NAME

pkg repo - create a package repository catalogue

SYNOPSIS

pkg repo [**-lq**] [**-m** *meta-file*] [**-o** *output-dir*]
<*repo-path*> [rsa:<*rsa-key*> | signing_command: <*the command*>]

pkg repo [**--{list-files,quiet}**] [**--meta-file** *meta-file*] [**--output-dir** *output-dir*]
<*repo-path*> [rsa:<*rsa-key*> | signing_command: <*the command*>]

DESCRIPTION

pkg repo is used to create a catalogue of the available packages in a repository. **pkg repo** catalogues are necessary for sharing your package repository, and are intrinsic to the operation of **pkg install** or **pkg upgrade**.

The repository files created by **pkg repo** consist of a number of compressed tar archives stored typically at the top level of the repository filesystem. Of these, *meta.txz* must exist at the apex of the repository filesystem. This is a well-known name that is hard-wired into pkg(8).

meta.txz contains at least one file: *meta* which contains a key to the location and format of the other files comprising the catalogue information. Other files may have arbitrary names as defined in *meta*, but conventionally the following names are used.

digests.txz contains *digests* which lists the cryptographic checksums for each of the packages in the repository. This is downloaded when **SIGNATURE_TYPE** is set to *FINGERPRINTS* in the repository configuration.

filesite.txz contains *filesite.yaml* which is a database of all of the files present in all of the packages in the repository, containing filenames, file sizes and checksums as described in pkg-repository(5). Generating *filesite.txz* involves significant additional system resources and is not usually done.

packagesite.txz similarly contains at least one file *packagesite.yaml*, which lists selected metadata for each of the packages in the repository as described in pkg-repository(5). This is the key file containing the working data used by pkg(8) and includes the run-time dependencies for each package, plus shared library dependencies and similar data that are used by pkg(8) to solve package dependency problems.

In addition to the files already mentioned, the *.txz* archives may also contain cryptographic signatures. These will be produced when the internal signature mechanism of **pkg repo** is enabled.

Repository users download these files to their local machines, where they are processed into per-

repository sqlite databases for fast lookup of available packages by programs such as pkg-install(8).

To create a package repository catalogue, specify the top-level directory beneath which all the packages are stored as *repo-path*. **pkg repo** will search the filesystem beneath *repo-path* to find all the packages it contains. Directories starting with '.' or named *Latest* are not traversed.

The repository files will be created in the top-level repository directory unless relocated by specifying **-o** *output-dir* or **--output-dir** *output-dir*.

Optionally, the repository catalogue may be cryptographically signed. This is enabled either by specifying the path to an RSA private key as the *rsa-key* argument or by using an external command.

If *rsa-key* is used, the SHA256 of the repository is signed using the provided key. The signature is added into the repository catalogue. The client side should use **SIGNATURE_TYPE** set to PUBKEY and **PUBKEY** set to a local path of the public key in its repository configuration file.

An external command can be useful to create a signing server to keep the private key separate from the repository. The external command is passed the SHA256 of the repository catalogue on its stdin. It should output the following format:

```
SIGNATURE
signature data here
CERT
public key data here
END
```

When using an external command, the client's *pkg.conf* must have **SIGNATURE_TYPE** set to FINGERPRINTS and **FINGERPRINTS** set to a directory having a *trusted/myrepo* containing a fingerprint style representation of the public key:

```
function: sha256
fingerprint: \"sha256_representation_of_the_public_key\"
```

See the *EXAMPLES* section and pkg.conf(5) for more information.

Signing the catalogue is strongly recommended.

OPTIONS

The following options are supported by **pkg repo**:

-l, **--list-files**
Generate list of all files in repo as filesite.txz archive.

-m *meta-file*, **--meta-file** *meta-file*
Use the specified file as repository meta file instead of the default settings.

-o *output-dir*, **--output-dir** *output-dir*
Create the repository in the specified directory instead of the package directory.

-q, **--quiet**
Force quiet output.

FILES

See pkg.conf(5).

ENVIRONMENT

PKG_REPO_HASH When set, rename packages with the short hash of contents appended to the filename.

PKG_REPO_SYMLINK When set, create a symlink between the short hash filename and the regular filename.

SEE ALSO

pkg_create(3), pkg_printf(3), pkg_repos(3), pkg-keywords(5), pkg-lua-script(5), pkg-repository(5), pkg-script(5), pkg-triggers(5), pkg.conf(5), pkg(8), pkg-add(8), pkg-alias(8), pkg-annotate(8), pkg-audit(8), pkg-autoremove(8), pkg-check(8), pkg-clean(8), pkg-config(8), pkg-create(8), pkg-delete(8), pkg-fetch(8), pkg-info(8), pkg-install(8), pkg-lock(8), pkg-query(8), pkg-register(8), pkg-rquery(8), pkg-search(8), pkg-set(8), pkg-shell(8), pkg-shlib(8), pkg-ssh(8), pkg-stats(8), pkg-triggers(8), pkg-update(8), pkg-updating(8), pkg-upgrade(8), pkg-version(8), pkg-which(8)

EXAMPLES

Create an RSA key pair:

```
% openssl genrsa -out repo.key 2048
% chmod 0400 repo.key
% openssl rsa -in repo.key -out repo.pub -pubout
```

Create a repository and sign it with a local RSA key. The public key would be shared on all client servers with **SIGNATURE_TYPE** set to PUBKEY and its path set via **PUBKEY** setting in the repository configuration file:

```
pkg repo /usr/ports/packages repo.key
```

Create a repository and sign it with an external command. The client should set, via the repository configuration file, **SIGNATURE_TYPE** to FINGERPRINTS and **FINGERPRINTS** to a path containing a file with the SHA256 of the public key:

```
# On signing server:
% cat > sign.sh << EOF
#!/bin/sh
read -t 2 sum
[ -z "$sum" ] && exit 1
echo SIGNATURE
echo -n $sum | /usr/bin/openssl dgst -sign repo.key -sha256 -binary
echo
echo CERT
cat repo.pub
echo END
EOF

# On package server:
% pkg repo /usr/ports/packages signing_command: ssh signing-server sign.sh
# Generate fingerprint for sharing with clients
% sh -c ’( echo "function: sha256"; echo "fingerprint: \"$(sha256 -q repo.pub)\""; ) > fingerprint’
# The ’fingerprint’ file should be distributed to all clients.

# On clients with FINGERPRINTS: /usr/local/etc/pkg/fingerprints/myrepo:
$ mkdir -p /usr/local/etc/pkg/fingerprints/myrepo/trusted
# Add ’fingerprint’ into /usr/local/etc/pkg/fingerprints/myrepo/trusted
```