NAME

eix - a set of utilities for searching, diffing and updating a binary cache of your local portage-trees

SYNOPSIS

eix [*common options*] [*OPTIONS*] *EXPRESSION*

eix-update [*common options*] [*eix-update options*]

eix-diff [*common options*] [*OLD-CACHE*] [*NEW-CACHE*]

eix-sync

eix-postsync

eix-test-obsolete

eix-remote

eix-installed-after

eix-installed

eix-etcat

eix-functions.sh

eix-header

eix-drop-permissions [**--**] [*command to execute*]

masked-packages [*options*] *category*/*name*-*version*[**:***slot*][**::***repo*] ...

versionsort [**-n**|**-p**|**-f**|**-v**|**-r**|**-V**] [*(rubbish)package*-]*version* ...

DESCRIPTION

eix-update generates a binary cache from your local portage-tree and overlays. **eix** searches this cache for packages that match the restricting criteria you specify in *EXPRESSION*; if you specify no such restriction, all packages are output, of course. **eix-diff** compares tw o binary caches and finds packages that were added, removed or for which the highest stable versions has changed.

All of these programs and scripts read the configuration files described later. **eix-sync** has optionally an additional separate configuration file.

eix-sync can sync the portage/overlay trees and compare them with the old cache using **eix-diff**. To get more help on **eix-sync** call **eix-sync -h** and see the documentation of */etc/eix-sync.conf* below. Note that the content of the latter can also be stored in the variable **EIX_SYNC_CONF**.

eix-postsync is a simple alternative to **eix-sync**. It is similar to**eix-update**, b ut backups the cache before so that **eix-diff** can be used to show the differences to the previous syncing. To execute **eix-postsync** automatically after **emerge --sync** with >=portage-2.3.7, you can create the directory /etc/portage/postsync.d/ and symlink **eix-postsync** there. The main disadvantage of using **emerge --sync** with that symlink (compared to **eix-sync**) is that in the exceptional situation when a new upgrade of eix should no longer support the current database format then **eix-diff** will not work for the first syncing after the upgrade (unless you manually called **eix-update** before syncing).

eix-test-obsolete is a script which calls **eix** several times to display the output of **eix -tTc** in a more organized manner.

eix-remote can sync an eix database from an external server and add/remove it to the current database. This can be used to create a local database of packages in all registered repositories/overlays, not just those in overlays installed locally. By default it uses a separate cachefile. With this default you should regularly call **eix-remote add1** and/or **eix-remote add2** after **eix-update** so that the separate cachefile is synchronized with your main database. (If you use **eix-sync** this is done by default.) If you use the same cachefile you can save remote-data over **eix-update** calls by setting **KEEP_VIRTUALS=true** in */etc/eixrc*. By default, this script drops its permissions using **eix-drop-permissions**. To get more help on **eix-remote** call **eix-remote -h**.

eix-installed-after is a simple script (with many comments) demonstrating some possibilities how a custom eix output format can be used. It outputs packages installed after (or before) the last (or first) emerge of a specific package/version. To get more help on this script call **eix-installed-after -h**.

eix-functions.sh outputs helper functions used by **eix-sync** and **eix-remote**. You might want to use them for your own similar scripts. You can either "eval" the output of this program. Note that after this, you should usually call **ReadFunctions** for initialisation.

eix-header is a helper program for **eix-remote** (and possibly your own scripts). It can check database file headers and can find/output overlay paths or labels stored in these files. For details how to use it, call **eix-header -h**.

eix-drop-permissions is a helper program for **eix-remote** (and possibly your own scripts). It executes the command given as argument with permissions dropped according to **EIX_USER**, **EIX_GROUP**, **EIX_UID**, **EIX_GID** (see later). Note that also the binaries **eix**, **eix-diff**, and **eix-update** drop their permissions according to these variables as soon as possible, i.e., you might have to adjust these variables if you use **eix** (in particular **eix-update**) in a nonstandard setting.

eix-installed is a simple script which outputs all installed packages (in their exact version) and can check for packages installed with/without repository or buildtime information (cf. the description of **CHECK_INSTALLED_OVERLAYS** and **USE_BUILD_TIME**). To get more help on **eix-installed** call **eix-installed -h**.

eix-etcat is a wrapper script which calls eix with a configuration such that it resembles the behaviour of **etcat -v** from old app-portage/gentoolkit versions or its updated version from https://github.com/proteusx/etcat/ (e.g. app-portage/etcat from the mv overlay). The output is analogousand the default input accepts a sequence of (exact) package names (the specification of the category is optional). It might be instructive to look at the simple wrapper script if you want to write your own input/output format: The output format can be seen by calling **eix --print FORMAT_ETCAT** and **eix --print FORMAT_VERSION_ETCAT**.

masked-packages is a helper tool which matches the arguments against a list of masks. Details can be found near the end of the manpage.

versionsort is a helper tool for scripts which cuts the version strings from its arguments and outputs them sorted according to the portage rules of version sorting. Details can be found near the end of the manpage.

OPTIMIZATION, MEMORY, SECURITY

If you installed eix from the gentoo repository, then very likely the eix ebuild **will not** have installed eix with the **CXXFLAGS** and **LDFLAGS** which are recommended for optimization and security of eix by the eix maintainer. Also, eix can be configured to use less memory if required. See the section**INST ALLATION** for how to install eix in the way the maintainer recommends.

EXAMPLES

The following examples are some useful but perhaps unusual applications of eix. They are listed here to give you an idea what unexpected things you can do with eix. Since the examples are meant to be copied from this manpage they are listed early. To understand how they work, you should read the full manpage, of course. For more examples see e.g. the script **eix-installed-after** which contains many comments.

cmd | **eix '-|*' --format '<markedversions:NAMESLOT>'**

Assuming *cmd* creates a list of the form **category/name-version** or **=category/name-version**, the above prints the corresponding list **category/name** or **category/name:SLOT** (depending on whether **SLOT**s need to be distinguished).

cmd | **eix '-|*' --format '<markedversions:NAMEASLOT>'**

Similar to the above, but print always **category/name:SLOT** even if **SLOT** may be redundant. (Mnemonic: **A**lways).

eix '-I*' --format '<installedversions:NAMEVERSION>'

Print out installed packages, using the format **category/name-version**. Of course, you could replace **NAMEVERSION** also by **NAMESLOT** or **NAMEASLOT** to get another format. The only purpose of the **I** option in this context is to speed up the output slightly.

eix '-I*' --format '<installedversions:EQNAMEVERSION>'

Similarly to the above but in the format **=category/name-version** which you can pipe directly to portage.

eix '-I*' --format '<installedversions:DATESORT>' | sort -n | cut -f2-3

Print out installed packages (with slots if necessary), sorted by installation date. If you want the output in another format, you have to modify **DATESORT** correspondingly (you can see the original definition in the output of **eix --dump**).

This sorting trick works as follows: The **DATESORT** variable outputs as the first column the seconds since epoch (**eix --print DATESORT** will show you that this happens by the variable **DATESORT_DATE** whose first entry is **%s**), and **sort** thus gets the correct order by sorting alphabetically. Finally the**cut** command gets rid of this first column which w as only needed for sorting.

In the above examples, things like **NAMEVERSION** and **DATESORT** are actually just names of variables which are predefined in eix (with **eix --dump** you see them). You can as well define your own variables and use them. To understand how this works, please read the manpage, especially the description of the **FORMAT** string.

OPTIONS

Common options

These options are common to **eix**, **eix-diff**, and **eix-update**.

-h, **--help**

Print a help screen and exit.

-Q, **--quick** (toggle) (not for **eix-update**)
Do (not) read the slots of installed versions which cannot be guessed (i.e. installed versions of packages with at least two different slots for which the installed version is not in the database anymore). Note that with this option, eix and eix-diff might report false positives for up-/downgrade recommendations for these packages.

--care (not for **eix-update**)
This deactivates **--quick**, and moreover, slots of installed version are always read instead of relying on the guess of the slot. This will in particular make sure that you get an up-/downgrade recommendation if a slot name of an installed version changes. Note that this will dramatically decrease the speed at the first call. (If your filesystem has a reasonable cache, later calls are almost not slower than without this option.)

--deps-installed (not for **eix-update**)
This is the same as **DEPS_INSTALLED=true**. Dependencies of installed version are always read, even if they are known from an available version. Note that this will dramatically decrease the speed at the first call. (If your filesystem has a reasonable cache, later calls are almost not slower than without this option.)

-q, **--quiet** (toggle)
Produce no output on stdout. For eix you can decrease execution time by combining this (depending on your needs) with either **--brief** or **--brief2** and by setting **COUNT_ONLY_PRINTED=false**. See also**NOFOUND_ST ATUS** and **MOREFOUND_STATUS**

--dump
Show the current eixrc-variables, and their defaults as comments; then exit.

--dump-defaults
Show the defaults of the eixrc-variables, and their current values as comments; then exit.

--print *VAR*
Print the specified variable *VAR* of eixrc or of portage settings, completely expanded as it would be used internally by eix; then exit. The exit status is nonzero if *VAR* is not known to eix (see **--known-vars**) This is mainly intended for usage in scripts or for debugging purposes. If you use it in scripts you might want to specify **PRINT_APPEND** to get trailing spaces (see description of **PRINT_APPEND**). Two special *VAR* values which can be accessed only in this way are **USE.profile** and **USE.make_conf** which contain the USE string of the profile or make.conf before the concatenation, respectively.

--known-vars
Print an alphabetical list of variables known to eix with **--print**. After each variable name a newline is output.

-V, **--version**
Print version and exit.

-n, **--nocolor**
Disables the use of ANSI color codes. This is useful for terminals that do not support ANSI colors. (This is automatically turned on if stdout is no tty, but can be overridden by using --force-color)

-F, **--force-color**
The opposite of --nocolor.

Special information options
The following special information options are only provided by the **eix** binary. They are one-shot exclusive options, i.e. eix outputs only the required data and then exits.

--color (always/never/auto)
Override all other color options and variables: Colorize always, never, or only if the output goes to some terminal

--ansi (also for eix-diff)
Define the 256 color palette suggested by Todd Larason (the author of the 256 color support for xterm). Use this if something some tool has redefined the default palette.

--256, **--256d**, **--256d0**, **--256d1**, **--256l**, **--256l0**, **--256l1**, **256b**
Print the 256 color palettes (assuming it follows ansi color scheme) and exit. With **--256d**/**--256l** (**--256d0**/**--256l0**, **--256d1**/**--256l1**) only the foreground palettes with dark/light background (or only the normal or bright dark/light foreground palette, respectively) is printed. With **--256b** only the background palette is printed.

--print-all-eapis
print all EAPIs used in some version.

--print-all-useflags
print all IUSE or REQUIRED_USE words used in some version.

--print-all-keywords
print all KEYWORDS used in some version.

--print-all-slots
print all SLOT strings used in some version.

--print-all-licenses
print all LICENSE strings used in some package.

--print-all-depends
print all words occurring in some **{,R,P,B}DEPEND**. This only works if **DEP=true** is active (and was so when the cachefile was created).

--print-world-sets
print the world sets.

--print-profile-paths
Outputs all paths of the current profile. To each Path **PRINT_APPEND** is appended. If **PRINT_APPEND** is empty, the null character is appended.

Output options

--nowarn (toggle)
Suppress some warnings concerning the portage configuration.

-x, **--versionsort** (toggle)
Prints available versions sorted by versions (slots). If sorted by slots, each new slot starts in a new line. This mode can be combined with**--v ersionlines** (**-l**) to start a new line even for every version.

-l, **--versionlines** (toggle)
Prints available versions in a (vertical) list. Only in this mode some additional information for each version is output. More precisely, depending on whether you have chosen **--verbose** and depending on the state of the configuration variables **VERSION_{IUSE,KEYWORDS,DEPS}_{NORMAL,VERBOSE}**, this can be the **KEYWORDS**, **IUSE**, **DEPEND**, **RDEPEND**, **PDEPEND**, **BDEPEND**, **IDEPEND** data. Note that if **IUSE** is not printed per version then it is only collected for the whole package which can give the wrong impression, since often **IUSE** changes dramatically with versions.

-c, **--compact**
Causes eix to use a compact layout for search results. Useful for obtaining a better overview in the case of a long list of results, and also to help speed up searching over slow connections such as a serial console.

-v, **--verbose**
Use a verbose layout with additional information about search results such as the license of a package.

-N, **--normal**
Use the normal layout which is the default if **DEFAULT_FORMAT** was not explicitly changed.

--xml, **--proto,** (toggle)
Output in XML or protobuf format.

For usage from an external program you will probably want to combine this with **--care**.

For **--xml**, you probably additionally want to export **LOCAL_PORTAGE_CONFIG** to make sure that the preferred user output setting do not influence your output.

The **--proto** format is independent of this variables: local and system settings are separate.

If some of these options is active, **OVERLAYS_LIST=none** and **--pure-packages** are automatically active so that your output is not clobbered with normal eix output.

The XML output format can be slightly influenced by the **XML_*** variables. The XML format used is documented in human-readable form in the files eix-xml.html or eix-xml.txt and in less human-readable form (namely as an xml-schema) in the file eix-xml.xsd.

The protobuf format is in the file eix.proto.

-*, **--pure-packages** (toggle)
(do not forget quoting if you use the short form from within a shell.) Omit printing of additional information (overlay names, number of found packages) after the packages. This might be useful for some shell scripts parsing the output

-#, **--only-names** (toggle)
As **--pure-packages**, but additionally print only the category and name of the packages found.

-0, **--brief** (toggle)
Print at most one package then stop. This option is typically faster when used with **COUNT_ONLY_PRINTED=false**. In the latter case, when you use fuzzy search, not necessarily the best match is output.

--brief2 (toggle)
As **--brief**, but print up to two packages.

Special options for eix

-t, **--test-non-matching**
Before other output, print entries of /etc/portage/package.* which do not match any existing version in the package database or which apparently have no meaning because they are empty (see **TEST_FOR_EMPTY**).

This option also lists all installed packages whose name is not in the database.

Note that this is essentially different from **-T** (see below). The latter only tests for packages**in the database** whether redundant entries in /etc/portage/package.* exist or whether the installed versions are available, respectively.

This option is best combined with **-T** to clean up /etc/portage/package.*

Or you can combine it with **-e** to have no other output.

If you have a reason to exclude certain entries/packages from this test, you can write those entries into a file /etc/portage/package.*.nonexistent where ***** is one of **accept_keywords** (or the obsolete **keywords**), **mask**, **unmask**, **use**, **env**, **license**, **accept_restrict**, **cflags**, and **installed**. These files

(and how to modify their filenames) are described later.

-R, **--remote**
Uses the value of **EIX_REMOTE1** as cache file name. Use this option if you want to see packages in the cache created and synchronized by **eix-remote**. You can make this option the default by setting **REMOTE_DEFAULT=1**.

-Z, **--remote2**
Uses the value of **EIX_REMOTE2** as cache file name. Use this option if you want to see packages in the cache created and synchronized by **eix-remote**. You can make this option the default by setting **REMOTE_DEFAULT=2**.

--cache-file *FILE*
Use *FILE* instead of **/var/cache/eix/portage.eix**.

Options for EXPRESSION

EXPRESSION is used to narrow which packages eix prints.

An EXPRESSION can contain Boolean operators and tests according to the following grammar:

EXPRESSION ::= [**--not** | **-!**] BRACE_OR_TEST |
EXPRESSION [**--and**| **-a**] EXPRESSION |
EXPRESSION [**--or** | **-o**] EXPRESSION

BRACE_OR_TEST ::= **--open**|**-(** EXPRESSION **--close**|**-)** |
TEST_WITH_OPTIONS

TEST_WITH_OPTIONS ::= [TEST_OPTIONS] [PATTERN]

Do not forget that **!**, **(**, **)** have to be quoted in shells so that eix actually receives these as parts of arguments!

If you want that an EXPRESSION starts with **-**, precede it with two further **--** symbols: This way, the EXPRESSION is not read as an option, and two additional **--** symbols are ignored. For example, **eix ---tool --or ---util** will output the packages containing **-tool** or **-util**.

The meaning of the logical operators should be obvious with perhaps some exceptions:

1. If neither **--and**|**-a** nor **--or**|**-o** is between two EXPRESSIONs, then one of them is implicitly assumed; whether this is **-a** or **-o** depends on the setting of the configuration variable **DEFAULT_IS_OR**.

2. The operators **-a** and **-o** have equal precedence and are left-associative. In particular, **X -o Y -a Z** fails if **Z** fails.

3. **--not**|**-!** negates only the result of the subsequent BRACE_OR_TEST.

4. If PATTERN is omitted, it defaults to the empty PATTERN. For instance, with the default settings, **eix** without arguments will match all packages, since the empty string is contained in every name. On the other hand, **eix -e** should (usually) output nothing, because no package name should *exactly be* the empty string.

5. Note that the syntax implies that PATTERN always ends an expression. TEST_OPTIONS after PATTERN always start a new expression (i.e. an implicit **--and** or **--or** is inserted, depending on **DEFAULT_IS_OR**). In particular**eix -e f oo** has a different meaning than **eix foo -e**. The latter means the same as **eix foo --and -e** or **eix foo --or -e**, depending on **DEFAULT_IS_OR**.

6. Observe that TEST_OPTIONS can consist of several options. All of these options are applied simultaneously, i.e. in a sense these options are logically braced and combined with **and** (independently of **DEFAULT_IS_OR**). This is somehow ambiguous, since PATTERN can be omitted. This ambiguity is resolved by interpreting successive TEST_OPTIONS always as a part of one TEST_WITH_OPTIONS. For instance, in **eix -I -O -e foo** all options are considered as part of a common *EXPRESSION* (not as four *EXPRESSION*s as would be the case for **eix -I ” -O ” -e ” foo**). On the other hand, in**eix -I --not -e** the **--not** will cause the subsequent TEST_OPTION **-e** to be considered as part of a new *EXPRESSION*. Options other than TEST_OPTIONS or a logical option (like **-!**, **-(**, **-)**, **-a**, **-o**) are ignored in this respect. For instance, **eix -I -c -e** generates only one *EXPRESSION*, since **-c** is neither a TEST_OPTION nor a logical option and therefore plays no role for the interpretation of *EXPRESSION*.

7. The TEST_OPTIONS can specify things like the **match algorithm** and also **operation selection**. All these specifications refer **only** to the current TEST_WITH_OPTIONS, in particular, they are only active for the next **PATTERN**.

Note that besides using this expression syntax a different way of implicit (though slow) package selection on various other criteria is possible by defining **FORMATSTRING** (see below) using conditionals such that it outputs an empty string for the undesired packages.

Here are the admissible TEST_OPTIONS:

-I, **--installed**

Only match installed packages. Please do not use this as a replacement for **eix-installed -a** (or **qlist -ICv** or **equery**), it is not the same: Packages that are installed, but no longer in the portage tree (or an overlay) are not listed here. However, you should better not have such packages at all (better put these packages in overlays in case you need to reinstall them). To find such packages, you can use **eix -te** (or **eix -tI** to get listed both) but be aware that eix -t does not obey the usual **FORMAT** rules for these packages. So you better do not use this in scripts unless you know what you are doing.

If you really want to use this option as a substitute for equery in scripts, you might want to combine it with some of

--format --only-names

--format ’<installedversions:NAMEVERSION>’ --pure-packages

--format ’<installedversions:EQNAMEVERSION>’ --pure-packages

--format ’<installedversions:NAMESLOT>’ --pure-packages

--format ’<installedversions:NAMEASLOT>’ --pure-packages

--format ’<installedversions:DATESORT>’ --pure-packages

-i, **--multi-installed**

Only match packages which are installed in at least two different versions. Usually, this means that these versions are slotted (at installation time).

-d, **--dup-packages**

Only match duplicated packages, for example, if sys-foo/bar exists both in the official portage tree and a local overlay. If**DUP_P ACKAGES_ONLY_OVERLAYS** is set (see below), the instances must be in two different local overlays.

-D, **--dup-versions**

Only match packages with duplicated versions, for example, if sys-foo/bar-0.2.1 exists both in the official portage tree and a local overlay. If**DUP_VERSIONS_ONL Y_OVERLAYS** is set (see

below), both instances must be in overlays.

-1, **--slotted**
Only match packages with a nontrivial slot, i.e. where SLOT is nonempty and different from "0".

-2, **--slots**
Only match packages with at least two different slots. In contrast to -1, a package is not shown here, if only one slot is available with e.g. slot-name "4.3".

-u, **--upgrade**, **--upgrade+**, **--upgrade-**
Only match packages which have at least one slotted version installed which is not the best version within that slot. This means usually that you should either upgrade or downgrade that package.

However, the test takes also **UPGRADE_TO_HIGHEST_SLOT** (see below) into account.

If you use **--upgrade+** or **--upgrade-**, then the test acts as if **LOCAL_PORTAGE_CONFIG** is **true** or **false**. Otherwise, this decision is based upon**UPGRADE_LOCAL_MODE**.

If you want to see only packages with downgrade recommendations, you might make use of the **FORMATSTRING** features described below.

--stable, **--testing**, **--non-masked**, **--system**, **--profile**
Only match packages which have at least one version which is stable (and non-masked), testing or stable (and non-masked), non-masked, or in system or profile, respectively. If several of these options are combined in one test, it is the same version which must satisfy all.

--stable+, **--testing+**, **--non-masked+**, **--system+**, **--profile+**
This is as the above ones just that the test acts as if **LOCAL_PORTAGE_CONFIG=true**.

--stable-, **--testing-**, **--non-masked-**, **--system-**, **--profile-**
This is as the above ones just that the test acts as if **LOCAL_PORTAGE_CONFIG=false**.

--installed-unstable, **--installed-testing**, **--installed-masked**
Only match packages which have at least one non-stable, testing, or masked version installed (the test acts as if **LOCAL_PORTAGE_CONFIG=false**). If several of these options are combined in one test, it is the same version which must satisfy all.

--world
Only match @world packages. This is analogous to "emerge @world", i.e. it includes not only packages in the world file but also in world sets and the @system set. If you do not want to have all included, choose the appropriate alternative option.

--world-file
This only matches packages from the world file or from the @system set.

--world-set
This only matches packages from world_set or from the @system set.

--selected
Only match @selected packages. This is analogous to "emerge @selected", i.e. it includes not only packages in the world file but also in world_sets (if you have @system contained in the world_sets file, then the behaviour is of course identical to **--world**). If you do not want to have both included, choose the appropriate alternative option.

--selected-file
This only matches packages from the world file.

--selected-set
This only matches packages from world_set.

--binary
Only match packages with a binary file (*.tbz2, *.gpkg.tar, or *.xpak) in PKGDIR. The version of the binary file must either match the version of an available version or of an installed version.

(However, note that if no version is available, the package cannot be found either). Just the existence of the corresponding *.tbz2, *.gpkg.tar, or *.xpak file is checked: Whether portage can actually use the file depends also on metadata within that file (like USE settings) which is not considered by eix.

--multi-binary*NR*
Only match packages with a version with at least *NR* files matching **--binary**.

--nonvirtual
Only match packages with at least one version from the main tree or a non-virtual overlay.

--virtual
Only match packages with at least one version in a virtual overlay.

-O, **--overlay**
Only match packages with at least one version in an overlay.

--in-overlay *overlay*
Only match packages with at least one version in an overlay matching *overlay*.

If this option is repeated, the additional *overlay* arguments are joined to the list of admissible overlays.

overlay may be either a wildcard pattern or a number. Note that if you use the default **OVERLAYS_LIST=all-used-renumbered** you do not see the correct overlay numbers; to get a list of the correct overlay numbers you can e.g. call

OVERLAYS_LIST=all eix --not

or in scripts better

OVERLAYS_LIST=all PRINT_COUNT_ALWAYS=never eix -!

The special values **0** or **$PORTDIR** match the "main" tree (which in this connection is considered as the 0'th overlay).

If *overlay* is empty (or omitted if **--in-overlay** is the last option) it matches all overlays except for the "main" tree (i.e. **--in-overlay ''** is the same as **-O**).

--only-in-overlay *overlay*
Only match packages which have only versions in an overlay matching *overlay*.

If this option is repeated, the additional *overlay* arguments are joined to the list of admissible overlays.

overlay may be either a wildcard pattern or a number, as in **--in-overlay**. In particular, **--only-in-overlay ''** matches all packages which are not in the official portage tree but only in some overlay.

-J, **--installed-overlay**
Only match packages which have been installed from some overlay. To get a completely reliable result you should set **CHECK_INSTALLED_OVERLAYS** to true (which is not the default because it dramatically slows down the test). See **CHECK_INSTALLED_OVERLAYS** for details.

--installed-from-overlay *overlay*
This is analogous to **--in-overlay** with the difference that only packages are matched which have at least one version installed from *overlay*. For instance, **--installed-from-overlay 0** will only match those packages which have at least one version which was installed from the regular portage tree. As for -J, you should set**CHECK_INST ALLED_OVERLAYS** to true to get a completely reliable result.

--installed-in-some-overlay
Only match packages with at least one installed version number which is also in some overlay.

--installed-in-overlay *overlay*
This is analogous to **--in-overlay** with the difference that only packages are matched which have at least one installed version in *overlay*. For instance, **--installed-in-overlay 0** will only match those packages which have at least one version which is also in the regular portage tree.

--restrict-fetch
Only match packages which have at least one version with RESTRICT=fetch. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--restrict-mirror
Only match packages which have at least one version with RESTRICT=mirror. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--restrict-primaryuri
Only match packages which have at least one version with RESTRICT=primaryuri. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--restrict-binchecks
Only match packages which have at least one version with RESTRICT=binchecks. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--restrict-strip
Only match packages which have at least one version with RESTRICT=strip. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--restrict-test
Only match packages which have at least one version with RESTRICT=test. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--restrict-userpriv
Only match packages which have at least one version with RESTRICT=userpriv. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--restrict-installsources
Only match packages which have at least one version with RESTRICT=installsources. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--restrict-bindist
Only match packages which have at least one version with RESTRICT=bindist. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--restrict-parallel
Only match packages which have at least one version with RESTRICT=parallel. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--properties-interactive
Only match packages which have at least one version with PROPERTIES=interactive. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--properties-live
Only match packages which have at least one version with PROPERTIES=live. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--properties-virtual
Only match packages which have at least one version with PROPERTIES=virtual. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

--properties-set
Only match packages which have at least one version with PROPERTIES=set. If used with other PROPERTIES/RESTRICT tests, the version must satisfy all simultaneously.

-T, **--test-obsolete**
Only match obsolete packages.

Packages are obsolete if they have redundant entries in /etc/portage/package.* (if **TEST_FOR_REDUNDANCY** is true) or if not all installed versions exist (if **TEST_FOR_NONEXISTENT** is true).

What is considered as redundant is defined by the **REDUNDANT_IF**-variables below, and what is considered as non-existent is defined by the **NONEXISTENT_IF**-variables. Note that the test for versions from obsolete overlays works only reliable if you set **CHECK_INSTALLED_OVERLAYS** to true (which is not the default because it dramatically slows down the test). See **CHECK_INSTALLED_OVERLAYS** for details.

Note that this options only tests packages in the database - in particular, you will not find entries for e.g. renamed or removed (from the portage tree) packages with this option. Use **-t** to find the latter.

Therefore, this option is best combined with -t to find also other types of obsolete entries.

If you have a reason to exclude certain packages from this test, you can write those entries into a file (or directory) /etc/portage/package.nowarn This file (and how to specify alternative/additional files) is described later.

-|, **--pipe**
(Recall that a shell will not pass an unquoted | sign, so quote properly).

Only match packages/masks from standard input; usually you will want to use this in a pipe, redirected e.g. from emerge -pv (similar to genlop -p). Actually, any (space or newline separated) words are considered which contain a / symbol.

Each such word is considered as a mask (in the form of the first entry in /etc/portage/package.* files), and due to a heuristics package versions can also specified without an explicit leading = symbol. (This can lead to ambiguities; use**--pipe-mask** if you need a reliable result.)

All available packages/versions which match a mask from the input will in the output be considered as marked. For details on the latter, see the **marked** and **markedversions:*** formatstrings.

Even if **--pipe** occurs more than once, the standard input is of course only read once, but interpreted repeatedly for each occurrence (i.e. if the first --pipe matches, all others will also match).

If you want to use the standard input only for marking but not for selection, you can choose an expression like

eix something -a "-(" --pipe -o "-)"

If the option **--pipe-name** or **--pipe-version** occurs after this option in the argument list, **--pipe** acts as if it were **--pipe-name** or **--pipe-version**, respectively.

--pipe-mask
This is analogous to **--pipe** with the difference that only masks are allowed in the input, i.e. if package versions are specified the leading = is non-optional. This avoids ambiguities in the input if a part of the name looks like a version number. You should use this option instead of **--pipe** if you want a reliable result!

Match field selection

The subsequent options define the fields that the pattern should be tested on. Multiple fields may be specified in one expression (the expression matches if the pattern matches at least one of the specified fields). If you do not specify some of these options, the default is chosen according to some heuristic depending on the form of your pattern. In most cases, the default match field will be **--name**, but if your pattern looks "special" like e.g. "cat/n" or "@set", the default match field will change to **--category-name**, **--set**, **--description**, **--homepage**, or **--license**. Details of the heuristic are specified by the**DEFAULT_MATCH_FIELD** configuration variable explained later. Depending on the configuration of that variable, it might even be non-optional to specify some of the following options for every query.

-y, **--any**

Any match field is searched. This is the same as specifying simultaneously all of **--name**, **--description**, **--category**, ... Thus, you may want to use this if you want to be sure to not miss any matches. However, be warned that the output list will often be much longer than you might expect. (If you are curious, you can find with options**-vl** and piping to grep the often une xpected reason why a match occurs with **-y**.) To exclude at least the dependency strings from matches, you might want to combine this option with setting **DEP=false**, e.g.

DEP=false eix --any SIP

-s, **--name**

e.g. "eix"

-S, **--description**

e.g. "Small utility for searching .."

-C, **--category**

e.g. "app-portage"

-A, **--category-name**

e.g. "app-portage/eix"

-H, **--homepage**

e.g. "https://github.com/vaeth/eix/"

-L, **--license**

e.g. "GPL-2"

--available-deps, **--available-depend**, **--available-rdepend**, **--available-pdepend**, **--available-bdepend**, **--available-idepend**

This test can only be successful if **DEP=true** is used (and if **DEP=true** was used when the cachefile was created). It matches against **DEPEND**, **RDEPEND**, **PDEPEND**, **BDEPEND**, or **IDEPEND** (or with **--available-deps** any of those) of any available version of the package. **--available-deps** is the same as specifying all of **--available-depend --available-rdepend --available-pdepend --available-bdepend --available-idepend**. Note that it is matched against the string without any interpretation or modification of the latter; only word separators become single spaces. In particular, the string to match against can look like

app-admin/fam nls? (sys-devel/gettext) =dev-libs/apr-1* !!dev-db/libpq

Therefore, the match is not only against dependent packages but also against blockers and/or conditionals and various ways of specifying versions.

--installed-deps, **--installed-depend**, **--installed-rdepend**, **--installed-pdepend**, **--installed-bdepend**, **--installed-idepend**

This is similar to the corresponding option --available-* but with the difference that the corresponding dependency string of any installed versions of the package is matched. In contrast to --available-* this check does not involve the setting of **DEP**. You might probably want to combine these options with **--deps-installed** to make sure that indeed the content of the installed version is

read.

Note that dependencies of installed versions are usually much simpler, e.g. not containing conditionals.

--deps, **--depend**, **--rdepend**, **--pdepend**, **--bdepend**, **--idepend**
These are shortcuts for specifying the corresponding --available-* and --installed-* options simultaneously.

--set Name of a local package set of a version in the database (i.e. corresponding to a file in **/etc/portage/sets**, **/etc/portage/sets.eix**, or from another directory specified in **EIX_LOCAL_SETS_ADD**; see the comments to **EIX_LOCAL_SETS**). The "profile", "system" and "world" sets are intentionally excluded here, since these should be tested with **--profile[+-]**, **--system[+-]**, **--world**, **--world-all**, or **--world-sets**.

--src-uri
SRC_URI of a version in the database

--eapi EAPI of a version in the database, e.g. "6"

--installed-eapi
EAPI of an installed version

--slot Slotname of a version in the database, e.g. "kde-4"

--fullslot
Slotname of a version in the database, possibly with its subslot, e.g. "3/1.4"

--installed-slot
Slotname of an installed version. Recall that without option**--car e** (or **CAREMODE=true**) the slotname might be guessed.

--installed-fullslot
Slotname of an installed version, possibly with its subslot. Recall that without option **--care** (or **CAREMODE=true**) the slotname might be guessed.

-U, **--use**
A useflag defined by IUSE in some version by some of the ebuilds of the package. You will usually want to combine this option with -e

--installed-with-use
A useflag enabled during installation of the package. Of course, this flag can only match if the package is installed. Note that the same restrictions hold as for **-I**, i.e. only packages will be matched which are still in the database.

--installed-without-use
A useflag disabled during installation of the package. Of course, this flag can only match if the package is installed. Note that the same restrictions hold as for **-I**, i.e. only packages will be matched which are still in the database.

Match algorithm

The subsequent options define the algorithm by which the match field should be tested against the expression. Only one algorithm can be chosen for an expression. If you do not specify some of these options, the default is chosen according to some heuristic depending on the form of your search pattern and according to the selected match field. In most cases, the default will be **--regex** (or **--regex-case** if your expression contains capital letters) unless your expression "looks" like a glob pattern or a substring (in which case the corresponding algorithm will be the default), or if the selected match field refers only to USE-flags, sets, EAPI, or SLOT which perhaps most people would expect to match the whole string. Details of the heuristic are specified by the **DEFAULT_MATCH_ALGORITHM** configuration variable explained later. Depending on the configuration of that variable, it might even be non-optional to specify some of the following options for every query.

-e, **--exact**
Pattern is an exact (full) string. For example, "eix -e gcc" will only show packages with the name "gcc".

-b, **--begin**
Pattern is the beginning of the string. For example, "eix -b gcc" will show the package with the name "gcc" but also e.g. "gcc-config".

--end Pattern is the end of the string.

-z, **--substring**
Pattern occurs somewhere within the string.

-f [*N*], **--fuzzy** [*N*]
Do a fuzzy search with a maximal levenshtein-distance of *N* (default) for the full string. Note that this command slows down search speed.

-p, **--pattern**
pattern is a wildcard-pattern (for the full string). See **fnmatch**(3) and/or **glob**(7) for further information. Be sure to use single quotes around patterns (to prevent the shell from intercepting any wildcards).

-r, **--regex**
pattern is a regexp, ignoring case. Only a substring must be matched (unless ^ or $ are used); the empty pattern matches everything. For further information, please read **regex**(7). Again, be sure to use single quotes around patterns.

--regex-case
As **--regex**, but does not ignore case.

Defining layouts (see **FORMATSTRING** below)

--format *FORMATSTRING*
Defines the *FORMATSTRING*. Since eix-0.29.6, all other possibilities to modify the format string like the setting of **DEFAULT_FORMAT** or **FORMAT** are overridden by this option. In contrast, if this option is not used, whether the **FORMAT**, **FORMAT_VERBOSE** or **FORMAT_COMPACT** variable is used depends on the setting of **DEFAULT_FORMAT** and/or whether the options **--verbose**, **--compact**, or **--normal** are used.

Special options for eix-update

-H, **--nostatus**
Do not update the status line.

--force-status
Update the status line even if output is not a terminal or if TERM does not begin with a TERM_STATUSLINE word.

-o *outputfile*, **--output** *outputfile*
With this option, **eix-update** will write the eix database to *outputfile* instead of **/var/cache/eix/portage.eix**. When this option is used, the current**umask** will be honoured: other - wise, the **umask** is forced to **002** for creating the file.

-a *overlay*, **--add-overlay** *overlay*
This is similar to adding *overlay* to **PORTDIR_OVERLAY** in /etc/portage/make.conf or to **ADD_OVERLAY** but has the advantage that you need not modify some of those, and you can also use spaces in *overlay*. Overlays added by this option come after overlays added by **KEEP_VIRTUALS**. If*o verlay* is already contained in the list of overlays, this option has no effect. It is explicitly admissible to use this option repeatedly to add several overlays.

-x *overlay*, **--exclude-overlay** *overlay*
This is similar to adding *overlay* to **EXCLUDE_OVERLAY** but has the advantage that you need not modify the latter, and you can also use spaces in *overlay*. *overlay* is considered as a mask. All

matching overlays (even those added by later **--add-overlay** options) are excluded from the list of overlays. The**POR TDIR** directory is considered as any other overlay which can be excluded (in this case, the first *overlay* in the list will be stored as **PORTDIR**). It is explicitly admissible to use this option repeatedly to exclude several overlays.

-m *overlay method*, **--override-method** *overlay method*
Change the cache method of *overlay* (the **PORTDIR** directory is an allowed *overlay*) to *method*. *overlay* is considered as a mask, i.e. it may contain wildcards. If *overlay* does not match anything in the list of overlays, this option has no effect. This option is similar to adding the entry "*overlay method*" at the end of the **OVERRIDE_CACHE_METHOD** variable. It is explicitly admissible to use this option repeatedly to override cache methods for several overlays. The last matching override takes precedence. In particular, the option overrides the content of **OVERRIDE_CACHE_METHOD**.

-r *overlay-path overlay-label*, **--repo-name** *overlay-path overlay-label*
The overlay *overlay-path* obtains the label *overlay-label*, independent of any other settings. This may be overridden by **REPO_NAMES**. In contrast to**REPO_N AMES**, *overlay-path* is not a pattern but the exact path.

-v --verbose
Output the effectively used cache method for each ebuild. This produces a lot of output and is mainly useful for debugging if you are wondering why eix-update is faster/slower than expected.

OUTPUT

Slots

In contrast to usual output of versions in emerge, **eix** can also print slot names if they are nonempty and different from "0". Whether this happens is determined by the content of the **FORMATSTRING**.

If slots are printed, the slot name is separated from the version number either with a colon, or the slot names are written in braces. You can choose the preferred modes with the **COLON_SLOTS** variable.

If **PROPERTIES** or **RESTRICT** are set in the ebuild, this is shown by default in the version string; details can be influenced by setting of configuration variables.

Some Examples:

4.1.1:4.1 or **4.1.1(4.1)**
This is version 4.1.1 which will be installed into the slot "4.1".

3.14p:GNAT-3.14p or **3.14p(GNAT-3.14p)**
This is version 3.14p which will be installed into the slot "GNAT-3.14p".

2.0.0_rc1-r6
This is version 2.0.0_rc1-r6, and SLOT is either empty or "0".

1.0*ilvs^fmpbstuidP{tbz2,gpkg:3,pak:2}
This is version 1.0 which is subject to PROPERTIES="interactive live virtual set" as well as RESTRICT="fetch mirror primaryuri binchecks strip test userpriv installsources bindist parallel" Moreover, a *.tbz2, three *.gpkg.tar, and two *.xpak files (binary packages without or with FEATURES=binpkg-multi-instance or BINPKG_FORMAT=gpkg, respectively) exist for that version in **PKGDIR**.

5.0-r3(5.0R3)^f or 5.0-r3:5.0R3^f
This is version 5.0-r3 which will be installed into SLOT 5.0R3 and has fetch restrictions.

Masking

If you used gentoo for more than a week you're probably going to immediately recognize the format of the masking in the version-strings. Nevertheless, we are going to explain it here by some examples. We

describe here of course only the default setting; details can be influenced by setting of configuration variables.

[P]2.95.3-r8
If a mask for the package was found in the packages-files from your profile, but this version does not match it, the version is determined to be "masked by profile".

[M]4.0.0_alpha20050213
The version matches a mask from /etc/portage/package.mask, $PORTDIR/profiles/package.mask or a package.mask from your profile. Portage calls this "masked by package.mask".

[m]4.1.4
The version matches a local mask (from /etc/portage/package.mask), but it is neither "masked by profile" nor masked in $PORTDIR/profiles/package.mask.

{P}2.95.3-r8
The version was originally "masked by profile", but this was locally changed in /etc/portage/profile/packages.

{M}4.0.0_alpha20050213
The version was originally masked in $PORTDIR/profiles/package.mask, but this was locally changed in /etc/portage/package.unmask.

***3.3.3** This means the version is "masked by missing keyword" but stable for an alien architecture.

~*3.3.3 This version is "masked by missing keyword", stable on no architecture, but unstable on an alien architecture.

****3.3.3** This means the version is "masked by missing keyword" for all architectures.

(**)3.4.3-r2
That version originally had no keyword, but this was locally changed (in /etc/portage/package.{accept_,}keywords or by ACCEPT_KEYWORDS).

-0.8.14 Masked by -ARCH.

~3.3.5.20050130
The version would be "masked by ~keyword".

(~)3.3.5.20050130
The version was originally "masked by ~keyword", but this was locally changed (in /etc/portage/package.{accept_,}keywords or by ACCEPT_KEYWORDS).

[M]~1.0.9626
The version is both, "masked by package.mask" and "masked by ~keyword".

[m](~)4.1.4-r1
The version was originally only "masked by ~keyword", but this was locally changed (in /etc/portage/package.{accept_,}keywords or by ACCEPT_KEYWORDS). However, the version is locally masked (in /etc/portage/package.mask).

3.3.1 Finally, this would be a stable version (which would be stable also without the local settings).

eix-diff

The output of **eix-diff** is completely determined by configuration variables (**DIFF_FORMAT_NEW**, **DIFF_FORMAT_DELETE**, **DIFF_FORMAT_CHANGED** and a lot of variables which - at least in their default setting - is used by them via delayed substitution, see below). The following explanation by means of examples therefore can only describe the behavior in the default setting of the current eix version. Although this default format was not changed for quite a while, no stability of the defaults is guaranteed for future eix versions.

[N] >> foo/bar (~1.0): description of foo/bar
The package **foo/bar** has freshly appeared in the tree.

[*N] >> foo/bar (1.0): description of foo/bar
The package **foo/bar** has freshly appeared in the tree. Moreover, it has a version (1.0) which could be installed without any sort of unmasking/keywording.

<< foo/bar ({M}1.0): description of foo/bar
foo/bar was removed from the tree; the previous version **1.0** was masked, but it is currently not masked anymore (probably, because the developer has cleaned up the package.mask file with the removal of the package).

[*>] == foo/bar (1.0): description of foo/bar
The status of package **foo/bar** has changed in the tree (for your settings): It has gained a version (1.0) which could be installed without any sort of unmasking/keywords while in the previous tree foo/bar had no such version. Moreover, the > means that one slot has gained a higher version. In this case, it is actually the same change which is responsible for both symbols **>** and *****.

[><] == foo/bar (1.1(1) 2.0(2) -> 1.0(1) 2.1(2)): description
The status of package **foo/bar** has changed in the tree (for your settings): The symbols on the left mean that one slot has gained a higher version which could be installed without modifying masks/keywords, another slot had such a highest version removed. If you consider the version strings, it becomes clear that slot **2** has obtained a new version (the previous stable in this slot was **2.0**, the new one is **2.1**), and that the previously highest stable version **1.1** of slot **1** was removed or masked (the current stable version in this slot is now **1.0**).

[U?] == foo/bar (1.1(1)@01.01.2009; 1.1(1) -> 2.0(2)): description
The status of package **foo/bar** has changed in the tree (for your settings): The symbols on the left mean that a slot you have installed can be upgraded (without modifying masks/keywords), and that another slot you have installed was removed/masked. Looking at the versions on the right, it becomes clear that the installed version **1.1** of Slot **1** has been removed or masked, and that there is no other installed version of slot **1**. However, a new stable version **2.0** in slot **2** has appeared (i.e. slot **2** did previously not exist or had no stable version).

Since no version of slot **2** was installed yet, eix cannot decide in this situation whether the symbol "**U**" is really appropriate: Since eix does no dependency tracking, it does not know whether the new slot would be pulled in e.g. by your world file, or whether there is only some dependency to your old slot. Therefore, the symbol "**U**" is only shown in this situation if **UPGRADE_TO_HIGHEST_SLOT=true** or if the package is listed in **/etc/portage/package.slot_upgrade_allow**.

Actually, the output **[U?><] == foo/bar ...** w ould be more consistent, because in addition one slot has gained a higher stable version and the highest stable version of another slot was removed, but since typically this is implicitly implied by "**U**" or "**?**", respectively, it was a design decision of the default setting that if **U** or **?** is output then the symbols < and > will not be output. You can of course build a different **DIFF_FORMAT_HEADER_CHANGED** string which follows another policy.

FORMATSTRING

A formatstring can contain conditional blocks, package properties, colors and normal strings. The normal strings are output as expected (with special escape sequences being recognized, see below); for size calculations for e.g. tabs, UTF8 encoding is supposed.

Conditional blocks

Conditions are very simple: A property is expanded and the resulting string is tested against another string. If they are the same, the condition is true and the block is executed. Conditions can be negated so that the else-part is executed if the condition is true, and the if-part if the condition is false. The else-part can be also be completely left out.

{[**!**]*PROPERTY*[**=***RHS*]**}***TCODE***{}**

Execute *TCODE* if the string resulting from expanding *PROPERTY* is equal to *RHS*. The**!** w ould negate the behaviour.

{[**!**]*PROPERTY*[**=***RHS*]**}***TCODE***{else}***FCODE***{}**

Execute *TCODE* if the string resulting from expanding *PROPERTY* is equal to *STRING*. If it's not, execute *FCODE*.

RHS is either a property (if enclosed in <>) or a variable (if prefixed with $) or a string (otherwise or if quoted).

PROPERTY can be either one of the Package properties specified below, or it can be a variable access. A variable access has the form **$***VARIABLE*. *VARIABLE* need not be initialized; its content defaults to the empty string.

To change *VARIABLE* during runtime use this syntax:

{[**!**]*****VARIABLE*[**=***RHS*]**}**

This sets the runtime variable **VARIABLE** to *RHS*. With **!**, the result is either 1 or empty, depending on whether *RHS* is nonmpty. If the trailing part (including the**=** sign) is omitted, there is a special meaning: **{***VARIABLE***}** sets *VARIABLE* to 1, **{!***VARIABLE***}** sets *VARIABLE* to the empty string.

Package properties

Names that refer to specific properties of the package that is currently printed. If used to print a property, the name **must be enclosed in square brackets** (i.e. "<name>").

name, **category**, **homepage**, **licenses**

The name, category, homepage and licenses for the current package.

availableversions:*VARIABLE*, **availableversions:***VARIABLE*:*VARSLOTS*

For each version, print the content of the configuration/environment variable with name *VARIABLE*, interpreting it as a format string. If the second form is used and at least one slot of the package is nontrivial, then *VARSLOTS* is used instead of *VARIABLE*, and the versions are sorted according to slots.

To avoid a misunderstanding: It is not possible to enter the required format directly after the colon. Instead, the required format must be stored in a new variable, and *VARIABLE* and *VARSLOTS* are only the names of these variables.

Useful examples for *VARIABLE* are **NAMEVERSION**, **EQNAMEVERSION**, **EQNAMEVERSION**, **ANAMESLOT**, **ANAMEASLOT**, **NAMESLOT**, **NAMEASLOT** or **DATESORT**. Here, **ANAMESLOT** and **ANAMEASLOT** are meant to be used in the second form, i.e. in **availableversions:ANAMESLOT:ANAMESLOT** or **availableversions:ANAMEASLOT:ANAMEASLOT** (Mnemonic: **A**SLOT prints the slot **a**lways). Moreover, **NAMESLOT**, **NAMEASLOT** and **DATESORT** makes sense only for installed versions. See**eix --dump** to see the variables in detail.

markedversions:*VARIABLE*, **markedversions:***VARIABLE*:*VARSLOTS*

This is analogous to **availableversions** with the difference that only marked (and available) versions are printed.

bestversion:*VARIABLE*, **bestversion*:***VARIABLE*, **bestslotversions:***VARIABLE*, **bestslotversions*:***VARIABLE*, **bestslotupgradeversions:***VARIABLE*, **bestslotupgradeversions*:***VARIABLE*

This is analogous to **availableversions** with the difference that only the best version resp. the best versions of each slot are printed. For the variants with ***** also unstable versions are accepted. For the variants with **upgrade** only those versions are selected which probably will appear after the upgrade.

installedversions:*VARIABLE*
This is analogous to **availableversions** with the difference that only installed versions are printed.

first, **last**, **slotfirst**, **slotlast**, **oneslot**
This must occur only in *VARIABLE* in the context of version printing. You can use these flags to test whether you are currently printing the first or last version of the package (usually to output some additional text in these cases). Similarly, when the printing is sorted according to slots you can test whether the current version is the first or last of the current slot or if there is only one slot at all. All these properties are empty if the condition is not satisfied, otherwise 1. To ease reusing code, when printing is not sorted according to slots, then **slotfirst**/**slotlast** is equivalent to **first**/**last** and **oneslot** is 1.

srcuri, **havesrcuri**
This must occur only in *VARIABLE* in the context of version printing. It outputs the **SRC_URI** of an available version or 1, if **SRC_URI** is nonempty, respectively.

eapi This must occur only in *VARIABLE* in the context of version printing. It outputs the **EAPI** of an available or installed version.

slot, **isslot**, **fullslot**, **isfullslot**, **subslot**, **issubslot**
This must occur only in *VARIABLE* in the context of version printing. It outputs the slot/subslot/both of the current version. **isslot**, **isfullslot**, **issubslot** just test whether the slot, fullslot, or subslot are trivial and outputs 1 in this case, otherwise nothing.

overlayver, **overlaynum**, **overlayvername**, **virtual**
This must occur only in *VARIABLE* in the context of version printing. It outputs the overlay of the current version. **overlayver** is the colored index of the overlay and empty if the whole package is from only one overlay; you should use **overlaykey** to print the overlay in such a case. **o verlaynum**, in contrast, contains the number of the overlay without colors unconditionally (and is empty only if the version is not from an overlay). **overlayplainname** is the label (name) (or path if the label is unknown or empty) of the overlay; **overlayplainname*** considers also the main repository as an overlay, here. **o verlayvername** and **overlayvername*** are analogous, but return an empty string if the whole package is from the same respository. **virtual** is nonempty if and only if the version is from a virtual overlay.

versionkeywords, **versionkeywords***, **versionekeywords**
This must occur only in *VARIABLE* in the context of version printing. It outputs the full keywords of the current version. If the data is modified by the profile, the modified data is printed with**v ersionkeywords***. The special form **versionekeywords** prints the latter if it differs from **versionkeywords**; otherwise, it is empty.

isbestupgrade, **isbestupgrade***, **isbestupgradeslot**, **isbestupgradeslot***
This must occur only in *VARIABLE* in the context of version printing. It returns 1 if the current version is the best resp. best in the current slot for upgrading. For the variants with * also unstable versions are considered.

installedversion, **markedversion**
This must occur only in *VARIABLE* in the context of version printing. It returns 1 or the empty string depending on whether the current available version is installed or marked, respectively. The test for **markedversions** always fails in the context of <installedversions:...>.

ishardmasked, **washardmasked**, **isprofilemasked**, **wasprofilemasked**, **ismasked**, **wasmasked**, **isstable**, **wasstable**, **isunstable**, **wasunstable**, **isalienstable**, **wasalienstable**, **isalienunstable**, **wasalienunstable**, **ismissingkeyword**, **wasmissingkeyword**, **isminuskeyword**, **wasminuskeyword**, **isminusunstable**, **wasminusunstable**, **isminusasterisk**, **wasminusasterisk**
This must occur only in *VARIABLE* in the context of version printing. This returns 1 or empty depending whether the version has the current stability property according to the local or default configuration, respectively.

isbinary
This must occur only in *VARIABLE* in the context of version printing. It returns 1 or empty depending on whether there is a corresponding *.tbz2, *gpkg.tar, or *.xpak file for the version.

istbz This must occur only in *VARIABLE* in the context of version printing. It returns 1 or empty depending on whether there is a corresponding *.tbz2 file for the version.

isgpkg This must occur only in *VARIABLE* in the context of version printing. It returns 1 or empty depending on whether there is at least one corresponding *.gpkg.tar file for the version.

ispak This must occur only in *VARIABLE* in the context of version printing. It returns 1 or empty depending on whether there is at least one corresponding *.xpak file for the version.

isgpkgmulti
This must occur only in *VARIABLE* in the context of version printing. It returns 1 or empty depending on whether there are at least two corresponding *.gpkg.tar files for the version.

ismultipak
This must occur only in *VARIABLE* in the context of version printing. It returns 1 or empty depending on whether there are at least two corresponding *.xpak files for the version.

gpkgcount
This must occur only in *VARIABLE* in the context of version printing. It returns the number of *.gpkg.tar files for the version or empty if there are none.

pakcount
This must occur only in *VARIABLE* in the context of version printing. It returns the number of *.xpak files for the version or empty if there are none.

restrict, **restrictfetch**, **restrictmirror**, **restrictprimaryuri**, **restrictbincheck**, **restrictstrip**, **restricttest**, **restrictuserpriv**, **restrictinstalledsources**, **restrictbindist**, **restrictparallel**
This must occur only in *VARIABLE* in the context of version printing. It returns 1 or empty depending on whether some or the corresponding **RESTRICT** attribute is set for the version.

properties, **propertiesinteractive**, **propertieslive**, **propertiesvirtual**, **propertiesset**
This must occur only in *VARIABLE* in the context of version printing. It returns 1 or empty depending on whether some or the corresponding **PROPERTIES** attribute is set for the version.

depend, **rdepend**, **pdepend**, **bdepend**, **depend***, **rdepend***, **pdepend***, **bdepend***, **idepend***
This must occur only in *VARIABLE* in the context of version printing. It returns the respective **DEPEND**, **RDEPEND**, **PDEPEND**, **BDEPEND**, **IDEPEND** string of an available package (the string is shortened if possible if the variant with ***** is used). This is empty unless **DEP=true** is set (and was set when the cachefile was created).

havedepend, **haverdepend**, **havepdepend**, **havebdepend**, **haveidepend**, **havedeps**
This is like the above with the difference that only 1 is returned if the value of the corresponding string (resp. of at least one of these) is nonempty.

havemaskreasons, **maskreasons**, **maskreasons***
This must occur only in *VARIABLE* in the context of version printing. For available versions, **maskreasons** outputs the mask reasons and **havemaskreasons** whether there are some. Separators used are **FORMAT_MASKREASONS_LINESKIP** and **FORMAT_MASKREASONS_SEP**; for **maskreasons*** separators used are <FORMAT_MASKREASONSS_LINESKIP> and **FORMAT_MASKREASONSS_SEP**.

haveuse, **use**, **use***, **use0**
This must occur only in *VARIABLE* in the context of version printing. For available versions, **use** outputs the **IUSE** variable. For installed versions, **use** outputs the **USE** flags and whether they are set (in the later case, the content of **FORMAT_BEFORE_SET_USE**, **FORMAT_AFTER_SET_USE**, **FORMAT_BEFORE_UNSET_USE**, **FORMAT_AFTER_UNSET_USE** is printed at the appropriate places). The **use*** variant treats **USE_EXPAND** variables separately and outputs them, using the content of **FORMAT_BEFORE_USE_EXPAND_START**,

FORMAT_BEFORE_USE_EXPAND_END, **FORMAT_AFTER_USE_EXPAND**, **FORMAT_BEFORE_IUSE_EXPAND_START**, **FORMAT_BEFORE_IUSE_EXPAND_END** and **FORMAT_AFTER_IUSE_EXPAND** at the appropriate places. The **use0** variant suppresses the output of **USE_EXPAND** variables. **haveuse** can be used to test whether an output would be non-empty (in this case it is 1, otherwise empty).

requireduse, **haverequireduse**
This must occur only in *VARIABLE* in the context of version printing. For available versions, it outputs the **REQUIRED_USE** variable, or **1** if that variable is not empty, respectively. This is empty unless **REQUIRED_USE** is set (and was set when the cachefile was created).

date:*VAR*
This must occur only in *VARIABLE* in the context of version printing. It outputs the installation date; the strftime() format for the date is read from the variable *VAR*.

version,plainversion,**revision**
This must occur only in *VARIABLE* in the context of version printing. It outputs the version (man 5 ebuild: $PVR) in text form. plainversion outputs the version without revision ($PV), and revision only the revision ($PR). In contrast to portage, revision is empty if there was no revision specified.

installed, **best**, **best***
Is 1 or empty, depending on whether at least one version of the package is installed or has a best stable or best unstable version.

versionlines
Is 1 or empty, depending on whether the --versionlines flag was passed.

slotsorted
Is 1 or empty, depending on whether the --versionsort flag was passed.

color Is 1 or empty, depending on whether colors/markers are printed. For instance if the output is redirected to a terminal and no opposite option is used, this will be empty.

setnames, **allsetnames**
The name of all local sets to which the package belongs, separated by spaces. With **allsetnames** also the "system" and "profile" set names are included.

binary Is 1 or empty, depending on whether at least one version (available or installed) has a corresponding *.tbz2, *.gpkg.tar, or *.xpak file. See the remarks for **--binary**.

mainrepo
Is 1 or empty, depending on whether at least one available version belongs to the main repository 0 (typically: gentoo).

overlaykey, **overlayname**
If all versions are in the same overlay, "[overlaykey]" (in corresponding colors) or "overlaylabel" is output.

havevirtual, **havenonvirtual**
Is 1 or empty, depending on whether at least one available version is from a (non)virtual overlay.

system expands to 1 if the package is in @system

profile expands to 1 if the package is in @profile

world expands to 1 if the package is in the world file

world_sets
expands to 1 if the package is in the world sets

marked
expands to 1 if the package name was passed with the **--pipe** option (no matter whether a version matches or not, see also **havemarkedversion**). This is mainly only useful in tests.

havemarkedversion
Is 1 or empty, depending on whether at least one available version of the package is marked. Note that it is possible that a package is marked although none of its available versions is marked.

slots, **slotted**
expands to 1 if there are at least two slots resp. at least one nontrivial slot.

colliuse, **colliuse***, **colliuse0**, **havecolliuse**
The collected IUSE flags (i.e. their union) of all available versions of the package, separated by spaces. The**colliuse*** v ariant outputs USE_EXPAND variables separately, using **FORMAT_BEFORE_COLL_EXPAND_START**, **FORMAT_BEFORE_COLL_EXPAND_END** and **FORMAT_AFTER_COLL_EXPAND** at the appropriate places. The **colliuse0** variant omits the output of USE_EXPAND variables. havecolliuse expands to 1 if colliuse is nonempty; this is faster than colliuse.

havebest, **havebest***
Expands to 1 if the package has a best stable/unstable version.

upgrade, **upgradeorinstall**, **downgrade**, **recommend**, **recommendorinstall**
upgrade expands to 1 if the package is installed and at least one slot can be upgraded (or the best stable version is a new slot and **UPGRADE_TO_HIGHEST_SLOT** is true). The other variables test similarly whether the package can be upgraded or newly installed, should be downgraded, can/should be upgraded/downgraded, or can/should be upgraded/downgraded/installed, respectively. The variable **RECOMMEND_LOCAL_MODE** determines whether these tests obey **LOCAL_PORTAGE_CONFIG**.

bestupgrade, **bestupgradeorinstall**, **bestdowngrade**, **bestrecommend**, **bestrecommendorinstall**
as above with the difference that only the best stable version of the package is taken into account (and not all slots).

better, **worse**, **differ**, **bestbetter**, **bestworse**, **bestdiffer**
this can only be used in conditionals in **DIFF_FORMAT_CHANGED**. **better** expands to 1 if the new package has a new slot or a better stable version (or the same best stable version but from a different overlay number) for some slot than the old package. **w orse** means analogously that the old package had at least one better slot or a slot which is not available in the new package. **differ** means that not all best stable slots of the old and new package coincide. The corresponding **best*** versions have an analogous meaning with the difference that only the best stable version is taken into account (and not all slots). The variable **RECOMMEND_LOCAL_MODE** determines whether these tests obey **LOCAL_PORTAGE_CONFIG**.

old..., **new...**
this can only be used in **DIFF_FORMAT_CHANGED**. You can prefix any property name with **old** and **new**, and the value corresponds to that property, taking the old resp. new data into account. If neither**old** nor **new** is specified, the ne w version is assumed. For example, **oldavailableversions:VAR** will output the previous available versions (using the variable **VAR** to determine the output format) while **newavailableversions:VAR** and **availableversions:VAR** both will print the available versions of the current (i.e. new) package.

Escape sequences

\n \r \t \b \a
These are interpreted in the usual way as in printf. Within **FORMATSTRING**, tabs are calculated into spaces, assuming that the **FORMATSTRING** is in UTF8.

\C<*COLUMN*>
This special escape sequence is only admissible in **FORMATSTRING**. Here,*COLUMN* is a number (which must be enclosed in the braces). During output, this is interpreted by eix in such a way that if the column *COLUMN* is not yet reached during output of the current line, so many spaces are output that this is the case. Thus, in a sense, this can be considered as a powerful

tabulator which does not leave the interpretation to the terminal. For the calculation of the length of the current line, it is assumed that the **FORMATSTRING** is in UTF8.

Colors

(*COLORSTRING*|*COLORSTRING*|...**)**

The effective *COLORSTRING* chosen depends on **COLORSCHEME?**. It should have the following form:

*MARKER_OR_COLOR***;***MARKER_OR_COLOR***;**...

MARKER_OR_COLOR is either a marker attribute or a color name, optionally followed by **,1** (or **,0**) to specify that bright color should (not) be used.

Available marker attributes are: *none bold italic underline blink inverse*

Available colors are: *default black red green yellow blue purple cyan gray 0 ... 255*

Note that *gray* does not work as required on all terminals and that *0 ... 255* are interpreted usefully only on 88/256 color terminals.

The first color in *COLORSTRING* specifies the foreground color, the second the background color.

If a foreground color is specified, all colors and attributes are reset. The empty string is defined as the color "default". In particular, **()** means that all colors and attributes are reset.

If you want that nothing is actually output or reset, specify the attribute **(none)**.

Examples:

eix --format '{installed}(yellow,1;underline){else}(yellow,0){}<name>()\n' ...

If the package **...** is installed, print the name underlined in bright yellow, else in normal yellow.

INSTFORMAT='{first}:{}<version> <date:DATEFORMAT>{!last}\n\t{}' DATEFORMAT='%x' eix --format '<category>/<name><installedversions:INSTFORMAT>\n' 'autom*'

This will print for every **autom*** package the category name, and if it is installed, it will also print the installed versions and the installation day. One could have obtained the same result by omitting **{!last}\t\n{}** and writing instead **{first}:{else}\t\n{}**, since it is of course the same whether **\t\n** is printed at the end of each version except the last or at the beginning of each version except the first.

FORMAT='{downgrade}%{FORMAT_ALL}{}' eix -I

This will print all installed packages for which there are downgrade recommendations. Note that **FORMAT='{downgrade}%{FORMAT}{}'** does not work, because this would be a self-referential definition in the delayed substitution: Of course, a variable cannot be defined by an expansion of itself. For this reason, the content of **FORMAT** and similar variables has become very simple since eix-0.13.4: It is just **%{FORMAT_ALL}** (and similarly for other variables). So one can easily insert the full definition of the original value of **FORMAT** (which we have done in the above example).

Note that if you want to have compact output in the above example, you cannot just append the option **-c**, because appending this option just has the effect that for the format string the variable **FORMAT_COMPACT** is used instead of **FORMAT**. So, to have compact output, you should use either

FORMAT_COMPACT='{downgrade}%{FORMAT_ALL_COMPACT}{}' eix -Ic

or even simpler and with the same effect:

FORMAT='{downgrade}%{FORMAT_ALL_COMPACT}{}' eix -I

A much simpler possibility to avoid the self-reference is in the above example usage of the **--format** option:

eix --format '{downgrade}%{FORMAT}{}' -I

or, for compact output,

eix --format '{downgrade}%{FORMAT_COMPACT}{}' -I

respectively.

FILES

/etc/eix-sync.conf

This file stores commands and configurations to apply with **eix-sync**. Comment lines start with**#**. Lines can have the following forms and will be executed in the given order before **emerge --sync** is executed.

Note that the content of the variable **EIX_SYNC_CONF** is appended to that file. By default, the latter expands to **EIX_SYNC_OPTS** so that also that variable is appended to that file. Both variables can be used to modify the defaults set in */etc/eix-sync.conf*. Be aware that a lot code of these lines is evaluated when **eix-sync** is executed, so take care about security!

option(s)
: Use *option(s)* as default for eix-sync (in front of all other options). As usual, *option(s)* must start with "**-**". *option(s)* will be evaluated by the shell in the **eix-sync** script, so take care about security and be sure to quote shell command separators properly!

Name
: Call **layman -s** *Name*. Note that**layman** is pro vided by **app-portage/layman** to sync overlays.

*****
: Call **layman -S** (i.e. all overlays are synced with layman).

!*command*
: Evaluate *command* within the **eix-sync** shell script (in the same shell). *command* must exit successfully or **eix-sync** will print a warning. (Thus,**!layman** *Name* is more or less equi valent to *Name*.) Close e.g. the command with "; true" if you want that the exit status is ignored. The warning becomes even a fatal error with the option **-F**. You can set **FATAL_HOOKS** to a non-empty/empty value to force/cancel **-F** for the currently and subsequently executed hooks. If you want only the current hook to exit with an error, you can call **die** explicitly.

  You can use this feature to unpack e.g. an overlay before calling **layman** or to apply local fixes after **layman** was called. You can also set the variable **have_changed=:** here to tell eix-sync that some overlay has changed. (Otherwise eix will stop with exit code 3 if the main portage tree did not change.)

  In contrast to earlier eix versions, *command* does not output anything (use einfo if you want that), and it is not evaluated within a subshell, i.e. you can also modify environment variables or begin/end redirection as you like. The disadvantage is that you can also easily override internal eix-sync variables or functions by accident; if in doubt, put your command in braces (...) to start it in a subshell.

!!*command*
: This is similar to **!** *command* with the difference that *command* is unconditionally executed, i.e. even if options -d, -u, or -l are used. It is meant to set environment variables for the other programs.

~*command*

This is only important if eix-sync options **-s** or **-2** are used. In this case, *command* will be executed before the first call of **rsync**; the output of *command* is evaluated within the **eix-sync** shell. If *command* or the evaluation of its output are not successful, **eix-sync** will print a warning (or error out if **-F** is active, i.e. if **FATAL_HOOKS** is nonempty). This can be used to execute e.g. **keychain** and to return the content of the appropriate *~/.keychain/*-sh* file or to return export-commands for the current **SSH_AUTH_SOCK** and **SSH_AGENT_PID**. It is also admissible that the output of *command* is a command which modifies the variables **PORTAGE_RSYNC_OPTS**, **PORTAGE_RSYNC_EXTRA_OPTS**, **PORTDIR**, **PORTDIR_SERVER**, **PORTDIR_CLIENT**, **SERVER**, or **CLIENT**. These variables are filled with defaults when *command* is called and will later be used for the **rsync** command(s) with their obvious meaning.

@*command*

Adds a hook for *command*; the actual execution of *command* is postponed until **emerge --sync** was (successfully) called.

@@*command*

Adds a hook for *command*; the actual execution of *command* is postponed until **emerge --sync** and the subsequent **eix-update** was (successfully) called.

Here is a **schematic** overview of the order of hooks/commands:

!! hooks

cp /var/cache/eix/portage.eix /var/cache/eix/previous.eix

layman calls and **!** hooks in order of */etc/eix-sync.conf*

~ hooks

emerge --sync

@ hooks

eix-update

@@ hooks

eix-diff

Some examples of useful lines in */etc/eix-sync.conf*:

-F Always error out on layman/hook errors. This was always the case in earlier eix versions.

-C --ignore-default-opts

Use this line if you have --ask in EMERGE_DEFAULT_OPTS in /etc/portage/make.conf but do not want to get asked for eix-sync.

-r -M Use this if you use e.g. **PORTDIR_CACHE_METHOD=assign**, and **FEATURES=metadata-transfer** is inactive or disabled (the latter is the default in newer portage versions). Depending on your needs you might want to use also **-r** or **-M** alone or replace **-M** by the line

@StatusInfo...**[***statusline***]**

Announce that the next command will be ... (sets hardstatus line and output info for user.)

@Statusline...

As above, but set only hardstatus line.

@einfo...
As above, but output only info to user.

@eix-remote fetch1
Download remote1 data with every call of **eix-sync**.

@emerge --regen
to use **emerge --regen** instead of **emerge --metadata**.

@egencache --repo=local --update
Update the metadata cache of the overlay with repository name "local" before eix-update is called but after calling emerge --sync. This can make sense if you want to use cache method **metadata-md5** or **metadata-flat** for that overlay and want to make sure that this metadata cache is up-to-date. The disadvantage in case of **metadata-flat** is that this **egencache** command is executed even if "local" was not changed (which is usually the case for local repositories).

!egencache --repo=foo --update
Update the metadata cache of the overlay with repository name "foo"; this makes sense if you had earlier a line **foo** or a line **!command to update foo** for updating this repository (by layman or by some other command) and if you want to use cache method **metadata-flat** for that overlay although the metadata cache is broken or not contained in that repository.

!!exec >/var/log/eix-sync.log ; chown portage: /var/log/eix-sync.log || true
Send the output to a logfile (with correct permissions). The "true" at the end is to force continuing even if the "chown" failed. If you do not want to redirect the output of the final **eix-diff**, you might want to combine this with

@@exec >/dev/tty
Output the **eix-diff** stat to the terminal, even if redirection occurred.

@@exit 0
Do not execute **eix-diff** at all.

!!export FORCE_USECOLORS=${FORCE_USECOLORS:-true}
Unless FORCE_USECOLORS is set differently in your environment, eix output will be colored even if you use redirection.

~keychain --quiet ~/.ssh/id_rsa ; cat ~/.keychain/"`hostname`-sh"

/etc/eixrc
Global configuration file. The variables in **~/.eixrc** or from the environment can override the variables set in this file. See **~/.eixrc**.

EIXRC
If this environment variable is set, its value is used instead of the filename */etc/eixrc* to read the configuration data. In this case, the file ~/.eixrc is ignored (but you can of course source it if you want it).

EIX_SYNC_OPTS, EIX_SYNC_CONF, EIX_REMOTE_OPTS, EIX_LAYMAN_OPTS, EIX_TEST_OBSOLETE_OPTS
Although these variables are usually set in ~/.eixrc (and are therefore described in the corresponding section), these variables are pointed out here because they are exceptionally security relevant: They (or at least part of them) are evaluated by the shell if you call the scripts with the corresponding name. Thus, be sure that they are not compromised when you call these scripts with root permissions.

~/.eixrc
Per-user configuration file. The variables in this file can be overridden by environment variables. You can use a shell-like syntax to set the following variables. In particular, you can source other files and you can use auxiliary variables to set other variables.

However, if you use auxiliary variables as usual, you will only see the substituted values with --dump or --dump-defaults and you cannot replace the inserted values e.g. by setting them in the environment.

For this reason, you can refer to variables besides the usual shell-way also in the syntax **%{***VARIABLE***}** (the braces here are not optional). This means that the substitution is delayed until all configuration files and environment variables are read, and the substitution will not show up with --dump or --dump-defaults.

This concept is called delayed substitution/reference, and provides also some additional features:

Special symbols for delayed substitution

%{% This must be used if you need **%{** in a variable text (otherwise delayed substitution would take place).

VARIABLE
If a delayed reference uses a variable name starting with *****, this ***** is replaced by **EIX_**, **DIFF_**, **UPDATE_**, or **DROP_**, depending on whether it is used from **eix**/**eix-diff**/**eix-update**, or **eix-drop-permissions**, respectively. This way you can have different defaults for these programs.

For example, the delayed reference **%{***VARIABLE***}** will insert the expansion of **EIX_***VARIABLE*, **DIFF_***VARIABLE*, **UPDATE_***VARIABLE*, or **DROP_VARIABLE**, respectively.

The \ and ***** attributes can be combined (order plays no role), and you can also combine it with an *APPENDIX*.

As a practical example, if you want that **EIX_USER** has the value "nobody" when called from **eix-drop-permissions** (e.g. from **eix-remote**) but the default value "portage" otherwise, you can set:

EIX_USER=%{*EIX_USER}

EIX_EIX_USER=portage

DIFF_EIX_USER=portage

UPDATE_EIX_USER=portage

DROP_EIX_USER=nobody

VARIABLE
If a delayed reference uses a variable name starting with \, all \ or [\n\r\t] symbols of the (expanded) *VARIABLE* content are considered as escaped.

In particular, the delayed reference **%{***VARIABLE***}** can be used in string list variables like **CACHE_METHOD** or **EIX_LOCAL_SETS**, ensuring that *VARIABLE* produces at most one entry "as is", even if it should contain spaces or \ characters.

The \ and ***** attributes can be combined (order plays no role), and you can also combine it with an *APPENDIX*.

***VARIABLE** *APPENDIX*

This is only recognized if *APPENDIX* starts with one of the symbols **,** or **;**. In this case, **APPENDIX** is appended to the content of **VARIABLE** and in addition in front of all **|** symbols within that content. (The **APPENDIX** can be combined with the \ and ***** attributes.) **APPENDIX** is taken as-is, and it must not contain the symbol **}**. The purpose of **APPENDIX** is to make writing colors simpler. Example:

COLOR_DEFAULT=blue|27

FORMAT="(%{COLOR_DEFAULT,1)A (%{COLOR_DEFAULT;invert})B"

BAD_FORMAT=(%{COLOR_DEFAULT},1;invert)A

In this example, **FORMAT** will expand to

(blue,1|27,1)A (blue;invert|27;invert)B

while **BAD_FORMAT** will expand to

(blue|27,1;invert)A

The latter is probably not what was intended if **COLORSCHEME?=0**.

Conditional blocks in delayed references

If you want to substitute several variables completely differently, depending on the state of a (Boolean) variable, you can use conditional blocks.

This is somewhat analogous to Conditional Blocks in FORMATSTRING: If the referenced variable expands finally to the Boolean value true (**true**/**1**/**yes**/**y**/**on**) (resp. to something nonempty if you preceed *VARIABLE* with an additional **?**) the result is true and the corresponding block of the string is expanded. Conditions can be negated so that the else-part is expanded if the condition is true, and the if-part if the condition is false. The else-part can be also be completely left out. The special variable names ******VARIABLE* (instead of *VARIABLE*) can also be used in these conditionals.

%{?*VARIABLE***}***TCODE***%{}**

Expand *TCODE* if *VARIABLE* expands to true.

%{??*VARIABLE***}***TCODE***%{}**

Expand *TCODE* if *VARIABLE* expands to a nonempty string.

%{!*VARIABLE***}***TCODE***%{}**

Expand *TCODE* if *VARIABLE* expands to false.

%{!?*VARIABLE***}***TCODE***%{}**

Expand *TCODE* if *VARIABLE* expands to the empty string.

%{?*VARIABLE***}***TCODE***%{else}***FCODE***%{}**

Expand *TCODE* if *VARIABLE* expands to true, otherwise *FCODE*.

%{??*VARIABLE***}***TCODE***%{else}***FCODE***%{}**

Expand *TCODE* if *VARIABLE* expands to a nonempty string, otherwise *FCODE*.

%{!*VARIABLE***}***TCODE***%{else}***FCODE***%{}**
Expand *TCODE* if *VARIABLE* expands to false, otherwise *FCODE*.

%{!?*VARIABLE***}***TCODE***%{else}***FCODE***%{}**
Expand *TCODE* if *VARIABLE* expands to the empty string, otherwise *FCODE*.

A conditional block must occur completely within one variable, i.e. it is not possible to e.g. substitute the **%{}** symbol from another variable via delayed reference (but in *TCODE*/*FCODE* delayed references can be used).

Note that any variables you add for delayed substitution are only output with --dump if they are actually used (i.e. referenced in some of the other variables). If you want to output them anyway, e.g. for comments or easier later change, you can collect references to them in the **DUMMY** variable.

The following variables do not contain those which are used only in delayed references. To get a full description of the latter (and of the defaults), please use **eix --dump**.

Some remarks concerning the stability of names in future eix versions: As far as possible and reasonable, it is attempted to keep all meanings of the names, at least for the variables in the subsequent list. For variables for delayed substitution, changes are more likely to occur. As a rule of thumb: The more "elementary" a variable is (e.g. a variable defining just a single character) the less likely is that it is changed or removed in a future eix version. On the other hand, variables with delayed substitution in the "middle range" (that is, those using other variables for delayed substitution) are more likely to be changed in their meaning or to be removed - depending which extensions in the format will be necessary to implement future extensions of eix.

DUMMY (string)
This variable has no direct influence on the programs, but the content of this variable can be used to collect delayed references to (otherwise unused) variables so that they are printed with --dump or --dump-defaults.

WIDETERM (true/false/auto/empty)
This variable is only used for delayed substitution. It defines whether you have a wide terminal (bigger than 80 columns). In **WIDETERM** is empty or **auto** then **COLUMNS** is used to set its value for delayed substitution.

COLUMNS (integer/auto/empty)
This defines the terminal width if **WIDETERM** is empty. If**COLUMNS** is empty or **auto** then a heuristic is used for **WIDETERM**. If the heuristic fails then the substituted value of **WIDETERM** is empty (which acts as **false** for delayed substitution).

EIX_SYNC_OPTS, **EIX_SYNC_CONF (string)**
The content of **EIX_SYNC_CONF** is appended to the file *@SYSCONFDIR/eix-sync.conf* See the description of that file for details. By default, **EIX_SYNC_CONF** is a delayed substitution of **EIX_SYNC_OPTS** so that typically you can also use **EIX_SYNC_OPTS** with the same purpose. Be aware that if one of these variables is compromised, almost anything can happen if you call **eix-sync** with root permissions, so be aware of security risks! Quote these variables carefully if you use them!

EIX_REMOTE_OPTS, **EIX_LAYMAN_OPTS**, **EIX_TEST_OBSOLETE_OPTS (string)**
The content of this variable is evaluated and used as arguments for the scripts **eix-remote**, **eix-layman**, and **eix-test-obsolete**, respectively. So be sure to quote these variables correctly and be

aware of security risks!

EIXRC_SOURCE (string)
This path is prepended to source commands in */etc/eixrc*. It is meant to be set in the environment but can also be set in */etc/eixrc*. In the latter case, it will override the setting from the environment as soon as it is read until all files are sourced. Note that when this variable takes effect, no delayed substitution is performed yet.

EIX_PREFIX (prefix-string) (prefix-string means that this is a string, but if it has the value '/' it is changed to '')

This is essentially meant to be set in the environment if you want to use a chroot: It is prefixed to the path where */etc/eixrc* is searched. If it is unset, the value of the environment variable **PORTAGE_CONFIGROOT** is prefixed. The default value (typically empty) of this variable is used, if neither is set in the environment.

Moreover, this variable is used for delayed substitution to determine the prefix of a lot of paths in the subsequent variables; see **eix --dump** to see in detail in which other variables it occurs.

EPREFIX (prefix-string)
The default value of this variable stems from EIX_PREFIX, appended with a configure-specific default (for prefix-portage). This variable has no meaning on its own, but it is used for delayed substitution to determine the prefix of a lot of paths in the subsequent variables (if their default is not changed); see **eix --dump** to see in detail in which other variables it occurs. In the current default, it influences all paths with the following exceptions:

TMPDIR (string)
This is used for delayed substitution in **EIX_TMPDIR**. Usually it is set by the environment variable. eix exports this variable initialized with the value of **EIX_TMPDIR**.

EIX_TMPDIR (string)
If this variable is nonempty, this directory is used instead of */tmp* for temporary files. eix exports this variable as **TMPDIR**. The default is via delayed substitution to **TMPDIR**.

/usr/bin/eix-functions.sh

~/.eixrc

cachefile path(s) passed in the commandline

PORTAGE_PROFILE (only the variable, not the link)

PORTDIR

Overlay paths

The last three items can be modified by **EPREFIX_TREE**.

The purpose of **EPREFIX** is to allow a quasi-chroot analogously to prefix-portage.

ROOT (prefix-string)

EPREFIX_TREE (prefix-string)

EPREFIX_ROOT (prefix-string)

These are actually not internal variables of eix but they are just used for delayed substitution for the following variables analogously to **EPREFIX** (see above).

The purpose of **ROOT** is to follow roughly portage's usage of this variable. Note that variables in /etc/portage/make.conf do not influence eix configuration variables. In particular, a **ROOT=***something* command in /etc/portage/make.conf does not influence eix. You must set it instead in the environment or in an eix configuration file.

You can easily customize to which paths the **EPREFIX** or **ROOT** variables apply: Simply use in the appropriate following variables either the delayed references **%{EPREFIX}**, **%{ROOT}**, nothing, or **%{EPREFIX_ROOT}** (which in turn is defined as a delayed reference so that you can easily change what to do if **EPREFIX** and **ROOT** are both nonempty: You might e.g. concatenate these paths or use only one of them). You can of course also use other variables as delayed references, e.g. you might want to set

EIX_CACHEFILE=%{EPREFIX_PROFILE}/var/cache/eix/portage.eix

if you feel that the eix cachefile should depend (only) on the profile root.

PORTAGE_CONFIGROOT (prefix-string)
This path is prepended to the /etc paths. The purpose is to respect PORTAGE_CONFIGROOT in an analogous way portage does. If you set this variable in the environment it will also change the path where */etc/eixrc* is searched. (Note that reading */etc/eixrc* occurs before delayed substitution takes effect.)

MAKE_GLOBALS (string)
If this file exists, it is used instead of **%{PORTAGE_CONFIGROOT}/etc/make.globals** The default value corresponds to >=portage-2.2* behavior.

PORTAGE_REPOS_CONF (string)
This file is used to set the default repository information. The file */etc/portage/repos.conf* is read only after this.

PORTAGE_REPOSITORIES (string)
If this variable is nonempty, its content is used as by portage as a substitute for all repos.conf files.

EPREFIX_PORTAGE_EXEC (prefix-string)
This path might be used as a prefix for */usr/bin/ebuild*.

EPREFIX_SOURCE (prefix-string)
This path is prepended to paths occurring as the argument of source commands in /etc/portage/make.conf and /etc/make.globals.

EPREFIX_INSTALLED (prefix-string)
This is prepended to the path where eix expects information about installed packages.

EPREFIX_PORTAGE_CACHE (prefix-string)
This prefix is prepended to the portage cache.

EPREFIX_ACCESS_OVERLAYS (prefix-string)
This prefix is prepended to overlays when their files are accessed.

EPREFIX_PORTDIR (prefix-string)
This path is prepended to **PORTDIR**.

EPREFIX_OVERLAYS (prefix-string)
This prefix is prepended to all entries of **PORTAGE_OVERLAY**.

EPREFIX_PROFILE (prefix-string)
This prefix is prepended to **PORTAGE_PROFILE** (the variable, not the link).

EPREFIX_VIRTUAL (prefix-string)
This is prepended to overlays in the eix database to test whether they exist.

EIX_CACHEFILE (string)
The eix cachefile, usually **%{EPREFIX}/var/cache/eix/portage.eix**

EIX_PREVIOUS (string)
The previous eix cachefile for eix-diff and eix-sync, usually **%{EPREFIX}/var/cache/eix/previous.eix**

EIX_REMOTE1, **EIX_REMOTE2 (string)**
The eix cache used when **-R** or **-Z** is in effect. If the string is nonempty, eix-remote uses this file for the output. The default value is **%{EPREFIX}/var/cache/eix/remote.eix** or **%{EPREFIX}/var/cache/eix/remote.eix**, respectively.

REMOTE_DEFAULT (integer)
The value 1, 2, or 0 means that eix option **-R**, **-Z**, or neither is the default.

EIX_REMOTEARCHIVE1, **EIX_REMOTEARCHIVE2 (string)**
This is a local copy of the remote archive used by eix-remote. If it is empty, only temporary files are used.

EBUILD_DEPEND_TEMP (string)
Path to the file which is generated by **ebuild depend**.

EIX_WORLD (string)
The file eix considers as the world file. Note that usually this file is only readable if you are a member of the portage group. To avoid needing this permissions you can use **SAVE_WORLD**.

EIX_WORLD_SETS (string)
The file eix considers as the world_sets file. The same remarks as for **EIX_WORLD** hold true.

EIX_LOCAL_SETS (string list)(the meaning of string list is explained in the next paragraph)
This is a list of directories which contain locally defined sets. The directories are read in the given order; files with set-names which have been read earlier are ignored: In this sense earlier entries in **EIX_LOCAL_SETS** take precedence over later entries.

Relative directories (i.e. those not starting with /) are understood relative to **$PORTDIR**. Entries in **EIX_LOCAL_SETS** starting with the special symbol ***** are interpreted in a special way: They are interpreted as several entries, the ***** symbol being replaced successively by the paths to the overlays in reverse order (the latter in view of the fact that in a sense earlier entries of **EIX_LOCAL_SETS** override later entries while for **PORTDIR_OVERLAY** one expects the converse).

The default of this variable contains **/etc/portage/sets**, **/etc/portage/sets.eix**, **sets** (i.e. roughly **$PORTDIR/sets** according to the previous paragraph), ***/sets** (i.e. roughly **$PORTIDR_OVERLAY/sets** according to the previous paragraph) as well as **%{EIX_LOCAL_SETS_ADD}**. The reason for the latter is that you can just add additional directories to the variable **%{EIX_LOCAL_SETS_ADD}** in */etc/eixrc*. Reasons why you might want to do this is e.g. if you defined another multiset directory (e.g. for some overlay or for treatment with "world-candidate = False") in /etc/portage/sets.conf or in the sets.conf file of some overlay.

For all string list variables, the possible separators between different entries are [\t\r\n]. If you want to use such a separator within an entry, you have to escape it by \. Moreover, all \ symbols have to be escaped analogously, since the escapes of escapes and escapes of the separator symbols are removed. If you want to insert a variable "as-is", you can make use of delayed substitution: **%{***VAR***}** will insert the value of *VAR* with all symbols from [\ \t\r\n] being escaped. (See delayed substitution for more details).

EAPI_REGEX
This regular expression matches the recognized EAPIs in .ebuild suffixes which will be recognized according to GLEP 55. You might need to modify this expression locally according to the installed portage version (in order to make sure that the installed portage can parse the corresponding EAPIs). An exceptional case is if that variable is empty: In this case all EAPI-suffixed ebuilds are ignored.

SAVE_WORLD (true/false)
If you set this to true, the information of your world file is stored in /var/cache/eix/portage.eix. Note that this means that anybody who can read this file has the information about the content of your world file. Make sure that this is really what you want if you set this to true.

CURRENT_WORLD (true/false)
If false, then the world file information stored in /var/cache/eix/portage.eix is used, even if your current world file is readable. Otherwise, your current world file (if it is readable) overrides that information.

EIX_USER, **EIX_GROUP**, **EIX_UID**, **EIX_GID (string/integer)**
eix, **eix-diff**, **eix-update**, **eix-drop-permissions**, and **eix-remote** change to the permissions of **EIX_USER**/**EIX_GROUP** (using the ID-numbers **EIX_UID**/**EIX_GID** if the former have no valid name) as soon as possible. To avoid this change, use an invalid name (e.g. the empty name) and a non-positive ID-number.

If **EIX_GROUP** is valid or **EIX_GID** is positive, then also the grouplist is changed according to the following rule: If **EIX_USER** is a valid name, the full grouplist of that user is chosen; otherwise, the grouplist is shrinked to a singleton.

REQUIRE_DROP (true/false/root)
If **true**, it is required that the dropping of permissions according to **EIX_{USER,GROUP,UID,GID}** is successful, see **DROP_FATAL**. If**r oot**, this is required only if the current UID is 0.

NODROP_FATAL (true/false)
If the requirement of **REQUIRE_DROP** is not met, a warning (**false**) or an error (**true**) is raised.

PORTAGE_ROOTPATH (string)
If nonempty, this variable is passed unchanged to ebuild.sh for cache method **ebuild***.

DEFAULT_ARCH (string)
The default ARCH if none is specified by the profile.

NOFOUND_STATUS (integer)
This value is used as exit status if there are 0 matches. The value of **COUNT_ONLY_PRINTED** is honoured.

MOREFOUND_STATUS (integer)
This value is used as exit status if there are 2 or more matches. The value of **COUNT_ONLY_PRINTED** is honoured.

EIX_LIMIT, **EIX_LIMIT_COMPACT (integer)**
As a safety measurement for typos there are never more packages displayed than specified here: There are two different variables, depending on whether **--compact** is active. There is no limit if the output is not sent to a terminal or if the value of the variable is **0**.

QUICKMODE (true/false)
If true, eix and eix-diff will use **--quick** by default.

CAREMODE (true/false)
If true, eix and eix-diff will use **--care**.

USE_BUILD_TIME (true/false)
Whether to use the BUILD_TIME entry (if it exists) of the portage database instead of the directory timestamp (which usually is the installation time). The difference is usually only important for packages installed from *.tbz2, *.gpkg.tar, or *.xpak files, but in most cases the build time is more relevant (it is also more reliable). Unfortunately, the build time is available only for packages built and emerged with >=portage-2.2_rc63; you can use **eix-installed [no-]buildtime** to check for which versions this is (not) the case. Reading the build time is always slower than using the directory timestamp (even if the build time is not available). Thus, you might want to set this variable to **false** if speed of eix is more important to you than the buildtime information.

QUIETMODE (true/false)
If true, eix and eix-diff will use **--quiet** by default.

PRINT_APPEND (string)
This string is appended to the output of **--print**. Standard escape sequences in this string like \n are interpreted. The default is a newline for reasonable output in an interactive shell. Note that command substitution in shell scripts removes all trailing spaces so that this newline does not

harm (but trailing spaces of the variable would be removed in the shell script anyway). Hence, in order to read variables from eix in a shell script without omitting trailing spaces, you should use a visible symbol for PRINT_APPEND and use for instance something like **VAR=‘PRINT_APPEND=x eix --print VAR‘ ; VAR=${VAR%x}**

DEFAULT_FORMAT (normal/compact/verbose)
Defines whether **--compact** or **--verbose** is on by default. Since this variable can be changed by the user in **/etc/eixrc** or **~/.eixrc**, this might have undesired side effects for scripts setting **FORMAT** with the intention to get a fixed **FORMATSTRING**: Such scripts should better use the **--format** option (which ignores **DEFAULT_FORMAT** since eix-0.29.6), use the **--normal** option (available since eix-0.29.6), and/or export **DEFAULT_FORMAT=normal** to override such user settings.

DIFF_ONLY_INSTALLED (true/false)
If true, eix-diff will only consider version changes for installed packages.

DIFF_NO_SLOTS (true/false)
If true, eix-diff will not consider slots for version changes.

DIFF_SEPARATE_DELETED (true/false)
If true, eix-diff will print deleted packages in a section on their own. Otherwise, eix-diff will mix deleted and changed packages "alphabetically".

NO_RESTRICTIONS (true/false)
If false, RESTRICTION and PROPERTIES data is output.

RESTRICT_INSTALLED (true/false)
If true, fetch and mirror restrictions for installed versions are calculated.

CARE_RESTRICT_INSTALLED (true/false)
If true, fetch and mirror restrictions for installed versions are always fetched from disk, even if it could be read from a corresponding version. This is slower but more reliable, i.e. it will also find out whether restrictions were changed.

DEPS_INSTALLED (true/false)
If true, depdendency information for installed versions are always fetched from disk, even if it could be read from a corresponding version. The option**--deps-installed** does the same.

DEP (true/false)
If true, store/use **DEPEND**, **RDEPEND**, **PDEPEND**, **BDEPEND**, **IDEPEND** (e.g. shown with eix -lv). Usage of **DEP** roughly doubles disk resp. memory requirements.

SRC_URI (true/false)
If true, store/use **SRC_URI**. Usage of**SRC_URI** roughly doubles disk resp. memory requirements.

REQUIRED_USE (true/false)
If true, store/use **REQUIRED_USE** (e.g. shown with eix -l). Usage of **REQUIRED_USE** increases disk and memory requirements.

FORMAT, **FORMAT_COMPACT**, **FORMAT_VERBOSE (string)**
Define the normal, compact and verbose layout for results printed by **eix**. See**FORMA T-STRING**. If one of these variables is set in a script, it is recommended to set simultaneously the corresponding value of **DEFAULT_FORMAT**.

Since eix-0.13.4 these variables just expand to delayed substitution of the variables **FORMAT_ALL**, **FORMAT_ALL_COMPACT**, or **FORMAT_ALL_VERBOSE**, respectively. The intention of this simple definition is that it is very easy to refer to the default definition when you change the definition of **FORMAT**.

FORMAT_TEST_OBSOLETE (string)
The format used by the **eix-test-obsolete** script; the default value is the delayed substitution of **FORMAT_ALL_COMPACT**. It is not admissible to use delayed substitution of**FORMA T** here; if you want this use instead delayed substitution of **FORMAT_ALL**.

DIFF_FORMAT_NEW, **DIFF_FORMAT_DELETE**, **DIFF_FORMAT_CHANGED (string)**
Define the format for packages that were added, removed or for which the highest stable versions has changed. This is only used by **eix-diff**. See**FORMA TSTRING**. Since eix-0.13.4 these variables just expand to delayed substitution of the variables **DIFF_FORMAT_ALL_NEW**, **DIFF_FORMAT_ALL_DELETE**, or **DIFF_FORMAT_ALL_CHANGED**, respectively.

FORMAT_INSTALLATION_DATE, **FORMAT_SHORT_INSTALLATION_DATE**
Define the strftime() format used to print the installation date (in normal resp. short form).

FORMAT_INSTALLED_USE
Define the printf-like format used to print useflags of installed packages. If this string is empty, the processing is slightly faster since then these data are not read.

FORMAT_BEFORE_SET_USE, **FORMAT_AFTER_SET_USE**, **FORMAT_BEFORE_UNSET_USE**, **FORMAT_AFTER_UNSET_USE**
These strings are printed before/after the set/unset USE flags of installed versions are printed.

FORMAT_BEFORE_USE_EXPAND_START, **FORMAT_BEFORE_USE_EXPAND_END**, **FORMAT_AFTER_USE_EXPAND**, **FORMAT_BEFORE_IUSE_EXPAND_START**, **FORMAT_BEFORE_IUSE_EXPAND_END**, **FORMAT_AFTER_IUSE_EXPAND**
These strings are printed at the start/end or after **USE_EXPAND** variables are output with **use*** for installed or available versions, respectively.

FORMAT_BEFORE_COLL_EXPAND_START, **FORMAT_BEFORE_COLL_EXPAND_END**, **FORMAT_AFTER_COLL_EXPAND**
These strings are printed at the start/end or after **USE_EXPAND** variables are output with **colliuse***.

FORMAT_MASKREASONS_LINESKIP, **FORMAT_MASKREASONS_SEP**, **FORMAT_MASKREASONSS_LINESKIP**, **FORMAT_MASKREASONSS_SEP**
There string are used as separator for new lines resp. new reasons for <maskreasons> and <maskreasons*>, respectively.

COLOR_OVERLAYKEY, **COLOR_VIRTUALKEY**, **COLOR_KEYEND**, **COLOR_OVERLAYNAME**, **COLOR_OVERLAYNAMEEND**, **COLOR_NUMBERTEXT**, **COLOR_NUMBERTEXTEND**

These colors are used before/after normal/virtual overlay keys, for printing the overlays at the end or when printing the number of packages.

NOCOLORS (true/false)
Do not use colors.

NOSTATUSLINE (true/false)
Do not update status line.

NOPERCENTAGE (true/false)
Do not show percentage progress.

FORCE_COLORS (true/false)
Use colors even when stdout is not a terminal.

FORCE_STATUSLINE (true/false)
Update the status line even when stdout is not a terminal.

FORCE_PERCENTAGE (true/false)
Show the percentage progress even when stdout is not a terminal.

STYLE_VERSION_SORTED (true/false)
Defines whether **--versionsort** is on by default.

STYLE_VERSION_LINES (true/false)
Defines whether **--versionlines** is on by default.

DUP_PACKAGES_ONLY_OVERLAYS (true/false)
Defines whether duplicate package checks occur only among overlays, i.e. a package is only considered as duplicate if it occurs in at least two different overlays.

DUP_VERSIONS_ONLY_OVERLAYS (true/false)
Defines whether duplicate version checks occur only among overlays, i.e. a version is only considered as duplicate if it occurs at least twice in overlays.

DEFAULT_IS_OR (true/false)
If several pattern arguments occur without logical concatenation (-a or -o), eix assumes an implicit concatenation. If this variable is true, it assumes that this concatenation is -o (or), otherwise -a (and).

OVERLAYS_LIST (all/all-if-used/all-used/all-used-renumbered/no)
People with many different overlays do not want to see all overlays listed at the end but only those which are really used. Here, you can customize the behaviour. The value is interpreted as follows:

all-if-used/if-used/if
Display all overlays, if at least one is used. This was the default behaviour before eix-0.6.0.

used-renumbered/renumber/renumbered/number
Display only the overlays actually used, numbering them "correctly" (i.e. if only two overlays are needed, number them [1] and [2]). The disadvantage is that overlays get different numbers for different queries. However, the order of the numbering is consistent.

all-used/only-used/used
Display only the overlays actually used, keeping the numbering consistent over all queries (on the same database).

no/false
Never display a list of overlays.

anything else
List all overlays for every query (even if not even one is needed).

TERM (string)
The current terminal. Usually this is set by the environment variable.

TERM_STATUSLINE (string)
This is a list of (space-separated) words; if one of these words matches the beginning of TERM, it is assumed that the terminal supports a status line.

TERM_SOFTSTATUSLINE (string)
This is a list of (space-separated) words; if one of these words matches the beginning of TERM, and if the statusline is active, also a soft status line (used e.g. by screen but not by tmux) will be output.

EXIT_STATUSLINE (string)
If this is nonempty, it is used as the exit statusline. An optional leading space in this string is ignored.

TERM_ALT1, **TERM_ALT2**, **TERM_ALT3 (string list)**
This is a list of regular expressions; if one of it matches **TERM**, then **COLORSCHEME1**, **COLORSCHEME2**, or **COLORSCHEME3**, respectively, is used instead of **COLORSCHEME0**, the biggest matching number taking precedence.

You are of course free to assign any meaning, but the intention of the defaults is that no match means 8/16 colors, **TERM_ALT1** specifies the 256 color terminals, **TERM_ALT2** specifies 88 color terminals, and **TERM_ALT3** is free for the user (therefore overriding everything else).

The default expansion of **TERM_ALT1** and **TERM_ALT2** include the values of **TERM_ALT1_ADD** and **TERM_ALT2_ADD** via delayed substitution. So when you just want to add something to the default lists, add it to **TERM_ALT?_ADD**.

EXAMPLE 1: If you want to force 8/16 colors on all terminals you can set **TERM_ALT3=.** to select all terminals for the **COLORSCHEME3** default (which is **COLORSCHEME0**, see below). Users of Ethan Schoonover's **solarized** colorscheme should only do this if they want to force keeping the original 16 solarized colors. Setting **SOLARIZED** (see the separate description) has influence anyway.

EXAMPLE 2: If you want to tell eix that TERM=rxvt means a 256 color terminal, set **TERM_ALT1_ADD=rxvt**. (This is no longer the default, because there are rxvt versions which support only 88 colors and set TERM=rxvt; thus assuming 256 colors would produce completely broken colors.)

SOLARIZED (true/light/dark/false)

This variable is only used in delayed substitution in the default values of the **COLORSCHEME?** variables: If **SOLARIZED** is not **false** (i.e. **true**, **light**, and **dark**, are currently equivalent for eix) then the default in these variables chooses a colorscheme which assumes that the terminal output uses Ethan Schoonover's Solarized colorscheme (light or dark). There is even a solarized scheme available for 256 colors: If your terminal supports it, additional colors besides the 16 solarized colors are used. This is somewhat against Ethan Schoonover's original philosophy about solarized, but eix carries so much information in colors that this extension appears reasonable. It is attempted to keep the usage of additional colors as nonintrusive as possible, e.g. to use these colors mainly for single symbols/flags.

COLORSCHEME0, **COLORSCHEME1**, **COLORSCHEME2**, **COLORSCHEME3 (list of integers)**

Which of these variables is actually used depends on which **TERM_ALT?** matches **TERM**; if none matches, then **COLORSCHEME0** is used.

The integer value of this variable determines the color scheme to be used; if the variable contains two integer numbers (separated by spaces) then the first/second number is used if darkmode is on/off (cf. **DARK** and **TERM_DARK**).

The value 0 means that the first *COLORSTRING* of each color specification is used; for higher numbers a correspondingly later *COLORSTRING* is used. If the number is too high, the first *COLORSTRING* is used.

The default values of the **COLORSCHEME?** variables honour the values of the variable **SOLARIZED**: If **SOLARIZED** is set then the default is to use a colorscheme which was developed based on Ethan Schoonover's Solarized system colors (both, light and dark mode are supported; in 256 color capable terminals additional colors were added). Note that this only gives a reasonable output if the terminal was previously configured to use the solarized colors as system colors.

The numbers in the default settings are:

0: Dark basic scheme (for black background)

1: Dark scheme for 256 colors

2: Light basic scheme (for white background)

3: Light scheme for 256 colors

4: Solarized extended for 256 colors (light or dark)

5: Solarized original (16 colors, light or dark)

The variables **BG0**, **BG1**, **BG2**, **BG3** are honoured by the defaults with delayed substitution to force the background color for the corresponding schemes. The default of the **BG?** variables is "none": No background color should be forced. If you want to force the correct background colors, set the variables as follows:

BG0=black

BG1=black

BG2=white

BG3=white

(Depending on your eix version, some of these are even the defaults). However, this can have undesired effects on transparent terminals or when scrolling is necessary, see the **BUGS** section. For these side effects it can be desirable/undesirable - depending on your taste and terminal - to set **RESET_ALL_LINES=false** or **RESET_ALL_LINES=true** (the default can depend on the eix version).

The default values of the **COLORSCHEME?** variables are:

COLORSCHEME0: 0 2; meant for 8/16 color terminals

COLORSCHEME1: 1 3; meant for 256 color terminals

COLORSCHEME2: as **COLORSCHEME0**, meant for 88 color terminals

COLORSCHEME3: as **COLORSCHEME0**, meant to be a sane default, to be modified by the user.

DARK (true/false/auto)

The value of this variable determines whether darkmode is on/off for the color scheme selection in the **COLORSCHEME?** variables. For the value **auto** the result depends on a heuristic based on **TERM_DARK** and **COLORFGBG_DARK**, see below.

TERM_DARK (string list)

This is a list of pairs of the form *regular-expresssion 1 dark-mode 1 regular-expresssion 2 dark-mode 2 ...* where *dark-mode* is one of **true**, **false**, **true***, **false***; optionally, a default *dark-mode* can be specified as a last entry (if none is specified, the default is **true***). If**D ARK=auto** this is used to determine the darkmode as follows:

The first **regular-expresssion** matching **TERM** is considered, and the corresponding **dark-mode** describes what to do (if no **regular expression** matches then **dark-mode** is assumed to be **true**): The values **true** (or **true***) and **false** (or **false***) mean that the darkmode is on or off, respectively. However, for **true*** and **false*** the variable **COLORFGBG** is respected in addition: If **COLORFGBG** is nonempty then the darkmode is on or off, depending on whether **COLORFGBG** matches an expression from **COLORFGBG_DARK**. Note that**COLORFGBG** is an en vironment variable set by some terminals like **rxvt** or **konsole** which describes the background color.

COLORFGBG_DARK (string list)

This is a list of regular expressions. It is only used if**D ARK=auto**, the matching *dark-mode* in **DARK_TERM** is **true*** or **false***, and **COLORFGBG** is nonempty. In this case, the resulting darkmode is on or off, depending on whether **COLORFGBG** matches an element of this list. Note that **COLORFGBG** is an environment variable set by some terminals like **rxvt** or **konsole** which describes the background color.

LEVENSHTEIN_DISTANCE (integer)

Set default levenshtein-distance.

UPDATE_VERBOSE (true/false)

Whether eix-update -v is on by default (output of cache method per version).

EXCLUDE_OVERLAY (string list)

Set a list of wildcard patterns for overlay paths that are excluded from the index. See the**eix-update** option **--exclude-overlay**.

ADD_OVERLAY (string list)
Set a list of overlays that are added to the index. See the**eix-update** option **--add-o verlay**.

EXPORT_PORTDIR_OVERLAY (true/false)
If true and overlays are excluded/added, a correspondingly modified **PORTDIR_OVERLAY** is exported. This means that in a sense also the corresponding eclasses of these overlay are excluded/added for cache methods eix and eix*.

CACHE_METHOD_PARSE (string)
This string is appended to all cache methods using parse/parse*/ebuild/ebuild*. Its default contains **#metadata-md5#metadata-flat**. Unless you have a very special setup this is always what you want: It means that if a current metadata (in a metadata/*cache subdirectory) is available for the ebuild, this is used instead of parsing/executing the ebuild.

The latter is in principle not completely reliable (parsing) or rather slow (executing), i.e. the metadata (if available and up-to-date) is always preferable. For more details see the section **SPEEDUP**.

PORTDIR_CACHE_METHOD, **OVERLAY_CACHE_METHOD (string)**
Set the type of the cache used by portage and for overlays. **PORTDIR_CACHE_METHOD** defaults to *metadata-md5-or-flat*, **OVERLAY_CACHE_METHOD** to *parse|ebuild**.

For speed reasons it is preferable to use metadata also for overlays. More details can be found in the section **SPEEDUP**.

Security Warning: If you do not completely trust the .ebuilds in your overlays, you should set **OVERLAY_CACHE_METHOD=parse**.

You might want to set temporarily **OVERLAY_CACHE_METHOD=eix*::˜** - as will be explained later, eix will then take by default the overlay data from the previous eix database.

The available cache methods are described below. You might want to specify a different cache method only for some overlays. This can be done with the following variables:

CACHE_METHOD, **OVERRIDE_CACHE_METHOD (string list)**
These variables are string lists of the form "*overlay method overlay method* ...". The cache method of *overlay* is set to the subsequent *method*, thus overriding the default settings of **OVERLAY_CACHE_METHOD** (or of **PORTDIR_CACHE_METHOD** if *overlay* is the **PORTDIR** directory). *overlay* is interpreted as a wildcard pattern which should match the *path* (symbolic links resolved) of the overlay (it is not possible to refer to repository's name here - you must use the true *path*). Later entries override earlier ones: The last matching entry takes precedence.

The difference between **CACHE_METHOD** and **OVERRIDE_CACHE_METHOD** is that the former applies immediately while the latter can also be used to extend/override the changes made implicitly if **KEEP_VIRTUALS** is used (see below).

The default definitions of **CACHE_METHOD** and **OVERRIDE_CACHE_METHOD** contain **%{ADD_CACHE_METHOD}** and **%{ADD_OVERRIDE_CACHE_METHOD}**, respectively. According to the mechanism of delayed substitution explained in another section, this means that the variables **ADD_CACHE_METHOD** or **ADD_OVERRIDE_CACHE_METHOD** can also be used to override the cache method locally. If you modify**CA CHE_METHOD** or **OVERRIDE_CACHE_METHOD** in */etc/eixrc*, it is recommended to add "**%{ADD_CACHE_METHOD}**" or "**%{ADD_OVERRIDE_CACHE_METHOD}**" (note the

space in front of **%**) at the end of the respective modified definitions, so that these variables can still be used for local overriding.

The following cache methods are available:

metadata-flat or **metadata-flat:***PATH*
: Use the metadata-cache located inside the portage-tree ($PORTDIR/metadata/cache).
If you provide *PATH*, it overrides the above path; in this case, it must be the full path (i.e. no prefix is used). You might want to provide *PATH* if you use some package manager to generate the metadata in the corresponding directory.

metadata-md5 or **metadata-md5:***PATH*
: This is similar to **metadata-flat** with the difference that the files within ($PORTDIR/metadata/md5-cache) are used where an md5sum is used instead of the timestamp to check whether the metadata is up-to-date. This should be the preferred method for overlays if the overlay supports it, that is, if its **metadata/layout.conf** file contains the line **cache-format=md5-dict**. (Note that only portage reads**metadata/lay out.conf**; eix does not.)

metadata-assign or **metadata-assign:***PATH*
: This is similar to **metadata-flat** with the difference that the files within the metadata cache are expected to be in an "assignment format" (TYPE=value) which is the case in some alt-gentoo trees.

metadata-md5-or-flat, **metadata-md5-or-flat:***PATH*
: This is similar to **metadata-flat** with the difference that it is first attempted to find each category in $PORTDIR/metadata/md5-cache. If*P ATH* is specified, this method is equivalent to **metadata-md5**:*PATH*.

metadata-md5-or-assign, **metadata-md5-or-assign:***PATH*
: This is similar to **metadata-assign** with the difference that it is first attempted to find each category in $PORTDIR/metadata/md5-cache. If*P ATH* is specified, this method is equivalent to **metadata-md5**:*PATH*.

sqlite
: This is an extremely fast cache method if you are using portage with the sqlite backend, see http://wikigentoo.ksiezyc.pl/TIP_speed_up_portage_with_sqlite.htm or https://wiki.gentoo.org/wiki//etc/portage/modules (older links to this topic were *http://gentoo-wiki.com/TIP_speed_up_portage_with_sqlite* or *http://gentoo-wiki.info/TIP_speed_up_portage_with_sqlite* or *http://en.gentoo-wiki.com/wiki/Portage_SQLite_Cache* though apparently none of these do work anymore).

  Note that in contrast to the default **metadata** cache method you must use **emerge --metadata** before you call **eix-update** with this method.

  Since the support of this cache method requires **sqlite** to be installed, this method is not necessarily compiled in. You must have emerged **eix** with the appropriate USE flag (or in a manual installation used *./configure --with-sqlite* before compilation) if you want support for this method.

  As for all other cache methods, only those categories enabled by profile/category (in the main tree or some overlay) are read. If you do not want this, use **sqlite*** (see below).

sqlite*
: This is analogous to the cache method **sqlite** with the difference that all categories found in the file are added, even those categories which were not enabled by some profile/categories file.

flat or **flat:***PATH*
This is similar to **metadata-flat** with the difference that the metadata is expected in the directory *PATH$PORTAGE_OR_OVERLAY_DIR*. If*P ATH* is omitted, it defaults to **/var/cache/edb/dep**.

You can use this cache method with <portage-2.1 and the default backend.

assign or **assign:***PATH*
This is analogous to **flat** with the difference that the files are expected to be in an "assignment format" (TYPE=value). This is the case if you use portage-2.1 with the default backend.

If you are using >=portage-2.1 and the default backend, you might want to use this one if you do not have access to the portage tree when you run eix-update. Note that in contrast to the default **metadata-*** cache methods you must run **emerge --metadata** before you call **eix-update** with this method. You will probably want to use a corresponding option in eix-sync in this case.

repo-flat or **repo-flat:***PATH* or **repo-assign** or **repo-assign:***PATH*
This is analogous to **flat**/**assign** with the difference that the metadata is expected in the directory *PATH*/*$repo_name*. If*$r epo_name* is empty, the name **x-***XXX* is assumed where *XXX* is the last nonzero path component of the portage/overlay directory. This corresponds to the naming scheme of paludis. If this gives a wrong path for some overlay, you can still manually override this with e.g. **O VERRIDE_CACHE_METHOD=’***bad_overlay metadata-assign***:***correct_path***’**.

parse[#metadata-method**]...**
Get the information from the ebuilds, parsing it using some heuristics. Hence, this method has no security risk but possibly some other problems. For example, if variables are only set in eclasses, this method will not see them. Examples of problems with this method is missing SLOT information for typical ebuilds from kde-base or stupid version numbers for gcc cross-compilers. This is the cache-method **none** from older eix versions (before 0.11.1).

It is optionally possible to append one or several strings of the form **#***metadata-method* where *metadata-method* is any of the above mentioned cache methods (excluding **sqlite**). In this case, the metadata-methods are used to check whether they contain newer information than the ebuild. If this is the case, the metadata is used instead of the ebuild (the first matching metadata wins).

Usually this is what you want, especially for overlays, since the metadata is more reliable than the results of cache method **parse**. Of course, this is only useful if the overlay *foo* contains metadata which is regularly updated by the overlay maintainer using

egencache --repo=*foo* --update

(whether this is done for layman overlays depends on the overlay maintainer; for your local overlays you can of course use the above command manually).

Since this is usually what you want, the string in **CACHE_METHOD_PARSE** is appended by default to the cache method **parse**.

Of course, if you know in advance that the metadata information of the overlay is up-to-date, it is (slightly) faster to use directly to use the corresponding metadata method (usually **metadata-flat**). The latter makes sense if you e.g. call **egencache** from a script which you use to sync the overlay (e.g. in eix-sync).

parse*[#metadata-method]...
This is essentially the same as cache method **parse** with the difference that variables are not expanded in variable definitions. This is the cache-method **none*** from older eix versions (before 0.11.1) and cache-method **none** from very old eix versions (before 0.7.1)

ebuild[#metadata-method]...
If no portage cache is available (e.g. for overlays) this is the most compatible but also the slowest method. The information is obtained via "/usr/bin/ebuild ... depend" from the ebuild. Since all ebuilds will get executed by bash, this may be a security risk if you do not trust all ".ebuild" scripts or your environment variables. Intentionally, the environment is not cleared before the actual execution so that you can pass further variables to the ebuild. However, this can also lead to unexpected behaviour or even a security risk since many bash scripts may be tricked with strange environments. Use**en v -i eix-update** when this causes problems.

Do not use this method if you do not completely trust all .ebuilds for which the method applies!

Concerning the optional appendices *#metadata-method*, see the description of the cache method **parse**. Note that if corresponding metadata is available this will be essentially faster than executing the ebuild.

ebuild*[#metadata-method]...
This is a slightly faster and slightly less compatible version of **ebuild**: The information is obtained via the undocumented "/usr/lib/portage/ebuild.sh". Thus, instead of executing a python program for each ".ebuild" as for cache-method **ebuild**, "only" a lengthy shell-script and the ebuild itself is executed (hence, also this method is unsafe if you do not trust all ".ebuild" scripts). Most environment variables except for portage variables and PATH are cleared; PORTAGE_ROOTPATH is exported; some .ebuild-specific variables like $P are set when the ebuild is executed. This method is not quite as compatible as the method **ebuild**, and its success may depend more on the portage version. However, this method is considerably faster than **ebuild** and stable enough to treat e.g. typical ebuilds from kde-base.

Do not use this method if you do not completely trust all .ebuilds for which the method applies!

parse|ebuild, **parse*|ebuild**, **parse|ebuild***, **parse*|ebuild*** [#metadata-method]...
This is a mixture of **parse**/**parse*** and **ebuild**/**ebuild***. Each ebuild is first scanned as with method **parse**/**parse***. If the obtained result has missing information or appears strange, the ebuild is treated as with cache method **ebuild**/**ebuild***. As a rule of thumb, this method is much faster than **ebuild**/**ebuild*** but still much slower than **parse**/**parse***. It has the same security risks as **ebuild**/**ebuild***, of course.

Do not use this method if you do not completely trust all .ebuilds for which the method applies!

eix or **eix:***FILE* or **eix:***FILE*:*overlay*
Use the cachefile *FILE* previously generated by **eix-update**. If omitted or empty, *FILE* defaults to /var/cache/eix/portage.eix.

As for all other cache methods, only those categories enabled by profile/category (in the main tree or some overlay) are read. If you do not want this, use **eix*** (see below).

If *overlay* is not given or empty, then only the "main" tree in *FILE* is read - overlays within *FILE* are ignored. Otherwise only the part corresponding to *overlay* is read from *FILE*. Here, "corresponding" means the first overlay from *FILE* matching the wildcard

pattern *overlay* is used. "Matching" means that first the label is tested, then the path and finally also the number of the overlay within *FILE*. Note that *overlay* has in general no relation with the current overlay names or order - only the names/order stored in *FILE* play a role here.

There are two exceptional values for *overlay* which are treated by different rules:

If *overlay* has the special value "~", then the name of the current overlay label is used implicitly as the *overlay* argument; If the current overlay label is empty or does not match, then the current overlay path is used instead.

If *overlay* has the special value "*", then **all** overlays in *FILE* are read. Note this is usually not what you want since it means that if *FILE* originally contained several overlays, this overlay structure will appear "flattened".

eix* or **eix*:***FILE** or **eix*:***FILE**:*overlay*
This is analogous to the cache method **eix** with the difference that all categories found in *FILE* are added, even those categories which were not enabled by some profile/categories file.

This cache method is useful if *FILE* contains information for some overlay directory for which the corresponding profile/categories on the local machine is unknown or not necessarily up-to-date.

This is used for **eix-remote**.

Note that in particular with **PORTDIR_CACHE_METHOD=eix*::~**, the overlay data is by default just "copied" from the previous eix cachefile.

KEEP_VIRTUALS (true/false)
If true, eix-update will keep all virtual overlays from the previous database if that existed. This has the same effect as appending for each virtual overlay of the previous database the entry "*overlay-name*" to **ADD_OVERLAY** and appending a corresponding entry "*overlay* eix*::*overlay*" to **CACHE_METHOD**. Note that this means that this option might override settings made in **CACHE_METHOD**, but it might be overridden by settings from **OVERRIDE_CACHE_METHOD**.

REPO_NAMES
This variable is a string list of the form "*dir-pattern overlay-label dir-pattern overlay-label* ...". When a new cachefile is created, the overlay matching *dir-pattern* obtains the label *overlay-label*, independent of the content of its profiles/repo_name file. This variable can also associate label names to virtual overlays which do not possess such a file. In particular, this variable also overrides overlay labels set by KEEP_VIRTUALS. Later entries override earlier ones: The last matching entry takes precedence.

LOCAL_PORTAGE_CONFIG (true/false)
If false, /etc/portage and **ACCEPT_KEYWORDS** (from make.conf or the environment) are ignored. Since eix-0.7.9, it is recommended to leave this value "true", because setting it to "false" will just hide some informations.

ALWAYS_ACCEPT_KEYWORDS (true/false)
If true, **ACCEPT_KEYWORDS** is used even without **LOCAL_PORTAGE_CONFIG**, e.g. to determine the ‘default’ stability.

UPGRADE_LOCAL_MODE (+ or **local**/**-** or **non-local**/anything else**)**
If this variable is **+** / **-**, the **--upgrade** option of eix will always match as if **LOCAL_PORTAGE_CONFIG** was set to **true** / **false**.

RECOMMEND_LOCAL_MODE (+ or **local**/**-** or **non-local**/anything else**)**
If this variable is **+** / **-** the upgrade/downgrade recommendations as well as in eix-diff the tests for version changes will act as is **LOCAL_PORTAGE_CONFIG** was set to **true** / **false**.

UPGRADE_TO_HIGHEST_SLOT (true/false)
If true, all upgrade tests will give a positive result for an installed package for which not the slot with the best stable version is installed. Exceptions to this general policy can be specified in **/etc/portage/package.slot_upgrade_forbid** or **/etc/portage/package.slot_upgrade_allow**, respectively.

RECURSIVE_SETS (true/false)
If true, then sets and packages which are contained in an included set are considered as part of the parent set.

XML_KEYWORDS(full/effective/both/true/full*/effective*/none/false)
With --xml, this variable decides whether full/effective (or both) types of **KEYWORDS** are printed for each versions. Here, "full" is the**KEYW ORDS** string as specified in the ebuild and "effective" means its modification by the profile. The values **full***/**effective*** are similar to **full**/**effective**, but both types are printed if their values differ. **true**/**false** are equivalent to **full***/**none**.

XML_OVERLAY (true/false)
With --xml, this variable decides whether the overlay (i.e. its path) is output for each version. This has no influence for versions from overlays without label (repository name): For those versions the overlay is output unconditionally.

SORT_INST_USE_ALPHA (true/false)
If **true**, print the useflags of installed packages in alphabetical order. Otherwise, first those useflags are printed (in alphabetical order) which were set when the package was emerged, then the others are printed (in alphabetical order).

CHECK_INSTALLED_OVERLAYS (true/false/repository)
If **true**, always check from which overlay a package was installed. If **false**, only packages with versions in at least two trees are checked, i.e. only packages for which it appears reasonable from the database that it might have been installed from a different overlay. However, this information might be false if such a package was removed from an overlay or if a whole overlay is not in the database anymore.

The special value **repository** is a reasonable compromise: If repository data was stored during emerge (this is the case only with current portage versions; you can use **eix-installed [no-]repository** to check for which versions this is (not) the case) then this repository data is always used. Only if it is unreadable or does not match, the behaviour for that version is the same as in case **CHECK_INSTALLED_OVERLAYS=false**.

Especially in connection with option -T (if **NONEXISTENT_IF_OTHER_OVERLAY** is true) and with option -J the speed increase is enormously if you set this variable to **false** or **repository**, but you should be aware that the information used about the installed overlay is not completely reliable (for versions installed with old portage versions). In particular, the option -T will not detect if an installed version of a package actually stems from a redundant overlay if in the current database all versions of this package stem from one (other) location.

PRINT_COUNT_ALWAYS (true/false/never)
If true, always print the number of matches in the last line, even if this number is 0 or 1. If **PRINT_COUNT_ALWAYS=never**, then this last line is omitted completely. Both is normally not useful but might simplify writing certain scripts parsing the output of eix.

COUNT_ONLY_PRINTED (true/false)
If false, print only the number of matches, independently of whether the matches actually lead to some output. This might be useful for certain scripts if you are only interested in the number of matches and use e.g. **FORMAT=”** to speed things up.

DEFAULT_MATCH_FIELD (string list)

This is a list of strings of the form *regular_expression*[\n\r\t]*match_field* which is used to determine the default match field. The search pattern from the command line is matched against all the *regular_expression* values in this list in the given order. For the first match, the corresponding *match_field* value is used as the default match field. You can specify a last *match_field* (without a *regular_expression*) in this list which is used as the default fallback if all matches failed; if there is no such entry, then **name** is assumed. It may be convenient to use delayed substitution ${\VAR} for *regular_expression* to avoid manually inserting additional escapes.

The possible values for *match_field* are **name**, **category**, **category/name** (or **category-name**), **description**, **license**, **homepage**, **set**, **eapi**, **installed-eapi**, **slot**, **installed-slot**, **use** (or **iuse**), **with-use** (or **installed-with-use**), **without-use** (or **installed-without-use**), **src-uri** (or **srcuri**), **deps**, **depend**, **rdepend**, **pdepend**, **bdepend**, **idepend**, or **error**, corresponding to the analogous command line option for the match field. The special value **error** means that eix stops with an error message claiming that a match field is not autodetected and must be specified explicitly.

For testing scripts, it is recommended to use **DEFAULT_MATCH_FIELD=error** so that scripts do not rely on particular defaults of that variable: This default has already changed several times in the history of eix and probably will also eventually change in the future.

DEFAULT_MATCH_ALGORITHM (string list)

This is a list of strings of the form **(***match_field_specification***)***regular_expression*[\n\r\t]*match_algorithm* which is used to determine the default match algorithm.

The interpretation is essentially analogous to **DEFAULT_MATCH_FIELD** with the main difference that *match_algorithm* specifies the default match algorithm chosen. The possible values for *match_algorithm* are **regex**, **regexcase**, **pattern**, **substring**, **begin**, **end**, **exact**, **fuzzy**, or **error** which correspond to the analogous command line option for the match algorithm. The special value **error** means that eix stops with an error message claiming that a match algorithm is not autodetected and must be specified explicitly. If no other default match algorithm default is specified, then **regex** is used.

In each string one of the two parts **(***match_field_specification***)** and *regular_expression* can be omitted. If neither is omitted, they must both match simultaneously for the algorithm to be selected. The *match_field_specification* refers to the effectively selected match field(s), according to the following rules.

match_field_specification is a concatenation of words of the form **|***match_field*, **&***match_field*, and **!***match_field*, where *match_field* can have the same values as in **DEFAULT_MATCH_FIELD** (except for **error**). The order of these words plays no role.

eix combines all *match_field* values with the same leading symbol (**|**, **&**, or **!**) and thus obtains three (possibly empty) families of match fields. Now *match_field_specification* matches if all of the following holds:

1. If the first family (**|**) is nonempty, at least one match field of that family is effectively selected (think of a logical "or").

2. At least all match fields of the second family (**&**) are effectively selected (think of a logical "and").

3. If the third family (**!**) is nonempty, at most match fields from that family are selected (think of a logical "not (... or ... or ...)").

Thus, for instance, to specify that exactly all of **depend**, **rdepend**, **pdepend**, **bdepend**, **idepend**

are selected (i.e. **--deps**) and no other field is selected, you can use either of

(&deps!deps)

(&depend&rdepend&pdepend&bdepend&idepend!depend!rdepend!pdepend!bdepend!idepend)

On the other hand, if you want to specify that at least one of **depend**, **rdepend**, **pdepend**, **bdepend**, **idepend** are selected and no other than these fields are selected, you can use either of

(|deps!deps)

(|depend|rdepend|pdepend|bdepend|idepend!depend!rdepend!pdepend!bdepend!idepend)

For testing scripts, it is recommended to use **DEFAULT_MATCH_ALGORIRTHM=error** so that scripts do not rely on particular defaults of that variable: This default has already changed several times in the history of eix and probably will also eventually change in the future.

TEST_FOR_EMPTY (true/false)
Defines whether empty entries in /etc/portage/package.* are shown with -t.

TEST_KEYWORDS (true/false)
Defines whether /etc/portage/package.{accept_,}keywords is tested with -t.

TEST_MASK (true/false)
Defines whether /etc/portage/package.mask is tested with -t.

TEST_UNMASK (true/false)
Defines whether /etc/portage/package.unmask is tested with -t.

TEST_USE (true/false)
Defines whether /etc/portage/package.use is tested with -t.

TEST_ENV (true/false)
Defines whether /etc/portage/package.env is tested with -t.

TEST_LICENSE (true/false)
Defines whether /etc/portage/package.license is tested with -t.

TEST_RESTRICT (true/false)
Defines whether /etc/portage/package.accept_restrict is tested with -t.

TEST_CFLAGS (true/false)
Defines whether /etc/portage/package.cflags is tested with -t.

TEST_REMOVED (true/false)
Defines whether removed packages are tested with -t.

TEST_FOR_NONEXISTENT (true/false)
Defines whether non-existent installed versions are positive for -T. What is considered as non-existent is defined by the **NONEXISTENT_IF**-variables.

TEST_FOR_REDUNDANCY (true/false)
Defines whether redundant entries in /etc/portage/package.* are positive for -T. What is considered as redundant is defined by the **REDUNDANT_IF**-variables.

ACCEPT_KEYWORDS_AS_ARCH (full/true/false)
If full or true modify ARCH by ACCEPT_KEYWORDS. This determines which keywords are considered as ARCH or OTHERARCH. The value full also influences the original ARCH keywording.

NONEXISTENT_IF_OTHER_OVERLAY (true/false)
Defines whether versions are non-existent for TEST_FOR_NONEXISTENT if they come from a different overlay than the installed version.

NONEXISTENT_IF_MASKED (true/false)
Defines whether masked versions are non-existent for TEST_FOR_NONEXISTENT.

REDUNDANT_IF_DOUBLE (string)
Applies if /etc/portage/package.{accept_,}keywords lists the same keyword twice for some/all (un-/installed) versions.

string describes which versions should be tested. It can have the following values:

no or **false**
Do not test for this type of redundancy.

some It suffices that the redundancy occurs for some version in the database.

all The redundancy must occur for all versions in the database.

some-installed
It suffices that the redundancy occurs for some installed version. Uninstalled versions are ignored for this test.

all-installed
The redundancy must occurs at least for all installed versions of the package. If no version is installed, it must occur at least once.

some-uninstalled
It suffices that the redundancy occurs for some uninstalled version. Installed versions are ignored for this test.

all-uninstalled
The redundancy must occurs at least for all uninstalled versions. If all versions in the database are installed, it must occur at least once.

-some of the above or +some of the above
The test only applies if in addition no version (in case -) resp. at least some version (in case +) of the package is installed.

some of the above **or** some of the above
The result is positive if at least one of the two tests is positive. Instead of "**or**" also the symbols "|" or "||" can be used.

REDUNDANT_IF_DOUBLE_LINE (string, see above**)**
Applies if /etc/portage/package.{accept_,}keywords has two lines for identical targets, i.e. such that portage would drop the first of these lines. Note that lines with targets **foo/bar** and **=foo/bar-1** are considered as different in this context by portage (and thus also by eix) even if **foo/bar** would apply to version **1**. The latter redundancy can be found implicitly with **REDUNDANT_IF_DOUBLE_LINE**, **REDUNDANT_IF_MIXED**, and **REDUNDANT_IF_STRANGE**.

REDUNDANT_IF_MIXED (string, see above**)**
Applies if /etc/portage/package.{accept_,}keywords lists two different keywords, e.g. ~ARCH and *, for the versions in question.

REDUNDANT_IF_WEAKER (string, see above**)**
Applies if /etc/portage/package.{accept_,}keywords lists a keywords which can be replaced by a weaker keyword, e.g. ** or ~OTHERARCH or OTHERARCH in place of ~ARCH, or ~OTHERARCH in place of OTHERARCH, for the versions in question.

REDUNDANT_IF_STRANGE (string, see above**)**
Applies if /etc/portage/package.{accept_,}keywords lists a strange keyword, e.g. UNKNOWNARCH (unknown to the .ebuild and ARCH) or -OTHERARCH, for the versions in question.

REDUNDANT_IF_NO_CHANGE (string, see above**)**
Applies if /etc/portage/package.{accept_,}keywords provides keywords which do not change the availability keywords status for the versions in question.

REDUNDANT_IF_MASK_NO_CHANGE (string, see above**)**
Applies if /etc/portage/package.mask contains entries which do not change the mask status for the versions in question.

REDUNDANT_IF_UNMASK_NO_CHANGE (string, see above**)**
Applies if /etc/portage/package.unmask contains entries which do not change the mask status for the versions in question.

REDUNDANT_IF_DOUBLE_MASKED (string, see above**)**
Applies if /etc/portage/package.mask matches twice for the versions in question.

REDUNDANT_IF_DOUBLE_UNMASKED (string, see above**)**
Applies if /etc/portage/package.unmask matches twice for the versions in question.

REDUNDANT_IF_DOUBLE_USE (string, see above**)**
Applies if /etc/portage/package.use matches twice for the versions in question.

REDUNDANT_IF_DOUBLE_ENV (string, see above**)**
Applies if /etc/portage/package.env matches twice for the versions in question.

REDUNDANT_IF_DOUBLE_LICENSE (string, see above**)**
Applies if /etc/portage/package.license matches twice for the versions in question.

REDUNDANT_IF_DOUBLE_RESTRICT (string, see above**)**
Applies if /etc/portage/package.accept_restrict matches twice for the versions in question.

REDUNDANT_IF_DOUBLE_CFLAGS (string, see above**)**
Applies if /etc/portage/package.cflags matches twice for the versions in question. Note that this file is not supported by portage, but you might e.g. have built some support for it in your personal /etc/portage/bashrc file. Of course, this means that also no format for /etc/portage/package.cflags is defined. eix assumes that the format is analogous to /etc/portage/package.{keywords,use} (i.e. an entry is at most one line, with the matching version(s) at the beginning). As the other /etc/portage/package.* files, /etc/portage/package.cflags may also be a directory tree; in this case, all non-hidden files/subdirectories in this tree are read recursively, resolving all symbolic links.

REDUNDANT_IF_IN_KEYWORDS (string, see above**)**
Applies if /etc/portage/package.{accept_,}keywords contains a nonempty entry matching the versions in question (to find empty entries use -t). Of course, one will certainly not consider all matches as a redundancy (although one might misuse this option to simply list all matches). However, one might consider matching but non-installed packages as redundant. Hence, typically one might want to set this variable to the value **-some** or the equivalent value **-some-uninstalled** (or to **false** if one thinks that entries for uninstalled packages are "normal" and not redundant).

REDUNDANT_IF_IN_MASK (string, see above**)**
This is analogous to **REDUNDANT_IF_IN_KEYWORDS**, but for /etc/portage/package.mask.

REDUNDANT_IF_IN_UNMASK (string, see above**)**
This is analogous to **REDUNDANT_IF_IN_KEYWORDS**, but for /etc/portage/package.unmask.

REDUNDANT_IF_IN_USE (string, see above**)**
This is analogous to **REDUNDANT_IF_IN_KEYWORDS**, but for /etc/portage/package.use.

REDUNDANT_IF_IN_ENV (string, see above**)**
This is analogous to **REDUNDANT_IF_IN_KEYWORDS**, but for /etc/portage/package.env.

REDUNDANT_IF_IN_LICENSE (string, see above**)**
This is analogous to **REDUNDANT_IF_IN_KEYWORDS**, but for /etc/portage/package.license.

REDUNDANT_IF_IN_RESTRICT (string, see above**)**
This is analogous to **REDUNDANT_IF_IN_KEYWORDS**, but for /etc/portage/package.accept_restrict.

REDUNDANT_IF_IN_CFLAGS (string, see above**)**
This is analogous to **REDUNDANT_IF_IN_KEYWORDS**, but for /etc/portage/package.cflags. See the above comments about this file.

SLOT_UPGRADE_FORBID (string list)
This is a list of filenames/dirnames which serve as /etc/portage/package.slot_upgrade_forbid

SLOT_UPGRADE_ALLOW (string list)
This is a list of filenames/dirnames which serve as /etc/portage/package.slot_upgrade_allow

KEYWORDS_NONEXISTENT (string list)
This is a list of filenames/dirnames which serve as /etc/portage/package.{accept_,}keywords.nonexistent

MASK_NONEXISTENT (string list)
This is a list of filenames/dirnames which serve as /etc/portage/package.mask.nonexistent

UNMASK_NONEXISTENT (string list)
This is a list of filenames/dirnames which serve as /etc/portage/package.unmask.nonexistent

USE_NONEXISTENT (string list)
This is a list of filenames/dirnames which serve as /etc/portage/package.use.nonexistent

ENV_NONEXISTENT (string list)
This is a list of filenames/dirnames which serve as /etc/portage/package.env.nonexistent

LICENSE_NONEXISTENT (string list)
This is a list of filenames/dirnames which serve as /etc/portage/package.license.nonexistent

RESTRICT_NONEXISTENT (string list)
This is a list of filenames/dirnames which serve as /etc/portage/package.accept_restrict.nonexistent

CFLAGS_NONEXISTENT (string list)
This is a list of filenames/dirnames which serve as /etc/portage/package.cflags.nonexistent

INSTALLED_NONEXISTENT (string list)
This is a list of filenames/dirnames which serve as /etc/portage/package.installed.nonexistent

PACKAGE_NOWARN (string list)
This is a list of filenames/dirnames which serves as /etc/portage/package.nowarn

/etc/portage/sets.eix
This is a directory analogous to **/etc/portage/sets** (see the portage manpage). Since portage has some ways to define package sets which are not available for eix, you can use this directory to store (static) sets which you want that eix knows anyway (e.g. so that set entries in /etc/portage/package.{accept_,}keywords can be treated properly).

/etc/portage/package.slot_upgrade_forbid
/etc/portage/package.slot_upgrade_allow
Similarly as all other /etc/portage/package.*, this can be a file or directory. The entries of this file are of the form "category/name" (separated by newlines). The corresponding packages are treated as exceptions for **UPGRADE_TO_HIGHEST_SLOT**.

/etc/portage/package.accept_keywords.nonexistent
/etc/portage/package.keywords.nonexistent
/etc/portage/package.mask.nonexistent
/etc/portage/package.unmask.nonexistent
/etc/portage/package.use.nonexistent
/etc/portage/package.env.nonexistent
/etc/portage/package.license.nonexistent
/etc/portage/package.accept_restrict.nonexistent
/etc/portage/package.cflags.nonexistent

Similarly as /etc/portage/package.*, this can be a file or a directory. If an entry (separated by space or newline) matches the first word of a line in the corresponding /etc/portage/package.* file, this line is excluded from the -t tests (for names not in the database). You can use this to eliminate certain warning from -t.

/etc/portage/package.installed.nonexistent

This is similar to the other /etc/portage/package.*.nonexistent files/dirs with the difference that it eliminates messages from -t about installed packages which had been removed from the database. The entries of this file are of the form "category/name", but you can also omit the "category/" part (although this is not recommended).

/etc/portage/package.nowarn

Similarly as /etc/portage/package.*, this can be a file or a directory. With this file/dir you can switch off test for -T for particular packages. The format of the file is similarly to /etc/portage/package.use with the difference that you can switch on/off tests. Exactly those lines apply for which at least one matching version is available. For example, the lines

sys-kernel/*-sources no_change weaker

>sys-kernel/hardened-sources-2.6.40 -weaker

in this file will cause that -T does not find the package sys-kernel/*-sources if the only cause for it would be that **REDUNDANT_IF_NO_CHANGE** or **REDUNDANT_IF_WEAKER** is set. An exception to this rule is only made for **REDUNDANT_IF_WEAKER** and for hardened-sources, if the latter is available at least in version 2.6.40. The order in the file plays no role: The "-" always takes precedence if it occurs somewhere.

You can list a package several times in this file; the listed tests are then cumulative for the corresponding package.

Available tests are **in_keywords**, **no_change**, **double**, **mixed**, **weaker**, **double_line**, **in_mask**, **mask_no_change**, **double_masked**, **in_unmask**, **unmask_no_change**, **double_unmasked**, **in_use**, **double_use**, **in_env**, **double_env**, **in_license**, **double_license**, **in_restrict**, **double_restrict**, **in_cflags**, **double_cflags**, the meaning corresponding to the according **REDUNDNANT_IF_*** variable.

In addition there are the tests **nonexistent**, **masked**, **other_overlay** which correspond to the respective variables **TEST_FOR_NONEXISTENT**, **NONEXISTENT_IF_MASKED**, **NONEXISTENT_IF_OTHER_OVERLAY**.

/var/cache/eix/portage.eix

This is the binary database for eix. The path can be changed with the **EIX_CACHEFILE** variable (which by default honours **EPREFIX** via delayed reference).

/var/cache/eix/previous.eix
This is the previous version of /var/cache/eix/portage.eix, used by eix-diff and eix-sync. The path can be changed with the **EIX_PREVIOUS** variable (which by default honours **EPREFIX** via delayed reference).

/var/cache/eix/remote.tar.bz2 /var/cache/eix/remote2.tar.bz2
This is a local copy of the remote archive, used by eix-remote. The path can be changed with the **EIX_REMOTEARCHIVE1** or **EIX_REMOTEARCHIVE1** variable, respectively (which by default honour **EPREFIX** via delayed reference).

masked-packages

masked-packages is a helper tool for scripts which can match the arguments against a list of masked packages or versions. The masks are specified by the options and expected to be in the same format as in /etc/portage/package.mask The arguments must be in the form *category*/*name*-*version*[**:***slot*][**::***repo*]. By default, the arguments matching a mask are printed.

Examples:

masked-packages -q --file /etc/portage/package.accept_keywords app-portage/eix-99999999::mv
Returns successfully if an entry from **/etc/portage/package.accept_keywords** matches for the version **app-portage/eix-99999999::mv**.

masked-packages -m '=a/b-1*:1' -m '=c/d-1' a/b-0:1 a/b-1.1:1 a/b-1.2 c/d-1:2
Outputs only **a/b-1.1:1** and **c/d-1:2** since the other arguments do not match in version or slot, respectively.

Options

-h, **--help**
Print a help screen and exit.

-q, **--quiet** (toggle)
Do not print the arguments matching a mask, but instead show by the exit status whether an argument was matching. More precisely, the exit status is only successful if at least one argument is matching.

-Q, **--nowarn** (toggle)
Do not print warnings concerning bad syntax.

-m, **--mask** *MASK*
Add *MASK* to the list of masks.

-f, **--file** *FILE*
Add the lines of *FILE* to the list of masks. *FILE* can also be empty, **-** or a directory: An empty filename or **-** means that standard input is used; a directory is read recursively.

-F, **--read-file** *FILE*
Add words from *FILE* to the list of arguments. *FILE* can also be empty, **-** or a directory: An empty filename or **-** means that standard input is used; a directory is read recursively.

versionsort

versionsort is a helper tool for scripts which serves two purposes. It cuts the version strings and certain non-alphanumeric garbage from its arguments and interprets the arguments themselves as version strings (depending on some heuristics). Then it outputs the version strings in a sorted order, according to the portage rules of version sorting. If there is more than one argument, each version (including the last one) is followed by a newline. For example,

versionsort '>=gcc-4.4' 4.4_alpha0 '<sys-devel/gcc-4.05' 4.5

will output

4.05

4.4_alpha0

4.4

4.5

If you pass only one argument, versionsort is more sloppy about the version rules; even if the version part is not completely correct, it is guaranteed in this case that the output is string-equal to the version part. Thus, you can use versionsort with one argument to split the package name from the version part in a shell script by

split=1-font-adobe-75dpi-1.3-r1

version=‘versionsort "X$split"‘; name=${split%"-$version"}

Although currently it makes no difference, it is safer in such a case (concerning possible future extensions of the version format) to let the argument(s) start with a non-number (like **X** in the above example) to make sure that versionsort will really cut the version from the argument and cannot misinterpret the whole argument as a version.

Since eix-0.25.6, versionsort supports some options: With **-n**, **-p**, **-f**, **-v**, **-r**, **-V** only the name (without version appendix), name with version (without revision appendix), name with version including revision appendix, the pure version (without revision appendix), the revision, or version including appendix corresponding to the argument will be output, respectively. (Since eix-0.29.5 an arbitrary number of arguments is admissible after these options; if more than one argument is specified, on each output a newline is appended.) For example, after

split=1-font-adobe-75dpi-1.3-r1

PN=‘versionsort -n "$split"‘

P=‘versionsort -p "$split"‘

PF=‘versionsort -f "$split"‘

PV=‘versionsort -v "X$split"‘

PR=‘versionsort -r "X$split"‘

PVR=‘versionsort -V "X$split"‘

the variables **PN**, **P**, **PF**, **PV**, **PR**, and **PVR** have an analogous meaning as in ebuilds, namely **1-font-adobe-75dpi**, **1-font-adobe-75dpi-1.3**, **1-font-adobe-75dpi-1.3-r1**, **1.3**, **r1**, or **1.3-r1**, respectively. (if-**r1** would not have been specified, **PR** would be empty: This differs from ebuilds).

Be aware that **versionsort** expects only for the options **-n**, **p**, and **-f** that the argument contains a package name: For all other options (or without option), it is made clear by prepending of "X" that the argument is not a "pure" version number.

BUGS (and sort of FAQ)

For bugreports use either *https://github.com/vaeth/eix/issues/* or Gentoo's bugzilla *https://bugs.gentoo.org/*

Many cache methods are slowly in principle: Please use the hints given in the separate section **SPEEDUP** below.

eix cannot reliably detect the color capabilities and background color of your terminal. Here are a few simple methods how you can avoid some undesired color problems (to understand details of these methods look up the above description of the variables **TERM_ALT?**, **SOLARIZED**, **COLORSCHEME?**, **DARK**, **TERM_DARK**, and **COLORFGBG_DARK**.)

1. Use the color scheme corresponding to your actual background color by setting **DARK=true** or **DARK=false** in */etc/eixrc* (or adapt the heuristics in **TERM_DARK** to your system.)

2. Force the desired background color (or avoid to set it) by setting **BG0=black**, **BG1=black**, **BG2=white**, **BG3=white** (or **BG0=none**, **BG1=none**, **BG2=none**, **BG3=none**) in */etc/eixrc*. Note that forcing of the background color usually makes transparent terminals non-transparent and, moreover, can cause on some terminals some new scrolled-in lines to be in this background color. In order to (not) reset the forced background color before every newline you can set **RESET_ALL_LINES=false** or **RESET_ALL_LINES=true**, respectively (the default can depend on the eix version).

3. Force a desired color scheme on all terminals by setting **TERM_ALT3=.** in */etc/eixrc*; you may want to combine this with setting **COLORSCHEME3=0** (or **=1**, ..., **=5**). Note that the color schemes selected by the numbers 0 and 2 are very poor since they use only the few system colors and thus different things cannot always be distinguished by different colors. However, this can make sense e.g. if you frequently use different terminals and want to get used to only one scheme or if you want to force Ethan Schoonover's **original** solarized colorscheme. However, solarized users should in any case set **SOLARIZED** and use colorscheme 5 (or 4):

4. If you have configured your terminals to use Ethan Schoonover's Solarized colorscheme (light or dark) just set **SOLARIZED=true** (or light or dark - all is equivalent concerning eix). Only use the previous hint additionally if you do not like that there are additional colors used in 256 color capable terminals.

5. If you use rxvt which can do 256 colors but which does not set TERM=rxvt-256color or TERM=rxvt-unicode-256color, put **TERM_ALT1_ADD=rxvt** to */etc/eixrc*.

eix does not and probably never will support dependencies and/or useflags. In particular, this means that eix -u output will be in general different from the output of an emerge update command. You should rely only on the latter for your system. This holds in particular also for the upgrades of packages with slots. The variable **UPGRADE_TO_HIGHEST_SLOT** and manual exceptions in **/etc/portage/package.slot_upgrade_forbid** or **/etc/portage/package.slot_upgrade_allow**, respectively, can be used to manually work around some of these shortcomings somewhat.

eix does not and never will fully support all sets which portage does support. Currently, it appears that even the proposed way of PROPERTIES=set to put packages in the tree will never be supported by eix (because eix would need full support for dependencies and useflags to handle these). As a workaround you can manually define such additional sets in **/etc/portage/sets.eix**. Moreover, eix does not and probably never will support reading of sets.conf files. If you specified additional **sets/** directories e.g. in some overlay, you must add these additional directories manually to the **EIX_LOCAL_SETS** variable in */etc/eixrc*. The easiest way to do the latter is to put an entry like

EIX_LOCAL_SETS_ADD="*/path/to/overlay1/sets /path/to/overlay2/sets ...***"**

into */etc/eixrc* (see the description of **EIX_LOCAL_SETS** above).

eix-diff does never consider /etc/portage/profile. (Reason: The saved database contains only the masking state according to the original profile, but not the profile itself. On the other hand, /etc/portage/profile can only be interpreted when the profile is known.)

The output with the default **OVERLAYS_LIST=all-used-renumbered** is confusing when one wants to use the overlay number in some eix variable/command-argument.

There is no cache method setting which gets information from overlays (for which no portage cache metadata is available) fast and reliable - you must always choose between one of these two extremes. The default is the fast one, but it shows often false slots and has other problems.

All the **EPREFIX**/**ROOT** stuff is confusing. In particular, by the mere fact of allowing much of these variables, eix will always be vulnerable to local attacks if it is called with a possibly unsafe environment.

The previous default **KEEP_VIRTUALS=true** used to confuse people. However, with the new default, nobody will find out that this feature exists. :(

There are too many features: The documentation and configuration has become too complicated. On the other hand, there are still many things which cannot be configured...

SPEEDUP

All non-metadata cache methods for **eix-update** are either non-reliable (**parse**, **parse***) or rather slow and insecure (**ebuild**, **ebuild***). For this reason, it is recommended to generate and use metadata for the main portage tree as well as for all its overlays. The defaults of eix are such that this metadata is used if available (see **CACHE_METHOD_PARSE**) (configuration is only necessary if you use the sqlite backend of portage). Essentially, there are two types of possible metadata which can be used:

1. Metadata stored in each tree/overlay.

   This metadata is in the metadata/cache or metadata/md5-cache subdirectory of $PORTDIR or of the overlay's directory.

2. Metadata stored by portage for each tree/overlay in */var/cache/edb/*

   This metadata is updated with **emerge --metadata** (and its format depends on whether portage uses the sqlite backend).

The disadvantage of using metadata is that it has to be generated/updated before calling **eix-update** if the corresponding tree/overlay is updated and, in case of metadata in */var/cache/edb*, that it takes additional space.

Fortunately, you do not have to care about metadata in the main tree, since the metadata/{md5-,}cache subdirectory of $PORTDIR is automatically updated with **emerge --sync**. However, your local overlays and several overlays updated by layman (or maybe in a different way) do not contain metadata (and even if they do, this metadata is not always up-to-date), and thus metadata for these overlays should be generated to speed up **eix-update**.

Be aware that generating metadata is a security risk, since all ebuilds of the tree are executed when doing so! However, you must be trusting the overlays anyway if you use them...

I do **not** recommend to use **emerge --metadata** (although this could be automatically called at the correct

time by using **eix-sync** with the **-M** option: put e.g. the line **-M** into **/etc/eix-sync.conf**), because it would be a waste of disk space to store the metadata of the whole portage tree in duplicate into */var/cache/edb*.

Instead, I **do** recommend to update the metadata in the overlays at each **eix-sync**. This can also be automatically updated at the correct time, but it requires a more complicated setup:

For a local overlay, for simplicity assume it is in */usr/local/portage* (and that this path is contained in **PORTDIR_OVERLAY**), the procedure is as follows:

First you must give your overlay a name if you have not already done so: Put into the file */usr/local/portage/profiles/repo_name* (creating it if necessary) the name of your overlay. (This step should always be done for overlays, independent of eix or metadata). I assume in the following that this name is *my_local_overlay*.

Second, you should specify which type of metadata you want: Put into the file */usr/local/portage/metadata/layout.conf* (creating it if necessary) the line

cache-formats = md5-dict

and, unless you have a reason not to do so, also the line

thin-manifests = true

The former tells portage to create/read only the metadata in */usr/local/portage/metadata/md5-cache* (which is the new method and should be preferred over the other). The other line is not really related with metadata but just means that the Manifest files in your overlay will/need not contain checksums of files from */usr/local/portage*.

Now, whenever you want, the metadata can be updated by calling

egencache --repo=*my_local_overlay* **--update**

or (unless the overlay provides an up-to-date *profile/use.desc*)

egencache --repo=*my_local_overlay* **--update --update-use-local-desc**

Note that if the overlay uses eclasses from the main tree, the above command should be executed also if these eclasses change, i.e. after each **emerge --sync**. To call the above command automatically at the right time there are several methods:

One method is to put an executable shell script into the directory */etc/portage/repo.postsync.d/* which calls **egencache** in the appropriate manner in dependence of the repository name **$1**. For instance, you can generate an executable file */etc/portage/repo.postsync.d/50-egencache* with the following content:

```
#!/bin/sh

[ -z "$1" ] && exit

case $1 in

# For repositories "mv" and "local" update also profile/use.desc:

mv|local)
```

```
        exec egencache "--repo=$1" --update --update-use-local-desc;;

    # For all other repositories only update the metadata:

    *)

        exec egencache "--repo=$1" --update;;

    esac
```

If the above shell script is executable, **egencache** will be called after every **emerge --sync**.

An alternative is to call **egencache** automatically (only) at each call of **eix-sync**. To do the latter, put e.g. the following lines into */etc/eix-sync.conf*:

@StatusInfo 'Updating metadata for *my_local_overlay***'**

@egencache --repo=*my_local_overlay* **--update --update-use-local-desc**

In a similar manner, you can also treat e.g. overlays managed by layman: For each layman overlay which you use you should put corresponding calls to **egencache** into */etc/eix-sync.conf* (however, please be warned again that this is a security risk since it means that the ebuilds in the overlays will be executed with **eix-sync**).

No matter which method you use: For layman overlays there is an additional problem. Depending on the version control system of the overlay, layman might refuse to sync if you have written data into the metadata/ subdirectory, or layman might remove this subdirectory during syncing.

For overlays managed by git, this problem can be solved as follows: Create in the main directory of that overlay the file *.gitignore* (if it does not already exist) and add to it the lines:

/.gitignore

/metadata/

This will cause that the file .gitignore itself as well as the subdirectory metadata (and its content) are ignored (and thus not updated) by git/layman but only by your above call.

INSTALLATION

It is Gentoo policy to not modify user's **CXXFLAGS** and **LDFLAGS**. It is also Gentoo policy to not change default configs if this can be configured in another way.

For this reason, many USE-flags have been removed in recent gentoo ebuilds. For this reason, very likely your **./configure** settings for eix are **not appropriate** for eix. The maintainer of eix **strongly** recommends to export

EXTRA_ECONF="--enable-security --enable-new-dialect --enable-strong-optimization"

(At the end of this section it is explained how to do this most conveniently.) The purpose of this is to modify **CXXFLAGS** and **LDFLAGS** in a way which highly optimizes eix (and usually essentially decreases the binary size and corresponding memory usage) and which sets them such that it is reasonably safe (concerning *FLAGS) to run most code as root. Of course, some of the chosen *FLAGS are somewhat experimental, but the eix code tries hard to support all of them. Anyway, if this is too experimental for you, you can replace **--enable-strong-optimization** by **--enable-optimization**. You can also remove **--enable-new-**

dialect. Both of these modifications will usually result in less optimized code. Conversely, you can replace **--enable-security** by **--enable-strong-security** which will choose some (experimental) *FLAGS (if available) which will increase the security at the cost of a considerable loss of speed.

If you compile for a low-memory system, it might also be advisable to add

--without-dep-default

--without-src-uri-default

and/or

--without-required-use-default

to the above **EXTRA_ECONF**. The latter is similar to setting **DEP=false** and/or **REQUIRED_USE=false** in some file in **/etc/eixrc/** and thus decreases memory usage of eix by disabling storing of the corresponding data. Some more specialized options you might want to add to **EXTRA_ECONF** are

--enable-warnings --enable-strong-warnings --debugging

with their obvious meaning concerning *FLAGS. Further**EXTRA_ECONF** ar guments which had previously been accessible through USE-flags are

--enable-swap-remote (exchange the role of the two remote adresses used for **eix-remote**)

--enable-separate-tools (build separate small binaries for the tools instead of linking them into the eix binaries)

The eix maintainer recommends to configure your package manager to export the desired **EXTRA_ECONF** automatically when emerging eix. For portage, this can be done as follows. Put the line

app-portage/eix eix-extra-econf.conf

into the file */etc/portage/package.env* or into some file of that directory (creating this file/directory if it did not exist yet), and then create the file */etc/portage/env/eix-extra-econf.conf* with the corresponding line **EXTRA_ECONF="**...**"**.

Alternatively, install eix from the mv overlay. This also gives you the experimental option to build eix using the meson build system (instead of autotools). If this build system works on your system, this is recommended because it is faster and works nicer with ccache than autotools. At the time of writing this text, it is not possible to use meson with the ebuild from the main gentoo repository.

HISTORY

eix was formerly known as **portagedb**. The name was changed because a part of portage is also called portagedb, which was a bit confusing for everyone.

The functionality of eix-update was once triggered by using the -u switch on eix. It was than separated to provide better maintainability. Thus**update-eix** came to life. Meanwhile, it is the same e xecutable with functionality distinguished by its call name. and the original name update-eix was renamed into **eix-update**.

Also **eix-diff** was previously called **diff-eix**, **eix-remote** was previously called **update-eix-remote**, **eix-layman** was previously called **update-eix-layman**, and **eix-functions.sh** was previously called **functions-eix.sh** (and much earlier **update-eix-functions.sh**). The reason for all these renamings is to have a more

consistent naming scheme: All programs belonging to eix now start with eix-* (with the exception of **versionsort** which is more or less just an independent tool).

If you cannot get accustomed to this new naming scheme or have scripts depending on the old names, you can use symbolic links for the original names; this is explicitly supported and is not intended to be deprecated.

Previously, there was a **./configure** option to install such symlinks or a reminder about their removal automatically, but this support has been removed as of eix-0.24.0.

Since the introduction of the %{*VARIABLE} syntax in eix-0.8.0, it is not reasonable anymore to use different variable names for eix and eix-diff. Hence, all corresponding**DIFF_*** v ariables have vanished.

The cache method **metadata-flat** was previously just called **metadata**. The cache method**assign** w as previously called **backport** or **portage-2.1**. The cache method**flat** had pre viously the name **portage-2.0** which was even preferred. Anyway, the obsolete names are still supported.

portage-2.1 and portage-2.1.1 doesn't remove the old dep-cache, thus eix might find packages that are not in portage anymore if the **flat**/**assign** cache method is used. To circumvent this, eix-sync used to delete the old cache (rm -rf /var/cache/edb/dep/*). Since most people have no need to use this cache method anymore and deleting the old cache slows down the next portage run, this is not the default anymore (but still available as an option which can be put into */etc/eix-sync.conf*).

eix-sync used to default to gensync instead of layman. See the description of */etc/eix-sync.conf* how you can still use the deprecated gensync if you want to.

eix-sync no longer supports logging; options -v and -V have been removed. This avoids problems like no visible output with EMERGE_DEFAULT_OPTS=--ask. You should now use redirection when you want to use eix-sync in a cronjob.

The mechanism described now by **DEFAULT_MATCH_FIELD** has changed. The previous (less powerful) **MATCH_.*_IF** and **MATCH_ORDER** variables are not supported anymore.

The **ADD_CACHE_METHOD** and **ADD_OVERRIDE_CACHE_METHOD** variables are no longer built-ins but only used implicitly in delayed substitution in the default of **CACHE_METHOD** and **OVERRIDE_CACHE_METHOD**. In particular, setting the latter variables in */etc/eixrc* without adding the delayed substitution " **%{ADD_CACHE_METHOD}**" or " **%{ADD_OVERRIDE_CACHE_METHOD}**", respectively, will prevent the former variables to have any meaning.

Before eix-0.18.0, customizable version output properties like **<installedversions:***VAR***>** or **<availableversions:***VAR***>** did not exist. Instead there was a chaos of parameters for **<installedversions:*>** and a bulk of variants for printing available or installed versions for feeding the output to scripts like **<fullvailableversions>** etc. Now all this has vanished: The variables **NAMEVERSIONS**, **EQNAMEVERSION**, **ANAMESLOT**, **ANAMEASLOT**, **NAMESLOT**, **NAMEASLOT**, and **DATESORT** take the role of all the earlier variants (where **DATESORT** demonstrates a related example that was previously not available). See the description of these variables in **eix --dump** for details. To see the effect of such an example try e.g.

eix --format '<availableversions:ANAMESLOT:ANAMESLOT>' --pure-packages gcc

See also the comments for the **-I** option.

Since eix-0.20.0 the logical connectives for EXPRESSIONs have changed dramatically: Not only braces are now possible, also chains with **-a** and **-o** are now treated left-associative as most users would intuitively

expect. The negation **--not** is now treated as a logical operation starting a new BRACE_OR_TEST and no longer considered as part of TEST_OPTIONS (which was a source of confusion for many users). Now **--pipe** is really handled as part of TEST_OPTIONS and does not implicitly introduce logical connectives.

With eix-0.20.1 all the previously used files /etc/portage/package.*.nowarn are no longer supported by default: They were replaced by the single file/dir /etc/portage/package.nowarn. If you still want to use the previous files, set **OBSOLETE_NOWARN=true** in the environment. See the description of the variable **PACKAGE_NOWARN** (in the output of **eix --dump**) to understand why this works.

eix-installed was part of **eix-test-obsolete** before eix-0.22.4 which was rather confusing for users and for maintaining, since their task has nothing in common.

The default error-behavior of **eix-sync** has changed in eix-0.23.10: Previously, the **-F** option was automatic and could not be disabled.

In <eix-0.25.6 there was a **NEWLINE** variable which could add a magic newline after a package output. This hack had been only introduced to deal with obsolete FORMAT strings and has finally been removed.

The variables **COLORSTRING**, **COLORSTRING_ALT**, and **TERM_ALT** have been replaced by the more flexible variables **COLORSTRING?** and **TERM_ALT?**.

In **eix-remote** before eix-0.28.5 the current option **-x** was on by default. This would still be a reasonable default, but some users were confused by the feature (some even believed it is a bug) so that it is now necessary to configure this default manually (by setting **EIX_REMOTE_OPTS=-x** in */etc/eixrc*).

The tool **eix-header** did not exist before eix-0.28.6; instead eix had the options **--is-current**, **--print-overlay-path**, **--print-overlay-label**, **--print-overlay-data** which are now redundant and have been removed.

Since eix-0.29.6 the options **--format-verbose** and **--format-compact** have been removed, because their effect can be better obtained by exporting the environment variables **FORMAT_VERBOSE** or **FORMAT_COMPACT**, respectively. Instead, the option**--f ormat** overrides now the **FORMATSTRING** independently of the chosen format.

Since eix-0.29.6 the options **--verbose** and **--compact** are no longer toggling; for switching off, there is the new option **--normal**.

eix-functions.sh could previously only be sourced; only since eix-0.32.2, **eix-functions.sh** is also a program whose output is meant to be used with "eval".

TRANSLATION

For a simple fixed translation of the default **FORMATSTRING** to your language, it suffices to set the variables starting with I18N_... in your **/etc/eixrc** or **~/.eixrc**: Redirect the output of **eix --dump-defaults** to a file to get a list of all variables. Some of the last ones are those starting with I18N_.... you are looking for.

The variables starting with I18N_COLUMN_.... are special and can be used to adopt the columns to a new layout which might be necessary due to different lengths of the translations. (Note that the column counting will be garbled if you do not use UTF8 encoding for your language). You can see the default values of that variables in the corresponding variables C_COLUMN_.... which appear nearby (but which should not be modified for the purpose of translation). Normally, it should be sufficient to redefine **I18N_COLUMN_CONTENT** and **I18N_INST_COLUMN_CONTENT** to appropriate numbers. To make sure that all columns are correct, you should finally consider the output of all of the commands

eix -vxe binutils

eix -ve binutils

eix -le binutils -oe sun-jdk

eix -xle binutils

eix -vle binutils

eix -xvle binutils

eix -vle sun-jdk

Here, binutils means an **installed** and **slotted** package, while sun-jdk means a **masked** (with an explanation in package.mask) package.

Once you have your translation finished, you can send the changed variables to the eix maintainer or to eix on github or - even better: Add your language to po/LINGUAS, call contrib/make to generate a corresponding po/*.po file, and add your translation to that data in the file which refers to src/eixrc/def_i18n.cc

Of course, you are welcome to translate also other texts in that file: People might acknowledge even partial translations of the most important messages.

You can send your new po/*.po file to the eix maintainer or, even better: You can put your changes to github and make a pull request.

The ebuild has removed many previous USE-flags to follow gentoo policy. Therefore, the maintainer strongly recommends the usage of **EXTRA_ECONF** for installation, see Section **INSTALLATION**.

AUTHORS

Main authors of the programm:

Martin Väth <martin at mvath.de> (developer, current maintainer)

Emil Beinroth <emilbeinroth at gmx.net> (developer, previous maintainer)

Wolfgang Frisch <xororand at users.sourceforge.net> (inactive developer, initial author)

Roland Wittmann <linuxcommando at users.sourceforge.net> (inactive developer)

Many other contributors can be found in the **AUTHORS** file.

SEE ALSO

portage(5), **fnmatch**(3), **regex**(7), **emerge**(1), **esearch**(1), **qsearch**(1), **layman**(8)

The eix homepage *https://github.com/vaeth/eix/* provides further information and links.