Best practices for packaging database applications

Sean Finney

Abstract

This draft describes a set of guidelines and best practices to be implemented by the maintainers of database application packages. Pending a final draft, desire, and acceptance by the developer community at large, this may serve as the foundation for an official policy–or it may simply remain as it is.

Contents

Chapter 1

Database Applications

1.1 Scope

In this document a "database application" is any program that relies on some form of data storage outside the scope of the program's execution. This is primarily intended to encompass applications which rely on a relational database server or their own persistent storage mechanism, though effectively is a much larger set of applications. In the future this scope may have to be narrowed to avoid ambiguity and be more effective as a policy.

1.2 Terminology

For the purposes of this document, there are two types of databases: *persistent* and *cached*.

Persistent databases contain data that can not be entirely reconstituted in the case that the database is removed. Also included are databases that if removed would cause serious denial of service (making a system unstable/unusable) or security concerns. Applications using this category of databases are the primary focus of this document. Examples:

- relational database servers providing storage to other applications.
- web applications using a relational database
- openldap's slapd databases
- rrd files containing accumulated/historical data.

Cached databases are a specific group of databases which upon their removal could be sufficiently regenerated, and could be removed without causing serious denial of service or security concerns. Examples include:

- debconf responses
- locate database
- caching nameserver data
- apt's list of available packages

1.3 Placement of databases

Both persistent and cached databases fall under already defined guidelines in the FHS; persistent data must be placed under `/var/lib/packagename`, and cached data under `/var/cache/packagename`, respectively [1]. The remainder of this document primarily addresses the former.

[1] or an equivalently compliant location

1.4 Installation related issues

The following descriptions are divided into different parts, based on the action being performed. For each process, the acceptable behavior of database application packages is outlined.

1.4.1 Automatic Database Configuration/Prompting

It must always be assumed that the local admin knows more than any automated system. He/She must be given the ability to opt out of any "assistance" on the part of the package maintainer. Packages providing any such automated assistance may do so by default if and only if the opt-out debconf prompt is equal to or greater than priority high. With this in mind, directions for manually installing (and upgrading if relevant) the database must be included in the documentation for the package.

1.4.2 Database Installation

For packages providing automated assistance, database installation/configuration should be considered as part of the package installation process. A failure to install a database should be considered a failure to install the package and should result in an error value returned by the relevant maintainer script. Packages may provide a "try again" option to re-attempt configuration. Any such "try again" features here or elsewhere mentioned in this document must have a default negative response value, otherwise infinite loops could occur for noninteractive installs.

To properly handle package reinstallation and reconfiguration, any automated assistance must allow for a package to be reinstalled at the same version without removing or overwriting existing application data. Package reconfiguration may do so.

1.4.3 Database Upgrading

Occasionally a new upstream version of an application will require modifications to be made to the application's underlying database. If an automated system is to assist in such an upgrade, it should be considered as a part of the package upgrade process; failure to upgrade the database should be considered a failure to upgrade the package. This is the only way to safely guarantee the chance to reattempt the upgrade with respect to the underlying database.

Furthermore, any automated system that makes modifications to a database during upgrade must provide the ability to back-up the database before proceeding. Packages may perform such backups automatically, or prompt the admin via debconf. Failure to back up the database should also be considered a failure in the upgrade process of the whole package. As in the case of installation, automated assistance may provide a "try-again" feature to re-attempt the upgrade, but ultimately in the case of failure should cause a non-zero exit value to be returned to dpkg.

Note: if the database in question supports transactional operations, it is recommended to do so.

1.4.4 Database Removal

A package should consider databases in a spirit similar to configuration files or log files; they are something to which the administrator may have some need even when the software that created it is no longer present.

Packages may provide support for removing underlying databases, but it is highly recommended that the administrator is prompted with a chance to preserve the data before doing so.

1.5 Build-time and run-time tools

While not essential, a set of common tools for packaging and configuring these applications can make the life of the maintainer as well as the administrator much easier.

1.5.1 `dbconfig-common`

`dbconfig-common` is a common framework for packaging database applications. More information can be found in the dbconfig-common section (`https://www.debian.org/doc/manuals/dbconfig-common/`) of the Debian documention for maintainers, or under `/usr/share/doc/dbconfig-common`.

1.6 Related threads/discussions

- RFC: best practice creating database (`http://lists.debian.org/debian-devel/2004/10/msg00340.html`)
- RFC: common database policy/infrastructure (`http://lists.debian.org/debian-devel/2004/10/msg00962.html`)