 패턴을 여러 개 지정할 수 있습니다. 패턴을 사용하지 않을 경우 이미지에 설치된 모든 패키지에 대한 정보를 표시합니다.

`-g`를 사용할 경우 지정된 패키지 저장소 또는 아카이브를 작업에 대한 패키지 데이터의 소스로 사용합니다. 이 옵션은 여러 번 지정할 수 있습니다. `-g`를 사용하려면 `-r`이 필요합니다.

`-l`을 사용할 경우 설치된 패키지에 대해서만 정보를 표시합니다. 이 옵션이 기본값입니다.

`-r`을 사용할 경우 필요에 따라 이미지의 구성된 게시자 저장소에서 현재 설치되지 않은 패키지에 대한 정보를 검색하여 사용 가능한 최신 버전을 기반으로 패키지를 일치시킵니다. 이 옵션을 사용할 경우 패키지를 하나 이상 지정해야 합니다. `-r`을 사용하지 않을 경우 기본적으로 설치된 패키지만 표시됩니다.

`--license`를 사용할 경우 패키지에 대한 라이센스 텍스트를 표시합니다. 이 옵션은 `-l` 또는 `-r`과 함께 사용할 수 있습니다.

`contents [-Hmr] [-a` *attribute=pattern* `...] [-g` *path_or_uri* `...] [-o` *attribute*`,...] [-s` *sort_key*`] [-t` *action_type* `...] [`*pkg_fmri_pattern* `...]`
패키지 내용(작업 속성)을 표시합니다. 옵션 또는 피연산자를 사용하지 않을 경우 현재 이미지에 설치된 작업의 `path` 속성 값을 사전순으로 정렬하여 표시합니다. 작업 및 해당 속성에 대한 내용은 pkg(5) 매뉴얼 페이지의 "작업"을 참조하십시오. 아래의 의사 속성 이름 목록도 참조하십시오.

`-a`를 사용할 경우 옵션 인수에 명명된 속성을 가지며 값이 옵션 인수(속성 이름 뒤에 등호를 사용하여 표시)의 (glob) 패턴과 일치하는 해당 작업으로 출력을 제한합니다. `-a` 옵션을 여러 개 지정할 경우 옵션과 일치하는 작업이 표시됩니다.

`-g`를 사용할 경우 지정된 패키지 저장소 또는 아카이브에서 이 이미지에 설치할 수 있는 패키지에 대한 정보를 표시합니다. 설치할 수 있는 패키지에는 현재 설치된 패키지와 이 이미지의 설치 조건(예: 변형 및 페이싯 제한)을 충족하는 다른 패키지가 포함됩니다. 이 옵션은 여러 번 지정할 수 있습니다. `-g`를 사용하려면 `-r`이 필요합니다.

`-m`을 사용할 경우 이 이미지에 설치할 수 없는 작업을 포함하여 지정된 패키지의 모든 작업의 모든 속성을 표시합니다.

`-o`를 사용할 경우 나열된 속성들을 첫번째 속성 값에 따라 정렬하여 표시합니다. `-o` 옵션을 여러 번 지정할 수도 있고, 속성 이름을 쉼표로 구분하여 여러 개의 속성을 하나의 `-o` 옵션에 대한 인수로 지정할 수도 있습니다. 요청된 속성을 가진 작업만 표시됩니다.

`-r`을 사용할 경우 이 이미지에 구성된 게시자의 저장소에서 이 이미지에 설치할 수 있는 패키지의 사용 가능한 최신 버전에 대한 정보를 표시합니다. 설치할 수 있는 패키지에는 현재 설치된 패키지와 이 이미지의 설치 조건(예: 변형 및 페이싯 제한)을 충족하는 다른 패키지가 포함됩니다. 이 옵션을 사용할 경우 패키지를 하나 이상 지정해야 합니다.

`-s`를 사용할 경우 지정된 작업 속성을 기준으로 작업을 정렬합니다. 제공하지 않을 경우 기본적으로 경로 또는 `-o` 옵션에 의해 지정된 첫번째 속성을 기준으로 정렬합니다. `-s` 옵션은 여러 번 지정할 수 있습니다.

`-t`를 사용할 경우 지정된 유형의 작업만 나열합니다. 쉼표로 구분된 목록으로 여러 유형을 지정할 수 있습니다. 이 옵션은 여러 번 지정할 수 있습니다.

`-H`를 사용할 경우 목록에서 헤더를 생략합니다.

*pkg_fmri_pattern*을 사용할 경우 해당하는 명명된 패키지에 대한 정보만 표시합니다.

편의상 다음과 같은 특수한 의사 속성 이름을 여러 개 사용할 수 있습니다.

`action.hash`	작업에 페이로드가 있을 경우 작업의 해시 값입니다.
`action.key`	작업의 키 속성 값입니다. 예를 들어, `file` 작업의 경우 키 속성은 파일의 경로입니다. 일부 작업에는 키 속성이 없습니다.
`action.name`	작업의 이름입니다. 예를 들어, file 작업의 경우 `file`입니다.
`action.raw`	일치하는 작업의 모든 속성입니다.
`pkg.fmri`	작업을 포함하는 패키지의 전체 FMRI(예: `pkg://solaris/web/amp@0.5.11,5.11-0.169:20110705T153434Z`)입니다.
`pkg.name`	작업을 포함하는 패키지의 이름(예: `web/amp`)입니다.
`pkg.publisher`	작업을 포함하는 패키지의 게시자(예: `solaris`)입니다.
`pkg.shortfmri`	작업을 포함하는 패키지의 단축 형식 FMRI(예: `pkg://solaris/web/amp@0.5.11,5.11-0.169`)입니다.

`contents` 하위 명령과 `search` 하위 명령은 관련이 있습니다. 즉, 두 명령 모두 시스템에서 패키지 내용을 질의합니다. `contents` 하위 명령은 하나 이상의 설치된/설치 가능한 패키지의 작업을 지정된 옵션을 기반으로 출력을 필터링하여 표시합니다. `search` 하위 명령은 다른 방향에서 질의에 접근하여 사용자가 제공한 토큰을 포함하는 모든 패키지의 이름을 표시합니다.

각 하위 명령은 다른 하위 명령이 수행할 수 있는 일부 질의를 작성할 수 있습니다. 지정된 질의는 두 하위 명령 중 하나에서 보다 명확히 작성될 수 있으므로 하위 명령을 선택할 때는 주의해야 합니다.

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

*query*에 대한 일치 항목을 검색하고 결과를 표시합니다. 색인화되는 토큰은 작업에 종속되지만 컨텐츠 해시 및 경로 이름을 포함할 수 있습니다. 작업 및 해당 속성에 대한 내용은 pkg(5) 매뉴얼 페이지의 "작업"을 참조하십시오. 위의 `pkg contents`와 아래 `-o`에서 의사 속성 이름 목록도 참조하십시오.

기본적으로 질의는 정확히 일치될 일련의 용어로 해석됩니다. `?` 및 `*` 문자를 glob(3C) 스타일의 와일드카드로 사용할 수 있습니다. 그러면 질의 일치 유연성이 향상됩니다.

`-H`를 사용할 경우 헤더를 생략합니다.

`-I`를 사용할 경우 대소문자 구분 검색을 사용합니다.

기본적으로 `-a`를 사용할 경우 검색을 수행하고 일치 작업에 대한 정보를 표시합니다.

기본적으로 `search`는 현재 설치된 버전보다 오래된 패키지 및 현재 통합에 의해 제외된 패키지 버전에서 결과를 제거합니다. `-f`를 사용할 경우 패키지 버전에 관계없이 모든 결과를 표시합니다.

`-l`을 사용할 경우 이미지의 설치된 패키지를 검색합니다.

`-o`를 사용할 경우 결과 열을 제어할 수 있습니다. `-o` 옵션을 여러 번 지정할 수도 있고, 속성 이름을 쉼표로 구분하여 여러 개의 속성을 하나의 `-o` 옵션에 대한 인수로 지정할 수도 있습니다. 위에서 개략적으로 설명된 의사 속성 외에 다음 속성도 검색 결과에 대해 정의됩니다.

`search.match` 검색 질의와 일치한 문자열에 해당합니다.

`search.match_type` 검색 질의와 일치한 문자열을 포함하는 속성에 해당합니다.

`-p`를 사용할 경우 각 질의 용어와 일치하는 작업을 가진 패키지를 표시합니다. 이 옵션을 사용하는 것은 질의의 각 용어를 꺾쇠(`<>`)로 묶는 것과 동일합니다. `<>` 연산자에 대한 자세한 설명은 아래를 참조하십시오.

기본적으로, 그리고 `-r`을 사용할 경우 이미지의 게시자에 해당하는 저장소를 검색합니다.

`-s`를 사용할 경우 지정된 URI에 있는 pkg(5)를 검색합니다. 이 옵션은 여러 번 지정할 수 있습니다. 패키지 아카이브는 지원되지 않습니다.

`-l`과 `-r`(또는 `-s`)은 함께 지정할 수 있으며, 이 경우 로컬 검색과 원격 검색이 수행됩니다.

단순 토큰 일치 및 와일드카드 검색 외에 보다 복잡한 질의 언어도 지원됩니다. 작은따옴표 또는 큰따옴표(' 또는 ")를 사용하여 구문을 검색할 수 있습니다. pkg가 실제로 ' 또는 "를 확인하도록 셸을 고려해야 합니다.

AND 및 OR를 사용한 부울 검색이 지원됩니다. 필드 또는 구조화된 질의가 지원됩니다. 해당 질의에 대한 구문은 *pkg_name*: *action_type*:*key*:*token*입니다. 누락된 필드는 암시적으로 와일드카드로 처리됩니다. :basename:pkg 검색은 키가 basename이며 pkg 토큰과 일치하는 모든 패키지의 모든 작업 유형과 일치합니다. pkg_name 및 token 필드에는 명시적 와일드카드가 지원됩니다. action_type 및 key는 정확히 일치해야 합니다.

해당 작업을 포함하는 패키지로 작업을 변환하려면 <>를 사용하십시오. -a 옵션을 사용할 경우 token을 검색하면 token과 일치하는 작업에 대한 정보가 표시되지만, <token>을 검색하면 token과 일치한 작업을 포함하는 패키지 목록이 표시됩니다.

verify [-Hqv] [*pkg_fmri_pattern* ...]
현재 이미지에 설치된 패키지를 검증합니다. 관련 게시자에 대한 현재 서명 정책이 ignore이 아니면 정책을 기반으로 각 패키지의 서명이 검증됩니다. 서명 정책 적용 방법에 대한 설명은 아래의 "이미지 등록 정보"에 나오는 signature-policy를 참조하십시오.

-H를 사용할 경우 확인 출력에서 헤더를 생략합니다.

-q를 사용할 경우 어떤 내용도 인쇄하지 않지만 치명적 오류가 있으면 실패를 반환합니다.

-v를 사용할 경우 패키지와 관련된 정보 메시지를 포함합니다.

fix [--accept] [--licenses] [*pkg_fmri_pattern* ...]
pkg verify가 보고한 오류를 수정합니다. 설치된 패키지 컨텐츠에 대한 검증은 사용자 정의 컨텐츠 분석을 기반으로 수행되며 사용자 정의 컨텐츠 분석은 다른 프로그램과는 다른 결과를 반환할 수 있습니다.

--accept를 사용할 경우 업데이트 또는 설치된 패키지의 라이센스 계약 조건에 동의하는 것입니다. 이 옵션을 제공하지 않고 동의해야 할 패키지 라이센스가 있을 경우 작업이 실패합니다.

--licenses를 사용할 경우 이 작업의 일부로 설치 또는 업데이트할 패키지에 대한 모든 라이센스를 표시합니다.

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* ...)
파일을 전달된 그대로의 상태로 되돌립니다. 특정 값으로 태그가 지정된 모든 파일 또는 개별 파일을 되돌릴 수 있습니다. 파일 소유권과 보호도 함께 복원됩니다.

주의 – 몇몇 편집 가능한 파일을 해당 기본값으로 되돌리면 시스템을 부트할 수 없게 되거나 다른 오작동이 발생할 수 있습니다.

기타 모든 옵션의 경우 사용법 및 결과는 위의 install 명령을 참조하십시오.

mediator [-aH] [-F *format*] [*mediator ...*]
모든 조정자 또는 인수와 함께 지정된 특정 조정자의 현재 선택한 버전 및/또는 구현을 표시합니다.

-a를 사용할 경우 현재 설치된 패키지에 대해 설정할 수 있는 조정을 나열합니다.

-F를 사용할 경우 대체 출력 형식을 지정합니다. 지금은 tsv(탭으로 구분된 값)만 유효합니다.

-H를 사용할 경우 목록에서 헤더를 생략합니다.

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 ...*
현재 이미지의 지정된 조정자에 대한 버전 및/또는 구현을 설정합니다.

-I를 사용할 경우 사용하려는 조정된 인터페이스의 구현을 설정합니다. 기본적으로 버전이 지정되지 않을 경우 모든 구현 버전이 허용됩니다. 버전 없이 구현을 지정하려면 @ 기호를 추가하십시오.

-V를 사용할 경우 사용하려는 조정된 인터페이스의 버전을 설정합니다.

지정된 조정자 버전 및/또는 구현을 현재 사용할 수 없는 경우 지정된 조정자를 사용하는 링크가 제거됩니다.

기타 모든 옵션의 경우 사용법 및 결과는 위의 install 명령을 참조하십시오.

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 ...*
지정된 조정자의 버전 및/또는 구현을 시스템 기본값으로 되돌립니다.

-I를 사용할 경우 조정된 인터페이스의 구현만 되돌립니다.

-V를 사용할 경우 조정된 인터페이스의 버전만 되돌립니다.

기타 모든 옵션의 경우 사용법 및 결과는 위의 install 명령을 참조하십시오.

variant [-H] [variant.*variant_name ...*]
인수를 사용하지 않을 경우 이 이미지에 설정된 모든 변형의 현재 값을 표시합니다. 인수를 사용할 경우 이 이미지에 설정된 각각의 지정된 variant.*variant_name*의 값을 표시합니다.

-H를 사용할 경우 목록에서 헤더를 생략합니다.

변형에 대한 자세한 내용은 pkg(5) 매뉴얼 페이지의 "페이싯 및 변형"을 참조하십시오.

`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* `...`

현재 이미지에 설정된 지정된 변형의 값을 변경합니다.

옵션 사용법 및 결과는 위의 `install` 명령을 참조하십시오.

변형 값을 변경하면 패키지 컨텐츠가 제거, 업데이트 또는 설치될 수 있습니다. 또한 변형 값을 변경하면 새 이미지 구성에 맞게 전체 패키지가 설치, 업데이트 또는 제거될 수 있습니다. 변형에 대한 자세한 내용은 pkg(5) 매뉴얼 페이지의 "페이싯 및 변형"을 참조하십시오.

`facet [-H] [`*facet_name* `...]`

인수를 사용하지 않을 경우 `pkg change-facet` 명령을 사용하여 이 이미지에 명시적으로 설정된 모든 페이싯의 현재 값을 표시합니다. 인수를 사용할 경우 이 이미지에 설정된 각각의 지정된 *facet_name*의 값을 표시합니다.

`-H`를 사용할 경우 목록에서 헤더를 생략합니다.

페이싯에 대한 자세한 내용은 pkg(5) 매뉴얼 페이지의 "페이싯 및 변형"을 참조하십시오.

`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] ...`

현재 이미지에 설정된 지정된 페이싯의 값을 변경합니다.

페이싯은 `True` 또는 `False`로 설정할 수 있습니다. 페이싯을 `None`으로 설정하면 기본값인 `True`가 해당 페이싯에 적용됩니다. 즉, 페이싯에 적용되는 모든 작업이 설치됩니다. 작업에 대한 내용은 pkg(5) 매뉴얼 페이지의 "작업"을 참조하십시오.

옵션 사용법 및 결과는 위의 `install` 명령을 참조하십시오.

페이싯 값을 변경하면 패키지 컨텐츠가 제거, 업데이트 또는 설치될 수 있습니다. 또한 페이싯 값을 변경하면 새 이미지 구성에 맞게 전체 패키지가 설치, 업데이트 또는 제거될 수 있습니다. 페이싯에 대한 자세한 내용은 pkg(5) 매뉴얼 페이지의 "페이싯 및 변형"을 참조하십시오.

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

무시 목록의 지정된 패턴과 현재 일치하는 패키지 이름을 지정하여 패키지가 그룹 종속성의 대상인 경우 지정된 패키지를 무시합니다. 현재 설치되어 있지 않은 패키지만 무시할 수 있습니다. 패키지가 그룹 종속성의 대상일 경우 패키지를 제거하면 무시 목록에 놓입니다.

인수를 사용하지 않을 경우 해당 패키지에 대해 그룹 종속성을 가지는 패키지와 함께 무시된 각 패키지를 표시합니다.

필요한 경우 필수 종속성을 충족하기 위해 무시 목록에 있는 패키지가 설치됩니다. 해당 종속성이 제거되면 패키지가 제거됩니다.

`unavoid [`*pkg_fmri_pattern* `...]`
지정된 패키지를 무시 목록에서 제거합니다. 설치된 패키지의 그룹 종속성과 일치하는 무시 목록 패키지는 이 하위 명령을 사용하여 제거할 수 없습니다. 그룹 종속성과 일치하는 패키지를 무시 목록에서 제거하려면 패키지를 설치하십시오.

인수를 사용하지 않을 경우 무시된 패키지의 목록을 표시합니다.

`freeze [-n] [-c` *reason*`] [`*pkg_fmri_pattern*`] ...`
지정된 패키지를 지정된 버전으로 고정합니다. 버전을 지정하지 않을 경우 패키지가 설치되어야 하며 설치된 버전에서 고정됩니다. 고정된 패키지가 설치되거나 업데이트되는 경우 고정된 시점의 버전과 일치하는 버전으로 설치 또는 업데이트되어야 합니다. 예를 들어, 패키지가 1.2로 고정된 경우 1.2.1, 1.2.9, 1.2.0.0.1 등으로는 업데이트될 수 있지만 1.3이나 1.1은 될 수 없습니다. *pkg_fmri_pattern*에 표시된 게시자를 사용하여 일치하는 패키지를 찾습니다. 그러나 게시자 정보는 고정의 일부로 기록되지 않습니다. 패키지는 게시자가 아니라 해당 버전만 고려하여 고정됩니다. 이미 고정된 패키지를 고정하면 고정 버전이 새로 지정된 버전으로 바뀝니다.

패키지를 제공하지 않을 경우 현재 고정된 패키지에 대한 정보(패키지 이름, 버전, 패키지가 고정된 시간 및 관련 원인)가 표시됩니다.

패키지를 고정해도 패키지를 제거할 수 있습니다. 패키지를 제거하면 별도의 경고가 표시되지 않습니다.

`-c`를 사용할 경우 고정된 패키지와 함께 *reason*을 기록합니다. 고정으로 인해 설치나 업데이트가 실패할 경우 이유가 표시됩니다.

`-n`을 사용할 경우 패키지를 고정하지 않은 상태로 고정할 패키지 목록을 표시하여 테스트 작업 실행을 수행합니다.

`unfreeze [-n] [`*pkg_name_pattern* `] ...`
지정한 패키지에서 고정에 따른 제약 조건을 제거합니다. 제공한 버전은 모두 무시됩니다.

`-n`을 사용할 경우 패키지를 고정 해제하지 않은 상태로 고정을 해제할 패키지 목록을 표시하여 테스트 고정 해제 실행을 수행합니다.

`property [-H] [`*propname* `...]`
이미지 등록 정보를 표시합니다. 인수를 사용하지 않을 경우 모든 이미지 등록 정보에 대한 이름과 값을 표시합니다. 특정 등록 정보 이름 목록이 요청되면 해당 등록 정보에 대한 이름과 값을 표시합니다. 이미지 등록 정보에 대한 설명은 아래의 "이미지 등록 정보"를 참조하십시오.

`-H`를 사용할 경우 목록에서 헤더를 생략합니다.

`set-property` *propname propvalue*
기존 이미지 등록 정보를 업데이트하거나 새 이미지 등록 정보를 추가합니다.

`add-property-value` *propname propvalue*
기존 이미지 등록 정보에 값을 추가하거나 새 이미지 등록 정보를 추가합니다.

`remove-property-value` *propname propvalue*
값을 기존 이미지 등록 정보에서 제거합니다.

`unset-property` *propname* `...`
기존 이미지 등록 정보를 제거합니다.

`publisher [-HPn] [`*publisher* `...]`
게시자 정보를 표시합니다. 인수를 사용하지 않을 경우 모든 게시자, 해당 원본 URI 및 미러의 목록을 검색 기본 설정 순서대로 표시합니다. 특정 게시자가 요청되면 해당 게시자에 대한 상세 구성을 표시합니다.

`-H`를 사용할 경우 목록에서 헤더를 생략합니다.

`-P`를 사용할 경우 게시자 검색 순서의 첫번째 게시자만 표시합니다.

`-n`을 사용할 경우 사용으로 설정된 게시자만 표시합니다.

`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`
기존 게시자를 업데이트하거나 패키지 게시자를 추가합니다. 검색 순서에 영향을 끼치는 옵션을 지정하지 않을 경우 새 게시자가 검색 순서에 추가되어 마지막에 검색됩니다.

`-P` 또는 `--search-first`를 사용할 경우 지정된 게시자를 검색 순서의 첫번째로 설정합니다. 새 패키지를 설치하면 이 게시자가 먼저 검색됩니다. 이미 설치된 패키지에 대한 업데이트는 해당 게시자가 엄격히 유지되는 한 원래 패키지를 제공한 게시자와 동일한 게시자로부터 옵니다. `-P` 또는 `--search-first`를 `-p`와 함께 사용할 경우 추가된 게시자만 검색 순서의 첫번째로 배치됩니다.

`--non-sticky`를 사용할 경우 원래 이 게시자로부터 설치된 패키지에 대한 업데이트를 제공할 수 있는 이 게시자보다 더 높은 순위의 게시자를 지정합니다.

`--sticky`를 사용할 경우 이 게시자로부터 설치된 패키지에 대한 업데이트가 반드시 이 게시자로부터 오도록 지정합니다. 이 옵션이 기본 동작입니다.

--search-before를 사용할 경우 수정하려는 게시자가 지정된 게시자보다 먼저 검색되도록 게시자 검색 순서를 변경합니다. -p와 함께 사용할 경우 --search-before가 추가된 게시자에만 적용됩니다.

--search-after를 사용할 경우 수정하려는 게시자가 지정된 게시자 다음에 검색되도록 게시자 검색 순서를 변경합니다. -p와 함께 사용할 경우 --search-after가 추가된 게시자에만 적용됩니다.

--approve-ca-cert를 사용할 경우 지정된 인증서를 신뢰할 수 있는 CA 인증서로 추가합니다. 사용자가 승인한 CA 인증서에 대한 PEM 형식의 해시가 pkg publisher 명령의 상세 출력에 나열됩니다.

--revoke-ca-cert를 사용할 경우 PEM 형식의 지정된 해시가 있는 인증서를 해지된 것으로 처리합니다. 사용자가 해지한 CA 인증서의 해시가 pkg publisher 명령의 상세 출력에 나열됩니다.

--unset-ca-cert를 사용할 경우 승인된 인증서 목록 및 해지된 인증서 목록에서 지정된 해시가 있는 인증서를 제거합니다.

--set-property를 사용할 경우 기존 게시자 등록 정보를 업데이트하거나 새 게시자 등록 정보를 추가합니다.

--add-property-value를 사용할 경우 기존 게시자 등록 정보에 값을 추가하거나 새 게시자 등록 정보를 추가합니다.

--remove-property-value를 사용할 경우 기존 게시자 등록 정보에서 값을 제거합니다.

--unset-property를 사용할 경우 기존 게시자 등록 정보를 제거합니다.

-c 및 -k를 사용할 경우 클라이언트 SSL 인증서와 키를 각각 지정합니다.

-g(--add-origin)를 사용할 경우 지정된 URI 또는 경로를 지정된 게시자에 대한 원본으로 추가합니다. 이 옵션은 패키지 저장소 또는 아카이브의 위치여야 합니다.

-G(--remove-origin)를 사용할 경우 지정된 게시자의 원본 목록에서 URI 또는 경로를 제거합니다. 특수한 값 *를 사용하여 모든 원본을 제거할 수 있습니다.

--no-refresh를 사용할 경우 이미지 게시자가 사용 가능한 패키지 및 기타 메타데이터의 최신 목록을 검색하기 위해 저장소에 연결하려고 시도하지 않습니다.

--reset-uuid를 사용할 경우 게시자에 대해 이 이미지를 식별하는 새 고유 식별자를 선택합니다.

-m(--add-mirror)을 사용할 경우 URI를 지정된 게시자에 대한 미러로 추가합니다.

-M(--remove-mirror)을 사용할 경우 지정된 게시자에 대한 미러 목록에서 URI를 제거합니다. 특수한 값 *를 사용하여 모든 미러를 제거할 수 있습니다.

-p를 사용할 경우 지정된 저장소 URI에서 게시자 구성 정보를 검색합니다. 게시자를 지정하면 일치하는 게시자만 추가되거나 업데이트됩니다. 게시자를 지정하지 않으면 모든 게시자가 추가되거나 업데이트됩니다. 이 옵션은 -g, --add-origin, --G, --remove-origin, -m, --add-mirror, -M, --remove-mirror, --disable, --enable , --no-refresh 또는 --reset-uuid 옵션과 함께 사용할 수 없습니다.

-e(--enable)를 사용할 경우 게시자를 사용으로 설정합니다. -d(--disable)를 사용할 경우 게시자를 사용 안함으로 설정합니다. 사용 안함으로 설정된 게시자는 패키지 목록을 채울 때나 특정 패키지 작업(설치, 제거 및 업데이트)에서 사용되지 않습니다. 단, 이 경우에도 사용 안함으로 설정된 게시자에 대한 등록 정보는 설정하고 확인할 수 있습니다. 게시자가 하나만 존재하는 경우 해당 게시자를 사용 안함으로 설정할 수 없습니다.

/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*]

-p를 사용할 경우 지정된 저장소 URI에서 게시자 구성 정보를 검색합니다. 게시자를 지정하면 일치하는 게시자만 추가되거나 업데이트됩니다. 게시자를 지정하지 않으면 모든 게시자가 추가되거나 업데이트됩니다. -p 옵션과 함께 사용할 수 있는 다른 옵션에 대한 설명은 위의 pkg set-publisher를 참조하십시오. -p 옵션은 -g, --add-origin, -G, --remove-origin, -m, --add-mirror, -M, --remove-mirror, --disable, --enable, --no-refresh 또는 --reset-uuid 옵션과 함께 사용할 수 없습니다.

unset-publisher *publisher* ...

지정된 게시자에 연결된 구성을 제거합니다.

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

적용 가능한 이미지의 명령 내역을 표시합니다.

-H를 사용할 경우 목록에서 헤더를 생략합니다.

-t를 사용할 경우 형식이 %Y-%m-%dT%H:%M:%S(strftime(3C) 참조)인 쉼표로 구분된 시간 기록 목록에 대한 로그 레코드를 표시합니다. 시간 범위를 지정하려면 시작 시간 기록과 종료 시간 기록 사이에 하이픈(-)을 사용하십시오. 현재 시간에 대한 별칭으로 키워드 now를 사용할 수 있습니다. 지정한 시간 기록에 중복되는 시간 기록이나 날짜 범위가 포함되어 있으면 각 중복 기록 이벤트의 한 인스턴스만 출력됩니다.

-l을 사용할 경우 표준 형식과 함께 명령 결과, 명령이 완료된 시간, 사용된 클라이언트의 버전 및 이름, 작업을 수행한 사용자의 이름, 명령을 실행하는 중 발생한 오류를 포함하는 긴 형식의 로그 레코드를 표시합니다.

-n을 사용할 경우 지정된 수의 가장 최근 항목만 표시합니다.

-o를 사용할 경우 지정된 쉼표로 구분된 열 이름 목록을 사용하여 출력을 표시합니다. 유효한 열 이름은 다음과 같습니다.

be	이 작업이 시작된 부트 환경의 이름입니다.
be_uuid	이 작업이 시작된 부트 환경의 uuid입니다.
client	클라이언트의 이름입니다.
client_ver	클라이언트의 버전입니다.
command	이 작업에 사용된 명령줄입니다.
finish	이 작업이 완료된 시간입니다.
id	이 작업을 시작한 사용자 ID입니다.
new_be	이 작업으로 만들어진 새 부트 환경입니다.
new_be_uuid	이 작업으로 만들어진 새 부트 환경의 uuid입니다.
operation	작업의 이름입니다.
outcome	이 작업의 결과에 대한 요약입니다.
reason	이 작업의 결과에 대한 추가 정보입니다.
snapshot	이 작업 중 생성된 스냅샷입니다. 이는 작업이 성공적으로 완료된 후 스냅샷이 자동으로 제거되지 않은 경우에만 기록됩니다.
start	이 작업이 시작된 시간입니다.
time	이 작업을 수행하는 데 걸린 총 시간입니다. 1초 미만이 걸린 작업에 대해서는 0:00:00이 표시됩니다.
user	이 작업을 시작한 사용자 이름입니다.

command 또는 reason 열을 지정할 경우 출력 필드 구분을 유지하려면 해당 열이 -o 목록에서 마지막 항목이어야 합니다. 이러한 두 열은 동일한 history 명령에 표시될 수 없습니다.

부트 환경이 시스템에 더 이상 표시되지 않을 경우 be 또는 new_be에 대한 값 뒤에 별표(*)가 표시됩니다.

be 및 new_be에 대한 값을 확인하려면 be_uuid 또는 new_be_uuid 필드를 사용하여 현재 부트 환경 이름을 조회하십시오. 이후에 부트 환경의 이름이 바뀐 후 삭제된 경우 be 및 new_be에 대해 표시되는 값은 pkg 작업 시 기록된 값입니다.

purge-history
기존 내역 정보를 모두 삭제합니다.

rebuild-index
pkg search에 사용할 색인을 재구성합니다. 이는 일반 용도가 아닌 복구 작업입니다.

`update-format`
현재 버전에 대한 이미지의 형식을 업데이트합니다. 이 작업이 완료된 후에는 더 이상 이전 버전의 pkg(5) 시스템에서 이미지를 사용할 수 없습니다.

`version`
pkg(1)의 버전을 식별하는 고유 문자열을 표시합니다. 버전 간에 항상 이 문자열을 비교할 수 있는 것은 아닙니다.

`image-create [-FPUfz] [--force ] [--full | --partial | --user] [--zone] [-k` *ssl_key*`] [-c` *ssl_cert*`] [--no-refresh] [--variant` *variant_name*`=`*value* `...] [-g` *path_or_uri* `| --origin` *path_or_uri* `...] [-m` *uri* `| - -mirror` *uri* `...] [--set-property` *name_of_property* `=`*value*`][--facet` *facet_name*`=(True|False) ...] [(-p | --publisher) [`*name*`=] `*repo_uri*`] `*dir*
*dir*에 지정한 위치에 패키지 작업에 적합한 이미지를 만듭니다. 기본 이미지 유형은 -U(--user) 옵션으로 지정된 사용자입니다. 이미지 유형은 전체 이미지(--F 또는 --full) 또는 지정된 *dir* 경로를 묶는 전체 이미지에 링크된 부분 이미지(-P 또는 --partial)로 설정할 수 있습니다. -g 또는 --origin을 사용하여 추가 원본을 지정할 수 있습니다. --m 또는 --mirror를 사용하여 추가 미러를 지정할 수 있습니다.

패키지 저장소 URI는 -p 또는 --publisher 옵션을 사용하여 제공해야 합니다. 게시자 이름도 제공한 경우 이미지가 생성될 때 해당 게시자만 추가됩니다. 게시자 이름을 제공하지 않은 경우 지정한 저장소에서 인식하는 모든 게시자가 이미지에 추가됩니다. 초기 생성 작업이 끝나면 이 게시자와 관련된 카탈로그가 검색됩니다.

클라이언트 SSL 인증을 사용하는 게시자의 경우 -c 및 -k 옵션을 통해 클라이언트 키와 클라이언트 인증서를 등록할 수 있습니다. 이 키와 인증서는 이미지 생성 중 추가된 모든 게시자에 대해 사용됩니다.

비전역 영역 컨텍스트 내에서 이미지를 실행하려는 경우 -z(--zone) 옵션을 사용하여 적합한 변형을 설정할 수 있습니다.

-f(--force)를 사용할 경우 강제로 기존 이미지를 기반으로 이미지를 만듭니다. 이 옵션을 사용할 때는 신중해야 합니다.

--no-refresh를 사용할 경우 이미지 게시자가 사용 가능한 패키지 및 기타 메타데이터의 최신 목록을 검색하기 위해 저장소에 연결하려고 시도하지 않습니다.

--variant를 사용할 경우 지정된 변형을 표시된 값으로 설정합니다. 변형에 대한 자세한 내용은 pkg(5) 매뉴얼 페이지의 "페이싯 및 변형"을 참조하십시오.

--facet을 사용할 경우 지정된 페이싯을 표시된 값으로 설정합니다. 페이싯에 대한 자세한 내용은 pkg(5) 매뉴얼 페이지의 "페이싯 및 변형"을 참조하십시오.

--set-property를 사용할 경우 지정된 이미지 등록 정보를 표시된 값으로 설정합니다. 이미지 등록 정보에 대한 설명은 아래의 "이미지 등록 정보"를 참조하십시오.

이미지 등록 정보 다음 등록 정보는 이미지의 특성을 정의합니다. 이러한 등록 정보는 이미지의 용도, 컨텐츠 및 동작에 대한 정보를 저장합니다. 이미지에서 이러한 등록 정보의 현재 값을 보려면 `pkg property` 명령을 사용합니다. 이러한 등록 정보의 값을 수정하려면 `pkg set-property` 및 `pkg unset-property` 명령을 사용합니다.

`be-policy`
(문자열) 패키징 작업 중 부트 환경이 만들어질 시간을 지정합니다. 사용할 수 있는 값은 다음과 같습니다.

`default` 기본 BE 만들기 정책인 `create-backup`을 적용합니다.

`always-new` 다음 번 부트 시 활성으로 설정된 새 BE에서 모든 패키지 작업을 수행하여 모든 패키지 작업에 대한 재부트가 필요합니다. 백업 BE는 명시적으로 요청하지 않으면 생성되지 않습니다.

이 정책은 가장 안전하지만 재부트하지 않으면 패키지를 추가할 수 없기 때문에 대부분의 사이트 요구 사항에 비해 다소 엄격합니다.

`create-backup`
재부트가 필요한 패키지 작업의 경우 새 BE가 만들어지고 다음 번 부트 시 활성으로 설정됩니다. 패키지가 수정되거나 커널에 영향을 줄 수 있는 컨텐츠가 설치되고 작업이 라이브 BE에 영향을 주게 되면 백업 BE가 생성되지만 활성 BE로 설정되지는 않습니다. 백업 BE를 명시적으로 요청할 수도 있습니다.

이 정책은 새로 설치한 소프트웨어가 시스템 불안정을 야기하는 경우에 한해 위험을 가져올 수 있습니다. 이와 같은 경우는 거의 발생하지 않지만 가능성은 있습니다.

`when-required`
재부트가 필요한 패키지 작업의 경우 새 BE가 만들어지고 다음 번 부트 시 활성으로 설정됩니다. 백업 BE는 명시적으로 요청하지 않으면 생성되지 않습니다.

이 정책은 라이브 BE에 대한 패키징 변경으로 인해 추가 변경이 불가능할 경우 폴백할 수 있는 최근 BE가 없을 수 있으므로 위험성이 가장 높습니다.

`ca-path`
(문자열) SSL 작업에 대한 CA 인증서가 보관되는 디렉토리를 가리키는 경로 이름입니다. 이 디렉토리의 형식은 기본 SSL 구현에 따라 다릅니다. 인증된 CA 인증서를 보관하는 데 다른 위치를 사용하려면 다른 디렉토리를 가리키도록 이 값을 변경합니다. CA 디렉토리에 대한 요구 사항은 `SSL_CTX_load_verify_locations`(3openssl)의 `CApath` 부분을 참조하십시오.

기본값: `/etc/openssl/certs`

`check-certificate-revocation`
(부울) 이 등록 정보가 `True`로 설정되면 패키지 클라이언트는 인증서 발급 후 인증서가 취소되었는지 확인하기 위해 서명 확인에 사용되는 인증서의 CRL 배포 지점에 연결합니다.

기본값: `False`

`flush-content-cache-on-success`
(부울) 이 등록 정보가 `True`로 설정되면 패키지 클라이언트는 설치 또는 업데이트 작업이 완료될 때 컨텐츠 캐시에서 파일을 제거합니다. 업데이트 작업의 경우 소스 BE에서만 컨텐츠가 제거됩니다. 이 옵션을 변경하지 않을 경우 다음에 대상 BE에서 패키징 작업이 수행될 때 패키지 클라이언트는 해당 컨텐츠 캐시를 비웁니다.

이 등록 정보는 디스크 공간이 제한된 시스템에서 컨텐츠 캐시 크기를 작게 유지하는 데 사용할 수 있습니다. 이 등록 정보는 작업이 완료되는 데 걸리는 시간을 늘릴 수 있습니다.

기본값: `True`

`mirror-discovery`
(부울) 이 등록 정보는 mDNS 및 DNS-SD를 사용하여 링크 로컬 내용을 검색하도록 클라이언트에 알립니다. 이 등록 정보가 `True`로 설정되면 클라이언트는 동적으로 검색하는 미러에서 패키지 컨텐츠를 다운로드합니다. mDNS를 통해 내용을 보급하는 미러를 실행하려면 `pkg.depotd` (1M)를 참조하십시오.

기본값: `False`

`send-uuid`
(부울) 네트워크 작업을 수행할 때 이미지의 UUID(Universally Unique Identifier)를 보냅니다. 사용자가 이 옵션을 사용 안함으로 설정할 수는 있지만 일부 네트워크 저장소의 경우 UUID를 제공하지 않는 클라이언트와의 통신을 거부할 수도 있습니다.

기본값: `True`

`signature-policy`
(문자열) 이미지에 패키지를 설치하거나 이미지의 패키지를 업데이트, 수정 또는 확인할 때 매니페스트에 대해 수행할 검사를 결정합니다. 패키지에 적용되는 최종 정책은 이미지 정책과 게시자 정책의 조합에 따라 달라집니다. 두 정책을 조합할 경우 수준은 둘 중 더 엄격한 정책을 적용했을 때와 같습니다. 기본적으로 패키지 클라이언트는 인증서가 취소되었는지 여부를 확인하지 않습니다. 이러한 검사를 사용으로 설정하려면 `check-certificate-revocation` 이미지 등록 정보를 `True`로 설정합니다. 이 경우 클라이언트가 외부 웹 사이트에 연결해야 할 수도 있습니다. 사용할 수 있는 값은 다음과 같습니다.

`ignore`	모든 매니페스트에 대해 서명을 무시합니다.
`verify`	서명이 있는 모든 매니페스트가 유효하게 서명되었는지 확인하지만 설치된 모든 패키지에 대해 서명을 요구하지는 않습니다. 이것이 기본값입니다.
`require-signatures`	새로 설치된 모든 패키지가 유효한 서명을 적어도 하나 이상 포함하도록 요구합니다. 또한 `pkg fix` 및 `pkg verify` 명령은 설치된 패키지에 유효한 서명이 없을 경우 경고 메시지를 표시합니다.

require-names
: require-signatures와 동일한 요구 사항이 적용되지만 signature-required-names 등록 정보에 나열되는 문자열이 서명의 트러스트 체인을 확인하기 위해 사용되는 인증서의 공통 이름으로 나타나야 합니다.

signature-required-names
: (문자열 목록) 패키지의 서명을 검증하는 동안 인증서의 공통 이름으로 표시되어야 하는 이름의 목록입니다.

trust-anchor-directory
: (문자열) 이미지에 대한 신뢰 앵커를 포함하는 디렉토리의 경로 이름입니다. 이 경로는 이미지의 상대 경로입니다. 기본값은 ignore입니다.

use-system-repo
: (부울) 이 등록 정보는 이미지가 시스템 저장소를 이미지 및 게시자 구성에 대한 소스로 사용해야 하며 제공된 게시자와의 통신을 위한 프록시로 사용해야 할지 여부를 나타냅니다. 기본값은 False입니다. 시스템 저장소에 대한 자세한 내용은 pkg.sysrepo(1M)를 참조하십시오.

게시자 등록 정보 다음 등록 정보는 특정 게시자에 대한 서명 정책을 정의합니다. 동일한 이름의 이미지 등록 정보가 이미지의 서명 정책을 정의합니다. 특정 게시자에 대한 이러한 등록 정보의 현재 값을 보려면 pkg publisher *publisher_name* 명령을 사용합니다. 이러한 게시자 서명 정책 등록 정보의 값을 수정하려면 pkg set-publisher 명령의 --set-property 및 --unset-property 옵션을 사용합니다.

signature-policy
: (문자열) 이 등록 정보는 특정 게시자의 패키지에만 적용된다는 점을 제외하고 같은 이름의 이미지 등록 정보와 동일하게 작동합니다.

signature-required-names
: (문자열 목록) 이 등록 정보는 특정 게시자의 패키지에만 적용된다는 점을 제외하고 같은 이름의 이미지 등록 정보와 동일하게 작동합니다.

예

예 1 구성된 게시자를 사용하여 이미지 만들기

/aux0/example_root에 저장된 example.com 게시자를 사용하여 전체 이미지를 새로 만듭니다.

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

예 2 추가 원본 및 미러를 지정하여 이미지 만들기

추가 미러와 두 개의 추가 원본을 가지며 /aux0/example_root에 저장된 example.com 게시자를 사용하여 전체 이미지를 새로 만듭니다.

```
$ 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/ \
```

예 2 추가 원본 및 미러를 지정하여 이미지 만들기 *(계속)*

```
/aux0/example_root
```

예 3 구성된 게시자를 사용하지 않고 이미지 만들기

/aux0/example_root에 구성된 게시자를 사용하지 않고 전체 이미지를 새로 만듭니다.

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

예 4 패키지 설치

현재 이미지에 최신 버전의 widget 패키지를 설치합니다.

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

예 5 지정된 패키지 내용 나열

system/file-system/zfs 패키지의 내용을 나열합니다. 작업 이름, 파일 모드(정의된 경우), 크기(정의된 경우), 경로 및 대상(링크인 경우)을 표시합니다. 모든 작업에 사용 가능한 action.name 속성을 지정하면 여기에 적절하지 않은 모든 작업에 대한 행이 표시되므로 작업을 dir, file, link 및 hardlink 유형으로 제한합니다.

```
$ 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
...
```

예 6 두 패키지의 지정된 내용 나열

path 속성이 .desktop 또는 .png로 끝나는 작업의 패키지 이름 및 경로 속성으로 표시를 제한하여 web/browser/firefox 및 mail/thunderbird의 내용을 나열합니다.

```
$ 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
```

예 6 두 패키지의 지정된 내용 나열 *(**계속**)*

```
mail/thunderbird    usr/share/applications/thunderbird.desktop
web/browser/firefox usr/share/pixmaps/firefox-icon.png
mail/thunderbird    usr/share/pixmaps/thunderbird-icon.png
...
```

예 7 패키지 검색

패키지 데이터베이스에서 bge 토큰을 검색합니다.

```
$ 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
```

토큰은 /kernel/drv/ *arch*/bge를 나타내는 file 작업에 대한 기본 이름이자 드라이버 이름으로서 driver/network/bge 패키지에 있습니다.

예 8 지정된 패키지에 종속되는 패키지 검색

package/pkg에 종속되는 설치된 패키지를 검색합니다.

```
$ 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
```

예 9 종속성 검색

설치된 패키지에서 모든 incorporate 종속성을 검색합니다.

```
$ 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
...
```

예 10 게시자 추가

http://www.example.com/repo에 있는 저장소를 사용하여 example.com 게시자를 새로 추가합니다.

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

예 11 키와 인증서를 사용하여 게시자 추가

https://secure.example.com/repo에 있는 보안 저장소 및 /root/creds 디렉토리에 저장된 키와 인증서를 사용하여 example.com 게시자를 새로 추가합니다.

예 11 키와 인증서를 사용하여 게시자 추가 *(계속)*

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

예 12 추가 후 자동으로 게시자 구성

자동 구성을 통해 /export/repo에 있는 저장소를 사용하여 게시자를 새로 추가합니다.

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

예 13 추가 후 수동으로 게시자 구성

수동 구성을 통해 /export/repo/example.com에 있는 저장소를 사용하여 example.com 게시자를 새로 추가합니다.

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

예 14 서명된 모든 패키지 확인

서명된 모든 패키지를 확인하도록 이미지를 구성합니다.

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

예 15 모든 패키지에 서명 필요

모든 패키지가 서명되고 example.com 문자열이 트러스트 체인의 인증서 중 하나에 대한 공통 이름으로 표시되어야 하도록 이미지를 구성합니다.

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

예 16 지정된 게시자의 모든 패키지에 서명 필요

example.com 게시자로부터 설치된 모든 패키지가 서명되어야 하도록 이미지를 구성합니다.

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

예 17 트러스트 체인에 지정된 문자열 필요

유효한 것으로 간주되도록 서명 트러스트 체인에 표시되어야 할 이미지의 공통 이름 목록에 foo 문자열을 추가합니다.

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

예 18 지정된 게시자에 대한 트러스트 체인에서 문자열 제거

example.com 게시자에 대한 서명을 검증하기 위해 표시되어야 할 공통 이름 목록에서 foo 문자열을 제거합니다.

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

예 19 신뢰할 수 있는 CA 인증서 추가

/tmp/example_file.pem에 저장된 인증서를 example.com 게시자에 대한 신뢰할 수 있는 CA 인증서로 추가합니다.

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

예 20 인증서 해지

인증서가 example.com의 패키지에 대한 서명을 검증하지 않도록 example.com 게시자에 대한 해시가 a12345인 인증서를 해지합니다.

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

예 21 인증서 무시

사용자가 a12345 인증서를 추가 또는 해지했음을 pkg가 무시하도록 설정합니다.

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

예 22 패키지 다운그레이드

설치된 패키지 foo@1.1을 이전 버전으로 다운그레이드합니다.

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

예 23 충돌하는 패키지 설치 전환

두 개의 패키지가 충돌하는 경우 설치된 패키지를 변경합니다. 패키지 A가 패키지 B 또는 패키지 C에 종속되며 B와 C는 함께 사용할 수 없다고 가정합니다. A와 B가 설치된 경우 다음 명령을 사용하여 A를 제거하지 않고 B 대신 C를 사용하도록 전환합니다.

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

예 24 패키지 아카이브의 패키지 나열

패키지 아카이브에 있는 전체 패키지의 모든 버전을 나열합니다.

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

예 25 패키지 저장소의 패키지 나열

저장소에 있는 전체 패키지의 모든 버전을 나열합니다.

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

예 26 패키지 아카이브의 패키지에 대한 정보 표시

패키지 아카이브에 있는 최신 버전의 패키지에 대한 패키지 정보를 표시합니다. 패키지는 현재 설치되어 있을 수도 있고 설치되어 있지 않을 수도 있습니다.

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

예 27 패키지 아카이브에 있는 패키지의 내용 표시

패키지 아카이브에 있는 패키지의 내용을 표시합니다. 패키지는 현재 설치되어 있지 않습니다.

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

예 28 모든 게시자 원본 및 미러 제거

게시자에 대한 원본 및 미러를 모두 제거하고 새 원본을 추가합니다.

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

환경 변수

PKG_IMAGE
: 패키지 작업에 사용할 이미지가 포함된 디렉토리입니다. -R을 지정할 경우 무시됩니다.

PKG_CLIENT_CONNECT_TIMEOUT
: 각 시도에 대한 전송 작업 중 클라이언트가 작업을 중단하기 전까지 연결을 시도하기 위해 기다릴 시간(초)입니다. 0 값은 무기한 대기를 의미합니다.

  기본값: 60

PKG_CLIENT_LOWSPEED_TIMEOUT
: 전송 작업 중 클라이언트가 작업을 중단하기 전까지 lowspeed 제한(1024바이트/초) 아래로 떨어진 시간(초)입니다. 0 값은 작업을 중단하지 않음을 의미합니다.

  기본값: 30

PKG_CLIENT_MAX_CONSECUTIVE_ERROR
: 클라이언트가 작업을 중단하기 전까지의 최대 일시 전송 오류 수입니다. 0 값은 작업을 중단하지 않음을 의미합니다.

  기본값: 4

PKG_CLIENT_MAX_REDIRECT
: 전송 작업 중 연결이 중단되기 전까지 허용되는 최대 HTTP 또는 HTTPS 재지정 수입니다. 0 값은 작업을 중단하지 않음을 의미합니다.

  기본값: 5

PKG_CLIENT_MAX_TIMEOUT
: 클라이언트가 작업을 중단하기 전까지 수행되는 호스트당 최대 전송 시도 횟수입니다. 0 값은 작업을 중단하지 않음을 의미합니다.

  기본값: 4

http_proxy, https_proxy
: HTTP 또는 HTTPS 프록시 서버입니다.

종료 상태

다음 종료 값이 반환됩니다.

0 명령이 성공했습니다.

1 오류가 발생했습니다.

2 잘못된 명령줄 옵션이 지정되었습니다.

3 여러 명령이 요청되었지만 일부만 성공했습니다.

4 변경된 내용이 없어 수행할 작업이 없습니다.

5 라이브 이미지에 대해 요청된 작업을 수행할 수 없습니다.

6 설치 또는 업데이트하려는 패키지의 라이센스가 동의되지 않아 요청된 작업을 완료할 수 없습니다.

7 이미지를 현재 다른 프로세스가 사용하고 있어 수정할 수 없습니다.

99 예상치 않은 예외가 발생했습니다.

파일

보다 큰 파일 시스템 내에서는 pkg(5) 이미지가 임의적으로 배치될 수 있습니다. 다음 파일 설명에서 $IMAGE_ROOT 토큰은 상대 경로를 구별하는 데 사용됩니다. 일반적인 시스템 설치에서는 $IMAGE_ROOT가 /와 동일합니다.

$IMAGE_ROOT/var/pkg 전체 또는 부분 이미지에 대한 메타 데이터 디렉토리입니다.

$IMAGE_ROOT/.org.opensolaris,pkg 사용자 이미지에 대한 메타 데이터 디렉토리입니다.

특정 이미지의 메타 데이터 내에서 특정 파일 및 디렉토리에는 손상 복구 및 복구 중 유용한 특정 정보가 포함될 수 있습니다. $IMAGE_META 토큰은 메타 데이터가 포함된 최상위 레벨 디렉토리를 가리킵니다. 일반적으로 $IMAGE_META는 위에 지정된 두 개 경로 중 하나입니다.

$IMAGE_META/lost+found 패키지 작업 중 이동된 충돌하는 디렉토리 및 파일의 위치입니다. 제거된 디렉토리의 패키지화되지 않은 컨텐츠의 위치입니다.

$IMAGE_META/publisher 각 게시자에 대한 디렉토리를 포함합니다. 각 디렉토리는 게시자 특정 메타 데이터를 저장합니다.

$IMAGE_META 디렉토리 계층의 기타 경로는 개인용이므로 변경할 수 있습니다.

속성

다음 속성에 대한 설명은 attributes(5)를 참조하십시오.

속성 유형	속성 값
Availability	package/pkg

속성 유형	속성 값
Interface Stability	커밋되지 않음

참조

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

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

이름

pkgdepend – 이미지 패키징 시스템 종속성 분석기

개요

```
/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 ...
```

설명

pkgdepend는 패키지 종속성을 생성하고 확인하는 데 사용됩니다. 패키지는 다른 패키지의 파일에 종속될 수 있습니다. 일반적으로 pkgdepend는 두 가지 패스, 즉 파일 종속성 생성 및 파일이 종속된 패키지 확인에 사용됩니다.

generate 하위 명령은 패키지 내용을 검사하고 패키지에 필요한 외부 파일을 확인합니다.

resolve 하위 명령은 generate 단계에서 파일 목록을 가져온 후 패키지의 참조 세트를 검색하여 해당 종속 파일이 포함된 패키지의 이름을 확인합니다. 종속 파일이 검색된 패키지의 참조 세트는 현재 게시자의 시스템에 설치된 패키지입니다.

전달된 파일의 여러 구성 요소는 종속성 정보의 소스로 사용됩니다.

ELF — 전달된 파일의 ELF 헤더는 가져온 정보를 수정하는 -k 및 -D 옵션을 사용하여 종속성 정보에 대해 분석됩니다. ELF 종속성에 대한 자세한 내용은 ldd(1) 및 **Linker and Libraries Guide**를 참조하십시오.

스크립트 — 인터프리터를 참조하는 #! 행이 포함된 셸 스크립트는 해당 인터프리터를 전달하는 패키지에 대한 종속성을 생성합니다.

Python — Python 스크립트는 먼저 스크립트로 분석됩니다. 스크립트가 선언한 가져오기가 종속성 정보의 소스로 사용될 수도 있습니다.

하드 링크 — 매니페스트의 하드 링크는 링크 대상을 전달하는 패키지에 대한 종속성을 생성합니다.

SMF — require_all 종속성을 포함하는 전달된 SMF 서비스 매니페스트는 해당 FMRI를 제공한 SMF 매니페스트를 전달하는 패키지에 대한 종속성을 생성합니다.

옵션

다음 옵션이 지원됩니다.

-R *dir* — *dir*에 루트 지정된 이미지에 대해 작업을 수행합니다. 디렉토리를 지정하지 않았거나 디렉토리가 환경에 따라 달라지지 않는 경우 기본값은 /입니다. 자세한 내용은 "환경 변수" 절을 참조하십시오.

--help 또는 -? — 사용법 메시지를 표시합니다.

하위 명령

지원되는 하위 명령은 다음과 같습니다.

generate [-IMm] -d *dir* [-d *dir*] [-D *name=value*] [-k *path*] *manifest_file*
*manifest_file*에 의해 지정된 매니페스트의 파일에 대한 종속성을 생성합니다.

-I를 사용할 경우 *manifest_file* 내에서 충족되는 종속성을 표시합니다. -I 결과를 pkgdepend resolve와 함께 사용하지 마십시오.

-M을 사용할 경우 분석될 수 없는 파일 유형의 목록을 표시합니다.

-m을 사용할 경우 나중에 추가되는 검색된 종속성을 사용하여 원래 매니페스트를 반복합니다.

-d를 사용할 경우 매니페스트 파일을 검색할 디렉토리 목록에 *dir*을 추가합니다.

각 -D의 경우 ELF 파일 종속성에 대한 실행 경로에서 *name* 토큰을 확장할 방법으로 *value*를 추가합니다.

각 -k의 경우 커널 모듈을 검색할 실행 경로 목록에 *path*를 추가합니다. -k 인수를 사용하면 /kernel 및 /usr/kernel인 기본 경로가 제거됩니다.

작업 또는 매니페스트 속성 pkg.depend.runpath를 사용하여 작업 또는 매니페스트별로 실행 경로(예: -k 옵션에 의해 지정된 경로)를 지정할 수도 있습니다. pkg.depend.runpath 속성의 값은 사용할 경로의 문자열을 콜론으로 구분한 것입니다.

사용된 -k는 매니페스트 또는 작업의 pkg.depend.runpath 속성 세트로 대체됩니다.

분석하려는 파일에 대한 표준 시스템 실행 경로를 포함시키려는 경우 특수 토큰 $PKGDEPEND_RUNPATH를 pkg.depend.runpath 속성 값의 구성 요소로 사용할 수 있습니다.

자동으로 종속성이 생성되지 않도록 하려는 경우가 있습니다. 예를 들어, 패키지가 일련의 모듈을 가져오는 샘플 Python 스크립트를 전달하는 경우 샘플 스크립트로 가져온 해당 모듈은 샘플 스크립트를 전달하는 패키지에 대한 종속성이 아닙니다. 작업 또는 매니페스트 속성 pkg.depend.bypass-generate를 사용하여 지정된 파일에 대해 종속성이 생성되지 않도록 할 수 있습니다.

pkg.depend.bypass-generate 값은 파일 이름과 일치하는 Python 정규 표현식입니다. 정규 표현식은 파일 경로 시작 및 끝 부분에서 암시적으로 앵커됩니다. 다음 예에 지정된 값의 경우 this/that과는 일치하지만 something/this/that/the/other와는 일치하지 않습니다.

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

Python 정규 표현식 구문에 대한 자세한 내용을 보려면 pydoc re 명령을 사용하거나 http://docs.python.org/dev/howto/regex.html에 있는 보다 완전한 설명서를 참조하십시오.

pkgdepend generate 입력 매니페스트에 SMF 매니페스트 파일이 포함된 경우 해당 SMF 매니페스트 파일로 선언된 SMF 서비스나 인스턴스가 pkgdepend 출력에 포함됩니다. 이러한 SMF 서비스나 인스턴스는 set 작업의 형태로 org.opensolaris.smf.fmri 이름으로 포함됩니다.

`resolve [-moSv] [-d` *output_dir*`] [-s` *suffix*`]` *manifest_file* `...`
파일에 대한 종속성을 해당 파일을 전달하는 패키지에 대한 종속성으로 변환합니다. 종속성은 먼저 명령줄에 지정된 매니페스트에 대해 확인된 후 시스템에 설치된 패키지에 대해 확인됩니다. 기본적으로 각 매니페스트에 대한 종속성은 이름이 *manifest_file*`.res`인 파일에 배치됩니다.

`-m`을 사용할 경우 확인된 종속성을 추가하기 전에 제거된 `generate` 단계에서 생성된 종속성을 사용하여 매니페스트를 반복합니다.

`-o`를 사용할 경우 표준 출력에 결과를 기록합니다. 이 옵션은 사용자 소비에 사용됩니다. 파일에 이 출력을 추가하면 매니페스트가 잘못될 수 있습니다. 매니페스트 처리를 위한 파이프 행에는 `-o` 대신 `-d` 또는 `-s` 옵션을 사용하는 것이 좋습니다.

`-d`를 사용할 경우 *output_dir*의 별도 파일에서 제공된 각 매니페스트에 대해 확인된 종속성을 기록합니다. 기본적으로 각 파일의 기본 이름은 해당 파일에 기록된 종속성의 소스인 매니페스트의 이름과 동일합니다.

`-s`를 사용할 경우 각 출력 파일에 대해 확인된 종속성의 소스인 파일의 기본 이름에 *suffix*를 추가합니다. "."가 제공되지 않은 경우 *suffix* 앞에 추가됩니다.

`-S`를 사용할 경우 명령줄에 지정된 매니페스트에 대해서만 확인하고 시스템에 설치된 매니페스트에 대해서는 확인하지 않습니다.

`-v`를 사용할 경우 메타 데이터를 디버깅하는 추가 패키지 종속성을 포함합니다.

예

예 1 종속성 생성

`foo`(컨텐츠 디렉토리가 `./bar/baz`에 있음)에서 기록된 매니페스트에 대한 종속성을 생성하고 `foo.fdeps`에 결과를 저장합니다.

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

예 2 종속성 확인

`foo.fdeps` 및 `bar.fdeps`에서 상호 간에 대해, 그리고 현재 시스템에 설치된 패키지에 대해 파일 종속성을 확인합니다.

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

예 3 두 개의 매니페스트에 대한 종속성 생성 및 확인

두 개의 매니페스트(`foo` 및 `bar`)에 대한 파일 종속성을 생성하고 원래 매니페스트에 모든 정보를 보존합니다. 그런 다음 파일 종속성을 확인하고 `./res`에 결과 매니페스트를 배치합니다. 이러한 결과 매니페스트는 `pkgsend publish`와 함께 사용할 수 있습니다.

예 3 두 개의 매니페스트에 대한 종속성 생성 및 확인 *(계속)*

```
$ 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
```

예 4 ELF 파일 종속성에 대한 토큰에 값 추가

컨텐츠 디렉토리가 /에 있는 foo에서 기록된 매니페스트에 대한 종속성을 생성하면서 ELF 파일의 실행 경로에 있는 모든 PLATFORM 토큰을 sun4v 및 sun4u로 바꿉니다.

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

예 5 커널 모듈 디렉토리 지정

컨텐츠 디렉토리가 /에 있는 foo에서 기록된 매니페스트에 대한 종속성을 생성할 때 커널 모듈을 찾을 디렉토리로 /kmod를 지정합니다.

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

예 6 종속성 생성 무시

지정된 Python 스크립트에 대한 표준 Python 실행 경로에 opt/python을 추가하고 opt/python/foo/file.py로 전달된 파일에 대해 모든test Python 모듈에 대한 종속성 생성을 무시합니다.

usr/lib/python2.6/vendor-packages/xdg에서 전달된 파일에 대해 종속성이 생성되지 않도록 합니다.

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

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

환경 변수

`PKG_IMAGE` 패키지 작업에 사용할 이미지가 포함된 디렉토리를 지정합니다. `-R`이 지정된 경우 이 값은 무시됩니다.

종료 상태

다음 종료 값이 반환됩니다.

`0` 모든 요소가 작동되었습니다.

`1` 오류가 발생했습니다.

`2` 잘못된 명령줄 옵션이 지정되었습니다.

`99` 예상치 않은 예외가 발생했습니다.

속성

다음 속성에 대한 설명은 `attributes`(5)를 참조하십시오.

속성 유형	속성 값
Availability	`package/pkg`
Interface Stability	커밋되지 않음

참조

`pkg`(5)

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

이름 pkgdiff – 패키지 매니페스트 비교

개요

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

설명

pkgdiff는 두 개의 패키지 매니페스트를 비교하여 차이를 보고합니다. pkgdiff는 비교 전에 각 매니페스트 및 작업을 일관된 순서로 정렬합니다.

출력 형식은 다음과 같습니다.

+ complete_action 이 작업의 경우 *file2*에는 있지만 *file1*에는 없습니다.

- complete_action 이 작업의 경우 *file1*에는 있지만 *file2*에는 없습니다.

actionname keyvalue [*variant values, if any*]

- attribute1=value1 이 *attribute*, *value*의 경우 *file1*에는 있지만 *file2*에는 없습니다.

+ attribute2=value2 이 *attribute*, *value*의 경우 *file2*에는 있지만 *file1*에는 없습니다.

변형은 다르지만 유형과 키 속성 값은 동일한 작업은 비교를 위해 별도의 작업으로 처리됩니다. 따라서 속성을 변경하는 작업은 속성 변경 사항으로 표시되지 않고 전체 형식으로 표시됩니다.

옵션

다음 옵션이 지원됩니다.

-i *attribute* 비교 중 *attribute*(있을 경우)를 무시합니다. -i *hash*를 사용하여 파일 해시 값을 무시할 수 있습니다. 이 옵션은 -o 옵션과 함께 사용할 수 없습니다. 이 옵션은 반복할 수 있습니다.

-o *attribute* *attribute*의 차이만 보고합니다. 이 옵션은 -i 옵션과 함께 사용할 수 없습니다. 이 옵션은 작업에 대한 *attribute*에 영향을 끼치지 않는 작업 변경 사항을 무시합니다.

-v *name= value* 이 변형 값에 대한 차이만 계산합니다. 예를 들어, arch=sparc에 대한 차이만 계산합니다. 이 변형 태그는 비교 전에 모든 작업에 대해 제거됩니다. 값은 변형당 하나씩만 지정할 수 있습니다. 이 옵션은 여러 변형에 대해 반복할 수 있습니다.

종료 상태

다음 종료 값이 반환됩니다.

0 발견된 차이가 없습니다.

1 차이가 발견되었습니다.

>1 오류가 발생했습니다.

99 예상치 않은 예외가 발생했습니다.

속성

다음 속성에 대한 설명은 attributes(5)를 참조하십시오.

속성 유형	속성 값
Availability	package/pkg
Interface Stability	커밋되지 않음

참조

pkg(5)

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

이름

pkgfmt – 패키지 매니페스트 형식 지정

개요

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

설명

-c 또는 -d 옵션이 없는 pkgfmt는 80자에서 행을 줄 바꿈하고 유형을 기준으로 작업을 정렬하고 속성을 정렬하는 등의 일관된 방식으로 패키지 매니페스트의 형식을 지정합니다. 작업으로 구문이 분석되지 않는 행(예: 매크로, 설명 또는 변환)은 정렬된 순서로 나타나지 않습니다.

지정된 인수가 없을 경우 pkgfmt는 EOF까지 stdin을 읽은 후 형식이 지정된 매니페스트를 stdout에 기록합니다. 명령줄에 지정된 매니페스트는 해당 위치에서 형식이 지정됩니다.

-c 옵션이 있는 pkgfmt는 pkgfmt 스타일로 매니페스트의 형식이 지정되었는지 여부를 확인합니다. -d 옵션은 파일의 형식이 제대로 지정되지 않은 경우 차이를 표시합니다.

옵션

다음 옵션이 지원됩니다.

-c pkgfmt 스타일로 매니페스트의 형식이 지정되었는지 여부를 확인합니다.

-d 형식이 지정된 버전의 매니페스트 차이를 통합 형식으로 표시합니다.

-u 80자에서 행을 줄 바꿈하지 않습니다. 이 옵션은 패키지 매니페스트에 기존 텍스트 처리 도구를 적용하는 데 유용합니다.

종료 상태

다음 종료 값이 반환됩니다.

0 명령이 성공했습니다.

1 -c 또는 -d 옵션이 지정되었으며 하나 이상의 매니페스트가 pkgfmt 일반 형식이 아니거나 오류가 발생했습니다.

2 잘못된 명령줄 옵션이 지정되었습니다.

99 예상치 않은 예외가 발생했습니다.

속성

다음 속성에 대한 설명은 attributes(5)를 참조하십시오.

속성 유형	속성 값
Availability	package/pkg
Interface Stability	커밋되지 않음

참조

pkg(5)

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

이름 pkglint – 이미지 패키징 시스템 패키지 Lint

개요

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

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

설명

pkglint는 선택적으로 다른 저장소를 참조하는 하나 이상의 패키지 매니페스트에 대해 일련의 검사를 실행합니다.

pkglint는 패키지를 게시하기 전 패키지 작성 프로세스 도중에 사용해야 합니다. pkglint는 pkgsend(1) 또는 pkg.depotd(1M)의 일반 작업 중 수행하기에는 너무 부담이 될 수 있는 매니페스트에 대한 철저한 테스트를 수행합니다. pkglint 검사에는 중복 작업, 누락된 속성 및 비정상적인 파일 권한에 대한 테스트가 포함됩니다.

명령줄에서 공백으로 구분된 로컬 파일 목록으로 Lint 실행을 위한 매니페스트를 전달할 수도 있고, 저장소에서 매니페스트를 검색할 수도 있습니다.

저장소에서 매니페스트를 검색하면 첫번째 pkglint 실행 시 지정된 캐시 디렉토리에 pkg(5) 사용자 이미지가 만들어지고 채워집니다. -r 옵션을 제공할 경우 참조 저장소에 대해 이름이 *cache_dir*/ref_image인 사용자 이미지가 만들어집니다. -l 옵션을 제공할 경우 Lint 저장소에 대해 이름이 *cache_dir*/lint_image인 사용자 이미지가 만들어집니다. 해당 이미지에는 컨텐츠가 설치되지 않습니다. 해당 이미지는 pkglint가 저장소에서 매니페스트를 검색하는 데만 사용됩니다.

나중에 pkglint를 호출하면 캐시 디렉토리를 재사용할 수 있으며 -r 또는 -l 인수를 생략할 수 있습니다.

pkglint는 캐시 디렉토리에서 게시자를 구성하는 데 필요한 제한된 지원을 제공합니다. pkg(1)를 사용하면 해당 이미지에 대해 보다 복잡한 게시자 구성을 수행할 수 있습니다.

pkglint는 패키지 작성인이 지정된 매니페스트 또는 작업에 대한 검사를 무시할 수 있도록 허용합니다. True로 설정된 pkg.linted 속성을 포함하는 매니페스트 또는 작업은 해당 매니페스트 또는 작업에 대한 Lint 출력을 생성하지 않습니다.

pkglint 검사 이름의 하위 문자열을 사용하여 보다 세부적인 pkg.linted 설정을 지정할 수 있습니다. 예를 들어, pkg.linted. *check*.*id*를 True로 설정하면 지정된 매니페스트 또는 작업에 대해 이름이 *check*.*id* 인 모든 검사가 무시됩니다.

pkglintrc 파일을 지정하여 pkglint의 동작을 구성할 수 있습니다. 기본적으로 pkglint는 /usr/share/lib/pkg/pkglintrc 및 $HOME/.pkglintrc에서 구성 옵션을 검색합니다. -f 옵션을 사용하여 다른 구성 파일을 지정할 수 있습니다.

Lint 실행 중 발생한 오류 또는 경고는 stderr에 인쇄됩니다.

옵션

다음 옵션이 지원됩니다.

`-b` *build_no* Lint 및 참조 저장소에서 Lint 실행 중 사용되는 패키지 목록을 제한하는 데 사용할 빌드 번호를 지정합니다. `-b` 옵션을 지정하지 않을 경우 패키지의 최신 버전이 사용됩니다. `version.pattern` 구성 등록 정보도 참조하십시오.

`-c` *cache_dir* Lint 및 참조 저장소의 패키지 메타 데이터를 캐싱하는 데 사용할 로컬 디렉토리를 지정합니다.

`-l` *lint_uri* Lint 저장소의 위치를 나타내는 URI를 지정합니다. HTTP 및 파일 시스템 기반 게시가 모두 지원됩니다. `-l`을 지정할 경우 `-c`도 지정해야 합니다.

`-L` 알려진 Lint 검사 및 제외된 Lint 검사를 나열한 후 종료합니다. 각 검사에 대한 짧은 이름 및 설명을 표시합니다. `-v` 플래그와 함께 사용할 경우 설명 대신 검사 구현 방법을 표시합니다.

`-f` *config_file* *config_file* 구성 파일을 사용하여 `pkglint` 세션을 구성합니다.

`-p` *regexp* Lint 저장소에서 확인할 패키지 목록을 제한하는 데 사용할 정규 표현식을 지정합니다. `-b`에 대한 값(제공된 경우)과 일치한다는 가정하에 이 패턴을 무시하여 참조 저장소의 모든 매니페스트가 로드됩니다.

`-r` *repo_uri* 참조 저장소의 위치를 나타내는 URI를 지정합니다. `-r`을 지정할 경우 `-c`도 지정해야 합니다.

`-v` 구성 파일의 모든 `log_level` 설정을 대체하여 `pkglint`를 Verbose 모드로 실행합니다.

`--help` 또는 `-?` 사용법 메시지를 표시합니다.

파일

`pkglintrc` 구성 파일에 사용되는 키/값 인수는 다음과 같습니다.

`log_level` Lint 메시지를 내보낼 최소 레벨입니다. 이 레벨보다 낮은 Lint 메시지는 무시됩니다. 기본값은 `INFO`입니다.

로그 레벨은 `DEBUG`, `INFO`, `WARNING`, `ERROR` 및 `CRITICAL`(가장 사소한 것부터 심각한 순으로)입니다.

`do_pub_checks` `True`인 경우 게시된 패키지에 대해서만 처리될 수 있는 검사를 수행합니다. 기본값은 `True`입니다.

`pkglint.ext.*` `pkglint`의 플러그인 방식은 런타임에 다른 Lint 모듈이 추가될 수 있도록 허용합니다. `pkglint.ext.`로 시작하는 키에 사용되는 값은 완전하게 지정된 Python 모듈이어야 합니다. 자세한 내용은 "개발자" 절을 참조하십시오.

`pkglint.exclude`	수행될 일련의 검사에서 생략할 완전하게 지정된 Python 모듈, 클래스 또는 함수 이름의 공백으로 구분된 목록입니다.
`use_progress_tracker`	`True`인 경우 Lint 실행 중 매니페스트에 대해 반복될 때 진행 추적기를 사용합니다. 기본값은 `True`입니다.
`version.pattern`	Lint를 실행할 빌드 번호를 지정할 때 사용되는 버전 패턴입니다(`-b`). 구성 파일에서 지정되지 않은 경우 `-b` 옵션은 분기 접두어가 0이며 5.11 빌드의 모든 구성 요소와 일치하는 `*,5.11-0.` 패턴을 사용합니다.

개발자

`pkglint`와 하위 클래스 `pkg.lint.base.Checker`, 해당 하위 클래스 `ManifestChecker`, `ActionChecker` 및 `ContentChecker`에서 수행하는 일련의 검사를 확장합니다. 구성 파일의 새 `pkglint.ext.` 키에 해당 클래스를 포함하는 Python 모듈 이름을 추가합니다.

이러한 새 하위 클래스의 인스턴스는 시작 시 `pkglint`가 만듭니다. 특수한 키워드 인수 `pkglint_id`가 있는 각 하위 클래스 내 메소드가 Lint 세션 도중 호출됩니다. 이러한 메소드의 서명은 수퍼 클래스 내 해당 `check()` 메소드의 서명과 동일해야 합니다. `pkglint -L`이 인쇄한 설명으로 사용되는 `pkglint_desc` 속성에도 메소드가 지정되어야 합니다.

`Checker` 하위 클래스에 매개변수를 사용하여 동작을 조정할 수 있습니다. 권장되는 매개변수 이름 지정 규약은 *pkglint_id*.*name*입니다. 매개변수 값은 구성 파일에 저장되거나 `LintEngine.get_param()` 메소드를 사용하여 검색된 매니페스트 또는 작업에서 액세스될 수 있습니다. 매니페스트에서 매개변수에 액세스하면 `pkglint` 매개변수가 기존 작업 또는 매니페스트 값과 겹치지 않도록 접두어 `pkg.lint`가 키 이름 앞에 추가됩니다.

예

예 1 특정 저장소에 대한 첫번째 실행

지정된 저장소에 대해 `pkglint` 세션을 처음 실행합니다.

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

예 2 동일한 저장소에 대한 후속 실행

예 1에 사용된 것과 동일한 저장소에 대해 후속 세션을 실행합니다.

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

예 3 제한된 매니페스트 세트에 Lint 저장소 사용

Lint 저장소를 사용하여 `pkglint` 세션을 실행하고 확인할 일부 매니페스트를 지정합니다.

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

예 4 빌드 지정

지정된 빌드에 대해 Verbose 모드로 pkglint 세션을 실행합니다.

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

예 5 구성 파일 수정

일부 검사가 제외된 새 Lint 모듈에 대한 구성 파일입니다.

```
$ 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
```

종료 상태

다음 종료 값이 반환됩니다.

0 명령이 성공했습니다.

1 하나 이상의 Lint 검사가 출력을 내보냈습니다.

2 잘못된 명령줄 옵션이 지정되었습니다.

99 예상치 않은 예외가 발생했습니다.

속성

다음 속성에 대한 설명은 attributes(5)를 참조하십시오.

속성 유형	속성 값
Availability	package/pkg
Interface Stability	커밋되지 않음

참조

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

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

이름 pkgmerge – 이미지 패키징 시스템 패키지 병합 유틸리티

개요

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

설명 pkgmerge는 다중 변형 패키지를 만드는 데 사용되는 패키지 게시 도구입니다. 이 과정에서는 동일한 이름 및 버전(시간 기록 제외)과 패키지를 병합하고, 주어진 소스에 대해 지정된 변형 이름 및 값과 병합하려는 버전에서 고유한 작업의 태그를 지정한 후 대상 저장소에 새 패키지를 게시합니다. 각 소스의 패키지마다 최신 버전만 사용됩니다.

각 작업의 pkg.merge.blend 속성이 병합하려는 변형의 이름으로 설정된 경우 최종 출력에 표시되는 작업에 추가된 변형 태그가 나타나지 않도록 해당 작업은 병합 전에 다른 매니페스트에 복사됩니다. pkg.merge.blend 속성 자체가 출력 매니페스트의 작업에서 제거됩니다. 여러 패스 병합에 다른 값을 사용하여 이 속성을 반복할 수 있습니다.

입력 매니페스트의 동일한 경로로 전달되는 작업이 다를 경우 오류와 함께 pkgmerge가 종료됩니다.

옵션 다음 옵션이 지원됩니다.

-d *dest_repo*
병합된 패키지를 게시할 대상 저장소의 파일 시스템 경로 또는 URI입니다. 대상 저장소가 존재해야 합니다. pkgrepo(1)를 사용하여 새 저장소를 만들 수 있습니다.

-n
대상 저장소에 변경 사항을 적용하지 않은 상태로 테스트 실행을 수행합니다.

-s *variant= value*[,...],*src_repo*
이 소스의 패키지에 사용할 변형 이름 및 값으로, 이 뒤에는 패키지를 검색할 소스 저장소나 패키지 아카이브의 파일 시스템 경로 또는 URI가 옵니다. 여러 개의 변형은 쉼표로 구분하여 지정할 수 있습니다. 모든 소스에 대해 동일한 변형이 명명되어야 합니다. 이 옵션은 여러 번 지정할 수 있습니다.

--help 또는 -?
사용법 메시지를 표시합니다.

환경 변수 지원되는 환경 변수는 다음과 같습니다.

TMPDIR 프로그램 실행 중 임시 데이터가 저장될 디렉토리의 절대 경로입니다. 설정하지 않을 경우 기본적으로 /var/tmp에 임시 데이터를 저장합니다.

예

예 1 변형 이름 및 값 지정

검색 소스에 대해 지정된 주어진 변형 이름 및 값을 사용하여 지정된 소스에서 발견된 각 패키지의 태그를 지정합니다.

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

예 1 변형 이름 및 값 지정 *(**계속**)*

샘플 패키지:

```
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
```

작업 후 샘플 패키지:

```
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
```

예 2 패키지 병합 및 게시

지정된 소스의 각 패키지에 대한 최신 버전을 병합하고 대상 저장소에 새 패키지를 게시합니다.

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

소스 1의 샘플 패키지(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
```

소스 2의 샘플 패키지(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
```

병합된 패키지:

```
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
```

예 3 i386 및 SPARC 시스템용 디버그 및 비디버그 패키지 병합

i386 및 SPARC 시스템용의 일련의 디버그 및 비디버그 저장소에 있는 각 패키지의 최신 버전을 병합합니다.

```
$ 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 \
```

예 3 i386 및 SPARC 시스템용 디버그 및 비디버그 패키지 병합 *(계속)*

```
-s arch=i386,debug=true,/repo/i386-debug \
-d /path/to/target/repository
```

소스 1의 샘플 패키지(SPARC 비디버그):

```
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
```

소스 2의 샘플 패키지(SPARC 디버그):

```
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
```

소스 3의 샘플 패키지(i386 비디버그):

```
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
```

소스 4의 샘플 패키지(i386 디버그):

```
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
```

병합된 패키지:

```
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
```

예 4 pkg.merge.blend를 사용하여 병합

`pkg.merge.blend` 속성을 사용하여 충돌하지 않는 두 구조에 대한 패키지를 병합합니다.

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

소스 1의 샘플 패키지(SPARC):

예 4 pkg.merge.blend를 사용하여 병합 *(계속)*

```
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
```

소스 2의 샘플 패키지(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
```

병합된 패키지:

```
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
```

종료 상태

다음 종료 값이 반환됩니다.

`0` 명령이 성공했습니다.

`1` 오류가 발생했습니다.

`2` 잘못된 명령줄 옵션이 지정되었습니다.

`99` 예상치 않은 예외가 발생했습니다.

속성

다음 속성에 대한 설명은 `attributes`(5)를 참조하십시오.

속성 유형	속성 값
Availability	`package/pkg`
Interface Stability	커밋되지 않음

참조 pkgrepo(1), pkg(5)

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

이름

pkgmogrify – 이미지 패키징 시스템 매니페스트 transmogrifier

개요

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

설명

`pkgmogrify`는 소프트웨어 빌드 및 패키지 게시를 자동화할 때 필요한 일반적인 변환 과정을 단순화하기 위해 패키지 매니페스트를 프로그래밍 방식으로 편집할 수 있도록 합니다.

`pkgmogrify`는 다음을 제공합니다.

- 다양한 구조 및 플랫폼에서 단일 매니페스트를 원활하게 공유할 수 있도록 하는 매크로 대체
- 다른 매니페스트 또는 매니페스트 단편(예: 표준 구성 요소 및 변환) 포함
- 작업 속성 수정, 삭제 또는 추가 등의 패키지 작업 변환

옵션

다음 옵션이 지원됩니다.

`-D` *name= value*
값이 *value*인 *name*을 매크로로 정의합니다. 매크로는 입력 파일에 `$(macro)`로 표시됩니다. 변환을 더 이상 찾을 수 없을 때까지 대체가 반복됩니다. 일반적으로 사용되는 관용구는 다음과 같습니다.

- 행 시작 부분에 구조 관련 태그를 사용하여 다른 구조에 있는 매니페스트의 행 제거:

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

  SPARC 구조를 처리할 때 이 매크로가 빈 문자열로 설정됩니다. 다른 구조를 처리할 때 이 매크로가 명령줄에서 #으로 설정되므로 현재 구조에 있는 매니페스트에서 이 작업이 제거됩니다.

- 경로 이름의 플랫폼 관련 부분(예: 실행 파일 및 라이브러리에 대한 64비트 구조 디렉토리의 이름) 지정:

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

  이러한 매크로는 명령줄에서 적합한 값으로 설정됩니다. 미리 정의된 매크로 값은 없습니다.

`-I` *include_directory*
명령줄에 지정된 파일과 내장된 포함 지시어에 대한 검색 경로에 지정된 디렉토리를 추가합니다.

`-O` *outputfile*
지정된 파일에 매니페스트 출력을 기록합니다. 오류가 발생하거나 변환 지시어로 인해 중단 작업이 발생하는 경우 파일이 작성되지 않습니다. 기본적으로 매니페스트 출력은 `stdout`에 기록됩니다.

`-P` *printfile*
지정된 파일에 변환 지시어 인쇄 작업으로 인한 출력을 기록합니다. 오류가 발생하거나 변환 지시어로 인해 중단 작업이 발생하는 경우 파일이 작성되지 않습니다. 기본적으로 인쇄 출력은 `stdout`에 기록됩니다.

`-i`
파일의 포함 지시어를 무시합니다. 명령줄(또는 `stdin`)에 지정된 파일만 처리됩니다.

`-v`
변환 결과를 보여 주는 출력 매니페스트에 설명을 기록합니다. 이 정보는 디버깅에 유용합니다.

`--help` 또는 `-?`
사용법 메시지를 표시합니다.

내장된 지시어

매니페스트 파일에서 지원되는 지시어의 두 가지 유형은 포함 지시어와 변환 지시어입니다.

포함 지시어의 형식은 다음과 같습니다.

`<include` *file*`>`

이 지시어를 사용하면 `pkgmogrify`는 먼저 현재 디렉토리에서 이름이 `file`인 파일을 검색한 후 `-I` 옵션을 사용하여 지정된 디렉토리에서 해당 파일을 검색합니다. 해당 파일이 발견되면 지시어가 발생된 지점의 매니페스트에 파일 내용이 삽입됩니다. 해당 파일이 발견되지 않으면 `pkgmogrify`는 오류와 함께 종료됩니다.

변환 지시어의 형식은 다음과 같습니다.

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

이러한 지시어는 메모리에서 모든 입력을 읽을 때까지 누적된 후 발견된 순서대로 작업에 적용됩니다.

일치 조건의 형식은 다음과 같습니다.

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

지정된 *action-types* 중 하나가 일치해야 합니다. 지정된 모든 *attributes*가 일치해야 합니다. 사용되는 정규 표현식 구문은 Python의 구문입니다. Python 정규 표현식 구문에 대한 자세한 내용을 보려면 `pydoc re` 명령을 사용하거나 `http://docs.python.org/dev/howto/regex.html`에 있는 보다 완전한 설명서를 참조하십시오. 정규 표현식의 경우 시작 부분에서는 앵커되지만 끝 부분에서는 앵커되지 않습니다. 따라서 확장자에 따른 정규 표현식 일치 파일은 `.*`로 시작하고 `$`로 끝나야 합니다.

여러 조건은 공백으로 구분하여 지정할 수 있습니다.

사용 가능한 작업은 다음과 같습니다.

`add` 속성에 값을 추가합니다. 이 작업에는 두 개의 인수가 사용됩니다. 첫번째 인수는 속성 이름이며 두번째 인수는 값입니다.

`default` 존재하지 않을 경우 속성 값을 설정합니다. 이 작업에는 `add` 작업과 동일한 두 개의 인수가 사용됩니다.

`delete` 속성 값을 제거합니다. 이 작업에는 두 개의 인수가 사용됩니다. 첫번째 인수는 속성 이름입니다. 두번째 인수는 삭제된 속성 값과 일치시킬 정규 표현식입니다. 작업 일치에 사용되는 정규 표현식과 달리, 이 표현식은 앵커되지 않습니다.

`drop` 이 작업을 폐기합니다.

`edit` 작업의 속성을 수정합니다. 이 작업에는 세 개의 인수가 사용됩니다. 첫번째 인수는 속성 이름이며 두번째 인수는 속성 값과 일치하는 정규 표현식입니다. 세번째 인수는 정규 표현식에 의해 일치된 값의 일부분에 대해 대체되는 바꿀 문자열입니다. 작업 일치에 사용되는 정규 표현식과 달리, 이 표현식은 앵커되지 않습니다. 정규 표현식에서 그룹이 정의된 경우 바꿀 문자열에 `\1`, `\2` 등의 일반적인 정규 표현식 역참조를 사용할 수 있습니다.

`emit` 매니페스트 출력 스트림으로 행을 내보냅니다. 유효한 작업 문자열, 빈 문자열(공백 행 생성) 또는 설명(#, 임의 텍스트 순)이어야 합니다.

`exit` 매니페스트 처리를 종료합니다. 출력되는 매니페스트가 없고 적용되는 `print` 작업이 없습니다. 하나의 인수를 지정할 경우 해당 인수는 정수여야 하며 종료 코드로 사용됩니다. 기본값은 0입니다. 두 개의 인수를 지정할 경우 첫번째 인수는 종료 코드이며 두번째 인수는 `stderr`에 인쇄될 메시지입니다.

`print` `-P`를 사용하여 지정된 출력 파일에 메시지를 인쇄합니다.

`set` 속성 값을 설정합니다. 이 작업에는 `add` 작업과 동일한 두 개의 인수가 사용됩니다.

`delete` 및 `drop`을 제외한 모든 작업에는 컨텐츠가 출력 스트림으로 이동되는 인수가 사용됩니다(가능한 경우 선택적). 이러한 문자열에는 세 가지 다른 종류의 특수 토큰이 포함될 수 있습니다. 이러한 토큰은 각 작업의 고정 변환을 기반으로 하지 않는 정보가 출력에 포함될 수 있도록 허용합니다.

첫번째 종류의 대체에서는 퍼센트 기호 뒤의 괄호 안에 속성 이름을 삽입하여 작업이 현재 작업의 속성 값을 참조할 수 있도록 허용합니다. 예를 들어, `%(alias)`는 작업의 `alias` 속성을 참조합니다.

다양한 합성 속성이 존재합니다. 다음과 같은 두 개의 속성이 `pkgmogrify`에 대해 고유합니다.

- `pkg.manifest.filename`은 작업이 발견된 파일의 이름을 나타냅니다.
- `pkg.manifest.lineno`는 작업이 발견된 행을 나타냅니다.

다음과 같은 세 개의 합성 속성은 pkg(1)에서 사용되는 속성과 유사합니다.

- action.hash는 작업에 페이로드가 있을 경우 작업의 해시 값을 나타냅니다. 페이로드가 있는 작업의 경우 set 작업은 action.hash 속성에 대해 작업을 수행하여 작업의 해시를 변경할 수 있습니다.
- action.key는 키 속성의 값을 나타냅니다.
- action.name은 작업 유형의 이름을 나타냅니다.

값이 요청된 속성이 존재하지 않을 경우 pkgmogrify는 오류와 함께 종료됩니다. 오류와 함께 종료되지 않도록 하려면 속성 이름과 ;notfound=, 속성 값을 대체할 값을 차례로 지정하십시오. 예를 들어, %(alias;notfound='no alias')는 alias 속성 값이 존재할 경우 해당 속성 값을 인쇄하고 그렇지 않을 경우 no alias를 인쇄합니다.

값이 요청된 속성이 다중 값 속성일 경우 각 값이 공백으로 구분되어 인쇄됩니다. notfound 토큰과 마찬가지로, prefix, suffix 및 sep 토큰을 통해서도 이 동작을 변경할 수 있습니다. prefix로 표시된 문자열은 각 값 앞에 추가되며, suffix로 표시된 문자열은 각 값 뒤에 추가되고, sep는 특정 값의 접미어와 다음 값의 접두어 사이에 배치됩니다.

작업 속성과 마찬가지로, pkgmogrify 지시어는 괄호 대신 중괄호를 사용하여 패키지 속성을 참조할 수 있습니다(%{pkg.fmri}). 변환 지시어가 적용될 때 set 작업에 속성이 정의되어 있어야 합니다. 그렇지 않으면 위에 설명된 notfound로 처리됩니다. 처리가 패키지를 기술하는 매니페스트 파일 끝에 도달하면 다음 패키지를 위해 속성이 지워집니다.

이는 패키지 속성이 작업 속성인 것처럼 패키지 속성을 참조하고 패키지 속성에 대해 일치 및 일시적인 수정 작업을 수행하는 데도 유용합니다. 따라서 이러한 경우에 합성 작업 이름 pkg를 사용할 수 있습니다(pkgmogrify의 컨텍스트에서만).

pkgmogrify가 명령줄에 지정된 매니페스트 및 pkg.fmri 패키지 속성을 정의한 매니페스트에 대한 읽기 작업을 마치면 pkgmogrify는 속성이 패키지의 속성인 이 합성 pkg 작업을 만듭니다. <transform> 지시어가 기타 작업 유형과 마찬가지로 이 작업에 대해 일치할 수 있습니다.

pkg 작업의 작업은 메모리에만 존재하고 내보낸 매니페스트에 직접적인 영향을 끼치지 않는다는 점에서 특수합니다. 예를 들어, add, default 또는 set 작업을 통해 pkg 작업에 대한 속성을 설정하려고 시도하면 set 작업이 매니페스트에 추가되지 않습니다. 단, 일치시킬 다른 <transform> 지시어에는 사용될 수 있습니다. emit a pkg 작업을 시도하면 오류가 발생합니다. 패키지 속성을 추가하려면 대신 emit a set 작업을 시도하십시오.

세번째 종류의 대체는 역참조입니다. 이 대체는 edit 작업에서 사용 가능한 것과는 다르지만 -> 왼쪽의 변환 일치에 나열되는 그룹에 대한 참조입니다. 이는 일치 조건에 표시된 순서대로 %<1>, %<2> 등으로 표시됩니다.

처리 순서는 다음과 같습니다.

1. 입력 파일에서 행을 읽습니다.
2. 매크로를 적용합니다.
3. 추가 파일을 찾아 읽도록 <include ...> 및 <transform> 지시어를 처리합니다.
4. 모든 입력 파일이 누적된 후에는 입력의 각 행이 작업으로 변환되며 모든 변환이 적용됩니다.
5. 처리가 성공적으로 완료되면 출력이 기록됩니다.

예

예 1 SMF 매니페스트에 태그 추가

패키지가 라이브 시스템에 설치될 때 태그를 가져오도록 SMF(서비스 관리 기능) 매니페스트에 태그를 추가합니다.

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

예 2 파일 이동

usr/sfw/bin에서 usr/bin으로 파일을 이동합니다.

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

예 3 재부트 필요 지정

.conf 파일이 아닌 /kernel 아래의 파일에 reboot-needed 태그를 추가합니다. 이 예에서는 입력 파일에 표시된 순서대로 각 작업에 변환을 적용하는 방식을 활용합니다.

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

정규 표현식을 사용한 단일 변환 규칙에서도 완료할 수 있습니다.

예 4 Depend 작업으로 FMRI 속성 변환

통합에 포함되도록 패키지 속성 pkg.fmri를 depend 작업으로 변환합니다.

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

예 5 버그 번호 목록 인쇄

따옴표가 사용되고 접두어가 붙은 버그 번호의 쉼표로 구분된 목록을 인쇄합니다.

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

예 6 속성 누락 허용

속성이 누락된 경우에도 메시지를 인쇄할 수 있도록 합니다.

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

예 7 기본값 설정

기본 소유자, 그룹 및 권한 값을 설정합니다.

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

예 8 오래된 것으로 표시되지 않은 패키지에 종속성 추가

오래된 것으로 표시되지 않은 패키지의 경우 패키지를 전달하는 연결에 대한 통합에 종속성을 추가합니다. 이 변환 세트는 매니페스트를 읽은 후 발생해야 합니다. 그렇지 않으면 종속성을 항상 내보냅니다. pkg 수정 작업은 영구적으로 적용되는 것이 아니므로 pkg.obsolete=false와 일치하는 속성을 정리하지 않아도 됩니다.

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

예 9 오류가 있을 경우 종료 후 메시지 인쇄

매니페스트에 오래된 속성이 있을 경우 메시지와 함께 오류가 발생합니다.

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

예 10 적합한 로케일 페이싯 설정

고려 중인 경로 이름에 적합한 로케일 페이싯을 설정합니다.

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

종료 상태

다음 종료 값이 반환됩니다.

0 모든 요소가 작동되었습니다.

1 문제가 발생했지만 예상했던 문제입니다.

2 잘못된 명령줄 옵션이 지정되었습니다.

99 예상치 않은 처리 오류입니다.

파일

/usr/share/pkg/transforms 이 디렉토리에는 페이싯, 액추에이터 및 기타 속성을 설정하기 위한 유용한 변환과 함께 파일이 포함됩니다.

속성

다음 속성에 대한 설명은 attributes(5)를 참조하십시오.

속성 유형	속성 값
Availability	package/pkg
Interface Stability	커밋되지 않음

참조

pkg(1), pkg(5)

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

이름

pkgrecv – 이미지 패키징 시스템 컨텐츠 검색 유틸리티

개요

```
/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
```

설명

pkgrecv를 통해 사용자는 pkg(5) 저장소 또는 패키지 아카이브에서 패키지를 검색할 수 있습니다. pkgrecv는 선택적으로 검색된 패키지를 다른 패키지 저장소에 다시 게시하거나 아카이브할 수도 있습니다. 기본적으로 패키지는 pkg(1), pkg.depotd(1M) 및 패키지 게시 도구에서 사용하는 데 적합한 패키지 저장소 형식으로 검색됩니다.

pkgrecv 작업 후에 검색 색인을 구성하려면 저장소에서 pkgrepo refresh 또는 pkgrepo rebuild를 실행하십시오.

옵션

다음 옵션이 지원됩니다.

-a
: 검색된 패키지 데이터를 -d로 지정된 위치에 있는 pkg(5) 아카이브에 저장합니다. 아직 파일이 존재하지 않을 수 있습니다. 이 옵션은 파일 시스템 기반 대상에서만 사용할 수 있습니다. 요구 사항은 아니지만 .p5p 파일 확장자(예: archive.p5p)를 사용하는 것이 좋습니다. 이 옵션은 --raw와 함께 사용할 수 없습니다.

-c *cache_dir*
: 다운로드된 컨텐츠를 캐시하는 데 사용할 디렉토리의 경로입니다. 이 디렉토리를 제공하지 않을 경우 클라이언트가 자동으로 캐시 디렉토리를 선택합니다. 다운로드가 중단되고 캐시 디렉토리가 자동으로 선택되면 이 옵션을 사용하여 다운로드를 계속하십시오. 임시 데이터 저장소에 사용되는 위치를 설정하는 방법에 대한 자세한 내용은 아래의 "환경 변수" 절을 참조하십시오.

-d *path_or_uri*
: 패키지를 다시 게시할 대상의 파일 시스템 경로 또는 URI입니다. -a가 지정된 경우 대상은 아직 존재하지 않는 새 패키지 아카이브입니다. 그렇지 않으면 대상은 이미 존재하는 패키지 저장소여야 합니다. pkgrepo(1)를 사용하여 새 저장소를 만들 수 있습니다.

-h
: 사용법 메시지를 표시합니다.

-k
: 검색된 패키지 내용을 압축된 상태로 유지합니다. 다시 게시할 때는 이 옵션이 무시됩니다. 압축된 패키지 내용은 pkgsend(1)와 함께 사용하지 않아야 합니다.

-m *match*
: 다음 값을 사용하여 일치하는 동작을 제어합니다.

  all-timestamps
  : 최신 시간 기록뿐 아니라 일치하는 모든 시간 기록을 포함합니다(all-versions를 의미함).

  all-versions
  : 최신 버전뿐 아니라 일치하는 모든 버전을 포함합니다.

`-n` 변경 사항을 적용하지 않은 상태로 테스트 실행을 수행합니다.

`-r` 제공된 패키지 목록에 대한 모든 종속성을 반복적으로 검색합니다.

`-s` *src_repo_uri* 패키지 데이터를 수신할 pkg(5) 저장소 또는 패키지 아카이브의 위치를 나타내는 URI입니다.

`--cert` *file* HTTPS 저장소에서의 패키지 검색에 사용할 클라이언트 SSL 인증서 파일을 지정합니다.

`--key` *file* HTTPS 저장소에서의 패키지 검색에 사용할 클라이언트 SSL 키 파일을 지정합니다.

`--newest` 지정된 저장소에서 사용 가능한 패키지의 최신 버전을 나열한 후 종료합니다. `-s`를 제외한 기타 모든 옵션은 무시됩니다.

`--raw` `-d`로 지정된 위치에 있는 스템 및 버전으로 디렉토리 구조 세트의 원시 패키지 데이터를 검색하고 저장합니다. 이 옵션은 파일 시스템 기반 대상에서만 사용할 수 있습니다. 이 패키지 데이터를 사용하면 파일 내용을 수정하거나 추가 패키지 메타 데이터를 제공하여 편리하게 패키지를 수정하고 다시 게시할 수 있습니다. 이 옵션은 `-a`와 함께 사용할 수 없습니다.

예

예 1 최신 패키지 나열

이름이 `test`인 시스템의 저장소에서 사용 가능한 최신 패키지를 나열합니다.

```
$ 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
```

예 2 원시 패키지 데이터 검색

예 1의 `c++-runtime` 패키지를 `pkgsend publish`에 사용하는 데 적합한 형식으로 수신합니다.

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

예 3 시스템에서 종속성 검색

이름이 test인 시스템에서 editor/vim 패키지 및 모든 종속성을 수신합니다.

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

예 4 모든 버전 검색

이름이 test인 시스템에서 editor/vim 패키지의 모든 버전을 수신합니다.

```
$ 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
```

예 5 모든 버전 검색 및 원격으로 다시 게시

이름이 test인 시스템에서 library/zlib 패키지의 모든 버전을 수신하여 이름이 remote인 시스템의 원격 저장소에 다시 게시합니다.

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

예 6 저장소에서 종속성 검색

/export/repo에 있는 저장소에서 editor/gnu-emacs 패키지 및 모든 종속성을 수신합니다.

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

예 7 추가 패키지 검색

http://example.com:10000에 있는 저장소에서 아직 존재하지 않는 모든 패키지를 수신합니다.

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

예 8 패키지 아카이브 만들기

http://example.com:10000에 있는 저장소에서 editor/gnu-emacs 패키지 및 모든 종속성이 포함된 패키지 아카이브를 만듭니다.

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

예 9 아카이브에서 저장소로 패키지 복사

패키지 아카이브의 모든 패키지를 /export/repo에 있는 기존 저장소로 복사합니다.

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

환경 변수

지원되는 환경 변수는 다음과 같습니다.

`PKG_DEST` 패키지가 복사될 저장소나 패키지 아카이브의 파일 시스템 경로 또는 URI에 검색된 패키지를 저장할 디렉토리의 경로입니다.

`PKG_SRC` 패키지를 검색할 `pkg(5)` 저장소나 패키지 아카이브의 위치를 나타내는 URI 또는 파일 시스템 경로입니다.

`TMPDIR` 프로그램 실행 중 임시 데이터가 저장될 디렉토리의 절대 경로입니다. 설정하지 않을 경우 기본적으로 `/var/tmp`에 임시 데이터를 저장합니다.

종료 상태

다음 종료 값이 반환됩니다.

`0` 명령이 성공했습니다.

`1` 오류가 발생했습니다.

`2` 잘못된 명령줄 옵션이 지정되었습니다.

`3` 여러 명령이 요청되었지만 일부만 성공했습니다.

`99` 예상치 않은 예외가 발생했습니다.

속성

다음 속성에 대한 설명은 `attributes(5)`를 참조하십시오.

속성 유형	속성 값
Availability	package/pkg
Interface Stability	커밋되지 않음

참조

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

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

이름 pkgrepo – 이미지 패키징 시스템 저장소 관리 유틸리티

개요

```
/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
```

설명 pkgrepo는 pkg(5) 패키지 저장소를 만들고 관리할 수 있는 기능을 제공합니다. 패키지 저장소는 pkg(1)를 비롯하여 pkgsend(1) 또는 pkgrecv(1)와 같은 게시 클라이언트를 통해 패키지 데이터를 저장 및 검색할 수 있도록 하는 미리 정의된 일련의 디렉토리 및 파일입니다. 또한 패키지 저장소에 대한 네트워크 기반 액세스가 필요한 경우 pkg.depotd(1m)는 패키지 데이터를 저장 및/또는 검색할 수 있도록 저장소에 대한 클라이언트 액세스를 제공할 수 있습니다.

옵션 다음 옵션이 지원됩니다.

--help 또는 -? 사용법 메시지를 표시합니다.

하위 명령 지원되는 하위 명령은 다음과 같습니다.

create [--version *ver*] *uri_or_path*
지정된 위치에 pkg(5) 저장소를 만듭니다.

이 하위 명령은 파일 시스템 기반 저장소에서만 사용할 수 있습니다.

--version을 사용할 경우 지정된 버전과 호환되는 형식으로 저장소를 만듭니다. 기본적으로 버전 4 저장소가 만들어집니다. 지원되는 버전은 다음과 같습니다.

3 단일 게시자, 카탈로그 버전 1 및 검색 버전 1용 패키지 저장소를 지원합니다.

4 다중 게시자, 카탈로그 버전 1 및 검색 버전 1용 패키지 저장소를 지원합니다.

`add-publisher -s` *repo_uri_or_path publisher* `...`
지정된 게시자를 저장소에 추가합니다. 새 게시자에는 패키지 또는 내용이 없습니다.

이 하위 명령은 버전 4 파일 시스템 기반 저장소에서만 사용할 수 있습니다.

`get [-F` *format*`] [-p` *publisher* `...] -s` *repo_uri_or_path* `[`*section/property* `...]`
저장소 또는 게시자에 대한 등록 정보를 표시합니다.

기본적으로 각 등록 정보 및 해당 값은 별도의 행에 인쇄됩니다. 비어 있는 ASCII 문자열 값은 큰따옴표 쌍("")으로 표시됩니다. ASCII 문자열 값의 다음과 같은 본 셸 메타 문자 및 개행, 공백, 탭은 백슬래시 문자(\)로 제어되어야 합니다.

```
; & ( ) | ^ < > \ " ' `
```

"예" 절을 참조하십시오.

가능한 등록 정보, 용도 및 등록 정보별 값 목록은 아래의 `set` 하위 명령을 참조하십시오.

`-F`를 사용할 경우 대체 출력 형식을 지정합니다. *format*의 값은 `tsv`(탭으로 구분된 값), `json`(JavaScript 객체 표기법, 단일 행 사용) 또는 `json-formatted`(JavaScript 객체 표기법, 가독성을 위해 형식이 지정됨)일 수 있습니다.

`-H`를 사용할 경우 목록에서 헤더를 생략합니다.

`-p`를 사용할 경우 지정된 게시자에 대한 등록 정보를 표시합니다. 특수한 값 `all`은 모든 게시자에 대한 등록 정보를 표시합니다. 이 옵션은 여러 번 지정할 수 있습니다.

`-s`를 사용할 경우 지정된 URI 또는 파일 시스템 경로에 있는 저장소에서 작동합니다.

`info [-F` *format*`] [-H] [-p` *publisher* `...] - s` *repo_uri_or_path*
저장소가 확인한 패키지 게시자 목록을 표시합니다. 목록에는 게시자별 패키지 수, 게시자의 마지막 패키지 데이터 업데이트 시간, 게시자의 패키지 데이터 상태(예: 현재 처리되고 있는지 여부)가 포함됩니다.

`-F`를 사용할 경우 대체 출력 형식을 지정합니다. *format*의 값은 `tsv`(탭으로 구분된 값), `json`(JavaScript 객체 표기법, 단일 행 사용) 또는 `json-formatted`(JavaScript 객체 표기법, 가독성을 위해 형식이 지정됨)일 수 있습니다.

`-H`를 사용할 경우 목록에서 헤더를 생략합니다.

`-p`를 사용할 경우 지정된 게시자에 대해서만 데이터를 표시합니다. 제공하지 않을 경우 모든 게시자에 대한 데이터가 표시됩니다. 이 옵션은 여러 번 지정할 수 있습니다.

`-s`를 사용할 경우 지정된 URI 또는 파일 시스템 경로에 있는 저장소에서 작동합니다.

`list [-F` *format*`] [-H] [-p` *publisher* `...] - s` *repo_uri_or_path* `[`*pkg_fmri_pattern* `...]`
지정된 *pkg_fmri_pattern* 패턴과 일치하는 *repo_uri_or_path* 저장소의 패키지를 나열합니다. 패턴을 지정하지 않을 경우 저장소의 모든 패키지가 나열됩니다.

기본 출력에서 첫번째 열에는 패키지 게시자의 이름이 포함됩니다. 두번째 열에는 패키지 이름이 포함됩니다. 세번째 열은 패키지 상태를 보여 주는 플래그입니다. 상태 열의 `o` 값은 패키지가 오래된 것임을 나타냅니다. 상태 열의 `r` 값은 폐기 형식으로 패키지의 이름이 바뀌었음을 나타냅니다. 네번째 열에는 패키지의 릴리스 및 분기 버전이 포함됩니다. 릴리스 및 분기 버전에 대한 자세한 내용은 `pkg`(5)를 참조하십시오.

`-F`를 사용할 경우 대체 출력 형식을 지정합니다. *format*의 값은 `tsv`(탭으로 구분된 값), `json`(JavaScript 객체 표기법, 단일 행 사용) 또는 `json-formatted`(JavaScript 객체 표기법, 가독성을 위해 형식이 지정됨)일 수 있습니다.

`-H`를 사용할 경우 목록에서 헤더를 생략합니다.

`-p`를 사용할 경우 지정된 게시자의 패키지만 표시합니다. 제공하지 않을 경우 모든 게시자의 패키지가 표시됩니다. 이 옵션은 여러 번 지정할 수 있습니다.

`-s`를 사용할 경우 지정된 URI 또는 파일 시스템 경로에 있는 저장소에서 작동합니다.

`rebuild [-p` *publisher* `...] - s` *repo_uri_or_path* `[--no-catalog] [--no-index]`
저장소에 있는 모든 카탈로그, 검색 및 캐시된 기타 정보를 무시한 후 저장소의 현재 컨텐츠를 기반으로 다시 만듭니다.

`-p`를 사용할 경우 지정된 게시자에 대해서만 작업을 수행합니다. 제공하지 않거나 특수한 값 `all`을 지정할 경우 모든 게시자에 대해 작업이 수행됩니다. 이 옵션은 여러 번 지정할 수 있습니다.

`-s`를 사용할 경우 지정된 URI 또는 파일 시스템 경로에 있는 저장소에서 작동합니다.

`--no-catalog`를 사용할 경우 패키지 데이터를 재구성하지 않습니다.

`--no-index`를 사용할 경우 검색 색인을 재구성하지 않습니다.

`refresh [-p` *publisher* `...] - s` *repo_uri_or_path* `[--no-catalog] [--no-index]`
저장소에 있는 새 패키지를 카탈로그화하고 모든 검색 색인을 업데이트합니다. 이는 지연된 게시(`pkgsend`의 `--no-catalog` 또는 `--no-index` 옵션)에 사용됩니다.

`-p`를 사용할 경우 지정된 게시자에 대해서만 작업을 수행합니다. 제공하지 않거나 특수한 값 `all`을 지정할 경우 모든 게시자에 대해 작업이 수행됩니다. 이 옵션은 여러 번 지정할 수 있습니다.

`-s`를 사용할 경우 지정된 URI 또는 파일 시스템 경로에 있는 저장소에서 작동합니다.

`--no-catalog`를 사용할 경우 새 패키지를 추가하지 않습니다.

`--no-index`를 사용할 경우 검색 색인을 업데이트하지 않습니다.

`remove [-n] [-p` *publisher* `...] -s` *repo_uri_or_path pkg_fmri_pattern* `...`
다른 패키지가 사용하고 있지 않은 모든 참조된 파일을 비롯하여 지정된 패턴과 일치하는 패키지를 저장소에서 제거합니다.

주 – 관련 게시자에 대한 모든 검색 색인 데이터가 제거됩니다.

이 하위 명령은 파일 시스템 기반 저장소에서만 사용할 수 있습니다.

주의 – 이 작업은 되돌릴 수 없으며 다른 클라이언트가 저장소에 액세스하고 있는 동안에는 사용하지 않아야 합니다. 검색 작업 중 실패할 수 있기 때문입니다.

`-n`을 사용할 경우 패키지를 변경하지 않은 상태로 테스트 작업 실행을 수행합니다. 제거할 패키지 목록이 종료 전 표시됩니다.

`-p`를 사용할 경우 지정된 게시자에 대해서만 일치하는 패키지를 제거합니다. 제공하지 않을 경우 모든 게시자에 대해 일치하는 패키지가 모두 제거됩니다. 이 옵션은 여러 번 지정할 수 있습니다.

`-s`를 사용할 경우 지정된 URI 또는 파일 시스템 경로에 있는 저장소에서 작동합니다.

`set [-p` *publisher*`] - s` *repo_uri_or_path section/property* `=[`*value*`] ... or` *section/property* `=([`*value*`]) ...`
저장소 또는 게시자에 대해 지정된 등록 정보의 값을 설정합니다.

이 하위 명령은 파일 시스템 기반 저장소에서만 사용할 수 있습니다.

`-p`를 사용할 경우 지정된 게시자에 대해서만 등록 정보 데이터를 설정합니다. 게시자가 아직 존재하지 않을 경우 추가됩니다. 특수한 값 `all`을 사용하여 모든 게시자에 대한 등록 정보를 설정할 수 있습니다.

`-s`를 사용할 경우 지정된 URI 또는 파일 시스템 경로에 있는 저장소에서 작동합니다.

다음 형식 중 하나를 사용하여 등록 정보 및 값을 지정할 수 있습니다.

section/*property*=	등록 정보 값을 지웁니다.
section/*property*= *value*	등록 정보 값을 지정된 값으로 바꿉니다.
section/*property*=(*value1 value2 valueN*)	등록 정보 값을 값 목록으로 바꿉니다.

저장소 버전 3 및 4의 경우 저장소에 대해 다음 등록 정보를 설정할 수 있습니다.

`publisher/prefix` 기본 게시자의 이름을 나타내는 문자열입니다. 첫번째 문자는 a-z, A-Z 또는 0-9여야 합니다. 문자열의 나머지 부분에는 0-9, -, ., a-z 및 A-Z 문자만 포함될 수 있습니다. 이 값은 게시자 둘 이상의 패키지가 존재하거나 패키지가 저장소에 게시되었지만 게시자가 지정되지 않은 경우 사용해야 할 게시자를 나타냅니다.

저장소 버전 3 및 4의 경우 저장소의 개별 게시자에 대해 다음 등록 정보를 설정할 수 있습니다.

`publisher/alias`	클라이언트가 저장소의 구성 데이터를 사용하여 게시자를 추가할 때 사용해야 할 기본 별칭을 나타내는 문자열입니다. 첫번째 문자는 a-z, A-Z 또는 0-9여야 합니다. 문자열의 나머지 부분에는 0-9, -, ., a-z 및 A-Z 문자만 포함될 수 있습니다.
`repository/collection_type`	값은 이 저장소에 제공된 패키지의 유형을 나타내는 `core` 또는 `supplemental`일 수 있습니다. `core` 유형은 저장소의 패키지가 선언한 모든 종속성이 저장소에 포함되어 있음을 나타냅니다. `core` 유형은 주로 운영 체제 저장소에 사용됩니다. `supplemental` 유형은 다른 저장소에 있는 패키지에 종속되거나 해당 패키지와 함께 사용되는 패키지가 저장소에 포함되어 있음을 나타냅니다.
`repository/description`	저장소의 용도 및 컨텐츠를 기술하는 일반 텍스트의 단락입니다.
`repository/detailed_url`	저장소에 대한 추가 정보를 제공하는 문서(예: 웹 페이지)의 위치를 나타내는 URI입니다.
`repository/legal_uris`	저장소에 대한 추가 법적 정보를 제공하는 문서의 위치(URI) 목록입니다.
`repository/mirrors`	저장소 패키지 내용의 복사본은 포함하지만 패키지 메타 데이터는 포함하지 않는 저장소의 위치(URI) 목록입니다.
`repository/name`	저장소의 이름을 포함하는 일반 텍스트 문자열입니다.
`repository/origins`	저장소 패키지 메타 데이터 및 내용의 전체 복사본을 포함하는 저장소의 위치(URI) 목록입니다.
`repository/refresh_seconds`	각 업데이트 검사 후 저장소에서 업데이트된 패키지 데이터를 확인하기 전에 클라이언트가 기다릴 시간(초)을 나타내는 정수 값입니다.
`repository/registration_uri`	저장소에 액세스하는 데 필요한 인증서를 얻는 데 사용되어야 할 리소스의 위치를 나타내는 URI입니다. 등록 웹 페이지를 예로 들 수 있습니다.

repository/related_uris — 사용자가 관심을 가질 수 있는 패키지가 포함된 저장소의 위치(URI) 목록입니다.

여기서는 설명되지 않지만 get 하위 명령의 출력에 나열되는 등록 정보는 내부용으로 예약된 것이므로 설정하지 않아야 합니다.

version
pkg(5) 시스템의 버전을 식별하는 고유한 문자열입니다. version 작업으로 생성된 값은 정렬할 수 없으며 동등 이외의 비교에 안전하지 않습니다.

예

예 1 패키지 저장소 만들기

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

예 2 정보 표시

저장소의 게시자 및 패키지 수에 대한 요약을 표시합니다.

```
$ 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
```

예 3 카탈로그 및 검색 데이터 재구성

저장소의 카탈로그 및 검색 데이터를 재구성합니다.

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

예 4 카탈로그 및 검색 데이터 새로 고침

저장소의 카탈로그 및 검색 데이터를 새로 고칩니다.

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

예 5 모든 저장소 등록 정보 표시

```
$ 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
```

예 6 모든 게시자 등록 정보 표시

```
$ 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     ()
```

예 7 기본 게시자 설정

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

예 8 게시자 등록 정보 설정

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

예 9 저장소에 새 게시자 추가

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

종료 상태

다음 종료 값이 반환됩니다.

`0` 명령이 성공했습니다.

`1` 오류가 발생했습니다.

`2` 잘못된 명령줄 옵션이 지정되었습니다.

`3` 여러 명령이 요청되었지만 일부만 성공했습니다.

`4` 변경된 내용이 없어 수행할 작업이 없습니다.

`99` 예상치 않은 예외가 발생했습니다.

속성

다음 속성에 대한 설명은 `attributes`(5)를 참조하십시오.

속성 유형	속성 값
Availability	`package/pkg`
Interface Stability	커밋되지 않음

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

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

이름 pkgsend – 이미지 패키징 시스템 게시 클라이언트

개요

```
/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 ...]
```

설명 pkgsend는 패키지 매니페스트를 사용하여 새 패키지 및 새 패키지 버전을 이미지 패키징 저장소에 게시할 수 있도록 합니다. 저장소를 만들거나 관리하려면 pkgrepo(1)를 참조하십시오. 기존 저장소의 패키지를 기반으로 패키지 아카이브를 만들려면 pkgrecv(1)를 참조하십시오. 패키지 매니페스트에 대한 자세한 내용은 pkg(5)를 참조하십시오.

pkgsend 작업 후에 검색 색인을 구성하려면 저장소에서 pkgrepo refresh 또는 pkgrepo rebuild를 실행하십시오.

옵션 다음 옵션이 지원됩니다.

--help 또는 -? 사용법 메시지를 표시합니다.

하위 명령 지원되는 하위 명령은 다음과 같습니다.

generate [-T *pattern*] [--target *file*] *source* ...
각 *source*(예: SVR4 패키지, 디렉토리 또는 tar 파일)를 읽고 *source*를 기술하는 매니페스트를 stdout에 내보냅니다. 출력 매니페스트에서 file 및 dir 작업에 소유자가 root로 설정되고 그룹이 bin으로 설정됩니다.

그러면 출력 매니페스트에 주석을 추가하고, pkgdepend(1)를 사용하여 종속성을 추가 또는 분석하고, pkglint(1)를 사용하여 출력 매니페스트가 올바른지 확인한 후 publish 하위 명령으로 전달할 수 있습니다.

지원되는 소스는 다음과 같습니다.

- 파일 시스템 형식의 SVR4 패키지
- 데이터 스트림 형식의 SVR4 패키지
- tar 파일
- 디렉토리

소스에 있는 파일의 기본 이름이 -T를 사용하여 지정된 패턴과 일치하는 경우 해당 파일에 대한 작업에 파일의 시간 기록이 추가됩니다. *pattern*은 셸 일치 규칙을 사용합니다.

*	모든 항목과 일치합니다.
?	단일 문자와 일치합니다.
[*seq*]	*seq*의 임의의 문자와 일치합니다.

![*seq*] *seq*를 제외한 임의의 문자와 일치합니다.

지정된 소스가 디렉토리인 경우 단일 Inode에 대한 경로 이름이 여러 개 있으면 `file` 작업과 `hardlink` 작업을 구별할 수 있는 방법이 없습니다. 일반적으로 파일 시스템 영역에서 발견된 첫번째 작업이 file로 처리되며 나머지는 hardlink로 처리됩니다. 이는 파일 시스템 구현에 따라 임의적일 수 있습니다. 파일로 처리되어야 할 경로 이름을 지정하려면 `--target` 옵션에 대한 인수로 각 경로 이름을 전달하십시오. 다른 유형의 소스는 어떤 경로 이름이 file이며 어떤 경로 이름이 hardlink인지 표시할 수 있으므로 다른 유형의 소스에는 이 옵션이 적용되지 않습니다.

SVR4 패키지가 소스로 제공되면 `pkgsend`는 클래스 작업 스크립트가 있는 파일이 존재하지 않으며 사전 설치, 사후 설치, 사전 제거 또는 사후 제거 스크립트가 존재하지 않는지 확인합니다. `manifest` 클래스와 함께 설치된 SMF 매니페스트는 예외적입니다. `BASEDIR`은 재배치 가능한 모든 경로에서 제거됩니다.

SVR4 `DESC` 매개변수는 `pkg.description` 값으로 변환됩니다. SVR4 `NAME` 매개변수는 `pkg.summary` 값으로 변환됩니다.

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

제공된 소스에서 패키지에 대한 파일을 검색하여 지정된 패키지 매니페스트를 사용하는 패키지를 대상 패키지 저장소에 게시합니다. 매니페스트를 여러 개 지정할 경우 제공된 순서대로 조인됩니다. 매니페스트를 지정하지 않을 경우 `stdin`에서 매니페스트를 읽습니다.

`-b`를 사용할 경우 매니페스트에서 파일을 찾을 때 검색할 소스 목록에 지정된 번들을 추가합니다. 번들은 tar 파일, SVR4 패키지 등의 소스입니다. 이 옵션을 여러 번 지정하면 명령줄에 표시된 순서대로 소스가 검색됩니다. `-b`와 `-d`를 함께 지정할 경우 `-d` 소스가 먼저 검색됩니다. 지원되는 번들 및 사용 방법에 대한 자세한 내용은 위의 `generate` 하위 명령을 참조하십시오.

`-d`를 사용할 경우 매니페스트에서 파일을 찾을 때 검색할 소스 목록에 지정된 디렉토리를 추가합니다. 이 옵션을 여러 번 지정하면 명령줄에 표시된 순서대로 소스가 검색됩니다. 지원되는 소스 및 사용 방법에 대한 자세한 내용은 위의 `generate` 하위 명령을 참조하십시오.

`-s`를 사용할 경우 지정된 URI 또는 파일 시스템 경로에 있는 저장소에 패키지를 게시합니다. 게시 제한 및 제안 사항에 대한 자세한 내용은 아래의 "참고" 절을 참조하십시오. "환경 변수" 절도 참조하십시오.

`--no-catalog`를 사용할 경우 게시자의 카탈로그에 패키지를 추가하지 않습니다. 게시자 카탈로그에 대한 업데이트는 연속적으로 수행되어야 하므로 여러 패키지가 한 번에 게시될 때마다 이 옵션을 사용하는 것이 좋습니다. 게시가 완료되면 `pkgrepo(1)`의 `refresh` 하위 명령을 사용하여 개별 게시자 카탈로그에 새 패키지를 추가할 수 있습니다.

기타 모든 옵션의 경우 사용법 및 결과는 위의 `generate` 하위 명령을 참조하십시오.

환경 변수

PKG_REPO 대상 저장소의 경로 또는 URI입니다.

예

예 1 패키지 만들기 및 게시

pkgsend generate를 사용하여 패키지를 만들어 게시합니다.

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

foo.p5m의 시작 부분에 example.com 게시자에 대한 패키지 FMRI를 추가합니다.

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

결과 매니페스트는 다음과 같이 표시됩니다.

```
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
```

예 2 일반 패키지 만들기 및 게시

다음 행을 포함하는 example.com 게시자에 대한 매니페스트를 만듭니다.

```
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
```

패키지 게시:

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

예 3 기존 매니페스트 사용

게시 및 기존 매니페스트를 기반으로 파일 시스템을 사용하여 패키지를 게시합니다.

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

종료 상태

다음 종료 값이 반환됩니다.

0 명령이 성공했습니다.

1 오류가 발생했습니다.

2 잘못된 명령줄 옵션이 지정되었습니다.

99 예상치 않은 예외가 발생했습니다.

속성

다음 속성에 대한 설명은 attributes(5)를 참조하십시오.

속성 유형	속성 값
Availability	package/pkg
Interface Stability	커밋되지 않음

참조

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

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

주

게시 프로토콜 제한 사항으로 인해 크기가 128MB를 넘는 개별 패키지 파일을 게시할 때는 파일 시스템 기반 게시를 사용해야 합니다. 저장소에 대한 액세스 제어가 필요한 경우에도 파일 시스템 기반 게시가 권장됩니다.

파일 시스템 기반 게시를 사용할 때는 게시가 완료된 후 웹 인터페이스 또는 검색 응답에 변경 사항이 반영되도록 대상 저장소를 처리하고 있는 pkg.depotd 프로세스를 다시 시작해야 합니다. 자세한 내용은 pkg.depotd(1M)를 참조하십시오.

이름 pkgsign – 이미지 패키징 시스템 서명 유틸리티

개요

```
/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) ...
```

설명 pkgsign은 제공된 키 및 인증서를 사용하는 서명 작업을 추가하여 저장소의 적합한 위치에서 지정된 FMRI에 대한 매니페스트를 업데이트합니다. 수정된 패키지는 원래 시간 기록을 유지합니다.

옵션 다음 옵션이 지원됩니다.

-a를 사용할 경우 기본값 대신 서명 알고리즘 *hash_algorithm*을 사용합니다. 기본 서명 알고리즘은 rsa-sha256입니다. 지원되는 서명 알고리즘은 rsa-sha256, rsa-sha384, rsa-sha512, sha256, sha384 및 sha512입니다. 해시 알고리즘만 지정하는 서명 알고리즘을 사용하면 패키지의 매니페스트 해시가 서명 값이 됩니다. rsa 및 해시 알고리즘을 지정하는 서명 알고리즘을 사용하면 제공된 개인 키를 사용하여 서명된 매니페스트의 해시가 서명 값이 됩니다(-c 및 -k 옵션 참조).

-c를 사용할 경우 작업의 서명 값을 확인할 때 사용할 인증서로 *path_to_signing_certificate* 인증서를 추가합니다. -c 옵션은 -k 옵션과 함께만 사용할 수 있습니다.

-i를 사용할 경우 -c에 대한 인수로 지정된 *path_to_signing_certificate* 인증서를 검증할 때 사용할 인증서로 *path_to_intermediate_cert* 인증서를 추가합니다. -i를 여러 번 지정하여 여러 인증서를 제공할 수 있습니다.

-k를 사용할 경우 *path_to_private_key*에 저장된 개인 키를 사용하여 매니페스트를 서명합니다. -k 옵션은 -c 옵션과 함께만 사용할 수 있습니다. -k를 설정하지 않을 경우 매니페스트의 해시가 서명 값입니다.

-n을 사용할 경우 어떤 방식으로든 저장소를 변경하지 않는 테스트 실행을 수행합니다.

-s를 사용할 경우 *path_or_uri*의 저장소에 있는 패키지를 서명합니다.

--help를 사용할 경우 사용법 메시지를 표시합니다.

--no-index를 사용할 경우 서명된 매니페스트가 다시 게시된 후 저장소 검색 색인을 업데이트하지 않습니다.

--no-catalog를 사용할 경우 서명된 매니페스트가 다시 게시된 후 저장소 카탈로그를 업데이트하지 않습니다.

예

예 1 매니페스트의 해시 값을 사용하여 서명

매니페스트의 해시 값을 사용하여 `http://localhost:10000`에 게시된 패키지를 서명합니다. 이 방법은 테스트에 유용한 경우가 많습니다.

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

예 2 키 및 인증서를 사용하여 서명

매니페스트 해시 및 서명에 `rsa-sha384`를 사용하여 `/foo/bar`에 있는 파일 저장소에 게시된 패키지를 서명합니다. 서명 키는 `/key/usr2.key`에 있으며, 연결된 인증서는 `/key/usr2.cert`에, 인증서 검증에 필요한 인증서는 `/icerts/usr1.cert`에 있습니다.

```
$ 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
```

종료 상태

다음 종료 값이 반환됩니다.

`0` 명령이 성공했습니다.

`1` 오류가 발생했습니다.

`2` 잘못된 명령줄 옵션이 지정되었습니다.

`3` 여러 명령이 요청되었지만 일부만 성공했습니다.

`99` 예상치 않은 예외가 발생했습니다.

속성

다음 속성에 대한 설명은 `attributes`(5)를 참조하십시오.

속성 유형	속성 값
Availability	`package/pkg`
Interface Stability	커밋되지 않음

참조

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

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

이름 pm-updatemanager – 패키지 업데이트 응용 프로그램

개요

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

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

설명 pm-updatemanager는 시스템에 설치된 패키지에 대해 사용 가능한 업데이트를 확인하여 설치합니다.

주 – package/pkg, package/pkg/package-manager 또는 package/pkg/update-manager 패키지를 업데이트해야 할 경우 pm-updatemanager는 먼저 해당 패키지를 업데이트한 후 다시 시작하여 나머지 업데이트를 수행합니다.

옵션 다음 옵션이 지원됩니다.

-h 또는 --help	사용법 메시지를 표시합니다.
-d 또는 --debug	디버그 모드로 pm-updatemanager를 실행합니다.
-R 또는 --image-dir *dir*	자동으로 검색되는 이미지 대신 *dir*에 루트 지정된 이미지에서 작동합니다.

예

예 1 현재 이미지 업데이트

현재 이미지에서 pm-updatemanager를 호출합니다. 현재 이미지에 설치된 패키지에 대해 사용 가능한 모든 업데이트를 확인하여 설치합니다.

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

이 명령은 데스크탑 메뉴 옵션 System(시스템)>Administration(관리)>Update Manager(업데이트 관리자)가 호출하는 것과 동일한 명령입니다.

예 2 지정된 이미지 업데이트

/aux0/example_root에 저장된 이미지에서 pm-updatemanager를 호출합니다.

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

종료 상태 다음 종료 값이 반환됩니다.

0 모든 요소가 작동되었습니다.

1 오류가 발생했습니다.

2 잘못된 명령줄 옵션이 지정되었습니다.

속성 다음 속성에 대한 설명은 attributes(5)를 참조하십시오.

속성 유형	속성 값
Availability	package/pkg/update-manager

속성 유형	속성 값
Interface Stability	커밋되지 않음

참조 packagemanager(1), pkg(1), pkg(5)

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

주 소유하지 않은 이미지에서 작동하는 경우 충분한 권한이 있는 상태에서 pm-updatemanager를 호출해야 합니다. 일반적으로 이러한 경우 /usr/lib/pm-launch를 사용하여 pm-updatemanager를 호출합니다.

참 고

시스템 관리 명령

이름 pkg.depotd – 이미지 패키징 시스템 저장 서버

개요

```
/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]
```

설명 pkg.depotd는 이미지 패키징 시스템용 저장 서버입니다. 패키지 저장소에 포함된 데이터에 대한 네트워크 액세스를 제공합니다. 파일 시스템을 통해 저장소에 대한 직접 액세스를 지원하지 않는 클라이언트나 네트워크 액세스가 유일한 사용 가능 또는 기본 전송 방법인 클라이언트는 일반적으로 패키지 저장 서버를 사용합니다.

검색 클라이언트 pkg(1) 등의 클라이언트는 저장소에서 직접 또는 저장 서버를 통해 패키지 목록 및 패키지 메타 데이터를 검색할 수 있습니다. 게시 클라이언트 pkgsend(1)는 저장소로 직접 또는 저장 서버를 통해 패키지의 새 버전을 보낼 수 있습니다. pkgrepo(1)를 사용하면 저장 서버에 사용할 저장소를 만들거나 직접 또는 저장 서버를 통해 저장소를 관리할 수 있습니다.

일반적으로 pkg.depotd는 시스템에서 서비스로 실행됩니다. 패키지 및 소프트웨어 개발자는 테스트용으로 개인 복사본을 실행하고자 할 수 있습니다.

저장 서버는 고유의 액세스 제어 방법을 제공하지 않습니다. 기본적으로 연결할 수 있는 모든 클라이언트는 전체 패키지 데이터를 읽고 새 패키지 버전을 게시할 수 있습니다. SMF(서비스 관리 기능)에서 실행되는 경우를 제외하고 기본적으로 읽기 전용 모드로 실행합니다. 전개 컨텐츠와 함께 공용 저장 서버를 유지 관리하는 최상의 방법은 아래의 "참고" 절에서 설명됩니다.

Smf 등록 정보 일반적으로 pkg.depot 서버는 해당 서비스에 연결된 smf(5) 등록 정보를 통해 구성됩니다. 인식할 수 있는 등록 정보는 다음과 같습니다.

pkg/address
: (net_address) 연결을 위해 수신 대기할 IP 주소입니다. 기본값은 모든 활성 인터페이스에서 수신 대기하는 0.0.0.0(INADDR_ANY)입니다. 모든 활성 IPv6 인터페이스에서 수신 대기하려면 ::을 사용하십시오. 첫번째 값만 사용됩니다.

pkg/content_root
: (astring) 인스턴스가 정적 및 기타 웹 컨텐츠를 찾을 파일 시스템 경로입니다. 기본값은 /usr/share/lib/pkg입니다.

pkg/debug
: (astring) 사용으로 설정할 쉼표로 구분된 디버그 기능 목록입니다. 가능한 값은 다음과 같습니다.

  headers — 오류 로그에 모든 요청의 헤더를 기록합니다.

pkg/disable_ops	(astring) 저장 서버에 대해 사용 안함으로 설정할 쉼표로 구분된 작업 목록입니다. 작업은 *operation*[/*version*](예: catalog 또는 search_1)으로 지정됩니다.
pkg/image_root	(astring) 파일 정보가 파일 데이터 캐시로 사용될 이미지에 대한 경로입니다.
pkg/inst_root	(astring) 인스턴스가 저장소 데이터를 찾을 파일 시스템 경로입니다. file_root 또는 PKG_REPO가 제공되지 않은 경우 필요합니다. 기본값은 /var/pkgrepo입니다.
pkg/log_access	(astring) 저장 프로세스가 기록한 액세스 관련 정보에 대한 대상입니다. 가능한 값은 stderr, stdout, none 또는 절대 경로 이름입니다. stdout가 tty인 경우 기본값은 stdout입니다. stdout가 tty가 아닌 경우 기본값은 none입니다.
pkg/log_errors	(astring) 저장 프로세스가 기록한 오류 또는 기타 정보에 대한 대상입니다. 가능한 값은 stderr, stdout, none 또는 절대 경로 이름입니다. 기본값은 stderr입니다.
pkg/mirror	(boolean) 패키지 미러 모드 사용 여부를 설정합니다. true인 경우 게시 및 메타 데이터 작업이 사용 안함으로 설정되며 제한된 브라우저 사용자 인터페이스만 제공됩니다. pkg/readonly 등록 정보가 true인 경우 이 등록 정보는 true일 수 없습니다. 기본값은 false입니다.
pkg/port	(count) 인스턴스가 수신 패키지 요청을 수신 대기할 포트 번호입니다. SSL 인증서 및 키 정보가 제공되지 않은 경우 기본값은 80이며, 그렇지 않은 경우 기본값은 443입니다.
pkg/proxy_base	(uri) 이 등록 정보는 저장 서버에 대한 기본 URL을 변경하며 Apache 또는 역 프록시 구성의 기타 웹 서버에서 실행될 때 가장 유용합니다.
pkg/readonly	(boolean) 수정 작업(예: pkgsend(1)가 시작한 작업)을 사용 안함으로 설정할지 여부를 설정합니다. 검색 작업은 계속 사용할 수 있습니다. pkg/mirror 등록 정보가 true인 경우 이 등록 정보는 true일 수 없습니다. 기본값은 true입니다.

`pkg/socket_timeout` (count) 연결이 해제되기 전에 서버가 클라이언트의 응답을 기다릴 최대 시간(초)입니다. 기본값은 60입니다.

`pkg/sort_file_max_size` (count) 인덱서 정렬 파일의 최대 크기입니다. 저장 서버가 인덱싱을 위해 사용하는 RAM의 크기를 제한하거나 속도를 위해 크기를 늘리는 데 사용됩니다.

`pkg/ssl_cert_file` (astring) PEM 인코딩 인증서 파일에 대한 절대 경로 이름입니다. 기본값은 `none`입니다. 이 등록 정보는 `ssl_key_file`과 함께 사용해야 합니다. `ssl_cert_file`과 `/ssl_key_file`이 모두 제공된 경우에만 저장 서버가 SSL 요청에 응답합니다.

`pkg/ssl_dialog` (astring) `ssl_key_file` 해독에 사용되는 암호문을 얻을 때 사용할 방법을 지정합니다. 가능한 값은 다음과 같습니다.

`builtin`	암호문에 대한 프롬프트입니다. 이것이 기본값입니다.
`exec:`*/path/to/program*	암호문을 얻기 위해 지정된 외부 프로그램을 실행합니다. 프로그램에 대한 첫번째 인수는 ''이며 예약되어 있습니다. 프로그램에 대한 두번째 인수는 서버의 포트 번호입니다. 암호문은 `stdout`에 인쇄됩니다.
`smf:fmri`	FMRI와 관련된 서비스 인스턴스에서 `pkg_secure/ssl_key_passphrase` 등록 정보의 값을 검색하려고 시도합니다.

`pkg/ssl_key_file` (astring) PEM 인코딩 개인 키 파일에 대한 절대 경로 이름입니다. 이 등록 정보는 `ssl_cert_file` 등록 정보와 함께 사용해야 합니다.

	/ssl_key_file과 ssl_cert_file이 모두 제공된 경우에만 저장 서버가 SSL 요청에 응답합니다.
pkg/threads	(count) 요청을 처리하기 위해 시작된 스레드 수입니다. 기본값은 60입니다. 작은 규모의 배치에만 적합합니다. 이 값은 동시 클라이언트 수의 약 20배여야 합니다. threads의 최대 값은 5000입니다.
pkg/writable_root	(astring) 프로그램이 쓰기 액세스 권한을 가진 디렉토리에 대한 파일 시스템 경로입니다. 이 등록 정보는 패키지 정보에 대한 쓰기 액세스 권한 없이 저장 서버를 사용으로 설정하여 파일(예: 검색 색인)을 만드는 -readonly 옵션과 함께 사용됩니다.
pkg_secure/ssl_key_passphrase	(astring) pkg/ssl_key_file 해독에 사용할 암호입니다. 이 값은 solaris.smf.read.pkg-server 속성을 사용하여 보호되는 읽기 권한 부여입니다.

저장 서버에 대한 BUI(브라우저 사용자 인터페이스)의 표시 및 동작은 다음 등록 정보를 통해 제어됩니다.

pkg_bui/feed_description	(astring) RSS/기본 단위 피드에 대한 설명 단락입니다.
pkg_bui/feed_icon	(astring) RSS/기본 단위 피드를 시각적으로 표현하는 데 사용되는 작은 이미지의 경로 이름입니다. 경로 이름은 content_root에 상대적이어야 합니다. 기본값은 web/_themes/pkg-block-icon.png입니다.
pkg_bui/feed_logo	(astring) RSS/기본 단위 피드를 시각적으로 브랜딩하거나 식별하는 데 사용할 큰 이미지의 경로 이름입니다. 이 값은 content_root에 상대적이어야 합니다. 기본값은 web/_themes/pkg-block-icon.png입니다.
pkg_bui/feed_name	(astring) 저장소를 제공하는 저장 서버가 생성한 RSS/기본 단위 피드에 대한 짧은 설명이 포함된 이름입니다. 기본값은 “package repository feed”입니다.
pkg_bui/feed_window	(count) 저장소에 대한 피드가 마지막으로 생성되기 전의 시간(시)으로, 피드 생성 시 포함됩니다.

패키지 저장 서버를 pkg(5)의 로컬 클라이언트 이미지에 대한 미러 서버로 사용할 수도 있습니다. 그러면 LAN의 서브넷을 공유하는 클라이언트가 파일 캐시를 미러링할 수 있습니다. 클라이언트는 다른 클라이언트에서 파일을 다운로드할 수 있으므로 패키지

저장 서버에서 로드가 줄어듭니다. 이 기능은 smf(5)에 의해 구성된 대체 저장 서비스로 사용할 수 있습니다. 서비스 검색에 mDNS 및 dns-sd가 사용됩니다.

일반적으로 mDNS 미러는 해당 서비스에 연결된 smf(5) 등록 정보를 통해 구성됩니다. 인식할 수 있는 등록 정보는 다음과 같습니다.

pkg/image_root (astring) 파일 정보가 파일 데이터 캐시로 사용될 이미지에 대한 경로입니다. 기본값은 /입니다.

pkg/port (count) 인스턴스가 수신 패키지 요청을 수신 대기할 포트 번호입니다. 기본값은 80입니다.

옵션

pkg.depotd는 파일 또는 기존 smf(5) 서비스 인스턴스의 등록 정보 데이터에서 기본 구성 정보를 읽을 수 있습니다.

--cfg *source* 구성 데이터를 읽고 쓸 때 사용할 파일의 경로 이름 또는 smf:*fmri* 형식의 문자열입니다. 여기서 *fmri*는 구성 데이터를 읽어올 인스턴스의 서비스 FMRI(결함 관리 리소스 식별자)입니다. 지정된 파일의 형식에 대한 자세한 내용은 아래의 "저장 서버 구성"을 참조하십시오.

사용 가능한 기존 구성 소스가 없거나 --cfg를 사용하여 제공된 구성 파일에서 읽은 값을 대체하려는 경우 다음 옵션을 사용하여 저장 서버의 기본 동작을 변경할 수 있습니다.

-a address 위의 pkg/address를 참조하십시오.

--content-root *root_dir* 위의 pkg/content_root를 참조하십시오.

-d *inst_root* 위의 pkg/inst_root를 참조하십시오.

--debug *features* 위의 pkg/debug를 참조하십시오.

--disable-ops *op_list* 위의 pkg/disable_ops를 참조하십시오.

--image-root *path* 위의 pkg/image_root를 참조하십시오.

--log-access *dest* 위의 pkg/log_access를 참조하십시오.

--log-errors *dest* 위의 pkg/log_errors를 참조하십시오.

--mirror 위의 pkg/mirror를 참조하십시오.

-p *port* 위의 pkg/port를 참조하십시오.

--proxy-base *url* 위의 pkg/proxy_base를 참조하십시오. 빈 값이 제공된 경우 이 옵션은 무시됩니다.

--readonly 위의 pkg/readonly를 참조하십시오.

-s *threads* 위의 pkg/threads를 참조하십시오.

-s-ort-file-max-size *bytes* 위의 pkg/sort_file_max_size를 참조하십시오.

`--ssl-cert-file` *source*	위의 `pkg/ssl_cert_file`을 참조하십시오.
`--ssl-dialog` *type*	위의 `pkg/ssl_dialog`를 참조하십시오.
`--ssl-key-file` *source*	위의 `pkg/ssl_key_file`을 참조하십시오.
`-t` *socket_timeout*	위의 `pkg/socket_timeout`을 참조하십시오.
`--writable-root` *path*	위의 `pkg/writable_root`를 참조하십시오.

패키지 저장소에 대한 추가 관리 기능은 `pkgrepo`(1)가 제공합니다.

저장 서버 구성

`smf`(5) FMRI 대신 `--cfg` 옵션을 사용하여 구성 파일이 제공된 경우 저장 서버는 간단한 텍스트 형식으로 모든 구성 데이터를 읽고 씁니다. 구성 데이터는 위의 "SMF 등록 정보"에서 설명됩니다. 구성 데이터는 [*section*] 헤더, `name` = *value* 항목 순의 섹션으로 구성됩니다. 연속 항목은 RFC 822 스타일입니다. 연속 행을 공백으로 시작하여 여러 행으로 값을 나눌 수 있습니다.

구성 파일에 제공되지 않은 필수 값은 위의 "옵션"에 나열된 옵션을 사용하여 제공해야 합니다. 샘플 구성 파일은 다음과 같이 표시될 수 있습니다.

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

[pub_example_com]
feed_description = example.com’s software
  update log
```

예

예 1 저장 서버를 사용으로 설정

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

예 2 서버의 수신 포트 변경

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

예 3 미러를 사용으로 설정

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

환경 변수

`PKG_REPO`	제공할 저장소가 포함된 디렉토리를 지정합니다. `-d`가 지정된 경우 이 값은 무시됩니다.
`PKG_DEPOT_CONTENT`	저장 서버가 제공한 정적 컨텐츠가 포함된 디렉토리를 지정합니다. "Files" 아래에 나열되는 파일이 이 디렉토리에 존재해야 합니다. 단, 컨텐츠는 제공된 기본 컨텐츠와 다를 수 있습니다.

종료 상태

다음 종료 값이 반환됩니다.

`0` 작업이 성공했습니다.

`1` 오류가 발생했습니다.

`2` 잘못된 명령줄 옵션이 지정되었습니다.

`99` 예상치 않은 예외가 발생했습니다.

파일

`/usr/share/lib/pkg` 기본 프레젠테이션 컨텐츠 위치입니다. 대체 위치를 선택하려면 `pkg/content_root`를 수정합니다.

속성

다음 속성에 대한 설명은 `attributes`(5)를 참조하십시오.

속성 유형	속성 값
Availability	`package/pkg`
Interface Stability	커밋되지 않음

참조

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

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

주

`pkd.depotd` 서비스는 서비스 식별자 `svc:/application/pkg/server`의 SMF가 관리합니다.

mDNS 미러 서비스는 서비스 식별자 `svc:/application/pkg/dynamic-mirror`의 SMF가 관리합니다.

저장 서버에 대한 읽기 액세스를 제어하려는 경우 인증 방법(예: 클라이언트 기반 SSL 인증서 액세스)과 함께 `pkg`(1)가 기본적으로 지원하는 HTTP 역 프록시를 사용할 수 있습니다.

구성을 변경하거나 파일 시스템 기반 작업을 사용하여 패키지 데이터를 변경하면 변경 사항이 작업 및 출력에 반영될 수 있도록 저장 서버 프로세스를 다시 시작해야 합니다. 저장 서버 프로세스를 다시 시작하려면 다음 방법 중 하나를 사용하십시오.

- `application/pkg/server` 인스턴스를 다시 시작하려면 `svcadm`(1M)을 사용합니다.
- `kill`(1)을 사용하여 저장 서버 프로세스로 `SIGUSR1` 신호를 보냅니다. 그러면 프로세스는 그대로 유지하지만 모든 구성, 패키지 및 검색 데이터는 재로드하는 "정상적인 다시 시작"이 실행됩니다.

  `# kill -USR1` *pid*

이름

pkg.sysrepo – 이미지 패키징 시스템의 시스템 저장소 구성

개요

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

설명

pkg.sysrepo는 IPS(이미지 패키징 시스템)의 시스템 저장소용 구성 파일을 생성하는 데 사용됩니다. pkg.sysrepo는 svc:/application/pkg/system-repository SMF(서비스 관리 기능) 서비스에서 호출합니다. 구성 변경 사항은 SMF 서비스의 등록 정보에 적용되어야 합니다.

시스템 저장소는 중앙 집중식 프록시를 통해 참조 이미지에서 구성된 패키지 저장소에 대한 액세스를 제공합니다. 해당 참조 이미지에 적용된 게시자 구성 변경 사항은 시스템 저장소를 사용하도록 구성된 모든 클라이언트가 바로 확인합니다.

시스템 저장소는 주로 전역 영역에서 사용되는데 비전역 영역이 전역 영역에서 구성된 저장소에 액세스할 수 있도록 허용합니다. SMF 서비스 svc:/application/pkg/zones-proxyd 및 svc:/application/pkg/zones-proxy-client는 비전역 영역과 전역 영역 간의 전송을 제공합니다. 이 전송은 pkg(5)에서만 사용됩니다.

http, https 및 v4 파일 저장소만 지원됩니다. p5p 기반 파일 저장소 또는 보다 오래된 파일 저장소 형식은 지원되지 않습니다. 저장소 버전에 대한 자세한 내용은 pkgrepo(1)를 참조하십시오.

옵션

다음 옵션이 지원됩니다.

-c *cache_dir*
: 시스템 저장소가 구성된 게시자의 응답을 캐싱하는 데 사용할 디렉토리에 대한 절대 경로입니다.

  기본적으로 파일 캐시가 사용됩니다. 하지만 특수한 값 memory를 사용하여 인메모리 캐시가 사용되도록 지정할 수 있습니다. 특수한 값 None을 사용하면 시스템 저장소가 캐싱을 수행하지 않도록 지정할 수 있습니다. 이 설정은 config/cache_dir SMF 등록 정보를 사용하여 구성해야 합니다.

-p *port*
: 시스템 저장소가 요청 수신 대기에 사용할 포트입니다. 이 설정은 config/port SMF 등록 정보를 사용하여 구성해야 합니다.

-s *cache_size*
: 시스템 저장소의 최대 캐시 크기를 정의하는 정수 값(MB)입니다. 이 설정은 config/cache_max SMF 등록 정보를 사용하여 구성해야 합니다.

-w *http_proxy*
: 시스템 저장소가 HTTP 기반 패키지 저장소에 액세스하는 데 사용할 수 있는 웹 프록시를 정의하는 *scheme*://*hostname*[:*port*] 형식의 문자열입니다. 이 설정은 config/http_proxy SMF 등록 정보를 사용하여 구성할 수 있습니다.

-W *https_proxy*
: 시스템 저장소가 HTTPS 기반 패키지 저장소에 액세스하는 데 사용할 수 있는 웹 프록시를 정의하는 *scheme*://*hostname*[:*port*] 형식의 문자열입니다. 이 설정은 config/https_proxy SMF 등록 정보를

사용하여 구성할 수 있습니다.

예

예 1 시스템 저장소를 사용으로 설정

```
$ svcadm enable svc:/application/pkg/system-repository
```

종료 상태

다음 종료 값이 반환됩니다.

`0` 명령이 성공했습니다.

`1` 명령을 통해 유효한 구성을 쓰는 데 실패했습니다.

`2` 잘못된 명령줄 옵션이 지정되었습니다.

`99` 예상치 않은 예외가 발생했습니다.

속성

다음 속성에 대한 설명은 `attributes(5)`를 참조하십시오.

속성 유형	속성 값
Availability	`package/pkg`
Interface Stability	커밋되지 않음

참조

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

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

참 고

표준, 환경 및 매크로

이름

pkg – 이미지 패키징 시스템

설명

이미지 패키징 시스템인 pkg(5)는 소프트웨어 수명 주기 관리(설치, 업그레이드 및 제거)를 제공하는 프레임워크입니다. 이미지 패키징은 일련의 키/값 쌍 및 가능한 데이터 페이로드에 의해 정의된 작업 컬렉션인 패키지 단위로 소프트웨어를 관리합니다. 대부분의 경우 작업은 파일 시스템에서 발견된 파일이지만 설치 가능한 다른 객체(예: 드라이버, 서비스 및 사용자)를 나타내기도 합니다.

패키지 Fmri 및 버전

각 패키지는 pkg: 체계를 사용하여 FMRI(결함 관리 리소스 식별자)로 표시됩니다. 패키지의 전체 FMRI는 다음 형식의 체계, 게시자, 패키지 이름 및 버전 문자열로 구성됩니다.

```
pkg://solaris/system/library/c++-runtime@0.5.11,5.11-0.174.0.0.0.0.0:20110921T190358Z
```

solaris는 게시자입니다. system/library/c++-runtime은 패키지 이름입니다. 이름 공간은 계층적이며 길이가 임의적이지만 적용되는 제약이 없으며 이름이 임의적입니다. 게시자 정보는 선택적이지만 있을 경우 pkg://로 시작되어야 합니다. 게시자를 포함하는 FMRI는 "정규화"된 것으로 지칭되는 경우가 많습니다. 게시자 정보가 없을 경우 일반적으로 패키지 이름은 pkg:/로 시작되어야 합니다.

패키징 클라이언트가 게시자 정보를 포함하지 않는 FMRI의 체계를 생략할 수 있도록 허용하는 경우도 많습니다. 예를 들어, pkg:/system/library/c++-runtime을 system/library/c++-runtime으로 작성할 수 있습니다. 체계가 생략되면 클라이언트도 일치 용도로 사용할 패키지 이름의 마지막 구성 요소를 제외한 모든 부분을 생략할 수 있도록 허용합니다. 예를 들어, system/library/c++-runtime을 library/c++-runtime 또는 c++-runtime으로 작성할 수 있으며, 이는 이름이 c++-runtime인 패키지 또는 /c++-runtime으로 끝나는 패키지 이름과 일치됩니다.

게시자 이름은 개인, 개인 그룹 또는 조직을 하나 이상의 패키지에 대한 소스로 식별합니다. 게시자 이름이 충돌하지 않도록 하고 게시자를 쉽게 식별할 수 있도록 하려면 패키지 게시 엔티티를 게시자 이름으로 나타내는 도메인 이름을 사용하는 것이 가장 좋습니다.

패키지 이름 뒤에는 @ 기호로 구분된 버전이 옵니다. 버전은 문장 부호로 구분된 네 시퀀스의 숫자로 구성됩니다. 처음 세 시퀀스의 요소는 점으로 구분되며 시퀀스의 길이는 임의적입니다. 버전 구성 요소의 선행 0(예: 01.1 또는 1.01)은 허용되지 않습니다. 후행 0(예: 1.10)은 허용됩니다.

버전의 첫번째 부분은 구성 요소 버전입니다. 운영 체제에 긴밀하게 바인딩된 구성 요소의 경우 일반적으로 해당 운영 체제 버전의 uname -r 값입니다. 고유의 개발 수명 주기가 있는 구성 요소의 경우 이 시퀀스는 점으로 구분된 릴리스 번호(예: 2.4.10)입니다.

버전의 두번째 부분은 빌드 버전으로, 있을 경우 쉼표 앞에 와야 합니다. 빌드 버전은 패키지 내용이 성공적으로 실행되는 데 필요한 운영 체제 버전에 대한 최소 한도를 제공하여 패키지 내용이 작성된 운영 체제의 버전을 지정합니다.

버전의 세번째 부분은 분기 버전으로, 있을 경우 하이픈(-) 앞에 와야 합니다. 분기 버전은 공급업체 관련 정보를 제공하는 버전 구성 요소입니다. 분기 버전은 패키징 메타 데이터가 변경될 때 구성 요소 버전과 별개로 증분될 수 있습니다. 분기 버전에는 빌드 번호 또는 기타 정보가 포함될 수도 있습니다.

버전의 네번째 부분은 시간 기록으로, 있을 경우 콜론(:) 앞에 와야 합니다. 시간 기록은 패키지가 게시된 시간을 나타냅니다.

버전 간의 비교를 수행할 때 왼쪽에 있는 구성 요소가 동일하지 않은 경우 전체 버전의 구성 요소가 없는 것으로 간주됩니다. 따라서 "4.3"이 "4.2"보다 크므로 "4.3-1"이 "4.2-7"보다 높은 버전이며 "3"이 "1"보다 크므로 "4.3-3"이 "4.3-1"보다 높은 버전입니다.

표시되거나 필요한 정보의 양을 줄이기 위해 적합한 경우 시스템의 여러 요소가 FMRI를 표시할 때 FMRI를 요약하며 보다 짧은 형식의 입력값을 승인합니다. 일반적으로 체계, 게시자, 빌드 버전 및 시간 기록을 편집할 수 있습니다. 버전 정보를 모두 생략할 수 있는 경우도 있습니다.

작업

작업은 시스템에 설치 가능한 객체를 나타내며, 패키지의 매니페스트에 기술됩니다. 각 작업은 기본적으로 이름과 주요 속성으로 구성됩니다. 각각 버전 내역을 따르므로 각 작업의 조합은 고유 객체를 나타냅니다. 작업은 다른 속성을 가질 수 있습니다. 패키징 시스템에서 직접 해석하는 속성도 있고, 시스템 관리자 또는 최종 사용자에게만 유용한 속성도 있습니다.

작업은 다음과 같이 간단한 텍스트로 표현됩니다.

action_name attribute1=value1 attribute2=value2 ...

속성의 이름에는 공백, 따옴표 또는 등호(=)가 사용될 수 없습니다. 첫번째 등호 뒤의 모든 문자는 값에 속합니다. 값에는 공백, 따옴표 또는 등호가 모두 사용될 수 있습니다. 단, 공백은 작은따옴표 또는 큰따옴표로 묶여야 합니다. 작은따옴표는 큰따옴표로 묶인 문자열 안에서 제어되지 않아도 되며 큰따옴표는 작은따옴표로 묶인 문자열 안에서 제어되지 않아도 됩니다. 따옴표가 사용된 문자열이 종료되지 않도록 따옴표 앞에 백슬래시(\)를 붙일 수 있습니다. 백슬래시는 백슬래시로 제어될 수 있습니다.

속성의 이름은 여러 값을 사용하여 두 번 이상 지정될 수 있습니다. 이러한 속성은 순서가 지정되지 않은 목록으로 처리됩니다.

속성이 여러 개인 작업은 매니페스트 파일에 긴 행을 만들 수 있습니다. 완성되지 않은 각 행의 끝에 백슬래시를 붙여 해당 행을 줄 바꿈할 수 있습니다. 이 연속 문자는 속성/값 쌍 사이에 있어야 합니다. 속성, 속성 값 및 조합은 나눌 수 없습니다.

아래 나열되는 속성은 전체 세트가 아닙니다. 실제로, 작업에 연결될 수 있는 속성은 임의적이며 표준 속성 세트를 간편하게 확장하여 향후 개발을 통합할 수 있습니다.

패키징 컨텍스트 외부에서 추가 작업이 실행되도록 하는 작업 속성도 있습니다. 이러한 속성은 아래의 "액추에이터" 절에서 설명됩니다.

File 작업

file 작업은 일반 파일을 나타냅니다. file 작업은 페이로드를 참조하며 다음과 같은 네 개의 표준 속성을 가집니다.

path 파일이 설치된 파일 시스템 경로입니다. 이 속성은 file 작업의 키 속성입니다.

mode 파일의 액세스 권한(숫자 형식)입니다. 이러한 권한은 ACL이 아니라 단순한 권한일 뿐입니다.

owner 파일을 소유한 사용자의 이름입니다.

group 파일을 소유한 그룹의 이름입니다.

페이로드는 이름이 지정되지 않는다는 점에서 위치 속성으로, 작업 이름 뒤의 첫번째 단어입니다. 게시된 매니페스트에서는 파일 내용의 SHA-1 해시입니다. 게시해야 할 매니페스트에 있을 경우 페이로드를 찾을 수 있는 경로를 나타냅니다. pkgsend(1)를 참조하십시오. 값에 등호가 포함된 경우 위치 속성 대신 해시 속성을 사용할 수 있습니다. 동일한 작업에 위치 속성과 해시 속성을 함께 사용할 수 있습니다. 단, 해시는 같아야 합니다.

기타 속성은 다음과 같습니다.

preserve 이 속성은 파일이 설치되거나 마지막으로 업그레이드된 이후 파일 내용이 변경된 것으로 확인된 경우 업그레이드 시 파일 내용을 겹쳐 쓰지 않아야 하도록 지정합니다. 초기 설치 시 기존 파일이 발견된 경우 파일이 복원됩니다(/var/pkg/lost+found에 저장됨).

preserve의 값이 renameold인 경우 기존 파일의 이름이 확장자 .old로 바뀌며 새 파일이 해당 위치에 삽입됩니다.

preserve의 값이 renamenew인 경우 기존 파일이 그대도 유지되며 새 파일이 확장자 .new로 설치됩니다.

preserve의 값이 legacy인 경우 초기 패키지 설치를 위해 이 파일이 설치되지 않습니다. 업그레이드 시 기존 파일의 이름이 확장자 .legacy로 바뀐 후 새 파일이 해당 위치에 삽입됩니다.

preserve의 값이 true(또는 strawberry와 같이 위에 나열되지 않은 값)인 경우 기존 파일이 그대로 유지되며 새 파일이 설치되지 않습니다.

overlay 이 속성은 다른 패키지가 동일한 위치의 파일을 전달할 수 있도록 작업이 허용하지 여부 또는 작업이 다른 패키지를 오버레이하는 데 사용할 파일을 전달하는지 여부를 지정합니다. 이 기능은 자체 어셈블리(예: /etc/motd)에 참여하지 않으며 겹쳐 써도 문제가 없는 구성 파일에 사용됩니다.

overlay를 지정하지 않을 경우 여러 패키지가 파일을 동일한 위치로 전달할 수 없습니다.

overlay의 값이 allow인 경우 하나의 다른 패키지만 파일을 동일한 위치로 전달할 수 있도록 허용됩니다. preserve 속성이 함께 설정되지 않은 경우 이 값은 적용되지 않습니다.

overlay의 값이 true인 경우 작업이 전달하는 파일이 allow를 지정한 다른 작업을 겹쳐 씁니다. 설치된 파일에 대한 변경 사항은 오버레이 파일의 preserve 속성 값에 따라 보존됩니다. 제거 시 오버레이하려는 작업이 아직 설치된 상태인 경우 preserve 속성 지정 여부에 관계없이 파일 내용이 보존됩니다. 하나의 작업만 다른 작업을 오버레이할 수 있으며 mode, owner 및 group 속성이 일치해야 합니다.

"기호에 따라" 파일을 사용해 볼 수 있으며 종류에 따라 파일에는 추가 속성이 포함될 수 있습니다. ELF 파일의 경우 인식할 수 있는 속성은 다음과 같습니다.

elfarch — ELF 파일의 구조입니다. 이 속성은 파일이 작성된 구조에 대한 uname -p의 출력입니다.

elfbits — 이 속성은 32 또는 64입니다.

elfhash — 이 속성은 파일에서 "관심 있는" ELF 섹션의 해시입니다. 해당 섹션은 이진이 로드될 때 메모리에 매핑되는 섹션으로, 두 가지 이진의 실행 가능 동작이 다를지 여부를 확인할 때 고려해야 할 유일한 섹션입니다.

original_name — 이 속성은 패키지 간에 이동하거나 위치 간에 이동하거나 두 가지 모두에서 이동하는 편집 가능한 파일을 처리할 때 사용됩니다. 이 속성에는 원래 패키지 이름, 콜론, 원래 파일 경로 순의 형식이 사용됩니다. 삭제하려는 파일은 패키지와 경로 또는 original_name 속성(지정된 경우)의 값과 함께 기록됩니다. 동일한 패키징 작업의 일부로 삭제된 경우 설치하려는 편집 가능한 파일에 original_name 속성 세트가 있으면 해당 이름의 파일이 사용됩니다.

revert-tag — 이 속성은 하나의 세트로 되돌려야 할 편집 가능한 파일의 태그를 지정할 때 사용됩니다. revert-tag 값은 여러 개 지정할 수 있습니다. 해당 태그가 지정된 상태로 pkg revert가 호출되면 매니페스트에 정의된 상태로 파일이 되돌려집니다. pkg(1)를 참조하십시오.

Directory 작업

dir 작업은 파일 시스템 객체를 나타낸다는 점에서 file 작업과 유사합니다. dir 작업은 일반 파일 대신 디렉토리를 나타냅니다. dir 작업의 표준 속성은 file 작업과 동일한 네 개이며 path가 키 속성입니다.

디렉토리는 IPS에서 카운트되는 참조입니다. 명시적 또는 암시적으로 디렉토리를 참조하는 마지막 패키지가 더 이상 디렉토리를 참조하지 않을 경우 해당 디렉토리가 제거됩니다. 해당 디렉토리에 패키지화되지 않은 파일 시스템 객체가 포함된 경우 해당 항목은 $IMAGE_META/lost+found로 이동됩니다. $IMAGE_META에 대한 자세한 내용은 "파일" 절을 참조하십시오.

패키지화되지 않은 컨텐츠를 새 디렉토리로 이동하려는 경우 다음 속성이 유용할 수 있습니다.

`salvage-from` 이 속성은 복구된 항목의 디렉토리에 대한 이름을 지정합니다. 해당 속성을 가진 디렉토리는 만들어질 때 복구된 디렉토리 컨텐츠(있을 경우)를 상속합니다.

Link 작업 `link` 작업은 심볼릭 링크를 나타냅니다. `link` 작업의 표준 속성은 다음과 같습니다.

`path`
심볼릭 링크가 설치된 파일 시스템 경로입니다. 이 속성은 `link` 작업의 키 속성입니다.

`target`
심볼릭 링크의 대상으로, 링크가 확인되는 파일 시스템 객체입니다.

`mediator`
지정된 조정 그룹(예: `python`)에 참여한 모든 경로 이름이 공유하는 조정 이름 공간의 항목을 지정합니다. 링크 조정은 `mediator-version` 및/또는 `mediator-implementation`을 기반으로 수행할 수 있습니다. 지정된 경로 이름에 대해 조정된 모든 링크는 동일한 조정자를 지정해야 합니다. 단, 모든 조정자 버전 및 구현이 지정된 경로에서 링크를 제공해야 하는 것은 아닙니다. 조정이 링크를 제공하지 않을 경우 해당 조정이 선택될 때 링크가 제거됩니다. 특정 버전 및/또는 구현과 함께 `mediator`는 패키징 시스템에서 사용되도록 선택할 수 있는 조정을 나타냅니다.

`mediator-version`
`mediator` 속성에 의해 기술된 인터페이스의 버전(점으로 구분된 음수가 아닌 정수 시퀀스로 표시됨)을 지정합니다. `mediator`는 지정되었으며 `mediator-implementation`은 지정되지 않은 경우 이 속성이 필요합니다. 로컬 시스템 관리자는 명시적으로 사용할 버전을 설정할 수 있습니다. 지정된 값은 일반적으로 링크를 전달하는 패키지의 버전과 일치해야 합니다. 예를 들어, `runtime/python-26`은 필요하지 않지만 `mediator-version=2.6`을 사용해야 합니다.

`mediator-implementation`
`mediator-version`과 함께 또는 이 속성 대신 사용할 조정자의 구현을 지정합니다. 시스템 관리자가 명시적으로 지정하지 않은 경우 구현 문자열은 순서가 지정되지 않은 것으로 간주되며 문자열은 pkg(5)에 의해 임의적으로 선택됩니다.

값은 영숫자와 공백으로 구성된 임의적 길이의 문자열일 수 있습니다. 구현 자체의 버전을 지정할 수 있거나 버전이 지정된 경우 `@` 뒤의 문자열 끝에 버전(점으로 구분된 음수가 아닌 정수 시퀀스로 표시됨)을 지정해야 합니다. 구현의 버전이 여러 개 존재하는 경우 기본 동작은 최신 버전의 구현을 선택하는 것입니다.

특정 경로에 있는 구현 조정 링크의 인스턴스가 하나만 시스템에 설치된 경우 해당 인스턴스가 자동으로 선택됩니다. 나중에 경로에서 다른 링크가 설치되면 공급업체, 사이트 또는 로컬 대체가 적용되지 않거나 링크 중 하나가 조정된 버전인 경우 링크가 전환되지 않습니다.

`mediator-priority`
조정된 링크의 충돌을 해결할 때 일반적으로 pkg(5)는 `mediator-version`의 값이 가장 큰 링크를 선택하거나 `mediator-implementation`(불가능한 경우)을 기반으로 링크를 선택합니다. 이 속성은 일반적인 충돌 해결 프로세스에 대한 대체를 지정할 때 사용됩니다.

이 속성을 지정하지 않을 경우 기본 조정자 선택 논리가 적용됩니다.

값이 `vendor`인 경우 `mediator-priority`가 지정되지 않은 링크에 비해 해당 링크가 우선합니다.

값이 `site`인 경우 값이 `vendor`이거나 `mediator-priority`가 지정되지 않은 링크에 비해 해당 링크가 우선합니다.

로컬 시스템 관리자는 위에 설명된 선택 논리를 대체할 수 있습니다.

Hardlink 작업

`hardlink` 작업은 하드 링크를 나타냅니다. 이 작업은 `link` 작업과 동일한 속성을 가지며 `path`가 키 속성입니다.

Driver 작업

`driver` 작업은 장치 드라이버를 나타냅니다. `driver` 작업은 페이로드를 참조하지 않습니다. 드라이버 파일 자체는 `file` 작업으로 설치되어야 합니다. 인식할 수 있는 속성은 다음과 같습니다. 자세한 내용은 `add_drv`(1M)를 참조하십시오.

`name` 드라이버의 이름입니다. 일부 경우를 제외하고 일반적으로 드라이버 이진의 파일 이름입니다. 이 속성은 `driver` 작업의 키 속성입니다.

`alias` 이 속성은 드라이버의 별칭을 나타냅니다. 지정된 드라이버의 `alias` 속성은 두 개 이상일 수 있습니다. 특별한 할당 규칙이 필요하지 않습니다.

`class` 이 속성은 드라이버 클래스를 나타냅니다. 지정된 드라이버의 `class` 속성은 두 개 이상일 수 있습니다.

`perms` 이 속성은 드라이버의 장치 노드용 파일 시스템 권한을 나타냅니다.

`clone_perms` 이 속성은 해당 드라이버에 대한 복제 드라이버의 미러 노드용 파일 시스템 권한을 나타냅니다.

`policy` 이 속성은 장치에 대한 추가 보안 정책을 지정합니다. 지정된 드라이버의 `policy` 속성은 두 개 이상일 수 있지만 미러 장치 사양은 두 개 이상의 속성에 존재할 수 없습니다.

`privs` 이 속성은 드라이버에 사용되는 권한을 지정합니다. 지정된 드라이버의 `privs` 속성은 두 개 이상일 수 있습니다.

`devlink` 이 속성은 `/etc/devlink.tab`의 항목을 지정합니다. 값은 `\t`로 표시되는 탭을 사용하여 파일에 삽입될 정확한 행입니다. 자세한 내용은 `devlinks`(1M)를 참조하십시오. 지정된 드라이버의 `devlink` 속성은 두 개 이상일 수 있습니다.

Depend 작업

depend 작업은 패키지 간 종속성을 나타냅니다. 패키지는 다른 패키지에 종속될 수 있습니다. 첫번째 패키지의 기능이 작동 또는 설치되도록 하려면 첫번째 패키지에 두번째 패키지의 기능이 필요하기 때문입니다. 종속성은 선택적일 수 있습니다. 설치 시 종속성이 충족되지 않을 경우 패키징 시스템은 다른 제약 조건에 따라 종속 패키지를 설치하거나 최신 버전으로 업데이트하려고 시도합니다.

인식할 수 있는 속성은 다음과 같습니다.

fmri
: 종속 패키지를 나타내는 FMRI입니다. 이 속성은 dependency 작업의 키 속성입니다. fmri 값에는 게시자가 포함되지 않아야 합니다. 패키지 이름은 완전한 것으로 간주됩니다. require-any 유형의 종속성에는 fmri 속성이 여러 개 있을 수 있습니다. fmri 값에서 버전은 선택적입니다. 단, 일부 유형의 종속성의 경우 버전이 없는 fmri는 의미가 없습니다.

type
: 종속성의 유형입니다.

  값이 require인 경우 종속성이 필요하며 종속성의 버전은 fmri 속성에 지정된 버전과 같거나 이후 버전이어야 합니다. 버전을 지정하지 않을 경우 모든 버전이 종속성을 충족합니다. 필요한 종속성을 충족할 수 없을 경우 패키지를 설치할 수 없습니다.

  값이 optional인 경우 종속성(있을 경우)은 지정된 버전 레벨보다 크거나 같아야 합니다.

  값이 exclude인 경우 지정된 버전 레벨보다 크거나 같은 레벨에 종속성이 있으면 포함하는 패키지를 설치할 수 없습니다. 버전을 지정하지 않을 경우 종속성을 지정하는 패키지와 동시에 종속 패키지를 설치할 수 없습니다.

  값이 incorporate인 경우 종속성은 선택적이지만 종속 패키지의 버전에 대해서는 제약 조건이 적용됩니다. 아래의 "제약 조건 및 고정"을 참조하십시오.

  값이 require-any인 경우 require 종속성 유형과 동일한 규칙에 따라 여러 fmri 속성에 의해 지정된 여러 종속 패키지 중 하나가 종속성을 충족할 수 있습니다.

  값이 conditional인 경우 predicate 속성에 의해 정의된 패키지가 시스템에 있는 경우에만 종속성이 필요합니다.

  값이 origin인 경우 종속성(있을 경우)은 설치 전에 수정할 이미지의 지정된 값보다 크거나 같아야 합니다. root-image 속성의 값이 true인 경우 이 패키지를 설치하려면 /에 루트 지정된 이미지에 종속성이 있어야 합니다.

값이 group인 경우 패키지가 이미지 무시 목록에 있지 않는 한 종속성이 필요합니다. 오래된 패키지는 자동으로 그룹 종속성을 충족합니다. pkg(1)의 avoid 하위 명령을 참조하십시오.

값이 parent인 경우 이미지가 하위 이미지가 아니면 종속성이 무시됩니다. 이미지가 하위 이미지이면 종속성이 있어야만 상위 이미지에 존재할 수 있습니다. parent 종속성에 대해 일치하는 패키지 버전은 incorporate 종속성에 사용된 것과 동일합니다.

`predicate` conditional 종속성에 대한 술어를 나타내는 FMRI입니다.

`root-image` 위에 언급된 origin 종속성에 대해서만 적용됩니다.

License **작업**

license 작업은 패키지 내용에 연결된 라이센스 또는 기타 정보를 나타냅니다. 패키지는 license 작업을 사용하여 패키지 설치 프로그램에 라이센스, 보증 부인 또는 기타 지침을 전달할 수 있습니다.

license 작업의 페이로드는 패키지와 관련된 이미지 메타 데이터 디렉토리로 전달되며 사용자가 읽을 수 있는 텍스트 데이터만 포함해야 합니다. HTML 또는 다른 형식의 마크업은 포함하지 않아야 합니다. 속성을 통해 license 작업은 관련 페이로드가 표시되어야 하거나 동의가 필요함을 클라이언트에 알릴 수 있습니다. 표시 및/또는 동의 방법은 클라이언트에 따라 다릅니다.

인식할 수 있는 속성은 다음과 같습니다.

`license` 이 속성은 license 작업의 키 속성입니다. 이 속성은 사용자가 라이센스 텍스트 자체를 읽지 않고도 내용을 확인할 수 있도록 라이센스에 대한 의미 있는 설명을 제공합니다. 몇 가지 값 예는 다음과 같습니다.

- ABC Co. Copyright Notice
- ABC Co. Custom License
- Common Development and Distribution License 1.0 (CDDL)
- GNU General Public License 2.0 (GPL)
- GNU General Public License 2.0 (GPL) Only
- MIT License
- Mozilla Public License 1.1 (MPL)
- Simplified BSD License

license 값은 패키지 내에서 고유해야 합니다. 위의 예에 나타난 바와 같이, 설명에 라이센스 버전을 포함하는 것이 권장됩니다. 한 패키지에 여러 라이센스 하의 코드가 있는 경우 여러 license 작업을 사용합니다. 라이센스 속성 값의 길이는 64자를 초과할 수 없습니다.

`must-accept` `true`인 경우 사용자가 이 라이센스에 동의해야만 관련 패키지를 설치 또는 업데이트할 수 있습니다. 이 속성을 생략하는 것은 `false`로 설정하는 것과 동일합니다. 동의 방법(예: 대화식 또는 구성 기반)은 클라이언트에 따라 다릅니다.

`must-display` `true`인 경우 패키징 작업 중 클라이언트가 작업의 페이로드를 표시해야 합니다. 이 값을 생략하는 것은 `false`로 설정하는 것과 동일합니다. 이 속성은 저작권 경고에 사용되지 않고 작업 중 표시되어야 하는 실제 라이센스 또는 기타 자료에만 사용되어야 합니다. 표시 방법은 클라이언트에 따라 다릅니다.

Legacy 작업 `legacy` 작업은 레거시 패키징 시스템에 사용되는 패키지 데이터를 나타냅니다. 이 작업에 연결된 속성을 레거시 시스템의 데이터베이스에 추가하면 해당 데이터베이스를 질의하는 도구는 레거시 패키지가 실제로 설치되어 있는 것처럼 작동할 수 있습니다. 특히 이 속성은 패키지를 사용하여 종속성을 충족시킬 수 있도록 `pkg` 속성에 따라 명명된 패키지가 시스템에 설치되어 있음을 레거시 시스템에 충분히 인식시켜야 합니다.

`pkginfo`(4)의 매개변수에 따라 명명된 인식할 수 있는 속성은 다음과 같습니다.

`category` `CATEGORY` 매개변수에 대한 값입니다. 기본값은 `system`입니다.

`desc` `DESC` 매개변수에 대한 값입니다.

`hotline` `HOTLINE` 매개변수에 대한 값입니다.

`name` `NAME` 매개변수에 대한 값입니다. 기본값은 `none provided`입니다.

`pkg` 설치하려는 패키지의 약어입니다. 기본값은 패키지 FMRI의 이름입니다. 이 속성은 `legacy` 작업의 키 속성입니다.

`vendor` `VENDOR` 매개변수에 대한 값입니다.

`version` VERSION 매개변수에 대한 값입니다. 기본값은 패키지 FMRI의 버전입니다.

Set 작업 `set` 작업은 패키지 레벨 속성 또는 메타 데이터(예: 패키지 설명)를 나타냅니다.

인식할 수 있는 속성은 다음과 같습니다.

`name` 속성의 이름입니다.

`value` 속성에 지정된 값입니다.

`set` 작업은 패키지 작성인이 선택한 메타 데이터를 전달할 수 있습니다. 단, 패키징 시스템에 대해 특정 의미를 갖는 잘 정의된 속성 이름이 여러 개 있습니다.

`pkg.fmri` "설명" 절의 "패키지 FMRI 및 버전"을 참조하십시오.

`info.classification` `pkg`(5) 클라이언트가 패키지 분류에 사용할 수 있는 하나 이상의 토큰입니다. 값에는 체계(예:

"org.opensolaris.category.2008" 또는 "org.acm.class.1998")와 실제 분류(예: "Applications/Games")가 콜론(:)으로 구분되어 포함되어야 합니다.

`pkg.description` 패키지 내용 및 기능에 대한 자세한 설명(일반적으로 단락 또는 특정 길이)입니다.

`pkg.obsolete` `true`인 경우 패키지가 오래된 것으로 표시됩니다. 오래된 패키지는 set 작업 이외의 작업을 포함할 수 없으며 이름이 바뀐 것으로 표시되지 않아야 합니다.

`pkg.renamed` `true`인 경우 패키지의 이름이 바뀐 것입니다. 패키지에는 이 패키지의 이름이 바뀐 패키지 버전을 가리키는 depend 작업이 하나 이상 있어야 합니다. 패키지는 이름이 바뀐 패키지와 오래된 패키지로 동시에 표시될 수 없지만 그렇지 않은 경우 여러 개의 set 작업을 포함할 수 있습니다.

`pkg.summary` 패키지에 대한 한 행의 짧은 설명입니다.

Group 작업 `group` 작업은 `group`(4)에 정의된 대로 UNIX 그룹을 정의합니다. 그룹 암호는 지원되지 않습니다. 초기에 이 작업으로 정의된 그룹에는 사용자 목록이 없습니다. 사용자는 `user` 작업으로 추가할 수 있습니다. 인식할 수 있는 속성은 다음과 같습니다.

`groupname` 그룹 이름에 대한 값입니다.

`gid` 그룹의 고유한 숫자 ID입니다. 기본값은 100 미만의 첫번째 사용 가능한 그룹입니다.

User 작업 `user` 작업은 `/etc/passwd`, `/etc/shadow`, `/etc/group` 및 `/etc/ftpd/ftpusers` 파일에 정의된 대로 UNIX 사용자를 정의합니다. 이 속성으로 정의된 사용자는 해당 파일에 추가된 항목을 가집니다.

인식할 수 있는 속성은 다음과 같습니다.

`username` 사용자의 고유한 이름입니다.

`password` 사용자의 암호화된 암호입니다. 기본값은 `*LK*`입니다. `shadow`(4)를 참조하십시오.

`uid` 사용자의 고유한 UID입니다. 기본값은 100 미만의 첫번째 사용 가능한 값입니다.

`group` 사용자 기본 그룹의 이름입니다. `/etc/group`에 있어야 합니다.

`gcos-field` `/etc/passwd`에 있는 `gcos` 필드의 값입니다. 기본값은 `username`입니다.

`home-dir` 사용자의 홈 디렉토리입니다. 기본값은 `/`입니다.

`login-shell` 사용자의 기본 셸입니다. 기본값은 비어 있습니다.

`group-list` 사용자가 속한 두번째 그룹입니다. `group`(4)을 참조하십시오.

`ftpuser` `true` 또는 `false`로 설정할 수 있습니다. 기본값인 `true`를 사용하면 사용자가 FTP를 통해 로그인할 수 있습니다. `ftpusers`(4)를 참조하십시오.

`lastchg` 1970년 1월 1일에서 암호가 마지막으로 수정된 날짜 사이의 기간(일)입니다. 기본값은 비어 있습니다. `shadow`(4)를 참조하십시오.

`min` 암호 변경 간격에 필요한 최소 기간(일)입니다. 암호 에이징을 사용으로 설정하려면 이 필드를 0 이상으로 설정해야 합니다. 기본값은 비어 있습니다. `shadow`(4)를 참조하십시오.

`max` 최대 암호 유효 기간(일)입니다. 기본값은 비어 있습니다. `shadow`(4)를 참조하십시오.

`warn` 사용자에게 경고되는 암호 만료까지 남은 기간(일)입니다. `shadow`(4)를 참조하십시오.

`inactive` 해당 사용자에게 허용되는 비활성 기간(일)입니다. 이 속성은 시스템별로 카운트됩니다. 마지막 로그인에 대한 정보는 시스템의 `lastlog` 파일에서 제공됩니다. `shadow`(4)를 참조하십시오.

`expire` UNIX Epoch(1970년 1월 1일) 이후 기간(일)으로 표시되는 절대 날짜입니다. 이 수에 도달하면 더 이상 로그인을 사용할 수 없습니다. 예를 들어, 만료 값이 13514이면 2007년 1월 1일에 로그인이 만료됩니다. `shadow`(4)를 참조하십시오.

`flag` 비어 있음으로 설정합니다. `shadow`(4)를 참조하십시오.

액추에이터 일부 컨텍스트에서는 특정 작업 소개를 준비하는 동안 또는 특정 작업 소개를 완료한 후 추가 작업을 실행하는 것이 적합할 수 있습니다. 일반적으로 이러한 추가 작업은 라이브 시스템 이미지에만 필요하며 운영 체제에 따라 다릅니다. 패키지 설치 또는 제거와 관련된 여러 작업의 액추에이터가 동일한 경우 존재하는 액추에이터에 해당하는 작업이 설치 또는 제거 시 한 번만 실행됩니다.

액추에이터가 안전한 설치를 진행할 방법을 확인할 수 없을 경우 잘못 지정된 액추에이터로 인해 패키지 설치가 실패할 수 있습니다.

정의된 액추에이터는 다음과 같습니다.

`reboot-needed`
`true` 또는 `false`로 설정할 수 있습니다. 패키지 설치 중 이 액추에이터가 `true`로 설정된 작업이 설치되거나 업데이트되는 경우 재부트가 필요한 것으로 패키징 트랜잭션을 보급할 수 있습니다. 이미지가 라이브 시스템 이미지인 경우 특정 클라이언트 구현에는 추가 단계(예: 이미지 복제를 사용하여 전체 패키지 작업 수행)가 사용될 수도 있습니다.

`disable_fmri`, `refresh_fmri`, `restart_fmri`, `suspend_fmri`
해당 액추에이터 각각에는 패키지 설치 또는 제거 시 작동할 서비스 인스턴스의 FMRI에 대한 값이 사용됩니다. `disable_fmri`를 사용할 경우 `svcadm`(1M)의 `disable`

하위 명령에 따라 작업 제거에 앞서 지정된 FMRI가 사용 안함으로 설정됩니다. `refresh_fmri` 및 `restart_fmri`를 사용할 경우 svcadm(1M)의 개별 하위 명령에 따라 작업 설치, 업데이트 또는 제거 후 지정된 FMRI가 새로 고쳐지거나 다시 시작됩니다. 마지막으로 `suspend_fmri`를 사용할 경우 작업 설치 단계에 앞서 지정된 FMRI가 일시적으로 사용 안함으로 설정되었다가 해당 단계가 완료된 후 다시 사용으로 설정됩니다.

값에는 여러 서비스 인스턴스와 일치하는 패턴이 포함될 수 있습니다. 단, 값은 인스턴스를 지정하지 않은 상태로 암시적으로 처리하는 대신 svcs(1)가 승인한 glob를 명시적으로 처리해야 합니다.

제약 조건 및 고정 패키지가 새 버전으로 전환되거나 시스템에 추가되거나 시스템에서 제거되는 경우 선택되는 버전 또는 제거 허용 여부는 패키지에 대해 적용된 다양한 제약 조건에 따라 달라집니다. 해당 제약 조건은 다른 패키지가 종속성 형식으로 정의할 수도 있고 관리자가 고정 형식으로 정의할 수도 있습니다.

제약 조건의 가장 일반적인 형식은 위의 "Depend 작업"에 설명된 `require` 종속성에 의해 전달됩니다. 해당 제약 조건은 패키지가 다운그레이드되거나 제거되지 않도록 합니다.

운영 체제의 대부분의 요소는 **통합**이라는 패키지에 의해 캡슐화됩니다. 이러한 패키지는 주로 `incorporate` 종속성이 나타내는 제약 조건을 전달합니다.

앞서 설명된 대로 시스템에 통합된 패키지가 존재해야 하는 것은 아니지만 존재할 경우 이 패키지는 포함 최소 버전과 제외 최대 버전을 모두 지정합니다. 예를 들어, 종속 FMRI의 버전이 1.4.3이면 1.4.3보다 낮은 버전은 종속성을 충족하지 않으며 1.4.4 이상 버전도 종속성을 충족하지 않습니다. 단, 단순히 점으로 구분된 시퀀스를 확장한 버전(예: 1.4.3.7)은 허용됩니다.

통합 패키지는 강제로 시스템 요소를 동기식으로 업그레이드하는 데 사용됩니다. 이 패키지가 기본 요구 사항인 구성 요소(예: C 라이브러리 및 커널)도 있고, 특정 버전의 통합에서 참조할 수 있는 확인되고 테스트된 일련의 패키지 버전을 제공하는 용도로만 동기식 업그레이드가 사용되는 구성 요소(예: 종속성이 없는 간단한 Userland 구성 요소)도 있습니다.

통합은 패키지일 뿐이므로 제거할 수 있으며 전달된 모든 제약 조건의 수준도 낮출 수 있습니다. 하지만 제약 조건의 수준을 낮추는 것은 안전하지 않으므로 통합된 패키지에는 Oracle Solaris가 전달한 여러 통합이 필요합니다.

설치된 통합에서 허용하지 않는 버전으로 패키지를 업그레이드하려고 시도하면 요청을 충족시키기 위해 통합의 최신 버전을 찾지 않고 실패합니다. 제약 조건 자체를 이동해야 하는 경우 제약 조건을 지정하는 통합을 제거할 수 없으면 제약 조건의 적합한 버전을 지정하는 버전으로 통합을 업그레이드해야 합니다. 통합을 업그레이드하면 새 버전이 전달하는 제약 조건을 충족하지 않는 모든 통합된 패키지도 업그레이드됩니다.

시스템 관리자는 `pkg freeze` 명령을 사용하여 패키지를 제약할 수 있습니다. 버전을 제공하지 않을 경우 명명된 패키지는 시스템에 설치된 버전으로 제약됩니다. 버전이

지정된 패키지를 제공할 경우 이 관리 제약 조건 또는 고정은 `fmri` 속성이 제공된 패키지 버전의 값을 가진 위치에 incorporate 종속성이 설치된 것처럼 작동합니다.

고정은 패키징 시스템에 의해 자동으로 시작되지 않습니다. 제약 조건을 해제하려면 `pkg unfreeze` 명령을 사용합니다.

게시자 및 저장소 앞서 설명된 대로 게시자는 패키지 클라이언트가 패키지 공급자를 식별하는 데 사용하는 이름입니다. 게시자는 패키지 저장소 및/또는 패키지 아카이브를 사용하여 패키지를 배포할 수 있습니다. 현재 패키지 시스템이 지원하는 두 가지 유형의 저장소는 원본 저장소와 미러 저장소입니다.

원본은 모든 메타 데이터(예: 카탈로그, 매니페스트 및 검색 색인)와 하나 이상의 패키지에 대한 내용(파일)을 포함하는 패키지 저장소입니다. 이미지의 지정된 게시자에 대해 원본이 여러 개 구성된 경우 패키지 클라이언트 API는 패키지 데이터를 검색할 수 있는 최상의 원본을 선택하려고 시도합니다. 원본 저장소가 가장 일반적인 유형의 저장소이며 `pkgsend` 또는 `pkgrecv`가 패키지 저장소에서 사용될 때마다 암시적으로 만들어집니다.

미러는 패키지 내용(파일)만 포함하는 패키지 저장소입니다. 이미지의 지정된 게시자에 대해 미러가 하나 이상 구성된 경우 클라이언트 API는 먼저 패키지 내용 검색용 미러를 선택하고 패키지 내용을 검색할 수 있는 최상의 미러를 선택하려고 시도합니다. 미러에 연결할 수 없거나 필요한 내용이 없거나 미러가 너무 느린 경우 클라이언트 API는 구성된 원본 저장소에서 내용을 검색합니다. 미러는 `pkg.depotd`(1M)의 동적 미러 기능을 통해 신뢰할 수 있는 일련의 클라이언트와 내용을 공유하는 데 사용됩니다. 미러는 패키지 메타 데이터에 대한 액세스를 인증하는 데도 사용되지만 인증 없이 패키지 내용을 배포합니다. 예를 들어, 클라이언트는 액세스를 위해 SSL 키와 인증서 쌍이 필요한 `https` 원본으로 구성되었을 수도 있고, 패키지 내용을 제공하는 `http` 미러로 구성되었을 수도 있습니다. 따라서 패키지 내용 검색을 위한 인증 오버헤드를 피하면서 권한이 있는 클라이언트만 패키지를 설치 또는 업데이트할 수 있습니다. 이름이 `file`인 디렉토리와 해당 상위를 제외하고 저장소의 모든 하위 디렉토리를 제거하여 미러를 만들 수 있습니다. `pkg.depotd`(1M)의 미러 모드를 사용하여 원본 저장소를 미러로 규정할 수도 있습니다.

페이싯 및 변형 소프트웨어에는 선택적 구성 요소와 상호 배타적인 구성 요소가 포함될 수 있습니다. 선택적 구성 요소의 예로는 로켈과 설명서가 있습니다. 상호 배타적 구성 요소의 예로는 SPARC 또는 x86 및 디버그 또는 비디버그 이진이 있습니다.

IPS에서는 선택적 구성 요소를 **페이싯**이라 하고 상호 배타적 구성 요소를 **변형**이라고 합니다. 페이싯과 변형은 패키지 작업에서 태그로 지정됩니다. 각 페이싯 및 변형 태그에는 이름과 값이 있습니다. 단일 작업에 페이싯 및 변형 태그를 여러 개 지정할 수 있습니다. 페이싯 및 변형 태그를 여러 개 가진 구성 요소의 예로는 개발자가 사용하는 아키텍처 관련 헤더 파일이나 SPARC 전역 영역 전용 구성 요소가 있습니다.

변형 태그의 예는 `variant.arch=sparc`입니다. 페이싯 태그의 예는 `facet.devel=true`입니다. 페이싯과 변형은 종종 앞에 `facet.` 및 `variant.` 없이 지칭되기도 합니다.

페이싯과 변형은 특수한 이미지 등록 정보로 개별 패키지에 설정할 수 없습니다. 이미지에 설정된 페이싯과 변형의 현재 값을 보려면 pkg(1) 매뉴얼 페이지에 표시된 대로 pkg facet 및 pkg variant 명령을 사용합니다. 이미지에 설정된 페이싯과 변형의 값을 수정하려면 pkg change-facet 및 pkg change-variant 명령을 사용합니다.

페이싯은 부울이므로 true(사용) 또는 false(사용 안함)로만 설정할 수 있습니다. 기본적으로 모든 페이싯은 이미지에 true로 설정된 것으로 간주됩니다. 작업의 페이싯 태그는 true 값만 가져야 합니다. 다른 값은 정의되지 않은 동작을 일으킵니다. 이미지에 설정된 페이싯은 doc.man과 같은 전체 페이싯이거나 locale.*와 같은 패턴일 수 있습니다. 이는 페이싯 이름 공간의 일부분을 사용 안함으로 설정하고 그 안의 개별 페이싯만 사용으로 설정하려는 경우 유용합니다. 예를 들어, 다음 예제에 표시된 대로 모든 로켈을 사용 안함으로 설정한 후 한두 개의 특정 로켈만 사용으로 설정할 수 있습니다.

```
# pkg change-facet locale.*=false
[output about packages being updated]
# pkg change-facet locale.en_US=true
[output about packages being updated]
```

대부분의 변형은 원하는 수의 값을 가질 수 있습니다. 예를 들어, arch 변형을 i386, sparc, ppc, arm 또는 배포판이 지원하는 어떤 구조로든 설정할 수 있습니다. (i386 및 sparc만 Oracle Solaris에 사용됩니다.) 한가지 예외는 debug 변형입니다. debug 변형은 true 또는 false로만 설정할 수 있습니다. 다른 값은 정의되지 않은 동작을 일으킵니다. 파일 작업에 비디버그 및 디버그 버전이 모두 있는 경우, 다음 예제에 표시된 대로 양쪽 버전에 적용 가능한 debug 변형을 명시적으로 설정해야 합니다.

```
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
```

변형을 사용하여 패키지가 설치되도록 하려면 변형 값을 이미지에 설정해야 합니다. arch 및 zone 변형은 이미지를 만들어 초기 컨텐츠를 설치하는 프로그램에 의해 설정됩니다. debug.* 변형은 기본적으로 이미지에서 false입니다.

이미지에 설정된 페이싯과 변형은 특정 작업이 설치되는지 여부에 영향을 줍니다.

- 페이싯 또는 변형 태그가 없는 작업은 항상 설치됩니다.
- 페이싯 태그가 있는 작업은 태그와 일치하는 모든 페이싯 또는 페이싯 패턴이 이미지에 false로 설정되지 않는 한 설치됩니다. 어떤 페이싯이 true로 설정되거나 명시적으로 설정되지 않은 경우(true가 기본값) 작업이 설치됩니다.
- 변형 태그가 있는 작업은 모든 변형 태그의 값이 이미지에 설정된 값과 동일한 경우에만 설치됩니다.

- 페이싯 및 변형 태그가 모두 있는 작업은 페이싯과 변형이 모두 작업 설치를 허용하는 경우 설치됩니다.

고유의 페이싯 및 변형 태그를 만들 수 있습니다. 다음은 Oracle Solaris에서 흔히 사용되는 태그입니다.

변형 이름	가능한 값
variant.arch	sparc, i386
variant.opensolaris.zone	global, nonglobal
variant.debug.*	true, false

다음 목록은 Oracle Solaris에서 사용되는 페이싯 태그의 작은 샘플을 보여줍니다.

```
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
```

이미지 정책

이미지 정책은 부울 값이 있는 이미지 등록 정보에 의해 정의됩니다. `flush-content-cache-on-success` 및 `send-uuid` 등록 정보에 대한 설명과 해당 값을 보고 수정하는 방법은 pkg(1) 매뉴얼 페이지의 "이미지 등록 정보"를 참조하십시오.

파일

보다 큰 파일 시스템 내에서는 pkg(5) 이미지가 임의적으로 배치될 수 있으므로 `$IMAGE_ROOT` 토큰이 상대 경로를 구별하는 데 사용됩니다. 일반적인 시스템 설치에서는 `$IMAGE_ROOT`가 /와 동일합니다.

`$IMAGE_ROOT/var/pkg`	전체 또는 부분 이미지에 대한 메타 데이터 디렉토리입니다.
`$IMAGE_ROOT/.org.opensolaris,pkg`	사용자 이미지에 대한 메타 데이터 디렉토리입니다.

특정 이미지의 메타 데이터 내에서는 특정 파일 및 디렉토리에 손상 복구 및 복구 중 유용한 정보가 포함될 수 있습니다. `$IMAGE_META` 토큰은 메타 데이터가 포함된 최상위 레벨 디렉토리를 가리키는 데 사용됩니다. 일반적으로 `$IMAGE_META`는 위에 지정된 두 개 경로 중 하나입니다.

`$IMAGE_META/lost+found`	패키지 작업 중 이동된 충돌하는 디렉토리 및 파일의 위치입니다.
`$IMAGE_META/publisher`	각 게시자에 대한 디렉토리를 포함합니다. 각 디렉토리는 게시자 특정 메타 데이터를 저장합니다.

`$IMAGE_META` 디렉토리 계층의 기타 경로는 개인용이므로 변경할 수 있습니다.

속성

다음 속성에 대한 설명은 attributes(5)를 참조하십시오.

속성 유형	속성 값
Availability	package/pkg
Interface Stability	커밋되지 않음

참조

pkg(1), pkgsend(1), pkg.depotd(1m), pkg.sysrepo(1m), svcs(1), svcadm(1M)

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