PİSİ Packages: Version Policy v0.2

Eray Özkural and T. Barış Metin

November 18, 2005

Revision History

- v0.1: Barış Metin wrote the first version preparing the outline, detailed Source Version Section, and started the Section on Release Number.
- v0.2: Eray Özkural wrote a detailed introduction, added explanations of release and build numbers, reorganized a bit.

1 Introduction

This document explains the *version policy* that applies to PİSİ packages. Classically, the issue of distinguishing source and binary distributions unambiguously has not received a rigorous treatment in the context of LINUX distributions. We have identified several shortcomings of the usual practices of extending the original version with suffixes and prefixes, colorfully illustrated in the following common problems.

The problem of future downgrades The distribution chooses to use a previous version of the package in the next release. There is no way to indicate this, so ad-hoc solutions such as version prefixes are used. It is impossible to denote a future dependency that requires at least this distribution source release in this case, either.

The problem of redundant distributions A trivial patch has been applied to the source. While few binary packages have been affected by this change, all binary packages built from the source are redistributed.

The problem of underdetermined rebuilds There have been rapid changes in the system, and although no changes have been made to the package source, a new binary distribution must be prepared.

We have devised a slightly new approach in order to alleviate these problems. Our solution consists of encoding the history of source and binary package developments in separate version strings we call release and build numbers.

Since the source version is usually used by the users and developers to identify software, we retain the notion of a source version in PİSİ as a convenience.

In the following sections, we explain the components of our versioning scheme.

1.1 Source Version

Source version is the version number provided by the upstream maintainer of the source archive used in package. It must always be the same as the upstream version used.

Example: If the upstream archive name is *bash-3.0.tar.gz* the version number of the package is *3.0*

1.1.1 Version Suffixes

There is a pre-defined list of suffixes a package version can take.

- **alpha** Source/Package is in alpha state
- **beta** Source/Package is in beta state
- **pre** Source/Pacgage passed the beta state but stable version is not relased yet.
- **rc** Source/Package is a release-candidate.
- **p** Source/Package is released and some patches are applied after the release. This is the patch level.

The suffix should be written after the special separator character _. And there must allways be a number after a suffix. **Example**: packagename-1.0_beta1

The basic order of the priorities for suffixes is:
p > (no suffix) > rc > pre > beta > alpha.

The scope of a source version string is global in the literal sense. It shall not vary from repository to repository.

The support for these special suffixes as well as usual alphanumeric version string ordering has been implemented in PİSİ.

2 Identifying Package Sources

A PİSİ source has three identity elements written under `SOURCE` tag: name, source version, and source release number. We usually say just version and release number/release instead of source version and source release number, respectively. Name is available in the `<Name>` tag. Version and release are available in the last `<Update>` element of `<History>` tag of a `PSPEC`.

The name of a source package is constant throughout its revision history. The version is the original version, given by its programmers. Release is a positive integer. Name and release is sufficient to uniquely identify a particular PISI source revision. That is, version and release are independent.

2.1 Release Number

Release number is the number of the changes that are made to the package source since the initial version in the distribution source. A change can be a patch applied to the source archive, modification in the actions.py, pspec.xml or any file in the source package directory. This change is indicated in `<Update>` tags manually by the package maintainer.

The initial release of a package is by default `1`. The release number always increments by 1 in each revision in the `History`, even the slightest ones, but it never decrements.

The scope of the release number is a given distribution, regardless of its version, e.g. Pardus.

In the future, PISI will have strict checks for release numbers.

2.2 Dependency Specifications

We allow a package to use both source version and release to identify a particular version or a range of package versions.

3 Identifying Binary packages

A PISI binary package is produced from a PISI source package. It has a name that is constant throughout the history of the source package, and it inherits the source version and release number from the source package. However, a binary package has in addition a binary build number. Shortly, build number or just build. For each of the architecture targets, e.g. particular binaries, it also has an architecture tag.

A binary package is uniquely identified by its name, build number, and architecture regardless of the source version.

4 Build Number

Similarly to source release number, binary build number is the number of changes that are made to a binary package. By change, we mean any bit change. The existence of a change is tested by comparing the cryptographic checksums in files.xml with those of the previous build, and the build number is automatically determined by the PİSİ build system. The build number starts from 1 as in release number, and increments by one with each binary change.

The user never interferes with the build number himself. However, if the user fails to provide the previous build, then a package without a build number is built. A package without a build number is evaluated on the basis of release number, which is guaranteed to exist.

The scope of a build number is a given distribution build environment for a particular architecture, which may vary from repository to repository. Therefore, it is not used in dependency specifications. However, the system does assume that a build of a given package and architecture is unique in a given repository.

5 Package File Names

A PİSİ binary package file name contains all the components relevant to its identification, separated by dashes:

```
<binary name>-<source version>-<source release>-<binary build>.pisi
```

6 Future Work

In the future, it may be necessary to extend the notion of release number and build number to support branches and forks of a distribution. A proposal was to have CVS-like branching, but it was dismissed as unnecessary.