DPM Backing Tree

The `DPM Backing Tree` is the directory structure that stores all the information about the packages installed on the system.

Operations for installing packages are performed directly against this backing tree. Queries for package reporting are not performed directly against it. They are instead performed against the `DPM Caching Database` (`dpmdb`).

This directory structure exists at `/var/lib/dpm/storage` and has two reserved subdirectories:

1. packages
2. staging

The backing tree is used as a structured filesystem representation of data from packages managed by DPM and is the storage backing for the `dpmdb`.

packages

The `packages` subdirectory structure contains a directory for each installed package. The names of those directories are the sha256 checksum extracted from the `PACKAGE_DIGEST file for that package.

Inside each hash-named directory representing a package are the hooks, metadata, and signature archives copied from the package. Please consult the `DPM Package Format` specification for more details on what those are.

Here's an example of the packages tree on a system with only two packages installed:

```
packages
├──
4ffd74ecd2df9fb9748cc8ca91fcc8fc71323380e062b4f3a3f107db0ed192
26
│   ├── hooks
│   │   ├── POST-INSTALL
│   │   ├── POST-INSTALL_ROLLBACK
```

```
│   │   ├── POST-REMOVE
│   │   ├── POST-REMOVE_ROLLBACK
│   │   ├── POST-UPDATE
│   │   ├── POST-UPDATE_ROLLBACK
│   │   ├── PRE-INSTALL
│   │   ├── PRE-INSTALL_ROLLBACK
│   │   ├── PRE-REMOVE
│   │   ├── PRE-REMOVE_ROLLBACK
│   │   ├── PRE-UPDATE
│   │   └── PRE-UPDATE_ROLLBACK
│   ├── metadata
│   │   ├── ARCHITECTURE
│   │   ├── AUTHOR
│   │   ├── CONTENTS_MANIFEST_DIGEST
│   │   ├── DEPENDENCIES
│   │   ├── DESCRIPTION
│   │   ├── LICENSE
│   │   ├── NAME
│   │   ├── PACKAGE_DIGEST
│   │   ├── PROVIDES
│   │   ├── REPLACES
│   │   ├── SOURCE
│   │   └── VERSION
│   └── signatures
│       ├── contents.signature
│       ├── hooks.signature
│       └── metadata.signature
└── 
56213bb6cec8932c7adff33a97c2dfe7d3758c32dc46effb48b64b612f1e23
9a
    ├── hooks
    │   ├── POST-INSTALL
    │   ├── POST-INSTALL_ROLLBACK
    │   ├── POST-REMOVE
    │   ├── POST-REMOVE_ROLLBACK
    │   ├── POST-UPDATE
    │   ├── POST-UPDATE_ROLLBACK
    │   ├── PRE-INSTALL
    │   ├── PRE-INSTALL_ROLLBACK
    │   ├── PRE-REMOVE
    │   ├── PRE-REMOVE_ROLLBACK
```

```
│   ├── PRE-UPDATE
│   └── PRE-UPDATE_ROLLBACK
├── metadata
│   ├── ARCHITECTURE
│   ├── AUTHOR
│   ├── CONTENTS_MANIFEST_DIGEST
│   ├── DEPENDENCIES
│   ├── DESCRIPTION
│   ├── LICENSE
│   ├── NAME
│   ├── PACKAGE_DIGEST
│   ├── PROVIDES
│   ├── REPLACES
│   ├── SOURCE
│   └── VERSION
└── signatures
    ├── contents.signature
    ├── hooks.signature
    └── metadata.signature
```

While it would be more human-readable to use the package name as opposed to the `sha256` hash, using the name introduces limitations and potential vulnerabilities that are more simple to address in implementation if the hash is used.

staging

The staging directory is the extraction point for packages when they're being installed, updated or downgraded.

After optional package integrity and signature validations are checked for every file to be installed, the `content` archive contents are installed to the `rootfs_target` location (see: `DPM` Configuration Options) on the host. This process incorporates the `CONTENTS_MANIFEST_DIGEST` parameters set so as not to overwrite non-controlled files, rename on collision in that case, as well as user/group ownership, permissions etc.

From there, the appropriate hook is called to perform any post-operation tasks that the package creator built.

If any of this returns an error, the rollback hook is triggered, the staging area is cleared, and DPM returns an error that would indicate where the problem was — and no change to the backing tree is made.

If the content installation is successful, a new directory in staging whose name is the package digest checksum is created, the contents are removed from the staging area, and `hooks`, `metadata`, and `signatures` are moved to that new directory.

Once this new hash-named directory is populated in the staging area, it is moved as **one unit** to the `packages` directory tree, so that only complete transactions are stored in the DPM Backing Tree.

DPMDB Generation and Regeneration

Generation and Regeneration of the DPMDB

After the DPM Backing Tree is updated with the new entry, DPM will check if the `dpmdb` exists. If it doesn't, it will generate a new dpmdb from the backing tree. If the dpmdb already exists it will create a new entry in the `rpmdb`.

Restoration of the Backing Tree

While the backing tree is intended to be the source of truth on the system, in the event that the Backing Tree becomes corrupted or destroyed, a restore attempt can be performed from the RPMDB to regenerate the backing tree.

Transaction Recording

Whenever an install, update, downgrade, removal, or reinstall is performed, it's timestamped and logged to a data file in order at `/var/lib/dpm/storage/transactions`.

This provides data for a number of capabilities, largely surrounding auditing, but, could also be parsed to replay package management operations when rebuilding servers, so, there are some operational benefits as well.

This file contains these values for the following operations:

1. installing a package
    - operation type
    - TIMESTAMP at BEGINNING of operation
    - TIMESTAMP at END of operation
    - package NAME
    - package VERSION
    - COMPLETION STATUS

2. updating a package
    - operation type
    - TIMESTAMP at BEGINNING of operation
    - TIMESTAMP at END of operation
    - old package NAME being upgraded
    - old package VERSION being upgraded
    - new package NAME being upgraded to
    - new package VERSION being upgraded to
    - COMPLETION STATUS

3. downgrading a package
    - operation type
    - TIMESTAMP at BEGINNING of operation
    - TIMESTAMP at END of operation
    - current package NAME being downgraded
    - current package VERSION being downgraded
    - replacement package NAME being downgraded to
    - replacement package VERSION being downgraded to
    - COMPLETION STATUS

4. removing a package
    - operation type
    - TIMESTAMP at BEGINNING of operation
    - TIMESTAMP at END of operation
    - package NAME
    - package VERSION
    - COMPLETION STATUS

5. reinstall a package
    - operation type
    - TIMESTAMP at BEGINNING of operation
    - TIMESTAMP at END of operation
    - package NAME
    - package VERSION
    - COMPLETION STATUS

An example of this format looks as:

```
S 2025-02-05_20:05:04 2025-02-05_20:05:06 redshift 1.12 COMPLETE
R 2025-02-05_20:08:04 2025-02-05_20:08:12 gnome-keyring 46.2 COMPLETE
D 2025-02-05_20:10:13 2025-02-05_20:10:14 kernel 6.12.11 kernel 6.9.10 COMPLETE
U 2025-02-05_20:13:21 2025-02-05_20:13:28 slack 4.40.0 slack 4.41.105 COMPLETE
I 2025-02-05_21:37:38 2025-02-05_21:37:39 python3-pip 23.3.2 COMPLETE
```

The transactions log is not intended to be human-read but is deliberately kept simple enough to be able to be. It is intended to be consumed when generating RPMDB for operations that rollback transactions, or query transaction history or to be dumped by DPM to display as table output.