2. Overview of the Build System

Hierarchy

Description

List of the top-level directories:

Directory	Content
archives	Source tarballs downloaded for building components.
components	Hierarchy of directories containing packaging recipes.
doc	Documentation of the build system.
make-rules	Makefile rules used by components.
templates	Templates of component Makefile for different types of build toolchains.
tools	Scripts used by the build system.
transforms	Default rules used for the generation of IPS packages.

Categories

All components were originally laid out in one-level but are being reorganized by categories.

Whenever a component is added or updated, it should be placed under the directory indicates by the Categories layout.

Meta-packages

These components do not deliver any file content but define groups of software or logic to manage installation/removal/deprecation of packages.

They are all located within the directory 'components/meta-packages'.

Component structure

A component is comprised of a set of files providing rules for:

1. fetching source code,
2. configuring, building, and installing software,
3. preparing package content and generating one (or more) package.

The resulting package(s) are then published to a local repository.

Files

Name	Content
Makefile	Recipe for configuring, building, and installing software for the component.
*.license	Document containing all the licenses applicable to the packages.
*.p5m	One or more manifests describing package metadata and content.
history	Rules for package renaming and deprecation.
manifests/	Manifests generated automatically by the sample-manifest target.
patches/	Patches: indexed and name with a .patch extension.
test/	Test results for reproducible builds.

Makefile targets

Target	Function	When to use it?
clobber	Clean up the directory content and remove archives.	
clean	Clean up the directory.	
env-prep	Install dependencies listed in the REQUIRED_PACKAGES variable.	
prep	Fetch the source archives unpack, and apply patches.	
build	Configure and builds the software.	
install	Install to the prototype directory.	
sample-manifest	Generate sample manifest from the content found in the prototype directory.	
pre-publish	Run the publication stage without sending packages to the local repository.	
publish	Run the publication stage and publish to the local repository.	
REQUIRED_PACKAGES	Generate a list of **runtime** dependencies detected: **build** dependencies may need to be added automatically	

Testing

The testing framework can be used to make build reproducible and ensure that no regression is introduced when components are updated.

Support for the 'test' target should be ideally added to any component at creation or at update if it is not the case.

Makefile variables 'COMPONENT_TEST_*' affecting the execution of the test suite are declared in shared-macros.mk: most of these variables need not be changed.

Adding support for tests is covered in 3. Common Tasks: Reproducible builds and test suites

Recommendations

A list of recommendations for maintainers is listed at Best Practices.

IPS manifests

Manifest generation

Package actions

Content

Action	Definition	Usage
file		

dir		
link		
hardlink		
user		
group		
driver		

Metadata

Action	Definition	Usage
set		
depend		
license		
signature		

Transforms

Transforms are mainly used to set default attributes to files like ownership and permissions.

They are defined in oi-userland's transforms directory.

Examples

Set mode for binary executables:

```
<transform file path=usr/lib/$(MACH64)/e.+/utils/.+ -> default mode 0555>
```

Variants and facets

Incorporations

Linting

Package repositories

On-disk repositories

Remote repositories

Working with a pkg(5) server