DON Paths

Configuration

`/etc/don/config`

The main configuration file for DON.

Repository Configurations

`/etc/don.repos.d/<repository_name>.repo`

Each repository has a configuration file dedicated to it with various options.

Download Cache

`/var/cache/don/packages/<repo_name>/<package_name>`

Don downloads packages and caches them to reduce pull time and network load from redownloading.

Repository Metadata Cache

`/var/cache/don/repos/<repo_name>/metadata`

Don caches the metadata it downloads from repositories so that frequent searches do not create request burdens on the repository, with the added benefit of improving response time.

Transaction and Logistical Database

`/var/lib/don/dondb`

This stores transactions processed by DON as well as some base logistical information such as which repository which packages were downloaded from.

This information is used, for example, rolling back sets of package upgrades to undo a recent system update.

Lock File

`/var/lock/don.lock`

The lock file ensures that only one transaction is processed at a time.

Logging

`/var/log/don/don.log`
`SYSLOG`

DON optionally logs to either (or both) syslog and/or a main log file.

Package Blacklist Directory

`/etc/don/blacklist.d/`

This directory contains `.conf` files, each of which are a line-delimited list of regular expressions. If a package meets this regular expression, DON will refuse to install it and will skip over it in updates.

Be aware that should a package name your blacklisted package as a dependency in a transaction that DON will fail and refuse to install the package naming your package as a dependency.

Blacklisted package detection occurs at the end of the dependency resolution stage. Here's an example of the workflow:

Suppose you have blacklisted a package named `apples`. You have another package that depends on apples, called `bananas`. You run a `don update`, which will update all packages on the system that have an available upgrade package. `DON` will:

1. Detect that `bananas` has an update available.
2. During dependency resolution, discover that `apples` is a dependency.
3. Detect that `apples` is blacklisted.
4. Skip updating `bananas`, and continue with the other updates.
5. At the end of the update process for all packages getting updated, warn the user that `bananas` was skipped due to its dependency on a blacklisted

package.

Be cautious when blacklisting packages, as it doesn't just use package names, it uses regular expressions, so `'.*'` will prevent you from doing much at all with packages on your system. The expression matches against PROVIDES, REPLACES, and NAME.