Packaging Guidelines A

This appendix provides a list of criteria to use when building packages.

Many of the good packaging criteria present trade-offs among themselves. It will often be difficult to satisfy all requirements equally. These criteria are presented in order of importance; however, this sequence is meant to serve as a flexible guide depending on the circumstances. Although each of these criteria is important, it is up to you to optimize these requirements to produce a good set of packages.

Optimize for Client-Server Configurations

You should consider the various types of system software configurations (diskfull, diskless, and server) when laying out packages. Good packaging design divides the affected files to optimize installation of each configuration type. For example, the contents of `root` and `usr` should be segmented so that dataless and server configurations can easily be supported.

Package by Functional Boundaries

Packages should be self-contained and distinctly identified with a set of functionality. For example, a package containing UFS should contain all UFS utilities and be limited to only UFS binaries.

Packages should be organized from a customer's point of view into functional units.

A

Package Along Royalty Boundaries

Put code that requires royalty payments due to contractual agreements in a dedicated package or group of packages. Do not to disperse the code into more packages than necessary.

Package by Machine Dependencies

Keep machine dependent binaries in dedicated packages. For example, the kernel code should be in a dedicated package with each implementation architecture corresponding to a distinct package instance. This rule also applies to binaries for different architectures. For example, SPARC binaries would be in one package and binaries for an Intel machine would be in another.

Overlap in Packages

When constructing the packages, ensure that duplicate files are eliminated when possible. Unnecessary duplication of files results in support and version difficulties. If your product has multiple packages, constantly compare the contents of these packages for redundancies.

Sizing Considerations

Size is package-specific and depends on other criteria. For example, the maximum size of `/opt` should be considered. When possible, a good package should not contain only one or two files or contain extremely large numbers of files. There are cases where a smaller or larger package might be appropriate to satisfy other criteria.

Localization Software Packaging Guidelines

Localization specific items should be in their own package. An ideal packaging model would have a product's localizations delivered as one package per locale. Unfortunately, in some cases organizational boundaries may conflict with the functional or product boundaries criterion.

International defaults can also be delivered in a package. This would isolate the files necessary for localization changes and standardize delivery format of localization packages.