EAPI Cheat Sheet

Gentoo PMS team*

Version 5.0
20th September 2012

Abstract

An overview of the main EAPI changes in Gentoo, for ebuild authors. For full details, consult the Package Manager Specification found on the project page; this is an incomplete summary only.

Official Gentoo EAPIs are consecutively numbered integers (0, 1, 2, ...). Except where otherwise noted, an EAPI is the same as the previous EAPI. All labels refer to the PMS document itself, built from the same checkout as this overview.

EAPI 0

If there is no EAPI explicitly specified, EAPI 0 is assumed.

EAPI 1

Additions/Changes

IUSE defaults A USE flag can be marked as mandatory (if not disabled explicitly by user configuration) with a + sign in front. See IUSE-DEFAULTS on page 28.

Named slot dependencies Dependencies can explicitly request a specific slot by using the `dev-libs/foo:`*SLOT_name* syntax. See SLOT-DEPS on page 35.

*http://www.gentoo.org/proj/en/qa/pms.xml
[1]http://creativecommons.org/licenses/by-sa/3.0/

EAPI 2 (2008-09-25)

Additions/Changes

`SRC_URI` arrows Allows redirection of upstream file naming scheme. By using `SRC_URI="http://some/url -> foo"` the file is saved as `foo` in DISTDIR. See SRC-URI-ARROWS on page 37.

USE dependencies Dependencies can specify USE flag requirements on their target, removing the need for `built_with_use` checks.

[opt] The flag must be enabled.

[opt=] The flag must be enabled if it is enabled for the package with the dependency, or disabled otherwise.

[!opt=] The flag must be disabled if it is enabled for the package with the dependency, or enabled otherwise.

[opt?] The flag must be enabled if it is enabled for the package with the dependency.

[!opt?] The flag must be disabled if it is disabled for the package with the dependency.

[-opt] The flag must be disabled.

See USE-DEPS on page 34.

Blocker syntax A single exclamation mark as a blocker may be ignored by the package manager as long as the stated package is uninstalled later on. Two exclamation marks are a strong blocker and will always be respected. See BANG-STRENGTH on page 35.

`src_configure`, `src_prepare` Both new phases provide finer granularity in the ebuild's structure. Configure calls should be moved from `src_compile` to `src_configure`. Patching and similar preparation must now be done in `src_prepare`, not `src_unpack`. See SRC-PREPARE on page 40 and SRC-CONFIGURE on page 40.

Default phase functions The default functions for phases `pkg_nofetch`, `src_unpack`, `src_prepare`, `src_configure`, `src_compile` and `src_test` can be called via `default_`*phasename*, so duplicating the standard implementation is no longer necessary for small additions. The short-hand `default` function

calls the current phase's `default_` function automatically, so any small additions you need will not be accompanied by a complete reimplementation of the phase. See DEFAULT-PHASE-FUNCS on page 44 and DEFAULT-FUNC on page 69.

`doman` language support The `doman` installation function recognizes language specific man page extensions and behaves accordingly. This behaviour can be inhibited by the `-i18n` switch with EAPI 4. See DOMAN-LANGS on page 63.

EAPI 3 (2010-01-18)

Additions/Changes

Support for `.xz` Unpack of `.xz` and `.tar.xz` files is possible without any custom `src_unpack` functions. See UNPACK-EXTENSIONS on page 68.

Offset prefix Supporting installation on Prefix-enabled systems will be easier with this EAPI.

EAPI 4 (2011-01-17)

Additions/Changes

`pkg_pretend` Some useful checks (kernel options for example) can be placed in this new phase to inform the user early (when just pretending to emerge the package). Most checks should usually be repeated in `pkg_setup`. See PKG-PRETEND on page 38.

`src_install` The `src_install` phase is no longer empty but has a default now. This comes along with an accompanying `default` function. See SRC-INSTALL-4 on page 41.

`pkg_info` on non-installed packages The `pkg_info` phase can be called even for non-installed packages. Be warned that dependencies might not have been installed at execution time. See PKG-INFO on page 43.

`econf` changes The helper function now always activates `--disable-dependency-tracking`. See ECONF-OPTIONS on page 60.

USE dependency defaults In addition to the features offered in EAPI 2 for USE dependencies, a `(+)` or `(-)` can be added after a USE flag (mind the parentheses). The former specifies that flags not in IUSE should be treated as enabled; the latter, disabled. Cannot be used with USE_EXPAND flags. This mimics parts of the behaviour of `--missing` in `built_with_use`. See USE-DEP-DEFAULTS on page 36.

Controllable compression All items in the `doc`, `info`, `man` subdirectories of `/usr/share/` may be compressed on-disk after `src_install`, except for `/usr/share/doc/${PF}/html`. `docompress path ...` adds paths to the inclusion list for compression. `docompress -x path ...` adds paths to the exclusion list. See DOCOMPRESS on page 66.

`dodoc` recursion If the `-r` switch is given as first argument and followed by directories, files from there are installed recursively. See DODOC on page 62.

`doins` symlink support Symbolic links are now properly installed when using recursion (`-r` switch). See DOINS on page 63.

`nonfatal` for commands If you call `nonfatal` the command given as argument will not abort the build process in case of a failure (as is the default) but will return non-zero on failure. See NONFATAL on page 58.

`PROPERTIES` Is mandatory for all package managers now to support interactive installs.

`REQUIRED_USE` This variable can be used similar to the `(R|P)DEPEND` variables and define sets of USE flag combinations that are not allowed. All elements can be further nested to achieve more functionality.

Forbidden combination To prevent activation of `flag1` if `flag2` is enabled use "`flag2? ( !flag1 )`".

OR If at least one USE flag out of many must be activated on `flag1` use "`flag1? ( || ( flag2 flag3 ... ) )`".

XOR To allow exactly one USE flag out of many use "`^^ ( flag1 flag2 ... )`".

See REQUIRED-USE on page 28.

MERGE_TYPE This variable contains one of three possible values to allow checks if it is normal merge with compilation and installation (`source`), installation of a binary package (`binary`), or a compilation without installation (`buildonly`). See MERGE-TYPE on page 53.

REPLACING_VERSIONS, REPLACED_BY_VERSION These variables, valid in `pkg_*`, contain a list of all versions (`PVR`) of this package that we are replacing, and the version that is replacing the current one, respectively. See REPLACE-VERSION-VARS on page 55.

Removals/Bans

dohard, dosed Both functions are not allowed any more. See BANNED-COMMANDS on page 58.

No RDEPEND fall-back The package manager will not fall back to `RDEPEND=DEPEND` if `RDEPEND` is undefined. See RDEPEND-DEPEND on page 29.

S fallback changes The value of the variable `S` will not automatically be changed to `WORKDIR`, if `S` is not a directory, but abort. Virtual packages are the only exception. See S-WORKDIR-FALLBACK on page 38.

AA, KV These variables are not defined any more. See AA on page 49 and KV on page 53.

EAPI 5 (2012-09-20)

Additions/Changes

Sub-slots The `SLOT` variable and slot dependencies may contain an optional sub-slot part that follows the regular slot, delimited by a `/` character; for example `2/2.30`. The sub-slot is used to represent cases in which an upgrade to a new version of a package with a different sub-slot may require dependent packages to be rebuilt. If the sub-slot is not specified in `SLOT`, it defaults to the regular slot. See SUB-SLOT on page 35.

Slot operator dependencies One of the following operators can be specified after package atoms, which will affect updates of runtime dependencies:

`:*` Any slot value is acceptable. The package will not break when its dependency is updated.

`:=` Any slot value is acceptable, but the package can break when its dependency is updated to a different slot (or sub-slot).

See SLOT-OPERATOR-DEPS on page 35.

Profile `IUSE` injection Apart from the USE flags explicitly listed in `IUSE`, additional flags can be implicitly provided by profiles. See PROFILE-IUSE-INJECT on page 55.

At-most-one-of groups In `REQUIRED_USE` you can use "`?? ( flag1 flag2 ... )`" to allow zero or one USE flag out of many. See AT-MOST-ONE-OF on page 32.

Parallel tests The default for `src_test` runs `emake` without `-j1` now. See PARALLEL-TESTS on page 41.

`econf` changes The `econf` function now always passes `--disable-silent-rules` to `configure`. See ECONF-OPTIONS on page 60.

`has_version` and `best_version` changes The two helpers support a `--host-root` option that causes the query to apply to the host root instead of `ROOT`. See HOST-ROOT-OPTION on page 59.

`usex` Usage for this helper function is `usex` *<USE flag> [true1] [false1] [true2] [false2]*. If the USE flag is set, outputs *[true1][true2]* (defaults to `yes`), otherwise outputs *[false1][false2]* (defaults to `no`). See USEX on page 67.

`doheader` and `newheader` These new helper functions install the given header file(s) into `/usr/include`. The `-r` option enables recursion for `doheader`, similar to `doins`. See DOHEADER on page 62.

`new*` standard input The `newins` etc. commands read from standard input if the first argument is `-` (a hyphen). See NEWFOO-STDIN on page 64.

`EBUILD_PHASE_FUNC` This variable is very similar to `EBUILD_PHASE`, but contains the name of the current ebuild function. See EBUILD-PHASE-FUNC on page 52.

Stable use masking/forcing New files `use.stable.{mask,force}` and `package.use.stable.{mask,force}` are supported in profile directories. They are similar to their non-`stable` counterparts, but act only on packages that would be merged due to a stable keyword. See STABLEMASK on page 22.