(Re)Building a Next-GenSystem Package Manager

IPS, the Image Packaging System

Powered by

illumos

An Overview of IPS and its Concepts

OpenIndiana Project

The Component That Fascinated Me

The OpenSolaris Legacy

- A system born of innovation, designed for enterprise and stability.
  Key concepts: ZFS (CoW Filesystem), DTrace (Observability), Zones (Virtualization), Crossbow (Networking), SMF (Service Management).
  The Package Manager: Unlike anything before it.

Why IPS? The SVR4 Problems

- Stateful: Required complex, fragile scripts for upgrades. Non-Atomic: Failures could leave the system in an unrecoverable state.
  Image-Unaware: Treated the OS as a collection of loose files, not a coherent, managed image.

The Genesis of IPS

From SVR4 to the Modern Image

- Goal: Create a package manager that treats the Operating System like a **bootable image**, not just a set of archives.
  Creation Era: Developed by the OpenSolaris community and Sun Microsystems, deployed fully in Solaris 11 (c. 2011).
  Core Principle: **Repository-Centric Design.** The repository is the single source of truth for all system state and package metadata.

What IPS Solved

- Rollbacks: Integration with ZFS Boot Environments (BEs) makes package changes atomic and reversible.
  Dependency Hell: A sophisticated resolver that works against a precise, immutable record of every package version.
  Metadata: Shift from procedural (shell scripts) to declarative (manifests).

FMRI - The Canonical Identity

Fault Managed Resource Identifier (FMRI)

Component	Example	Role
Scheme	`pkg:`	Always denotes an IPS entity.
Publisher	`openindiana.org`	Source and maintainer of the package.
Name	`/web/server/nginx`	What the package *is* (unique).
Version	`@1.25.3-...`	**Crucial:** Multiple version components. Software + build + branch + timestamp.

Why FMRI?

- Immutability: Each FMRI is precise and globally unique. Rollback: Allows system to be reverted to a specific, known state. Precise Dependencies: Dependencies link to specific FMRIs, eliminating ambiguity.

FMRI is the UUID of package management.

Self-Assembly - The Design Philosophy

The Key Difference: The Image is Declared, Not Assembled during build

Traditional Approach

- Download a large, pre-built binary blob (archive). Unpack the onto the filesystem. Run scripts to configure.

Risk: Conflicts only discovered **during** extraction/script run.
Not much better than running a .exe downloaded from the Internet as root

IPS Self-Assembly Approach

- **Metadata-Centric:** Contents defined in a Manifest.
  Image-Time Resolution: Resolves the **entire** dependency graph **before** touching the system.
  Atomic Operation: The image (OS instance) is updated by pulling only the necessary file operations from the repository.
  Integrated into ZFS: The entire change is a single, atomic operation recorded as a new ZFS Boot Environment (BE).

Result: Guaranteed success or a 100% reversible failure.

Configuration Control: Facets and Variants

Facets (Optionality)

- **Definition:** Boolean properties for optional components.
  Example: `facet.doc.*`
  How it Works: Actions are conditionally included based on image settings.
  Benefit: Creates slim (production) or full (development) images.

Variants (Choice)

- **Definition:** Properties that define mutually exclusive choices.
  Example: **`variant.arch=i386`**
  Benefit: A single package definition can cover multiple architectures/compilation targets (e.g., Zabbix for MySQL or PostgreSQL).

Configuration Control: Mediators and Consolidations

Mediators (Version/Implementation Switching)

- **Definition:** Mechanisms to indicate a preferred version when multiple are installed.
  Example: The image can contain both `gcc-10` and `gcc-12`, with a mediator pointing to the **active** version.
  Benefit: Enables safe, instantaneous switching between different versions of core components **within the same image**.

Consolidations (System Integrity)

- **Definition:** Special, logical meta-packages that group related functional components (e.g., the core OS, a desktop environment, or a database stack).
  Example: consolidation/os/illumos-gate. Installing the consolidation ensures the entire associated subsystem is complete and maintained as one unit.
  Benefit: Guarantees system integrity and completeness across major functional areas, simplifying administrative updates and ensuring consistency by enforcing a single, known good state.

Key Principle

Facets, Variants, and Mediators are set on the **Image** (the system), not the individual **Package**.

The Future - IPS in Rust (pkg6)

(Re)Building for the Next Decade

Why Port to Rust?

- **Performance:** Move from C/Python implementation to a fast, with modern features.
  Safety: Eliminate large classes of bugs (memory management, concurrency) inherent in C. A package manager is system-critical. Compile it to one Static binary.
  Modernization: Clean, maintainable codebase that attracts new FOSS contributors.
  API: Opportunity to design cleaner, more ergonomic APIs for integration.

What Improvements Can We Make?

- **Algorithm:** Better dependency solver implementation (potentially leveraging modern SAT/SMT solvers).
  Repository: Enhanced metadata indexing and repository management performance.
  Integration: Easier integration with non-IPS tools and new OS features.

Goal: Retain the elegant design of IPS while gaining the performance and safety of a modern implementation.