DPM Repository Format

The DPM repository consists of metadata and packages. The metadata is used to generate a small file-based database that is downloaded, cached, and used by `DON` for repository operations.

Repository Caching

Ultimately, what DON downloads and caches from the repository to do what it does, is a file at `${repository_root}/metadata.db`.

In order to generate metadata.db, the repo creation tool, `dpm_create-repo` first generates a hierarchal tree structure. This is a sample repository tree used to generate metadata.db:

```
repository/
├── metadata.db
├── metadata/
...
│                                                                   ├──
4ffd74ecd2df9fb9748cc8ca91fcc8fc71323380e062b4f3...19226/
│   │   ├── metadata
│   │   │   ├── ARCHITECTURE
│   │   │   ├── AUTHOR
│   │   │   ├── CONTENTS_MANIFEST_DIGEST
│   │   │   ├── DEPENDENCIES
│   │   │   ├── DESCRIPTION
│   │   │   ├── LICENSE
│   │   │   ├── NAME
│   │   │   ├── PACKAGE_DIGEST
│   │   │   ├── PROVIDES
│   │   │   ├── REPLACES
│   │   │   ├── SOURCE
│   │   │   └── VERSION
│   │   └── signatures
│   │       ├── contents.signature
│   │       ├── hooks.signature
│   │       └── metadata.signature
│                                                                   ├──
```

```
56213bb6cec8932c7adff33a97c2dfe7d3758c32dc46effb...e239a/
│   │   ├── metadata
│   │   │   ├── ARCHITECTURE
│   │   │   ├── AUTHOR
│   │   │   ├── CONTENTS_MANIFEST_DIGEST
│   │   │   ├── DEPENDENCIES
│   │   │   ├── DESCRIPTION
│   │   │   ├── LICENSE
│   │   │   ├── NAME
│   │   │   ├── PACKAGE_DIGEST
│   │   │   ├── PROVIDES
│   │   │   ├── REPLACES
│   │   │   ├── SOURCE
│   │   │   └── VERSION
│   │   └── signatures
│   │       ├── contents.signature
│   │       ├── hooks.signature
│   │       └── metadata.signature
│   ├── REPO_MANIFEST
│   └── pubkeys
│       └── DHL2.pubkey
└── packages
    ├── aarch64
    │   ├── myapp-0.1.2-0.dhl2.aarch64.dpm
    │   └── myapp-selinux-0.1.2-0.dhl2.aarch64.dpm
    ├── noarch
    │   ├── myapp-0.1.2-0.dhl2.noarch.dpm
    │   └── myapp-selinux-0.1.2-0.dhl2.noarch.dpm
    └── x86_64
        ├── myapp-0.1.2-0.dhl2.x86_64.dpm
        └── myapp-selinux-0.1.2-0.dhl2.x86_64.dpm
```

It can be simplified as:

```
repository/
├── metadata/
│   ├── [package_digest_hash]/
│   │   ├── metadata/
│   │   │   └── [metadata files]
│   │   └── signatures/
│   │       └── [signature files]
│   ├── REPO_MANIFEST
```

```
│   ├── pubkeys/
│   │   └── [public signing keys]
│   └── metadata.db
└── packages/
    ├── aarch64/
    │   └── [architecture-specific packages]
    ├── noarch/
    │   └── [architecture-independent packages]
    └── x86_64/
        └── [architecture-specific packages]
```

At the top level, you will see a `metadata` directory and a `packages` directory. You will also see a metadata.db and REPO_MANIFEST at the top level.

Metadata Directory

The metadata tree in the repository structure closely resembles the `DPM Backing Tree`, and this is by design to facilitate what `DON` will do with them.

Once the DPM Repo metadata is created, it then populates the small database at `metadata.db` from that data.

The `DON` client downloads the repository metadata db, and caches this repository metadata locally with a configurable `TTL` (Time-to-Live). Once this cache expires, `DON` refreshes this cache from that repository at the time of an interaction (or unless the command to refresh it is run).

This metadata cache is for the purpose of listing package dependencies and file/package lookups so that the data to determine which package provides a file, or what files are provided by a package, is present in addition to what packages are required to be installed for any package in the repository to be installed.

There are a few subtle differences in the repository metadata from the DPM Backing Tree:

1. Hooks are not copied
2. A file called `REPO_MANIFEST` exists, generated by the `dpm_create_repo` tool at the time of the last update of the repository.

The REPO_MANIFEST File

The `REPO_MANIFEST` file is generated when the repository metadata is generated. During repository creation the utility finds all .dpm files recurvisely in the `packages` top-level directory and then calculates the cryptographic checksum to create a two column line delimited table file of the following format:

```
$CHECKSUM $file_path\n
```

This pairing of repository paths to package digest hashes compliments the pairing of package digest hashes to package metadata in the generated metadata tree just discussed.

Any file or directory that does not end in .dpm is skipped during the generation of the REPO_MANIFEST file.

The $file_path is relative to the top-level directory of the repository.

PUBKEYS

This directory contains the public keys and everything needed to validate package cryptographic signatures.

Packages Directory

The packages directory in the DPM repository is very simple. It is an arbitrary directory structure containing at least .dpm files.

If files that do not end in .dpm are included this is fine and these files will not be processed during a run of `dpm_create-repo`.

Repository Creation

After building out the packages directory structure, the user runs `dpm_create-repo`. This command generates the repository metadata tree and populates the `REPO_MANIFEST` inside it, before using that data to create `metadata.db`.

`dpm_create-repo` walks through the packages directory tree, extracting the package metadata, signatures archives to do this before rendering the

downloadable database.

Repository Usage

`DON` downloads cache repository metadata with a configurable TTL. It will then query the databases it has cached from all repositories when performing an operation.