Table of Contents

EAPI Usage and Description

The Package Manager Specification (PMS) is a standardization effort to ensure that the ebuild file format, the ebuild repository format (of which the portage tree is Gentoo's main incarnation) as well as behavior of the package managers interacting with these ebuilds is properly written down and agreed upon.

EAPI is a version defined in ebuilds and other package manager related files which inform the package manager about the file syntax and content. It is, in effect, the version of the package manager specification (PMS) that the file adheres to.

This section provides usage and descriptions of the different EAPIs.

Usage of EAPIs

An overview about the important features of each EAPI is provided in the appendix of the Package Manager Specification. The two-page leaflet can be printed out, consulted for reference and is available as **app-doc/pms** in the main tree.

If EAPI is undefined in an ebuild, then EAPI=0 is selected. You should set the EAPI variable, by specifying it at the top of the ebuild:

Note

Most developers prefer to set the EAPI version without quotes. However, the PMS allows single and double quotes as well.

# Copyright 1999-2015 Gentoo Foundation # Distributed under the terms of the GNU General Public License v2 # $Id$ EAPI=5 inherit eutils EAPI must only be defined in ebuild files, not eclasses. (eclasses may have EAPI-conditional code)

When writing new ebuilds developers can choose whathever EAPI they think is the best. Using the features of the latest EAPI is encouraged.

EAPI=1

- Default src_compile Phase Function

  Support for the **ECONF_SOURCE** variable, which is also supported by **econf**, has been added to the default **src_compile** implementation.
  src_compile() { if [[-x ${ECONF_SOURCE:-.}/configure]] ; then econf fi if [-f Makefile] || [-f GNUmakefile] || [-f makefile] ; then emake || die "emake failed" fi }

- SLOT dependencies

Any valid atom can be constrained to match a specific SLOT. This is accomplished by appending a colon to the atom, followed by a SLOT value.

SLOT dependency examples:

- **x11-libs/qt:3**
- **~x11-libs/qt-3.3.8:3**
- **>=x11-libs/qt-3.3.8:3**
- **=x11-libs/qt-3.3*:3**

- IUSE defaults

  Add + or - before the name of the use flag in IUSE to turn it on or off by default.
  The default USE-ordering is **USE_ORDER="env:pkg:conf:defaults:pkginternal:env.d"** (see man make.conf) Disabling default IUSE is pretty much useless as it does not override the profile and user config (make.conf and package.use) # Copyright 1999-2015 Gentoo Foundation # Distributed under the terms of the GNU General Public License v2 # $Id$ EAPI=1 IUSE="foo +bar"

EAPI=2

Helpers

- doman Language Support

  doman automatically detects language codes and puts it in the appropriate directory. doman foo.1 # will go into /usr/share/man/man1/foo.1 doman foo.lang.1 # will go into /usr/share/man/lang/man1/foo.1 with EAPI=2

Metadata

- Blocker Atoms

  - New Meaning for Old Syntax

    Blocker atoms which use the previously existing !atom syntax now have a slightly different meaning. These blocker atoms indicate that conflicting packages may be temporarily installed simultaneously. When temporary simultaneous installation of conflicting packages occurs, the installation of a newer package may overwrite any colliding files that belong to an older package which is explicitly blocked. When such file collisions occur, the colliding files cease to belong to the older package, and they remain installed after the older package is eventually uninstalled. The older package is uninstalled only after any newer blocking packages have been merged on top of it.

  - New !!atom Syntax

    A new !!atom syntax is now supported, for use in special cases for which temporary simultaneous installation of conflicting packages should not be allowed. If a given package happens to be blocked by a mixture of atoms consisting of both the !atom and !!atom syntaxes, the !!atom syntax takes precedence over the !atom syntax.

- USE Dependencies

  It is possible to depend on USE-flags of packages.

Examples:

- **foo[bar]** means that package foo must have USE-flag bar enabled
- **foo[bar,baz]** means that the package foo must have both the bar and baz USE-flags enabled
- **foo[-bar,baz]** means that the package foo must have the bar USE-flag disabled and baz USE-flag enabled
- **foo[bar?]** means **bar? (foo[bar]) !bar? (foo)**
- **foo[!bar?]** means **bar? (foo) !bar? (foo[-bar])**
- **foo[bar=]** means **bar? (foo[bar]) !bar? (foo[-bar])**
- **foo[!bar=]** means **bar? (foo[-bar]) !bar? (foo[bar])**

- Customization of Output File Names in SRC_URI

  A new syntax is supported which allows customization of the output file name for a given URI. In order to customize the output file name, a given URI should be followed by a "**->**" operator which, in turn, should be followed by the desired output file name. As usual, all tokens, including the operator and output file name, should be separated by whitespace.

  Example:
  SRC_URI="http://dl.google.com/earth/client/GE4/release_4_3/GoogleEarthLinux.bin -> GoogleEarthLinux-${PV}.bin"

Phases

- New **src_prepare** Phase Function

  A new src_prepare function is called after the **src_unpack** function, with cwd initially set to **$S**.

- New **src_configure** Phase Function

  The configure portion of the **src_compile** function has been split into a separate function which is named **src_configure**. The **src_configure** function is called in-between the **src_prepare** and **src_compile** functions.

  The default **src_configure** and **src_compile** functions in EAPI=2: src_configure() { if [[-x ${ECONF_SOURCE:-.}/configure]] ; then econf fi } src_compile() { if [-f Makefile] || [-f GNUmakefile] || [-f makefile] ; then emake || die "emake failed" fi }

- Execution Order of Phase Functions

  - **pkg_setup**
  - **src_unpack**
  - **src_prepare**
  - **src_configure**
  - **src_compile**
  - **src_test**

- **src_install**
- **pkg_preinst**
- **pkg_postinst**
- **pkg_prerm**
- **pkg_postrm**

- Default Phase Functions

  The default **pkg_nofetch** and **src_*** phase functions are now accessible via a function having a name that begins with **default_** and ends with the respective phase function name. For example, a call to a function with the name **default_src_compile** is equivalent to a call to the default **src_compile** implementation.

  The default phase functions are:

  - **default_pkg_nofetch**
  - **default_src_unpack**
  - **default_src_prepare**
  - **default_src_configure**
  - **default_src_compile**
  - **default_src_test**

- Default Phase Function Alias

  A function named "**default**" is redefined for each phase so that it will call the **default_*** function corresponding to the current phase. For example, a call to the function named "**default**" during the **src_compile** phase is equivalent to a call to the function named **default_src_compile**.

EAPI=3

- Gentoo Prefix support

  Support for the **EPREFIX**, **EROOT**, and **ED** variables. If an ebuild uses one of these, it must be EAPI3 aware. See Gentoo Prefix Techdocs [https://www.gentoo.org/proj/en/gentoo-alt/prefix/techdocs.xml#doc_chap2] for more information.

- unpack supports .xz and .tar.xz

  The **unpack** command supports xz-archives and xz-compressed tar files.

EAPI=4

Helpers

- utilities die on their own, unless the nonfatal command is used

  Ebuild functions all die on their own in EAPI=4. In case that this non-zero exit status is expected, you may call **nonfatal function [arg,...]**.

Example:
EAPI=1 ... src_install() { emake DESTDIR="${D}" install || die "make install failed" dodoc ChangeLog README } EAPI=4 ... src_install() { emake DESTDIR="${D}" install nonfatal dodoc ChangeLog README }

- recursive dodoc

  dodoc supports **-r** as the first argument, which leads **dodoc** to install the specified documentation directory recursively into the docdir.

  Example:
  src_install() { default dodoc ChangeLog dodoc -r doc/ }

- doins symlink supports

  Within EAPI=4, **doins** supports installing symlinks as symlinks when installing recursively. For older EAPIs, the symlink behaviour is undefined.

- dosed and dohard are banned

  The **dosed** and **dohard** commands are banned in this EAPI.

- econf adds --disable-dependency-tracking

  Within EAPI=4, **econf** adds **--disable-dependency-tracking** to the default configure options.

- controllable compression via docompress

  To compress files in the destination-folder **${D}**, the **docompress** command may be used in **src_install**. To control which items should be compressed and which shouldn't be compressed, you may include or exclude directories or plain files. The default inclusion list contains:

  - **/usr/share/doc**
  - **/usr/share/info**
  - **/usr/share/man**

  The default exclusion list contains:

  - **/usr/share/doc/${PF}/html**

  When a directory is in- or excluded, all files and directories in the given directories shall be added to the corresponding list. If a file is in- or excluded, the file shall be added to the corresponding list (exclusion is stronger than inclusion if a file is in both lists, the inclusion will be ignored).

  If the first argument of **docompress** is **-x**, the items specified will be added to the exclusion list, otherwise they will be added to the inclusion list.

  Note

  When **docompress** is called, it is *not* required that the paths specified as its arguments are pointing to existing files or directories. However, if a file still doesn't exist when **src_install** has completed, it will be ignored with a warning.

Metadata

- use dependencies default

In addition to the use-deps specified in EAPI=2, a **(+)** or **(-)** may be added to the use-dep to define a default-value in case the use-flag does not exist in the given package. The **(+)** means that this use-flag is assumed to be enabled, **(-)** the opposite.

Example:
DEPEND=" >=dev-libs/boost-1.32[boost(+)] sys-devel/gcc[openmp(-)]"

Phases

- new pkg_pretend phase

  The new **pkg_pretend** phase can be used to do sanity checks before the main phase function sequence is run (meaning this phase is executed after the package manager has calculated the dependencies and before installing them). This phase typically checks for a kernel configuration and may **eerror** and **die** when needed. There is no guarantee that the ebuild's dependencies are installed when this phase is called. As **pkg_pretend** is not called in the main phase function sequence, environment saving is not guaranteed.

  Example:
  # Copyright 1999-2015 Gentoo Foundation # Distributed under the terms of the GNU General Public License v2 # $Id$ EAPI=4 inherit linux-info ... CONFIG_CHECK="FUSE_FS" ERROR_FUSE_FS="this is an unrealistic testcase..." pkg_pretend() { if use kernel_linux ; then if [[-e "${ROOT}"/usr/src/linux/.config]] ; then if kernel_is lt 2 6 30 ; then check_extra_config fi fi fi }

- default src_install is no longer a no-op

  The default **src_install** function in EAPI=4:
  src_install() { if [[-f Makefile]] || [[-f GNUmakefile]] || [[-f makefile]] ; then emake DESTDIR="${D}" install fi if ! declare -p DOCS >/dev/null 2>&1 ; then local d for d in README* ChangeLog AUTHORS NEWS TODO CHANGES THANKS BUGS \ FAQ CREDITS CHANGELOG ; do [[-s "${d}"]] && dodoc "${d}" done elif declare -p DOCS | grep -q "^declare -a " ; then dodoc "${DOCS[@]}" else dodoc ${DOCS} fi }

- pkg_info for non-installed packages

  The **pkg_info** function may also be called by the package manager for non-installed packages. Ebuild writers should note that dependencies may not be available.

Variables

- REQUIRED_USE

  The **REQUIRED_USE** variable contains a list of assertions that must be met by the configuration of USE flags to be valid for this ebuild. In order to be matched, a USE flag in a terminal element must be enabled (or disabled if it has an exclamation mark prefix).

  Essentially, **REQUIRED_USE** is an analogue of **DEPEND** style syntax. For example, to state that some combination is forbidden, i.e. "if **foo** is set, **bar** must be unset":
  REQUIRED_USE="foo? (!bar)"

  To state "if **foo** is set, then at least one of **bar**, **baz**, and **quux** must be activated":
  REQUIRED_USE="foo? (|| (bar baz quux))"

  To state "exactly one of **foo**, **bar**, or **baz** must be set, but not several":
  REQUIRED_USE="^^ (foo bar baz)"

Note that the last relationship is that of an Exclusive OR (XOR). While an XOR could be formed from usual **DEPEND** syntax, a specific ^^ operator has been added for this case.

Finally, to state "at least one of **foo**, **bar**, or **baz** must be set":
REQUIRED_USE="|| (foo bar baz)" See section ::general-concepts/use-flags/#conflicting-use-flags for when (and when not) to use **REQUIRED_USE**.

- REPLACING_VERSIONS and REPLACED_BY_VERSION

  The **REPLACING_VERSIONS** variable contains a whitespace-separated list of all versions (**PVR**) of this package that are being replaced (uninstalled or overwritten) as a result of this install. It is a list, not a single optional value, to handle pathological cases such as installing **foo-2:2** to replace **foo-2:1** and **foo-3:2**.

  REPLACING_VERSIONS is valid in **pkg_preinst** and **pkg_postinst**. In addition, it may be available in **pkg_pretend** and **pkg_setup**, although you should take care to handle binary package creation and installation correctly when using it in these phases.

  The **REPLACED_BY_VERSION** variable contains the single version (**PVR**) of this package that is replacing us, if we are being uninstalled as part of an install, or an empty string otherwise. It is valid in **pkg_prerm** and **pkg_postrm**.

- MERGE_TYPE

  The **MERGE_TYPE** variable contains the type of package that is being merged. Possible values are:

  - ***source*** -

    if building and installing a package from source,

  - ***binary*** -

    if installing a binary package,

  - ***buildonly*** -

    if building a binary package without installing it.

- DOCS

  The **DOCS** variable is an array or whitespace-separated list of documentation files for the default **src_install** function to install using **dodoc**. If undefined, a reasonable default list is used. See the default **src_install** function above.

- AA and KV variables are gone

  The **AA** and **KV** variables are no longer set in EAPI=4.

- no more RDEPEND="${DEPEND}"

  When **RDEPEND** is unset, there will no longer be an automatic assignment of **RDEPEND="${DEPEND}"**.

EAPI=5

Metadata

- REQUIRED_USE supports new at-most-one-of operator

  The new at-most-one-of operator consists of the string **'??'**, and is satisfied if zero or one (but no more) of its child elements is matched.

- SLOT supports optional "sub-slot" part

  The **SLOT** variable may contain an optional sub-slot part that follows the regular slot and is delimited by a **/** character. The sub-slot must be a valid slot name. 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. When the sub-slot part is omitted from the SLOT definition, the package is considered to have an implicit sub-slot which is equal to the regular slot.

- Slot operators and sub-slots in dependencies

  A slot dependency may contain an optional sub-slot part that follows the regular slot and is delimited by a **/** character. This can be useful for packages installing pre-built binaries that require a library with a specific soname version which corresponds to the sub-slot. For example:
  RDEPEND="dev-libs/foo:0/3"

  Dependency atoms can use slot operators to clarify what should happen if the slot and/or sub-slot of a runtime dependency changes:

  - **:*** Indicates that any slot value is acceptable. In addition, for runtime dependencies, indicates that the package specifying the dependency will not break if the package matching the dependency is replaced by a different matching package with a different slot and/or sub-slot.

  - **:=** Indicates that any slot value is acceptable. In addition, for runtime dependencies, indicates that the package specifying the dependency will break unless there is available a package matching the dependency and whose slot and sub-slot are equal to the slot and sub-slot of the best installed version that had matched this dependency at the time when the package specifying this dependency had been installed.

  - **:slot=** Indicates that only a specific slot value is acceptable, and otherwise behaves identically to the **:=** operator.

    ### Note

    use **:slot/subslot** without a **=** to depend on a specific slot and sub-slot pair; it's a syntax error to use **:slot/subslot=** in an ebuild.

  The **:slot** dependency syntax continues to behave like in EAPI=4 or earlier, i.e. it indicates that only the specific slot value is acceptable, but the package will not break when the version matching the runtime dependency is replaced by a version with a different sub-slot.

  For example:
  RDEPEND="dev-libs/foo:2= >=dev-libs/bar-0.9:= media-gfx/baz:* x11-misc/wombat:0"

  means that the package should be rebuilt when **foo:2** or **>=bar-0.9** are upgraded to versions with different subslots, but that changes in subslots of **baz** or **wombat:0** should be ignored.

Profiles

- Profile stable USE forcing and masking

  In profile directories with an EAPI supporting stable masking, new USE configuration files are supported: **use.stable.mask**, **use.stable.force**, **package.use.stable.mask** and **package.use.stable.force**. These files behave similarly to previously supported USE configuration files, except that they only influence packages that are merged due to a stable keyword.

Helpers

- econf adds --disable-silent-rules

  This option will automatically be passed if **--disable-silent-rules** occurs in the output of **configure --help**.

- new* commands can read from standard input

  Standard input is read when the first parameter is **-** (a hyphen).

- New option --host-root for {has,best}_version

  This option **--host-root** will cause the query to apply to the host root instead of ROOT.

- New doheader helper function

  Installs the given header files into **/usr/include/**. If option **-r** is specified, descends recursively into any directories given.

- New usex helper function

  ### Example 1.

  ```
  USAGE: usex #USE flag# [true output] [false output] [true suffix] [false suffix]
  DESCRIPTION:
  If USE flag is set, echo [true output][true suffix] (defaults to "yes"),
   otherwise echo [false output][false suffix] (defaults to "no").
  ```

Phases

- src_test supports parallel tests

  Unlike older EAPIs, the default **src_test** implementation will not pass the -j1 option to emake.

Variables

- EBUILD_PHASE_FUNC

  During execution of an ebuild phase function (such as **pkg_setup** or **src_unpack**), the **EBUILD_PHASE_FUNC** variable will contain the name of the phase function that is currently executing.