Aegis
A Project Change Supervisor

Reference Manual

Peter Miller
millerp@canb.auug.org.au

.

This document describes Aegis version 3.15
and was prepared 3 May 1999.

NAME

aegis – project change supervisor

Aegis is distributed under the terms of the GNU General Public License. See the LICENSE section, below, for more details.

aegis (ee.j.iz) *n.,* a protection, a defense.

DESCRIPTION

Aegis is a CASE tool with a difference. In the spirit of the UNIX Operating System, *Aegis* is a small component designed to work with other programs.

Many CASE systems attempt to provide everything, from bubble charts to source control to compilers. Users are trapped with the components supplied by the CASE system, and if you don't like one of the components (it may be too limited, for instance), then that is just tough.

In contrast, UNIX provides many components of a CASE system – compilers, editors, dependency maintenance tools (such as make), source control tools (such as RCS). You may substitute the tool of your choice if you don't like the ones supplied with the system – gcc, jove, cake, to name just a few. *Aegis* adds to this list with software configuration management, and true to UNIX philosophy, *Aegis* does not dictate the choice of any of the other tools (although it may stretch them to their limits).

Enough hype, what is it that *Aegis* does? Just what is software configuration management? This question is sufficiently broad as to require a book in answer. In essence, *Aegis* is a project change supervisor. It provides a framework within which a team of developers may work on many changes to a program independently, and *Aegis* coordinates integrating these changes back into the master source of the program, with as little disruption as possible. Resolution of contention for source files, a major headache for any project with more than one developer, is one of *Aegis*'s major functions.

It should be noted that *Aegis* is a developer's tool, in the same sense as make or RCS are developer's tools. It is not a manager's tool – it does not provide progress tracking or manage work allocation.

BENEFITS

So why should you use *Aegis*?

Aegis uses a particular model of the development of software projects. This model has a master source (or baseline) of a project, and a team of developers creating changes to be made to this baseline. When a change is complete, it is integrated with the baseline, to become the new baseline. Each change must be atomic and self-contained, no change is allowed to cause the baseline to cease to work. "Working" is defined as passing it's own tests. The tests are considered part of the baseline. *Aegis* provides support for the developer so that an entire copy of the baseline need not be taken to change a few files, only those files which are to be changed need to be copied.

In order to ensure that changes are unable to cause the baseline to cease to work, *Aegis* mandates that changes be accompanied by at least one test, and that all such tests be known to complete successfully. These steadily accumulated tests form an ever increasing regression test suite for all later changes. There is also a mandatory review stage for each change to the baseline. While these requirements may be relaxed per-change or even per-project, doing so potentially compromises the "working" definition of the baseline.

The win in using *Aegis* is that there are *O(n)* interactions between developers and the baseline. Contrast this with a master source which is being edited directly by the developers - there are *O(n!)* interactions between developers - this makes adding "just one more" developer a potential disaster.

Another win is that the project baseline always works. Always having a working baseline means that a version is always available for demonstrations, or those "pre-release snapshots" we are always forced to provide.

The above advantages are all very well - for management types. Why should Joe Average Programmer use *Aegis*? Recall that RCS provides file locking, but only for one file at a time. *Aegis* provides the file locking, atomically, for the set of files in the change. Recall also that RCS locks the file the instant you start editing it. This makes popular files a project bottleneck. *Aegis* allows concurrent editing, and a resolution mechanism just before the change must be integrated, meaning fewer delays for J.A.Programmer.

Aegis also has strong support for geographically distributed development. It supports both push and pull models, and many distribution topologies. Aegis' normal development process is used to validate received change sets before committing them.

ARCHIVE SITE

The latest version of *Aegis* is available by HTTP from:

URL:	http://www.canb.auug.org.au/˜millerp/	
File:	aegis.html	# the Aegis page
File:	aegis.3.15.README	# Description, from tar file
File:	aegis.3.15.lsm	# Description, in LSM format
File:	aegis.3.15.ae	# the complete source, aedist format
File:	aegis.3.15.spec	# RedHat package specification
File:	aegis.3.15.tar.gz	# the complete source
File:	aegis.3.15.faq	# Frequently Asked Questions
File:	aegis.3.15.rm.ps.gz	# the Reference Manual, in PostScript
File:	aegis.3.15.ug.ps.gz	# the User Guide, in PostScript

This directory also contains a few other pieces of software written by me. Some are referred to in the *Aegis* documentation. Please have a look if you are interested.

Mirrors

See http://www.canb.auug.org.au/˜millerp/ for a list of mirror sites.

Aegis is also carried by `sunsite.unc.edu` in its Linux archives. You will be able to find Aegis on any of its mirrors.

URL:	ftp://sunsite.unc.edu/pub/Linux/devel/vc/	
File:	aegis.3.15.README	# Description, from tar file
File:	aegis.3.15.lsm	# Description, in LSM format
File:	aegis.3.15.spec	# RedHat package specification
File:	aegis.3.15.ae	# the complete source, aedist format
File:	aegis.3.15.tar.gz	# the complete source
File:	aegis.3.15.faq	# Frequently Asked Questions
File:	aegis.3.15.rm.ps.gz	# the Reference Manual, in PostScript
File:	aegis.3.15.ug.ps.gz	# the User Guide, in PostScript

This site is extensively mirrored around the world, so look for a copy near you (you will get much better response).

MAILING LIST

A mailing list has been created so that users of *Aegis* may exchange ideas about how to use *Aegis*. Discussion may include, but is not limited to: bugs, enhancements, and applications. The list is not moderated.

The address of the mailing list is

```
aegis-users@canb.auug.org.au
```

Please **do not** attempt to subscribe by sending email to this address. It is for content only.

How To Subscribe

To subscribe to this mailing list, send an email message to

```
majordomo@canb.auug.org.au
```

with a message body containing the single line

```
subscribe aegis-users
```

If you have an address which is not readily derived from your mail headers (majordomo is only a Perl program, after all) you will need to use a message of the form:

`subscribe aegis-users` *address*

where *address* is an email address to which you want messages sent.

No Files By EMail

The software which handles this mailing list **cannot** send you a copy of *Aegis*. Please use FTP or ftp-by-email, instead.

BUILDING

Instructions on how to build and test *Aegis* are to be found in the *BUILDING* file included in this distribution.

SOME HISTORY

The idea for *Aegis* did not come full-blown into my head in the shower, as some of my programs do, but rather from working in a software shop which used a simplistic form of something similar. That system was held together by chewing-gum and string, it was written in a disgusting variant of Basic, and by golly the damn thing worked (mostly). *Aegis* is nothing like it, owes none of its code to that system, and is far more versatile. It turns out that the system used is nothing new, and is described in many SCM textbooks; it is the result of systematically resolving development issues for large-ish teams.

Since that company decided to close down our section (the company was under attack by a hostile takeover bid) we all moved on simultaneously (all 60 of us), sometimes working together, and sometimes not, but always keeping in touch. With suggestions and conversations with some of them early in 1990, the manual entries for *Aegis* took shape, and formed most of the design document for *Aegis*.

Since getting the first glimmerings of a functional *Aegis* late in 1990 it is increasingly obvious that I never want to be without it ever again. All of my sources that I modify are instantly placed under *Aegis*, as is anything I distribute. All code I write for myself, and all new code I write for my employer, goes under *Aegis*. Why? Because it has fewer bugs!

Example: one of the sources I carry with me from job to job is "cook", my dependency maintenance tool. Cook had existed for 3 years before *Aegis* appeared on the scene, and I used it daily. When I placed cook under *Aegis*, I found 6 bugs! Since then I have found a few more. Not only are there now fewer bugs, but they never come back, because the regression test suite always grows.

Branching

In 1997 the full branching support was released (it took nearly 18 months to retro-fit. The underlying data structures for projects and change sets need to be merged. While I noticed back in 1990 that they were very similar, it wasn't until branch support design was well underways that they should have been the same data structure from the begining.

Geographically Distributed Development

In 1999 a conversation on the `aegis-users` mailing list resulted in the creation of *aedist*, a program which packages and unpackages Aegis changes so they can be sent by e-mail, or WWW or whatever. With 20:20 hindsight, this could have been done way back in 1991, because the basic idea builds on Aegis change process model.

Windows NT

Aegis depends on the underlying security provided by the operating system (rather than re-invent yet another security mechanism). However, in order to do this, Aegis uses the POSIX *seteuid* system call, which has no direct equivalent on Windows NT. This makes porting difficult. Single-user ports are possible (*e.g.* using Cygwin), but are not usually what folks want.

Compounding this is the fact that many sites want to develop their software for both Unix and Windows NT simultaneously. This means that the security of the repository needs to be guaranteed to be handled in the same way by both operating systems, otherwise one can act as a ‘‘back door’’ into the repository. Many sites do not have the same users and permissions (sourced from the same network register of users) on both Unix and Windows NT, making the mapping almost impossible even if the security models did actually correspond.

Most sites using Aegis and Windows NT together do so by running Aegis on the Unix systems, but building and testing on the NT systems. The work areas and repository are accessed via Samba or NFS.

LICENSE

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

RELEASE NOTES

For excruciating detail, and also acknowledgments of those who generously sent me feedback, please see the *etc/CHANGES.** files included in this distribution. A number of features have been added to *Aegis* with this release. A few of them are detailed here:

Version 3.15

• The "*aedist −receive*" command now accepts a **–delta** option, allowing a received change set to be applied to an historical version.

• There is now some information about managing super-projects and sub-projects in the *Branching* chapter of the User Guide.

• The *aenpr*(1) command now accepts a *−keep* option, so that you can re-attach projects moved after using the *aermpr −keep* command. See *aenpr*(1) for more information.

• The *aenpr*(1) command now accepts *−edit* and *−file* options, allowing you to specify project attributes when creating the project. See *aenpr*(1) for more information.

• If the project *developers_may_create_changes* attribute is true, the *aencu*(1) command now allows developers to destroy changes they created.

• There is a new *add_path_suffix* substitution, for manipulating search paths. See *aesub*(5) for more information.

• There are 3 new substitutions: ${bindir}, ${datadir} and ${libdir}. These are replaced by the *./configure* options of the same name (or the values calculated, if none were given to *./configure*). The old ${lib} substitution is deprecated in favour of the new ${datadir} substitution. See *aesub*(5) for more information.

• Some changes have been made which increases portability, particularly the Linux libc5 *vs* libc6 differences.

• Some changes have been made which increases portability, particularly for Windows NT. This isn't to say Aegis works under Windows NT yet, but it helps the porting efforts. Don't forget to run the mkpasswd and mkgroup utilities included in the Cygwin system.

Version 3.12

• The way the Apache config files are scanned for and read has been changed, to adapt to recent Apache changes. The *./configure* script will now find it more often.

• The "*aedist -receive*" command has been enhanced to be more robust about change sets without headers (some browsers *generously* strip them all off.

• A bug has been fixed in the "*aedist -receive*" command which sometimes caused decompression failures. An unfortunate interaction with the Windows NT support caused CRLF sequences in the compressed data to be mangled in some cases.

• The wrong include file was being used for zlib. This has been fixed, so it should build more easily now.

• The way MANPATH is handled on Linux has been improved in the *chsrc* and *profile* commands. It will not over-ride */etc/man.config* now.

• The *aegis.cgi* script has been made more robust in coping with *aedist* errors.

• The *symlink_exception* field of the project *config* file now accepts filename patterns, not simply literal filenames.

• There was a problem compiling with gcc 2.8, involving the <stdarg.h> header. This has been fixed.

Version 3.11

• The *aet*(1) command now accepts a `--force` option, forcing tests to be run, even if Aegis doesn't think they need to.

• The Aegis CGI interface has been enhanced so that you can download changes from the generated web pages listing the changes, using the *aedist* command.

• The *aedist --send* command now accepts a `--no-ascii-armor` option, which leaves off the MIME base 64 encoding. Useful for binary distributions and web servers.

• There is a new *trojan_horse_suspect* field in the project *config* file. This is used by *aedist --receive* to check for files which could be abused to carry Trojan horse attacks.

• The *aedist --receive* command now accepts a `--trojan` option which treats the incoming change set as suspect, and a `--no-trojan` option which treats the incoming change set as benign.

• The *aedist --receive* command now quotes filenames (if necessary) when executing commands, thus defending against filenames which contain semicolons.

• The *aenbru*(1) command has been implemented at last. At last! You no longer need to use the *aedbu* work-around.

• The *aedbu*(1) command now gives an error if you attempt to apply it to a branch.

• The *aermpr*(1) command may now be applied to a project with active branches, and will remove the branches as well (provided there are no active changes on any of the branches).

• The `dos_filename_required` and `windows_filename_required` fields of the project *config* file have been enhanced to reject the brain-dead Windows special filenames such as "aux" *et al*.

• The `${user}` and `${project}` substitutions have been enhanced to provide additional information when given an additional argument. Useful for file templates. See *aesub*(5) for more information.

• Several portability enhacements, notably the Windows filename incompatibility has been fixed, and also the Linux *stdlib.h* problem.

Version 3.10

• As of this release you must have **zlib** installed before you can build Aegis.

• There is a new *reuse_change_numbers* project attribute, letting you control whether *aenc* fills in holes in the change number sequence. Defaults to true if not set. See *aepattr*(5) for more information.

• There is a new *integrate_begin_exceptions* field in the project *config* file. This permits the user to specify file to be omitted when the integration directory copy/link is performed.

• The *aet*(1) command has been changed so that it does not exit with an error if you have a test exemption but no tests. This is no longer an error.

• There is a new *aedist*(1) command, which may be used to send and receive Aegis change sets via e-mail and the web.

• The *aeclone*(1), *aenbr*(1), *aenc*(1) commands now accept a **-output** option, a file to contain the automatically generated change number. This greatly assists in writing scripts. See the man pages for more information.

• The *aent*(1) command now accepts a **-output** option, a file to contain the automatically generated file name. See the man pages for more information.

• There is a new *compres_database* field in the project attributes, allowing the Aegis database to be stored in a compressed form (using the GNU Zip algorthm). Unless you have an exceptionally large project, coupled with fast CPUs and high network latency, there is probably very little benefit in using this feature. (The database is usually less than 5% of the size of the repository.) On slow networks, however, this can sometimes improve the preformance of file-related commands.

Version 3.9

• A bug in the merge command has been fixed. It no longer deletes all of your change source files if one of the merge commands fails.

• There is a new *tkaegis* command, using Tk/Tcl to give Aegis a GUI. Contributed by Graham Wheeler <gram@cdsec.com>. Please report *tkaegis* bugs and suggestions to Graham.

• The integrate pass command has been enhanced to cope with RCS and SCCS expanding keywords in source files (modifying the repository) on check-in. This can be ignored, or a warning can be issued, or it can be a fatal error (this is the default). See *aeip*(1) for more information.

• The worked example in the User Guide has (finally!) been updated to use the new branch numbering. Numerous spelling errors have been corrected.

• The developer section of the worked example chapter now also includes discussion of some common questions raised by folks evaluating Aegis. It covers insulating development directories from the baseline, partial check-in and collaboration.

• The *aesub*(5) man page now brings attention to the fact that the ${Copyright_Years} substitution contains spaces. You often need to quote it.

• The man pages which mention filename limitations, now also note that where underlying file-system has stricter filename length limitations than the *filename_maximum_length* field in the project *config* file, the file-system wins. Mention of this is now also present in *aedb*(1), *etc*; Linux UMSDOS is highlighted as problematic.

• Aegis can now collect code metrics. See *aeb*(1) and *aeipass*(1) for more information.

• There are three new report functions available: *quote_url*, *quote_html* and *unquote_url*. These are all for use when creating Aegis reports for the CGI interface. See *aer*(5) for more information.

• There are several new substitutions available. These include `subst`, `trim_extension`, `trim_directory`, and `trim_filename`. See *aesub*(5) for more information.

• The integrate_q.sh script now works correctly for branches.

• Numerous configure, make and install problems have been fixed for a variety of portability targets.

• The RPM spec file has been corrected to use appropriate file attributes.

Version 3.8

• Some users were unable to build the previous release, due to inconsistent wide character support by the various UNIX vendors. This has now been fixed.

• There are two new substitutions, *trim_directory* and *trim_extension*, which are useful for constructing file templates. These can be very useful in constructing skeletons of C++ classes.

• Some changes have been made to pathname handling to better cope with automounters. See *aegis*(1) for more information (see discussion of the AEGIS_AUTOMOUNT_POINTS environment variable). This assumes that paths below the automounter's mount directory are echoes of paths without it (*e.g.* `/home` is the trigger, and `/tmp_mnt/home` is where the NFS mount is performed, with `/home` appearing to be a symlink).

Version 3.7

• The *aeif*(1) and *aerf*(1) commands now have a new --reason option, to specify the failure reason on the command line, rather than in a file.

• Some file operations are now faster. Mostly, this applies to operations which mention many files, and to projects with large numbers of files. Smaller projects may not notice any improvement.

• There is a new **--delta-date** option to the *aecp*(1) command, allowing deltas to be extracted by date. This change also had the side-effect of making extraction by delta number more accurate on branches.

• There is a new **--base-relative** option to most of the file manipulation commands, *aecp*(1), *aenf*(1), *etc*. This option may be used to specify that relative filenames are relative to the base of the source tree, rather than the current directory. There is also a related user preference, see *aeuconf*(5) for more information.

• There is a new ‘‘aeclean’’ command. It can be used to clean your development directories of non-source files. See *aeclean*(1) for more information.

• The *aeb*(1) command now passes through arguments of the form *name***=***value*, on the assumption that these are variable assignments for th ebuild tool. Previously, they were ‘‘resolved’’ as if they were file names.

• A serious bug in the error and interrupt handling has been fixed. This bug would sometimes case Aegis to hang, and eventually run out of stack, when the user attempted to interrupt Aegis using ˆC.

Version 3.6

• The *diff3_command* field of the project *config* file has been replaced by a *merge_command* field. It works exactly the same way, but Aegis moves the files around first, so that the output replaces the change source file. This results in fewer ‘‘lost’’ merges. Those of you who have been hacking the *diff3_command* to move the files around will need to take the moves **out** when you rename the *diff3_command* field to be the new *merge_command* field.

• The columnizing functions used by the report generator and the listings has been enhanced to understand international character sets. This allows native character sets to be used in comments and descriptions, without getting gibberish (C escapes) in the output.

• There is a new *shell_safe_filenames* field in the project *config* file. This field controls whether filenames are required to be free of shell special characters. This field defaults to true if not set, so if you are using any ‘‘interesting’’ filenames, you may need to explicitly set this field to false. (You still can’t use spaces or international characters in filenames.)

• There is a new ${quote} substitution for insulating shell special characters in filenames in the commands in the project *config* file.

• A number of bugs relating to environment variables have been corrected; this will make the aereport and aefind commands behave more consistently, with respect to the aegis command.

• A bug has been fixed which caused the final newline of new test files to be omitted.

• A bug which prevented the ‘‘aeb -minimum’’ option from working in any non-trivial case has been fixed.

Version 3.5

• A bug was fixed in the lock waiting code. Aegis will now correctly wait for locks when there are several users blocking on the same lock.

Version 3.4

• There is a new ‘‘aegis -clone’’ command, used to replicate changes across branches. See *aeclone*(1) for more information.

• There is a new ‘‘-No-Wait’’ command line option, which asks for a fatal error if a lock cannot be obtained immediately; this applies to all commands which takes locks. See *aeuconf(5) for more information.*

Version 3.3

This release is a bug fix release, and mostly install and portability bug, at that.

• The problem with *errno* defines messing up *glue.c* has been fixed.

• Numerous fixes to the wide character support, to cope with the vagueries of wide character support on many platforms.

• The problem with the LINES and COLS environment variables messing up testing have been fixed. Some tests gave false negatives because of this.

• There is a new *aeb -minimum* option, for use with symbolic links, only, which has a minimal set of source file links, rather than everything in the baseline.

Version 3.2

• There are some additional reports available via the web interface. They are mainly to extract error causes and trends from the project history statistics.

• There have been a number of minor bug fixes concerning the handling of old 2.3 projects. This should ease transition for users with existing 2.3 projects.

• A bug in aecp -delta has been fixed, where Aegis was trying to find change state files one branch level too high.

• There is now a re-try performed when a stale NFS file handle error is detected. This should make it easier for some sites which are heavily networked.

• There have been some improvements to the way Control-C is handled. It should be more responsive when waiting for locks.

• Project administrators may now end development of a branch. Since branches can endure for months or years, the orginal branch creator may have moved on. This copes with this situation.

Version 3.1

Version 3.0 was not used by many sites. It was available as beta software for about a year, in numerous incarnations. Version 3.1 is the first completely stable version since adding full branching support.

Version 3.0

Version 3.0 is fully backwards compatible with earlier versions, however once a project has been used under 3.0, it will not be possible to revert, e.g. to version 2.3, without restoring the project's "info" directory from backup. While this was generally true of previous releases, any additional state information was usually undo-able with *vi*(1). This time the process is much more involved because the project state files and the change state files have been combined as a necessary step in implementing branches.

Version 3.0 Major New Features

• Aegis now has a feature known in the literature as long transactions, also known as branches. This allows appropriately created changes to be treated as if they were projects, and thus to have changes made to them. This allows a hierachy of changes within changes, to any desired depth. See the *Branching* chapter of the *Aegis User Guide* for more information.

• The project state files have been merged with the change state files. This is part of the implementation of branching. If you have written your own reports, you may need to alter them slightly. For example, in version 2.3 and earlier, reports accessed the project state file using

```
auto p, ps;
p = project[project_name()];
ps = p.state;
```

Because the project state has been moved into a change state, the *state* field above now points at a change state description, and most of the old project information is contained in the *branch* field within it. Reports access this information as

```
auto p, ps;
p = project[project_name()];
ps = p.state.branch;
```

Except for files, which were already present in the *cstate*, so access to the project file list need not change. See the new *aecstate*(5) for more information.

• The new project command now creates branches to match the version number specified. See *aenpr*(1) for more information.

• The error messages of Aegis have been internationalized. This affects how you build Aegis, and the environment Aegis runs in. See the *BUILDING* file for more information. The *cshrc* and *profile* shipped with this release set the LANG environment variable to ‘‘en’’ (for English) if you have not set it; otherwise ther error messages would be terse and uninformative.

• The *aet (1)* command can now suggest tests to be run. This is done by correlating the source files and test files from each change. See *aet*(1) for more information.

• There is now an *aereport*(1) command. The separates out the report functionality from the main body of the Aegis code, allowing the report generator to be used in places where more trust is required.

• There is an intranet Web interface, which is installed automaticly when the install script discovers a web server. This interface allows browsing of much of the Aegis meta-data, of all publicly accessible projects.

• There is now an *aefind*(1) command. This is very similar to the UNIX *find*(1) command, except that it finds in the unified directory stack of a change and its project. The introduction of full branch support can sometimes mean that finding a file may require looking in more than two directories; the *aefind*(1) command makes this simple again.

Version 3.0 Minor New Features

• There is now a -No_Pager option, to prevent listings and help from being redirected to a pager. There is also a user preference to more thoroughly disable paging, and a -PAGer option to override it. See *aegis*(1) and *aeuconf*(5) for more information.

• There is now a -No_PErsevere option to *aet*(1), allowing you to request that *aet*(1) stop after the first test failure. There is also a user preference to set this permanently if desired, and a -PErsever option to override it. See *aet*(1) and *aeuconf*(5) for more information.

• The copyright years attribute has been moved from being a project attribute to a change attribute. This is consistent with a number of other fields which have transparently moved from the project state files into the change state files, as a result of branching support. See *aeca*(1) and *aecattr*(5) for more information.

• There is a new *Search_Path* substitution, to support builds on branches. See *aesub*(5) for more information. As a side effect, you can also use it in the *test_command* field of the project *config* file, and thus have a search path to look down for data files for your tests.

• Test times are now remembered, so that tests are only run if they need to be. This allows you to keep working on a test, and Aegis only runs those that have not yet passed.

• Aegis now uses ‘‘fingerprints’’ to tell if files have changed, rather than simply relying on file modification time stamps. While this makes Aegis more robust, there is one caveat: it is recommended that 3.0 be installed when there are no changes in the ’being reviewed’ or ’awaiting integration’ states, in any project.

• There is now a log file prefernces control, allowing users to set their preferred logging behaviour. See *aeuconf*(5) for more information.

• It is now possible to specify the filename for new tests on the command line. See *aent*(1) for more information.

• It is not possible to specify a pattern for test filesnames. See *aepconf*(5) for more information.

• There is now a **−MAXimum** option to the *aeib* command, allowing you to keep obsolete derived files at integrate begin. This can avoid long integration build times for large projects.

Version 3.0 Bug Fixes

• Architecture names are now checked a 'develop end' time, to ensure there are no unknown variants. This fixes the mysterious ''you must build again'' problem.

• The *aecp*(1) and *aed*(1) commands now take a baseline read lock, to be more symetric with the *aeb*(1) command which has always done so. The *aeip*(1) command takes the complementary baseline write lock, ensuring the the baseline remains constant for the duration of builds, file copies, differences and merges. The manual entries for these commands have all been improved to document this behaviour. See *aeb*(1), *aecp*(1), *aed*(1) and *aeip*(1) for more information.

• There are now some reminder scripts in the library, which can be run from *crontab*(1). These are installed into the */remind* directory. These scripts can be used to remind users of changes in various states, such as those being developed or being reviewed.

• All of the commands which accept the *-Edit* option now also accept a *-Edit_BackGround* option, allowing edit commands to be piped in from the standard input.

• The *aecp*(1) command now accepts a *-INDependent* option, allowing files to be copied independent of any change (similar to the *-INDependent* option of the command.) See *aecp*(1) for more information.

• The *aecp*(1) command now accepts a *-Read_Only* option, allowing files to be copied into a change specifically to insulate it from baseline changes. Such files must be uncopied before development may end. See *aecp*(1) for more information.

• The *aenrls*(1) command is now used *only* to convert pre-3.0 projects into post-3.0 projects. This is because the full branching support in 3.0 makes it more useful to create a new release of a project by ending development on the branch of the previous release and starting development of a new branch numbered for the new release. See the *Branching* chapter of the *User Guide* for more information.

Version 2.3

• The merging behaviour of the *aed*(1) command has changed. If any files require merging, it only merges. In this way, merged files are not lost in the rest of the output. Also, there are now command line options and user preferences so that you can select to only merge or only difference. See *aed*(1) and *aeuconf*(5) for more nformation.

• It is now possible to assign symbolic names to project deltas. This means that you may now recreate earlier project baselines by name.

• All commands which accept a **-Edit** option now check for most errors before commencing the edit. This avoids wasted edits in many error cases.

• Fuzzy file name matches are now used to improve the error messages from aecp, aerm, etc.

• Version number separators in project names are preserved across new releases. Particularly, you can use a minus ('-') between the name and the major version number.

• A new ‘‘copyright_years’’ project attribute has been added. This is a list of years maintained at integrate begin time, to automate the insertion of list of copyright years into copyright messages and documentation. There is a new ${Copyright_Years} substitution and the copyright years are also listed in the ‘‘aegis -list version’’ listing. See *aesub*(5) and *ael*(1) for more information.

• It is now possible to specify patterns for acceptable and unacceptable filenames in the project *config* file. See *aepconf*(5) for more information.

• Four more functions have been added to the report language: *length*, *split*, *substr* and *wrap*. See *aer*(5) for more information.

• The tests distributed with are now more stable on very fast hosts. See the environment variables section of *aeb*(1) for more information.

• The *lib/config.example* directory of the distribution now contains files with example portions of the project *config* file. May thanks to David R Shue <shue@ll.mit.edu> for this suggestion.

Changes made in the previous release included:

Version 2.2

This release of Aegis provides 3 of the most commonly requested features: support for heterogeneous development, support for a greater range of DMTs, support for user-defined reports.

• Aegis now supports heterogeneous development. Now you can be sure that your project not only always builds and tests sucessfully, but that it does so across a configurable set of system or hardware architectures. See the *Heterogeneous Development* secion of the *Tips and Traps* chapter of the User Guide for more information.

• Aegis can now cope with a wider range of Dependency Maintenance Tools (DMTs). It now has the ability to fill development directories with symbolic links to all files in the baseline which are not present in the development directory. This allows DMTs to assume all files are present below the current directory, allowing DMTs such as *cake* and *GNU Make* to be used. See the *Dependency Maintenance Tool* section of the User Guide and *aeb*(1) for more information.

• Aegis now has a report generator, so you can create your own reports. Many "canned" reports are included in this distribution; of particular interest to many will be the *File_Activity* report, which details currently active files. See *aer*(1) for more information.

• Aegis is now configured using a shell script called *configure*, distributed with the package. This shell script is generated using GNU Autoconf. See the *BUILDING* file for more information.

• The *AEGIS* environment variable has been renamed *AEGIS_PATH*, to bring it in line with the AEGIS_PROJECT and AEGIS_CHANGE environment variable names. The old name will keep working for some time, but aegis will warn you.

• Filename lengths are now configurable. The 14 character portability limit is still the default, but a higher limit is configurable for each project, up to the filesystem filename limit. See *aepconf*(5) for more information.

• It is now possible to specify that filenames must be within the minimum character set mandated by POSIX. The default is as before, to allow any printing character. See *aepconf*(5) for more information.

• Limits on the length of project names have been relaxed. Project names are now only limited by the filesystem filename limit.

• It is now possible to specify the command to run tests, allowing a project to use a specialized test facility, rather than be forced to use shell scripts. See *aet*(1) and for more information.

• The commands which accept the *-Edit* now preserve the edited text in the event of a failure.

• The commands which delete files now accept a *-Interactive* option, which causes them to prompt the user for confirmation of file deletion. This can be made the default by an appropriate setting of the aliases or individual users preferences files. See *aenfu*(1), *aentu*(1), *aecpu*(1), and *aeuconf*(5) for more information.

• The *aecp*(1) command now accepts directory names, allowing whole directory trees to be copied into a change. The *aecpu*(1) command now has a **-UNChanged** option which allows the unchanged files to be uncopied.

• The *aeb* command now accepts file names, allowing partial builds to be performed. See *aeb*(1) for more information.

• There is a new *aechown*(1) command to facilitate reassigning the developer of a change which is in the *being developed* state.

• It is now possible for project administrators to assign changes to specific developers. See *aedb*(1) for more information.

Version 2.1

• Can now ask for history to maintained for file generated by the build. This is useful for generating patch files.

Version 2.0

• A new command has been added to facilitate changing the name of a file as part of a change. See *aemv*(1) for more information.

• It is now possible to list the locks currently held. See *ael*(1) for more information.

• If no other defaulting mechanism is specified, aegis will now attempt to guess the project name and change number from the pathname of the current directory. This only works from within development directories.

• The *aenc*, *aeca*, *aerfail*, *aeifail* and *aepa* commands now accept a **-Edit** command line argument. See the relevant manual pages for more information.

• The *aenpr* command now understands the **-MAJor** and **-MINOr** options, allowing the initial version of a project to be something other than 1.0.

• The *aed* command now understands the **-ANticipate** option. See *aed*(1) for more information.

• It is now possible to list all the outstanding changes of a project, or of all projects. See *ael*(1) for more information.

Version 1.4

• Support has been added for systems without the *seteuid* system call, or those with crippled implementations.

• Most of the unimplemented command variants have been finished. These include *New Change Undo*, *Develop Begin Undo* and *ReMove PRoject*. Most notable of the exceptions is *-ANticipate* option for the *-CoPy_file* and *-DIFFerence* command.

• The User Guide has been added to, making it a little more complete. It still needs more work, sigh.

• The code to handle automounters has been made more robust.

• The command substitutions have been vastly improved, and are now documented.

NAME

aegis – project change supervisor

The *aegis* program is distributed under the terms of the GNU General Public License. See the LICENSE section, below, for more details.

aegis (ee.j.iz) *n.,* a protection, a defence.

SPACE REQUIREMENTS

You will need up to 15MB to unpack and build the *aegis* program. (This is the worst case seen so far, most systems have binaries about 60% as big as this, 10MB is more typical.) Your mileage may vary.

SITE CONFIGURATION

The **aegis** package is configured using the *configure* shell script included in this distribution.

The *configure* shell script attempts to guess correct values for various system-dependent variables used during compilation, and creates the *Makefile* and *common/config.h* files. It also creates a shell script *config.status* that you can run in the future to recreate the current configuration.

Before You Start

Before you start configuring, it is worth reading the *OTHER USEFUL SOFTWARE* section, below.

The *configure* script checks for the internationalization library and functions. If your system does not have them, it is worth fetching and installing **GNU Gettext** before you run the *configure* script. Make sure that the *msgfmt* command from GNU Gettext appears earlier in your command search PATH than the existing system ones, if any (this is very important for SunOS and Solaris). You must do the GNU gettext install *before* running the *configure* script, or the error messages, even for English speakers, will be terse and uninformative. Remember to use the GNU gettext configure *--with-gnu-gettext* option if your system has native gettext tools.

The *configure* script checks for the compression library and functions. If your system does not have them, it is worth fetching and installing **GNU zlib** compression library before you run the *configure* script. (See `http://www.cdrom.com/pub/infozip/zlib/` for download.) This is essential, becaise Aegis will not build. without it. (**Note: this is not the same thing as** `zlib`**c which does something completely different.)**

The *configure* script checks for the regular expression library and functions. If your system does not have them, it is worth fetching and installing **GNU rx** compression library before you run the *configure* script. (Note: test 81 will fail if the POSIX regular expression functions are not available.)

Running Configure

Normally, you just *cd* to the directory containing *aegis*' source code and type

```
% ./configure
...lots of output...
%
```

If you're using *csh* on an old version of System V, you might need to type

```
% sh configure
...lots of output...
%
```

instead to prevent *csh* from trying to execute *configure* itself.

Running *configure* takes a minute or two. While it is running, it prints some messages that tell what it is doing. If you don't want to see the messages, run *configure* with its standard output redirected to */dev/null*; for example,

```
% ./configure --quiet
%
```

There is a known problem with GCC 2.8.3 and HP/UX. You will need to set `CFLAGS = -O` in the generated Makefile. (The configure script sets it to `CFLAGS = -O2`.) This is because the code optimization breaks the fingerprints. If test 32 fails (see below) this is probably the reason.

By default, *configure* will arrange for the *make install* command to install the **aegis** package's files in */usr/local/bin*, */usr/local/com/aegis*, */usr/local/lib/aegis*, */usr/local/man* and */usr/local/share/aegis*. There are a number of options which allow you to control the placement of these files.

`--prefix=`*PATH*
: This specifies the path prefix to be used in the installation. Defaults to */usr/local* unless otherwise specified.

`--exec-prefix=`*PATH*
: You can specify separate installation prefixes for architecture-specific files and architecture-independent files. Defaults to *${prefix}* unless otherwise specified.

`--bindir=`*PATH*
: This directory contains executable programs. On a network, this directory may be shared between machines with identical hardware and operating systems; it may be mounted read-only. Defaults to *${exec_prefix}/bin* unless otherwise specified.

`--datadir=`*PATH*
: This directory contains installed data, such as the documentation, reports and shell scripts distributed with Aegis. On a network, this directory may be shared between all machines; it may be mounted read-only. Defaults to *${prefix}/share/aegis* unless otherwise specified. An "aegis" directory will be appended if there is none in the specified path.

`--libdir=`*PATH*
: This directory contains installed data, such as the error message catalogues. On a network, this directory may be shared between machines with identical hardware and operating systems; it may be mounted read-only. Defaults to *${exec_prefix}/lib/aegis* unless otherwise specified. An "aegis" directory will be appended if there is none in the specified path.

`--mandir=`*PATH*
: This directory contains the on-line manual entries. On a network, this directory may be shared between all machines; it may be mounted read-only. Defaults to *${prefix}/man* unless otherwise specified.

`--sharedstatedir=`*PATH*
: This directory contains share state information, such as the Aegis lock file, and information on the location of the various Aegis projects. On a network, this directory may be shared between all machines; it **must** be mounted **read-write**. Defaults to *${prefix}/com/aegis* unless otherwise specified. An "aegis" directory will be appended if there is none in the specified path.

configure ignores any other arguments that you give it.

On systems that require unusual options for compilation or linking that the *aegis* package's *configure* script does not know about, you can give *configure* initial values for variables by setting them in the environment.

In Bourne-compatible shells, you can do that on the command line like this:

```
$ CC='gcc -traditional' LIBS=-lposix ./configure
...lots of output...
$
```

Here are the *make* variables that you might want to override with environment variables when running *configure*.

Variable: CC
: C compiler program. The default is *cc*.

Variable: INSTALL
: Program to use to install files. The default is *install* if you have it, *cp* otherwise.

Variable: LIBS
: Libraries to link with, in the form `-l`*foo* `-l`*bar*. The *configure* script will append to this, rather than replace it.

If you need to do unusual things to compile the package, the author encourages you to figure out how *configure* could check whether to do them, and mail diffs or instructions to the author so that they can be included in the next release.

Common Problem

It is very common that other packages, such as *gettext*, *rx* and *zlib* are installed using */usr/local* as the prefix. However, the configure script can't work this out, even when it, too, is using */usr/local* as the prefix.

To cope with this, you need to say

```
$ CFLAGS=-I/usr/local/include LDFLAGS=-L/usr/local/lib ./configure
...lots of output...
$
```

when running configure. Substitute the appropriate prefix if you are using something other than */usr/local*. Watch the output... it should now find your installed packages correctly.

PRIVILEGES

There are a number of items in the generated *Makefile* and *common/config.h* file which affect the way *aegis* works. If they are altered too far, *aegis* will not be able to function correctly.

AEGIS_MIN_UID
: This specifies the minimum unprivileged uid on your system. UIDs less than this may not own projects, or play any other role in an aegis project. The default value is 100.

AEGIS_MIN_GID
: This specifies the minimum unprivileged GID on your system. GIDs less than this may not own projects, or play any other role in an aegis project. The default value is 10.

AEGIS_USER_UID
: This is the owner of files used by *aegis* to record pointers to your projects. It is *not* used to own projects (i.e. it must be less than AEGIS_MIN_UID). If possible, the *configure* script tries to work out what value was used previously, but you must specify the --prefix option correctly for this to work. Because of operating system inconsistencies, this is specified numerically so that *aegis* will work across NFS. The default value is 3.

AEGIS_USER_GID

This is the group of files used by *aegis* to record pointers to your projects. It is *not* used as the group for projects (i.e. it must be less than AEGIS_MIN_GID). If possible, the *configure* script tries to work out what value was used previously, but you must specify the --prefix option correctly for this to work. Because of operating system inconsistencies, this is specified numerically so that *aegis* will work across NFS. The default value is 3.

DEFAULT_UMASK

When *aegis* runs commands for you, or creates files or directories for you, it will use the defined project umask. This is a project attribute, and may be altered using the *aepa*(1) command. The DEFAULT_UMASK is the umask initially given to all new projects created by the *aenpr*(1) command. The default value of DEFAULT_UMASK is 026. See the comments in the *common/config.h* file for an explanation of the alternatives.

It is required that *aegis* run set-uid-root for all of its functionality to be available. It is **not** possible to create an "aegis" account and make *aegis* run set-uid-aegis. This is because *aegis* does things as various different user IDs, sometimes as many as 3 in the one command. This allows *aegis* to use UNIX security rather than inventing its own, and also allows *aegis* to work across NFS. To be able to do these things, *aegis* must be set-uid-root. Appendix D of the *Aegis User Guide* explains why *aegis* must run set-uid-root; please read it if you have concerns.

BUILDING AEGIS

All you should need to do is use the

```
% make
...lots of output...
%
```

command and wait. When this finishes you should see a directory called *bin* containing three files: *aegis*, *aereport*, *aefind*, *aefp*, *find_sizes* and *fmtgen*.

aegis The *aegis* program is a project change supervisor.

aefp The *aefp* program may be used to ‘‘fingerprint’’ files.

aereport The *aereport* program is used to query Aegis’ database.

aefind The *aefind* program is used to find files.

find_sizes

The *find_sizes* program is a utility used to build the *aegis* package; it is not intended for general use and should not be installed.

fmtgen The *fmtgen* program is a utility used to build the *aegis* package; it is not intended for general use and should not be installed.

You can remove the program binaries and object files from the source directory by using the

```
% make clean
...lots of output...
%
```

command. To remove all of the above files, and also remove the *Makefile* and *common/config.h* and *config.status* files, use the

```
% make distclean
...lots of output...
%
```

command.

The file *aux/configure.in* is used to create *configure* by a GNU program called *autoconf*. You only need to know this if you want to regenerate *configure* using a newer version of *autoconf*.

OTHER USEFUL SOFTWARE

Before describing how to test *aegis*, you may need to grab some other free software, because the tests require it in some cases, and because it is generally useful in others.

cook This is a dependency maintenance tool (DMT). An example of a well-known DMT is *make*(1), however this old faithful is mostly not sufficiently capable to meet the demands placed on it by the *aegis* program, but *cook* certainly is. The *cook* package is written by the same author as *aegis*. The *cook* package is necessary if test 11 is to be meaningful. It is also used in the documentation. The *cook* program may be found at the same archive site as the *aegis* program. The *cook* program is available under the terms of the GNU General Public License.

GNU diff

If the *diff*(1) utility supplied by your flavor of unix does not have the **-c** option, you will need GNU diff to make patch files, if you want to publish software for FTP or on USENET. Context differences are also helpful for reviewing changes.

RCS This is a source control package, and is available from any of the GNU archives. The tests use RCS as the history mechanism, so it is necessary to have RCS for most of the tests to pass. (It is best to compile and install RCS *after* GNU diff. This is because the RCS configuration hardcodes the pathnames of the GNU diff utilities it needs into the RCS executables.)

GNU Gettext

Many systems do not yet supply the *gettext*(3) function. Aegis uses this function to internationalize its error messages. If your system does not have this function, you should fetch and install GNU Gettext *before* running the *configure* script. If you do not, Aegis will still work, but the error messages will be rather terse, even for English speakers. (You will be able to tell if your system has the internationalization library and functions, because the *configure* script will report finding `-lintl` and *(CWlibintl.h* and `msgfmt` in its running commentary.) Please note that the GNU Gettext implementation is likely to be superior to the one supplied with your system, if any. Remember to use the GNU gettext configure *--with-gnu-gettext* option if your system has native gettext tools.

Please note: if you install GNU gettext package into */usr/local* (for example) you must ensure that the Aegis *./configure* script is told to also look in */usr/local/include* for include files (CFLAGS), and */usr/local/lib* for library files (LDFLAGS). Otherwise the not been installed.

GNU Groff

This GNU software replaces the documentation tools which (sometimes) come with UNIX. They produce superior error messages, and support a wider range of functionality and fonts. The *aegis* User Guide was prepared with groff.

GNU Groff patch

There is a patch for GNU Groff available, which gives *groff*(1) and *gsoelim*(1) a **–I***path* option. This adds an include file search path, similar to *cpp*(1). The Makefile assumes this is available. The patch is at `http://www.canb.auug.org.au/~millerp/groff-1.10.patch.gz` or its mirrors.

bison This GNU software is a replacement for *yacc*(1). Some systems have very sick yaccs, and *bison* may be necessary if your system include files disagree strongly with your system's yacc. The generated *Makefile* will use bison if you have it.

fhist This software, available under the terms of the GNU General Public License, is a set of file history and comparison utilities. It was originally written by David I. Bell, and is based on the minimal difference algorithm by Eugene W. Myers. This copy is enhanced and maintained by the same author as *aegis*, and may be found at the same archive site, in the same directory.

rx This library provides POSIX regular expressions, for systems which don't have them. (Note: test 81 will fail if the POSIX regular expression functions are not available.)

zlib This library provides access to the GNU Zip (de)compression algorithm(s). It is essential to have this installed before you build Aegis. The home page may be found at `http://www.cdrom.com/pub/infozip/zlib/` if you need to download it. Note: this is not the same as **`zlibc`** which is Linux specific.

The tests also depend on the presence of a number of common UNIX programs, including but not limited to: *cc*, *cmp*, *diff*, **ed**, *find*, *make*, the *RCS* package, etc. Depending on your version of UNIX, some or all of these programs may be in optional packages. (This is especially true of Linux.) You need to ensure that these programs are correctly installed before you run the tests.

TESTING AEGIS

The *Aegis* program comes with a test suite. To run this test suite, use the command

```
% make sure
...lots of output...
Passed All Tests
%
```

The tests take a minute or two each, with a few very fast, and a couple very slow, but it varies greatly depending on your CPU.

Known Problems

The tests assume that the RCS commands "ci", "co", "rlog" and "rcs" are somewhere in the command search PATH.

The *test/00/t0011a.sh* file assumes the *cook*(1) command by the author is somewhere in the command search path. This test reproduces the example used in Chapter 3 of the User Guide. If the *cook*(1) command is not available, this test gives a pass result without testing anything.

If you are using HPUX and GCC, test 32 fails if you use -O2. You need to edit the Makefile to only optimize at -O, delete the objects and rebuild.

If you are using Sun's *tmpfs* file system as your */tmp* directory, the tests will fail. This is because the *tmpfs* file system does not support file locking. Set the *AEGIS_TMP* environment variable to somewhere else before running the tests. Something like

```
% setenv AEGIS_TMP /usr/tmp
%
```

is usually sufficient if you are using C shell, or

```
$ AEGIS_TMP=/usr/tmp
$ export AEGIS_TMP
$
```

if you are using Bourne shell. Remember, this must be done before running the tests.

If the tests fail due to errors complaining of "user too privileged" you will need to adjust the *AEGIS_MIN_UID* defined in the *common/config.h* file. Similarly for "group too privileged", although this is rarer. This error message will also occur if you run the tests as root: the tests must be run as a mortal each time.

If the POSIX regular expression functions are not available, test 81 will fail. The GNU rx library provides these. Installing it and re-configuring and re-building Aegis will solve the problem.

TESTING SET-UID-ROOT

If the *Aegis* program is not set-uid-root then it runs in "test" mode which gives you some confidence that *Aegis* is working before being tested again when it is set-uid-root. Two pass testing like this means that you need not trust your system to a set-uid-root program which is not known to work.

You will need to do a little of the install, to create the directory which will contain *Aegis*' lock file.

```
# make install-libdir
mkdir /usr/local/lib/aegis
chown 3 /usr/local/lib/aegis
chgrp 3 /usr/local/lib/aegis
chmod 0755 /usr/local/lib/aegis
mkdir /usr/local/com/aegis
chown 3 /usr/local/com/aegis
chgrp 3 /usr/local/com/aegis
chmod 0755 /usr/local/com/aegis
chown root bin/aegis
chmod 4755 bin/aegis
#
```

As you can see, the previous command also changed *Aegis* to be set-uid-root. Once this has been done, *Aegis* should be tested again, in the same manner as before.

```
% make sure
...lots of output...
Passed All Tests
%
```

You should test *Aegis* as a mortal in both passes, rather than as root, to be sure the set-uid-root functionality is working correctly.

It is required that *Aegis* run set-uid-root for all of its functionality to be available. It is **not** possible to create an "aegis" account and make *Aegis* run set-uid-aegis. This is because *Aegis* does things as various different user IDs, sometimes as many as 3 in the one command. This allows *Aegis* to use UNIX security rather than inventing its own, and also allows *Aegis* to work across NFS. To be able to do these things, *Aegis* must be set-uid-root. Appendix D of the *Aegis User Guide* explains why *Aegis* must run set-uid-root; please read it if you have concerns.

INSTALLING AEGIS

As explained in the *SITE CONFIGURATION* section, above, the *Aegis* package is installed under the */usr/local* tree by default. Use the `--prefix=`*PATH* option to *configure* if you want some other path.

All that is required to install the *Aegis* package is to use the

```
% make install
...lots of output...
%
```

command. Control of the directories used may be found in the first few lines of the *Makefile* file if you want to bypass the *configure* script.

The above procedure assumes that the *soelim*(1) command is somewhere in the command search *PATH*. The *soelim*(1) command is available as part of the *GNU Groff* package, mentioned below in the *PRINTED MANUALS* section. If you don't have it, but you do have the *cook* package, then a link from *roffpp* to *soelim* will also work.

The above procedure also assumes that the *$(prefix)/man/man1* and *$(prefix)/man/man5* directories already exist. If they do not, you will need to *mkdir* them manually.

USER CONFIGURATION

The *aegis* command is assumed to be in a generally accessible place, otherwise users will need to add the relevant directory to their PATH. Users should add

```
source /usr/local/lib/aegis/cshrc
```

to the end of their *.cshrc* file for the recommended aliases. (This may be in a different place, if you specified a --prefix option.)

There is also a *profile* for users of the Bourne shell (it assumes you have a version of the Bourne shell which has functions). Users should add

```
. /usr/local/lib/aegis/profile
```

to the end of their *.profile* file for the recommended aliases. (This *profile* assumes that users are using a Bourne shell which understands functions.)

The */usr/local/lib/aegis/state* file contains pointers to "system" projects. Users may add their own project pointers (to their own projects) by putting a search path into the *AEGIS_PATH* environment variable. The system part is always automatically appended by *Aegis*. The default, already set by the */usr/local/lib/aegis/cshrc* file, is *$HOME/lib/aegis*. Do not create this directory, *Aegis* is finicky and wants to do this itself.

Where projects reside is completely flexible, be they system projects or user projects. They are not kept under the */usr/local/lib/aegis* directory, this directory only contains pointers.

If you have a Web server, and the `aegis.cgi` was installed, you can set its *AEGIS_PATH* environment variable, if you want it to be able to see more projects than just the global projects. You do this by creating a */usr/local/lib/aegis.cgi.conf* file (there isn't one, by default) and setting the *AEGIS_PATH* environment variable in it. This is a fragment of Bourne shell script, not just the name.

PRINTED MANUALS

This distribution contains the sources to all of the documentation for *Aegis*, however the simplest way to get the documentation is by anonymous FTP; PostScript files of the User Guide and Reference Manual are available from the FTP sites listed in the README file.

The Reference Manual contains the README and BUILDING files, as well as all of the section 1 and section 5 manual pages. The Reference Manual is about 200 pages long.

The User Guide contains information about how to use Aegis, including a fully worked example. The User Guide is about 100 pages long.

TIME SYNCHRONIZATION

The *aegis* program uses time stamps to remember whether various events have happened and when. If you are using *Aegis* in a networked environment, typically a server and data-less workstations, you need to make absolutely sure that all of the machines agree about the time.

If possible, use the time daemon. Otherwise, use *rdate*(8) via *cron*(8) every hour or less.

GETTING HELP

If you need assistance with *Aegis*, please do not hesitate to contact the author at

Peter Miller <millerp@canb.auug.org.au>

Any and all feedback is welcome.

When reporting problems, please include the version number given by the

```
% aegis -version
aegis version 3.15
...
%
```

command. Please run this command to get the exact number, do not send the text of this example.

Runtime Checking

In the *common/main.h* file, there is a define of *DEBUG* in comments. If the comments are removed, extensive debugging is turned on. This causes some performance loss, but performs much run-time checking and adds the **-TRAce** command line option.

When the **-TRAce** command line option is followed by one or more file names, it turns on execution traces in those source files. It is usually best to place this on the end of the command line so that names of the files to be traced are not confused with other file names or strings on the command line.

Problem Reports

If you send email to the author, please include the following information:

1. The type of UNIX

   The author will need to know the brand and version of UNIX you are using, or if it is not UNIX but something else. The output of "uname -sr" is usually sufficient (but not all systems have it).

2. The Version Number

   In any information you send, please include the version number reported in the *common/patchlevel.h* file, or `aegis -vers` if you can get it to compile.

3. The Archive Site

   When and where you obtained this version of *Aegis*. If you tell me nothing else, tell me this (and, hopefully, why you did nothing else).

4. Unpacking

   Did you have problems unpacking *Aegis*? This probably isn't a problem with the .tar.Z distribution, but you could have obtained a shar format copy.

5. Building

   Did you have problems building *Aegis*? This could have been the instructions included, it could have been the configure script, it could have been the Makefile, or anything else.

6. Testing, Non-Set-Uid

   Did you have problems with the tests? You could have had problems running them, or some of them could have failed. If some tests fail but not others, please let me know *which* ones failed, and include the fact that *Aegis* was **not** set-uid-root at the time. The -k option to *make* can be useful if some tests fail but not others.

7. Testing, Set-Uid-Root

    Did you have problems with the tests when *Aegis* was set-uid-root? You could have had problems running them, or some of them could have failed. If some tests fail but not others, please let me know *which* ones failed, and include the fact that *Aegis* was set-uid-root at the time.

8. Installation

    Did you have problems installing *Aegis*? This could have been the instructions, or anything else.

At this point it would probably be a very good idea to print out the manual entries and read them carefully. You will also want to print a copy of the User Guide; if you don't gave groff, there should be a PostScript copy at the archive site. It is a known flaw that the User Guide is incomplete, it is something the author is working on "at this moment".

9. The Example Project

    After reading the User Guide, it is often useful to manually run through the example in chapter 3. You will need to do more than one change, hopefully several; the first change is not representative of the system. Did you manually do the example? Did you find flaws in the User Guide or manual entries?

10. Using Aegis

    Did you have problems using *Aegis*? This is a whole can of worms. If possible, include a shell script similar to the tests which accompany *Aegis*, which reproduces the bug. Exit code 1 on failure (bug), exit code 0 on success (for when bug is fixed).

11. The Source Code

    Did you read the code? Did you write some code? If you read the code and found problems, fixed them, or extended *Aegis*, these contributions are most welcome. I reserve the right to modify or reject such contributions.

The above list is inclusive, not exclusive. Any and all feedback is greatly appreciated, as is the effort and interest required to produce it.

LICENSE

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis – project change supervisor

SYNOPSIS

aegis *function* [*option*...]

aegis -Help

DESCRIPTION

The *aegis* program is a transaction base software configuration management system. It is used to supervise the development and integration of changes into projects.

FUNCTIONS

The following functions are available:

-Build

The *aegis -Build* command is used to build a project. See *aeb*(1) for more information.

-Change_Attributes

The *aegis -Change_Attributes* command is used to modify the attributes of a change. See *aeca*(1) for more information.

-Change_Directory

The *aegis -Change_Directory* command is used to change directory. See *aecd*(1) for more information.

-Change_Owner

The *aegis -Change_Owner* command is used to facilitate reassignment of the developer of a change in the *being developed* state. See *aechown*(1) for more information.

-CLone

The *aegis -CLone* command is used to exactly replicate a change, usually on another branch. See *aeclone*(1) for more information.

-CoPy_file

The *aegis -CoPy_file* command is used to copy a file into a change. See *aecp*(1) for more information.

-CoPy_file_Undo

The *aegis -Copy_File_Undo* command is used to remove a copy of a file from a change. See *aecpu*(1) for more information.

-Delta_Name

The *aegis -Delta_Name* command is used to add a symbolic name to a project delta. See *aedn*(1) for more information.

-Develop_Begin

The *aegis -Develop_Begin* command is used to begin development of a change. See *aedb*(1) for more information.

-Develop_Begin_Undo

The *aegis -Develop_Begin_Undo* command is used to cease development of a change. See *aedbu*(1) for more information.

-Develop_End

The *aegis -Develop_End* command is used to complete development of a change. See *aede*(1) for more information.

-Develop_End_Undo

The *aegis -Develop_End_Undo* command is used to recall a change for further development. See *aedeu*(1) for more information.

-DIFFerence

The *aegis -DIFFerence* command is used to find differences between development directory and baseline. See *aed*(1) for more information.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-Integrate_Begin

The *aegis -Integrate_Begin* command is used to begin integrating a change. See *aeib*(1) for more information.

-Integrate_Begin_Undo

The *aegis -Integrate_Begin_Undo* command is used to cease integrating a change. See *aeibu*(1) for more information.

-Integrate_Fail

The *aegis -Integrate_Fail* command is used to fail a change integration. See *aeif*(1) for more information.

-Integrate_Pass

The *aegis -Integrate_PASS* command is used to pass a change integration. See *aeip*(1) for more information.

-List

The *aegis -List* command is used to list interesting things. See *ael*(1) for more information.

-MoVe_file

The *aegis -MoVe_file* command is used to change the name of a file as part of a change. See *aemv*(1) for more information.

-New_Administrator

The *aegis -New_Administrator* command is used to add new administrators to a project. See *aena*(1) for more information.

-New_BRanch

The *aegis -New_BRanch* command is used to add a new branch to a project. See *aenbr*(1) for more information.

-New_BRanch_Undo

The *aegis -New_BRanch_Undo* command is used to remove a new branch from a project. See *aenbru*(1) for more information.

-New_Change

The *aegis -New_Change* command is used to add a new change to a project. See *aenc*(1) for more information.

-New_Change_Undo
The *aegis -New_Change_Undo* command is used to remove a new change from a project. See *aencu*(1) for more information.

-New_Developer
The *aegis -New_Developer* command is used to add new developers to a project. See *aend*(1) for more information.

-New_File
The *aegis -New_File* command is used to add new files to a change. See *aenf*(1) for more information.

-New_File_Undo
The *aegis -New_File_Undo* command is used to remove new files from a change. See *aenfu*(1) for more information.

-New_Integrator
The *aegis -New_Integrator* command is used to add new integrators to a project. See *aeni*(1) for more information.

-New_Project
The *aegis -New_Project* command is used to create a new project to be watched over by aegis. See *aenpr*(1) for more information.

-New_ReLeaSe
The *aegis -New_ReLeaSe* command is used to create a new project from an existing project. See *aenrls*(1) for more information.

-New_ReViewer
The *aegis -New_ReViewer* command is used to add new reviewers to a project. See *aenrv*(1) for more information.

-New_Test
The *aegis -New_Test* command is used to add a new test to a change. See *aent*(1) for more information.

-New_Test_Undo
The *aegis -New_Test_Undo* command is used to remove new tests from a change. See *aentu*(1) for more information.

-Project_Attributes
The *aegis -Project_Attributes* command is used to modify the attributes of a project. See *aepa*(1) for more information.

-Remove_Administrator
The *aegis -Remove_Administrator* command is used to remove administrators from a project. See *aera*(1) for more information.

-Remove_Developer
The *aegis -Remove_Developer* command is used to remove developers from a project. See *aerd*(1) for more information.

-ReMove_file
The *aegis -ReMove_file* command is used to add files to be deleted to a change. See *aerm*(1) for more information.

-ReMove_file_Undo
The *aegis -Remove_File_Undo* command is used to remove files to be deleted from a change. See *aermu*(1) for more information.

-Remove_Integrator
The *aegis -Remove_Integrator* command is used to remove integrators from a project. See *aeri*(1) for more information.

-ReMove_PRoject
The *aegis -ReMove_PRoject* command is used to remove a project. See *aermpr*(1) for more information.

-Remove_ReViewer
The *aegis -Remove_ReViewer* command is used to remove reviewers from a project. See *aerrv*(1) for more information.

-RePorT
The *aegis -RePort* command is used to generate reports from aegis' database. These reports may be written by users, or be distributed with aegis.

-Review_Fail
The *aegis -Review_Fail* command is used to fail a change review. See *aerf*(1) for more information.

-Review_Pass
The *aegis -Review_PASS* command is used to pass a change review. See *aerp*(1) for more information.

-Review_Pass_Undo
The *aegis -Review_Pass_Undo* command is used to rescind a change review pass. See *aerpu*(1) for more information.

-Test
The *aegis -Test* command is used to run tests. See *aet*(1) for more information.

-VERSion
The *aegis -VERSion* command is used to get copyright and version details. See *aev*(1) for more information.

All function selectors are case insensitive. Function selectors may be abbreviated; the abbreviation is the upper case letters. Function selectors must appear as the first command line argument.

OPTIONS

The following options are available to all functions. These options may appear anywhere on the command line following the function selectors.

-LIBrary *abspath*
This option may be used to specify a directory to be searched for global state files and user state files. (See *aegstate*(5) and *aeustate*(5) for more information.) Several library options may be present on the command line, and are search in the order given. Appended to this explicit search path are the directories specified by the *AEGIS_PATH* environment variable (colon separated), and finally, */usr/local/lib/aegis* is always searched. All paths specified, either on the command line or in the *AEGIS_PATH* environment variable, must be absolute.

-PAGer The output of listings and help is piped through the pager command given in the PAGER environment variable (or *more* if not set). This is the default if the command is in the foreground. This option may be used to override an y preference specified in the *aeuconf*(5) file.

-No_PAGer
This option may be used to ensure that the output of listings and help is not piped through a pager command. This option may be used to override an y preference specified in the *aeuconf*(5) file.

-Page_Length *number*
This option may be used to set the page length of listings. The default, in order of preference, is obtained from the system, from the *LINES* environment variable, or set to 24 lines.

-Page_Width *number*
This option may be used to set the page width of listings and error messages. The default, in order of preference, is obtained from the system, from the *COLS* environment variable, or set to 79 characters.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

The following options are available to all listings. These options may appear anywhere on the command line following the function selectors.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-UNFormatted
This option may be used with most listings to specify that the column formatting is not to be performed. This is useful for shell scripts.

The following options are available to *most* functions. These options may appear anywhere on the command line following the function selectors.

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

The *aegis* command understands the following environment variables:

AEGIS_PATH
: A colon-separated list of library directories. See the **-LIBrary** option for a description how this environment variable is used.

AEGIS_PROJECT
: Names a default project. See the **-Project** option for a description how this environment variable is used.

AEGIS_CHANGE
: Specifies a default change. See the **-Change** option for a description how this environment variable is used.

AEGIS_FLAGS
: This environment variable is used to hold *aeuconf*(5) information, and over-rides the settings in the users *.aegisrc* file. This is intended to be used within the tests distributed with aegis, but can also be of use within some shell scripts.

AEGIS_THROTTLE
: Specifies the number of seconds to delay execution within commands which set timestamps. This is intended to be used within the tests distributed with aegis, but can also be of use within some shell scripts.

AEGIS_AUTOMOUNT_POINTS
: A colon-separated list of directories which the automounter may use to mount file systems. Use with extreme care, as this distorts Aegis' idea of the shape of the filesystem.

  This feature assumes that paths below the automounter's mount directory are echoes of paths without it. *E.g.* When `/home` is the trigger, and `/tmp_mnt/home` is where the on-demand NFS mount is performed, with `/home` appearing to processes to be a symlink.

  This is the behavior of the Sun automounter. The AMD automounter is capable of being configured in this way, though it is not typical of the examples in the manual. Nor is it typical of the out-of-the-box Linux AMD configuration in many distributions.

COLS
: Specifies the page width for errors and listings. See the **-Page_Width** option for a description how this environment variable is used.

EDITOR
: Specifies the program use to edit files when the **-Edit** or **-Edit_BackGround** options are used. (See also the *VISUAL* environment variable.) Defaults to *vi* if not set.

LINES
: Specifies the page length for listings. See the **-Page_Length** option for a description how this environment variable is used.

PAGER
: Specifies the program to use to view listings and help. Not used if output is to a file or a pipe. Defaults to *more* if not set.

VISUAL
: Specifies the program use to edit files when the **-Edit** option is used. (See also the *EDITOR* environment variable.) Defaults to *vi* if not set.

SEE ALSO

aegis(5) aegis file format syntax

aecattr(5)
change attributes file format

aecstate(5)
change state file format

aedir(5) directory structures

aegstate(5)
aegis state file format

aepattr(5)
project attributes file format

aepconf(5)
project configuration file format

aepstate(5)
project state file format

aer(5) report script language definition

aesub(5)
available command substitutions

aeuconf(5)
user configuration file format

aeustate(5)
user state file format

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

ae_c – set change number

SYNOPSIS

ae_c *change-number*

DESCRIPTION

The *ae_c* command is an alias used to set the AEGIS_CHANGE environment variable. No checking of the argument is performed.

This can make changing the change you are working on quick and simple.

SEE ALSO

aegis(1) For information on environment variables.

ae_p(1) Set project name.

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

ae_p – set project name

SYNOPSIS

ae_p *project-name*

DESCRIPTION

The *ae_p* command is an alias used to set the AEGIS_PROJECT environment variable. No checking of the argument is performed.

This can make changing projects quick and simple.

SEE ALSO

aegis(1) For information on environment variables.

ae_c(1) Set change number.

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -Build – build a change

SYNOPSIS

aegis -Build [*option*...][*filename*...]

aegis -Build -List [*option*...]

aegis -Build -Help

DESCRIPTION

The *aegis -Build* command is used to build a project. The project configuration file is consulted for the appropriate build command, and that command is executed (see *aepconf*(5) for more information.) Output of the command is automatically logged to the *aegis.log* file at the root of the development directory tree. The build command will be executed with its current directory being the root of the development directory, irrespective of there the *aegis -Build* command was executed.

If the change is in the *being integrated* state, references to the development directory, above, should be read as the integration directory. Integration build commands are executed with the user and group set to the project's owning user and group. That is, it is not necessary for an integrator to log in as someone else, the project account for instance, in order to do an integration.

Process Side Effects

This command will cancel any test registrations, because building the project logically invalidates them. If the project *config* file was deleted, any diff registration will also be canceled.

PARTIAL BUILD

If files are named on the command line, these files are appended to the build command. This is known as a partial build. Partial builds are not legal in the *being integrated* state, but can often be useful in the *being developed* state. Partial builds are not recorded in the change status, because builds are decoupled from aegis it is not possible for aegis to know if any set of partial builds is equivalent to a full build.

Warning: no change state lock is taken for a partial build, only a baseline read lock.

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

SYMBOLIC LINKS

Many dependency maintenance tools, and indeed some compilers, have little or no support for include file search paths, and thus for the concept of the two-level directory hierarchy employed by *aegis*. (It becomes multi-level when aegis' branching functionality is used.) To allow these tools to be used, *aegis* provides the ability to maintain a set of symbolic links between the development directory of a change and the baseline of a project, so it appears to these tools that all of the project's files are present in the development directory.

Project Configuration

Two boolean fields of the project *config* file control the behavior of this feature. They are the *create_-symlinks_before_build* and *remove_symlinks_after_build* fields. See *aepconf*(5) for more information.

If both fields are set to *true*, the user never sees the symbolic links, because they are added purely for the benefit of the dependency maintenance tool during the execution of the *aeb*(1) command.

If only the *create_symlinks_before_build* field is set to *true* (the other will default to false) the symbolic links will be created at develop begin time (see *aedb*(1) for more information) and also maintained by each *aeb*(1) invocation. Note that the symbolic links are only maintained at these times, so project integrations during the course of development may leave the symbolic links in an inconsistent state until the next build.

When files are copied from the baseline into a change, using the *aecp*(1) command, the symbolic link pointing into the baseline, if any, will be removed before the file is copied.

Note: Using this functionality in either form has implications for how the rules file of the dependency maintenance tool is written. Rules must *remove* their targets before creating them. (Usually with an *rm -f* command.) This is to avoid attempting to write the result on the symbolic link, which will point at a read-only file in the project baseline. This is similar to the same requirement for using the *link_integration_-directory* field of the project *config* file.

User Configuration

There is a *symbolic_link_preference* field in the user configuration file (see *aeuconf*(5) for more information). This controls whether *aeb*(1) will verify the symbolic links before the build (default) or whether it will assume they are up-to-date. (This field is only relevant if *create_symlinks_before_build* is true.)

For medium-to-large projects, verifying the symbolic links can take as long as the build itself. Assuming the symbolic links are up-to-date can be a large time-saving for these projects. It may be advisable to review your choice of DMT in such a situation.

The *aedb*(1) command **does not** consult this preference. Thus, in most situations, the symbolic links will be up-to-date when the build is performed. The only Aegis function which may result in the symbolic links becoming out-of-date is the integration of another change, as this may alter the presence or absence of files in the baseline. In this situation, the default *aeb*(1) action is to ignore the user preference and the verify symbolic links.

There are two command line options which modify *aeb*(1) behavior further: the **--Verify_Symbolic_Links** option says to verify the symbolic links; and the **--Assume_Symbolic_Links** option says to assume the symbolic links are up-to-date. In each case the option over-rides the default and the user preference.

THE BASELINE LOCK

The baseline lock is used to ensure that the baseline remains in a consistent state for the duration of commands which need to read the contents of files in the baseline.

The commands which require the baseline to be consistent (these include the *aeb*(1), *aecp*(1) and *aed*(1) commands) take a baseline *read* lock. This is a non-exclusive lock, so the concurrent development of changes is not hindered.

The command which modifies the baseline, *aeip*(1), takes a baseline *write* lock. This is an exclusive lock, forcing *aeip*(1) to block until there are no active baseline read locks.

It is possible that one of the above development commands will block until an in-progress *aegis -Integrate_PASS* completes. This is usually of short duration while the project history is updated. The delay is essential so that these commands receive a consistent view of the baseline. No other integration command will cause the above development commands to block.

When aegis' branch functionality is in use, a read (non-exclusive) lock is taken on the branch baseline and also each of the "parent" baselines. However, a baseline write (exclusive) lock is only taken on the branch baseline; the "parent" baselines are only read (non-exclusive) locked.

METRICS

Aegis is capable of recording metrics as part of the file attributes of a change. This allows various properties of files to be recorded for later trend analysis, or other uses.

The specific metrics are not dictated by Aegis. It is expected that the integration build will create a metrics file for each of the source files the change. These metrics files must be in the format specified by *aemetrics*(5).

The name of the metrics file defaults to "*filename*,**S**", however it may be varied, by setting the *metrics_-filename_pattern* field of the project *config* file. See *aepconf*(5) for more information.

If such a metrics file exists, for each source file in a change, it will be read and remembered at integrate pass time. If it does not exist, Aegis assumes there are no relevant metrics for that file, and proceeds silently; it is not an error.

OPTIONS

The following options are understood:

name=*value*

Command line arguments of this form are assumed to be variable assignments for the build tool. They are passed through unchanged. They imply a partial build.

−BAse_RElative

This option may be used to cause relative filenames to be considered relative to the base of the source tree. See *aeuconf*(5) for the corresponding user preference.

−CUrrent_RElative

This option may be used to cause relative filenames to be considered relative to the current directory. This is usually the default. See *aeuconf*(5) for the corresponding user preference.

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

−MINImum
This option may be used to request a minimum set of symbolic links, when the *create_symlinks_to_baseline* functions are being used. This is useful if you want to simulate something like *aeib -minimum* in the development directory. This option is not meaningful if symbolic links are not being used.

-Not_Logging
This option may be used to disable the automatic logging of output and errors to a file. This is often useful when several aegis commands are combined in a shell script.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

−Verify_Symbolic_Links
This option may be used to request a scan for up-to-date symbolic links before and after the build. This is the default. See also the ‘‘symbolic_links_preference’’ field of *aeuconf*(5). This option is not meaningful if symbolic links are not being used.

−Assume_Symbolic_Links
This option may be used to request no scan for up-to-date symbolic links before and after the build, this is useful when you have just done a build and you know they are already up-to-date. See also the ‘‘symbolic_links_preference’’ field of *aeuconf*(5). This option is not meaningful if symbolic links are not being used.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option.

The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aeb 'aegis -b \!* -v'
sh$	aeb(){aegis -b $* -v}

ERRORS

It is an error if the change is not assigned to the current user.
It is an error if the change is not in one of the *being developed* or *being integrated* states.
It is an error if a partial build is requested and the change is in the *being integrated* state.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aedb(1) begin development of a change

aecp(1) file copy also takes a baseline read lock (non-exclusive)

aeib(1) begin integration of a change

aeip(1) integrate pass takes a baseline write lock (exclusive)

aet(1) run tests

aemetrics(5)
metrics values file format

aepconf(5)
project configuration file format

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -Change_Attributes – modify the attributes of a change

SYNOPSIS

aegis -Change_Attributes -File *attr-file* [*option*...]
aegis -Change_Attributes -Edit [*option*...]
aegis -Change_Attributes -List [*option*...]
aegis -Change_Attributes -Help

DESCRIPTION

The *aegis -Change_Attributes* command is used to set, edit or list the attributes of a change.

The output of the **-List** variant is suitable for use as input at a later time.

See *aecattr*(5) for a description of the file format.

OPTIONS

The following options are understood:

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Edit
Edit the attributes with a text editor, this is usually more convenient than supplying a text file. The *VISUAL* and then *EDITOR* environment variables are consulted for the name of the editor to use; defaults to *vi*(1) if neither is set. Warning: tries to be well behaved when faced with errors, the temporary file is left in your home directory. This option may not be used in the background, or when the standard input is not a terminal.

-Edit_BackGround
Edit the attributes with a dumb text editor, this is most often desired when edit commands are being piped into the editor via the standard input. Only the *EDITOR* environment variable is consulted for the name of the editor to use; it is a fatal error if it is not set.

-File *filename*
Take the attributes from the specified file.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aeca 'aegis -ca \!* -v'
sh$     aeca(){aegis -ca $* -v}
```

ERRORS

It is an error if the current user is not an administrator of the specified project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aepa(5) modify the attributes of a project

aecattr(5)
change attributes file format

aecstate(5)
: change state file format

aeuconf(5)
: user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -Change_Directory – change directory

SYNOPSIS

aegis -Change_Directory [*option*...][*relative-path*]

aegis -Change_Directory -List [*option*...]

aegis -Change_Directory -Help

DESCRIPTION

The *aegis -Change_Directory* command is used to obtain a path to change directory to. If the *relative-path* is supplied, this will be added to the output.

This command is usually used to calculate an argument for *cd*(1), however it can also be used to obtain an absolute path for change and project files.

OPTIONS

The following options are understood:

-BaseLine

This option may be used to specify that the project baseline is the subject of the command.

-BRanch *number*

This option may be used to specify a different branch for the origin file, rather than the baseline.

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Development_Directory

This option is used to specify that the development directory is the subject of the command. This is only useful for a change which is in the *being integrated* state, when the default is the integration directory.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*

This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse

This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-TRunk

This option may be used to specify the project trunk for the origin file, rather than the baseline.

-Verbose

This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aecd 'cd `aegis -cd \!* -v`'
sh$     aecd(){cd `aegis -cd $* -v`}
```

ERRORS

It is an error if the specified change is not in a state where it has a directory to change to.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aedb(1) begin development of a change

aeib(1) begin integration of a change

aerp(1) pass review of a change

aerf(1) fail review of a change

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -Change_Owner – set change owner

SYNOPSIS

aegis -Change_Owner *change-number user-name* [*option*...]
aegis -Change_Owner -Help
aegis -Change_Owner -VERSion

DESCRIPTION

The *aegis -Change_Owner* command is used to reassign a change from one developer to another.

A new development directory is created for the change in the new developers default area (see *aedb*(1) for more information) and the change files are copied across. Derived files are ignored, so a new build will be required. The old development directory will be deleted.

This command must be performed by a project administrator, and the new assignee must be a developer.

Warning: capricious use of this command will alienate developers very rapidly.

Development Directory Location

Please Note: Aegis also consults the underlying file system, to determine its notion of maximum file size. Where the file system's maximum file size is less than *maximum_filename_length*, the filesystem wins. This can happen, for example, when you are using the Linux UMSDOS file system, or when you have an NFS mounted an ancient V7 filesystem. Setting *maximum_filename_length* to 255 in these cases does not alter the fact that the underlying file systems limits are far smaller (12 and 14, respectively).

If your development directories (or your whole project) is on filesystems with filename limitations, or a portion of the heterogeneous builds take place in such an environment, it helps to tell Aegis what they are (using the project *config* file's fields) so that you don't run into the situation where the project builds on the more permissive environments, but fails with mysterious errors in the more limited environments.

If your development directories are routinely on a Linux UMSDOS filesystem, you would probably be better off setting *dos_filename_required = true*, and also changing the *development_directory_template field. Heterogenous development with various Windows environments may also require this.*

OPTIONS

The following options are understood:

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-DIRectory *path*

This option may be used to specify which directory is to be used. It is an error if the current user does not have appropriate permissions to create the directory path given. This must be an absolute path.

Caution: If you are using an automounter do not use 'pwd' to make an absolute path, it usually gives the wrong answer.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-Interactive

Specify that aegis should ask the user for confirmation before deleting each file. Answer the question *yes* to delete the file, or *no* to keep the file. You can also answer *all* to delete the file and all that follow, or *none* to keep the file and all that follow.

Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

If aegis is running in the background, the question will not be asked, and the files will be deleted.

-Keep

This option may be used to retain files and/or directories usually deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-No_Keep

This option may be used to ensure that the files and/or directories are deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*

This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse

This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-User *name*

This option is used to specify the user who is to develop the change.

-Verbose

This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait

This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait

This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%   alias aechown 'aegis -chown \!* -v'

sh$ aechown(){aegis -chown $* -v}

ERRORS

It is an error if the user issuing the command is not a project administrator.
It is an error if the change is not in the *being developed* state.
It is an error if the user given is not a developer.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller   E-Mail: millerp@canb.auug.org.au
/\/*   WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -CLEan – clean files from development directory

SYNOPSIS

aegis –CLEan [*option*...]
aegis -Help
aegis -VERSion

DESCRIPTION

The *aegis -CLEan* command is used to remove all files which are not change source files from a development directory. This can be used to obtain a ‘‘clean’’ development directory before a final build, to ensure that a change is ready to end development. A new build will be required.

All symbolic links will be removed from the development directory, even if *remove_symlinks_after_build* = *false* in the project *config* file. The symbolic links will be re-installed, if *create_symlinks_before_build* = *true*. This is to ensure that the symlinks are accurate, and that unnecessary ones are removed.

All special device files, pipes and sockets will be removed. These files cannot be source files, and it is expected that the following build will restore them.

All derived files created by previous builds of the change will be removed. It is expected that the following build will recreate them. Any temporary files you may have created in the development directory will also be removed.

The *develop_begin_command* in the project config file (see *aepconf*(5) for more information) will be run, if there is one. The *change_file_command* will be run, if there is one. The *project_file_command* will be run, if there is one.

You will be warned if any of the files are out-of-date and need to be merged. You will be warned if any files need to be differenced.

SYMBOLIC LINKS

Many dependency maintenance tools, and indeed some compilers, have little or no support for include file search paths, and thus for the concept of the two-level directory hierarchy employed by *aegis*. (It becomes multi-level when aegis’ branching functionality is used.) To allow these tools to be used, *aegis* provides the ability to maintain a set of symbolic links between the development directory of a change and the baseline of a project, so it appears to these tools that all of the project’s files are present in the development directory.

Project Configuration

Two boolean fields of the project *config* file control the behavior of this feature. They are the *create_symlinks_before_build* and *remove_symlinks_after_build* fields. See *aepconf*(5) for more information.

If both fields are set to *true*, the user never sees the symbolic links, because they are added purely for the benefit of the dependency maintenance tool during the execution of the *aeb*(1) command.

If only the *create_symlinks_before_build* field is set to *true* (the other will default to false) the symbolic links will be created at develop begin time (see *aedb*(1) for more information) and also maintained by each *aeb*(1) invocation. Note that the symbolic links are only maintained at these times, so project integrations during the course of development may leave the symbolic links in an inconsistent state until the next build.

When files are copied from the baseline into a change, using the *aecp*(1) command, the symbolic link pointing into the baseline, if any, will be removed before the file is copied.

Note: Using this functionality in either form has implications for how the rules file of the dependency maintenance tool is written. Rules must *remove* their targets before creating them. (Usually with an *rm -f* command.) This is to avoid attempting to write the result on the symbolic link, which will point at a read-only file in the project baseline. This is similar to the same requirement for using the *link_integration_-directory* field of the project *config* file.

User Configuration

There is a *symbolic_link_preference* field in the user configuration file (see *aeuconf*(5) for more information). This controls whether *aeb*(1) will verify the symbolic links before the build (default) or whether it will assume they are up-to-date. (This field is only relevant if *create_symlinks_before_build* is true.)

For medium-to-large projects, verifying the symbolic links can take as long as the build itself. Assuming the symbolic links are up-to-date can be a large time-saving for these projects. It may be advisable to review your choice of DMT in such a situation.

The *aedb*(1) command **does not** consult this preference. Thus, in most situations, the symbolic links will be up-to-date when the build is performed. The only Aegis function which may result in the symbolic links becoming out-of-date is the integration of another change, as this may alter the presence or absence of files in the baseline. In this situation, the default *aeb*(1) action is to ignore the user preference and the verify symbolic links.

There are two command line options which modify *aeb*(1) behavior further: the **--Verify_Symbolic_Links** option says to verify the symbolic links; and the **--Assume_Symbolic_Links** option says to assume the symbolic links are up-to-date. In each case the option over-rides the default and the user preference.

OPTIONS

The following options are understood:

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Not_Logging

This option may be used to disable the automatic logging of output and errors to a file. This is often useful when several aegis commands are combined in a shell script.

−MINIMum

This option may be used to request a minimum set of symbolic links, when the *create_symlinks_to_baseline* functions are being used. This is useful if you want to simulate something like *aeib -minimum* in the development directory. This option is not meaningful if symbolic links are not being used.

This option also says not to remove normal files which occlude project source files. This is a common technique used to temporarily over-ride project source files. The "*aecp -read-only" command would have been more appropriate.*

-Project *name*

This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-Verbose

This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait

This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -clone – make an exact copy of a change

SYNOPSIS

aegis –CLone [*option*...] *change-number* [change-number]
aegis -CLone -Help
aegis -CLone -VERSion

DESCRIPTION

The *aegis -CLone* command is used to create exact replicas of changes. This is of most use when a change need to be applied to several parallel branches.

One change number *must* be supplied. This is the change to be replicated. If any branch options are given (see below) the mandatory change number applies to the branch specified. If no branch is specified, the change applies to the project (implicit or explicit).

If the optional second change number is supplied, this is the change number to be created to hold the replica; if it is not supplied, the next available change number will be used.

If the change to be replicated has been completed, the appropriate file revisions will be extracted from history; otherwise the files will be copied from the development directory of the change to be copied. Be warned: if a file in the change which was cloned subsequently changes, those changes *will not* automagicly be tracked. It is best if changes are cloned at a stable time, such as one of the states after develop end, or even after integrate pass.

Development Directory Location

Please Note: Aegis also consults the underlying file system, to determine its notion of maximum file size. Where the file system's maximum file size is less than *maximum_filename_length*, the filesystem wins. This can happen, for example, when you are using the Linux UMSDOS file system, or when you have an NFS mounted an ancient V7 filesystem. Setting *maximum_filename_length* to 255 in these cases does not alter the fact that the underlying file systems limits are far smaller (12 and 14, respectively).

If your development directories (or your whole project) is on filesystems with filename limitations, or a portion of the heterogeneous builds take place in such an environment, it helps to tell Aegis what they are (using the project *config* file's fields) so that you don't run into the situation where the project builds on the more permissive environments, but fails with mysterious errors in the more limited environments.

If your development directories are routinely on a Linux UMSDOS filesystem, you would probably be better off setting *dos_filename_required = true*, and also changing the *development_directory_template field. Heterogenous development with various Windows environments may also require this.*

OPTIONS

The following options are understood:

-BRanch *number*

This option may be used to specify a different branch for the origin file, rather than the baseline.

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-DIRectory *path*
This option may be used to specify which directory is to be used. It is an error if the current user does not have appropriate permissions to create the directory path given. This must be an absolute path.

Caution: If you are using an automounter do not use ‘pwd‘ to make an absolute path, it usually gives the wrong answer.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

−Output *filename*
This option may be used to specify a filename which is to be written with the automatically determined change number. Useful for writing scripts.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TRunk
This option may be used to specify the project trunk for the origin file, rather than the baseline.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -CoPy_file – copy a file into a change

SYNOPSIS

aegis -CoPy_file [*option*...] *filename*...
aegis -CoPy_file -INDependent [*option*...] *filename*...
aegis -CoPy_file -List [*option*...]
aegis -CoPy_file -Help

DESCRIPTION

The *aegis -CoPy_file* command is used to copy a file into a change. The named files will be copied from the baseline into the development directory, and added to the list of files in the change. The version of files copied from the baseline is remembered.

This command may be used to copy tests into a change, not just source files. Tests are treated just like any other source file, and are subject to the same process.

Warning: If there are files in the development directory of the same name they will be overwritten by this command.

You may also name directories. All of the source files in the directories named, and all directories below them, will be copied from the baseline into the development directory, and added to the list of files in the change.

When copying files explicitly, it is an error if the file is already part of the change. When you name a directory, all of the source files in the project below that directory are copied, except any which are already in the change. It is an error if none of the files implicitly named by the directory can be used.

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

Process Side Effects

This command will cancel any build or test registrations, because adding another file logically invalidates them. If the project *config* file was added, any diff registration will also be canceled.

When the change files are listed (*aegis -List Change_Files -TERse*) the copied files will appear in the listing. When the project files are listed with an explicit change number (*aegis -List Project_Files -TERse -Change* N) none of the change's files, including the copied files, will appear in the terse listing. These two features are very helpful when calling aegis from within a DMT to generate the list of source files.

THE BASELINE LOCK

The baseline lock is used to ensure that the baseline remains in a consistent state for the duration of commands which need to read the contents of files in the baseline.

The commands which require the baseline to be consistent (these include the *aeb*(1), *aecp*(1) and *aed*(1) commands) take a baseline *read* lock. This is a non-exclusive lock, so the concurrent development of changes is not hindered.

The command which modifies the baseline, *aeip*(1), takes a baseline *write* lock. This is an exclusive lock, forcing *aeip*(1) to block until there are no active baseline read locks.

It is possible that one of the above development commands will block until an in-progress *aegis -Integrate_PASS* completes. This is usually of short duration while the project history is updated. The delay is essential so that these commands receive a consistent view of the baseline. No other integration command will cause the above development commands to block.

When aegis' branch functionality is in use, a read (non-exclusive) lock is taken on the branch baseline and also each of the "parent" baselines. However, a baseline write (exclusive) lock is only taken on the branch baseline; the "parent" baselines are only read (non-exclusive) locked.

TEST CORRELATIONS

The ‘‘aegis -Test -SUGgest’’ command may be used to have aegis suggest suitable regression tests for your change, based on the source files in your change. This automatically focuses testing effort to relevant tests, reducing the number of regression tests necessary to be confident that you have not introduced a bug.

The test correlations are generated by the ‘‘aegis -Integrate_Pass’’ command, which associates each test in the change with each source file in the change. Thus, each source file accumulates a list of tests which have been associated with it in the past. This is not as exact as code coverage analysis, but is a reasonable approximation in practice.

The *aecp*(1) and *aenf*(1) commands are used to associate files with a change. While they do not actively perform the association, these are the files used by *aeip*(1) and *aet*(1) to determine which source files are associated with which tests.

Test Correlation Accuracy

Assuming that the testing correlations are accurate and that the tests are evenly distributed across the function space, there will be a less than *1/number* chance that a relevant test has not been run by the ‘‘aegis -Test -SUGgest *number*’’ command. A small amount of noise is added to the test weighting, so that unexpected things are sometimes tested, and the same tests are not run every time.

Test correlation accuracy can be improved by ensuring that:

- Each change should be strongly focused, with no gratuitous file inclusions. This avoids spurious correlations.
- Each item of new functionality should be added in an individual change, rather than several together. This strongly correlates tests with functionality.
- Each bug should be fixed in an individual change, rather than several together. This strongly correlates tests with functionality.
- Test correlations will be lost if files are moved. This is because correlations are by name.

The best way for tests to correlate accurately with source files is when a change contains a test and exactly those files relating to the functionality under test. Too many spurious files will weaken the usefulness of the testing correlations.

OPTIONS

The following options are understood:

−BAse_RElative
This option may be used to cause relative filenames to be considered relative to the base of the source tree. See *aeuconf*(5) for the corresponding user preference.

−CUrrent_RElative
This option may be used to cause relative filenames to be considered relative to the current directory. This is usually the default. See *aeuconf*(5) for the corresponding user preference.

-BRanch *number*
This option may be used to specify a different branch for the origin file, rather than the baseline.

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-DELta *number*
This option may be used to specify a particular delta in the project's history to copy the file from, rather than the most current version. If the delta has been given a name (see *aedn*(1) for how) you may use a delta name instead of a delta number. It is an error if the delta specified does not exist. Delta numbers start from 1 and increase; delta 0 is a special case meaning ‘‘when the branch started’’.

-DElta_Date *string*
This option may be used to specify a particular date and time in the project's history to copy the file from, rather than the most current version. It is an error if the string specified cannot be interpreted as a valid date and time. Quote the string if you need to use spaces.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-INDependent
This option is used to specify that the copy is to be run independent of any particular change. The files will be copied relative to the current directory.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Not_Logging
This option may be used to disable the automatic logging of output and errors to a file. This is often useful when several aegis commands are combined in a shell script.

-Output *filename*
This option may be used to specify an output file of a file being copied from the baseline. Only one baseline file may be named when this option is used. The file name "-" is understood to mean the standard output. This option does not add the file to the set of change files. *No locks*

are taken when this option is used, not even the baseline read lock.

-OverWriting
This option may be used to force overwriting of files. The default action is to give an error if an existing file would be overwritten.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

−Read_Only
This option may be used to specify that the file is to be used to insulate the change from the baseline. The user does not intend to edit the file. These files must be uncopied before development may end.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-TRunk
This option may be used to specify the project trunk for the origin file, rather than the baseline.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aecp 'aegis -cp \!* -v'

sh$ aecp(){aegis -cp $* -v}

ERRORS

It is an error if the change is not in the *being developed* state.

It is an error if the change is not assigned to the current user.

It is an error if the file is already in the change and the **-OverWrite** option is not specified.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeb(1) build also takes a baseline read lock (non-exclusive)

aecpu(1) reverse action of aecp

aedb(1) begin development of a change

aedn(1) assign a name to a delta

aeip(1) integrate pass takes a baseline write lock (exclusive)

aemv(1) rename a file as part of a change

aenf(1) add a new file to a change

aerm(1) add files to be deleted to a change

aeuconf(1) user configuration file format

COPYRIGHT

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au

/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -CoPy_file_Undo – reverse action of aecp

SYNOPSIS

aegis -CoPy_file_Undo [*option*...] *filename*...
aegis -CoPy_file_Undo -List [*option*...]
aegis -CoPy_file_Undo -Help

DESCRIPTION

The *aegis -CoPy_file_Undo* command is used to delete files previously copied into a change. The named files will be removed from the list of files in the change.

The file is deleted from the development directory unless the **-Keep** option is specified. The **-Keep** option should be used with great care, as you can confuse tools such as *make*(1) by leaving these files in place.

You may name a directory to delete from the change all files in that directory tree previously copied into the change, other files in the tree will be ignored. It is an error if there are no relevant files.

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

Process Side Effects

This command will cancel any build or test registrations, because deleting a file logically invalidates them. If the project *config* file was deleted, any diff registration will also be canceled.

The difference file (`,D`) will also be removed, however any DMT derived files (e.g a `.o` file from a `.c` file) will not be removed. This is because aegis is decoupled from the DMT, and cannot know what these derived file may be called. You may need to delete derived files manually.

OPTIONS

The following options are understood:

–BAse_RElative
This option may be used to cause relative filenames to be considered relative to the base of the source tree. See *aeuconf*(5) for the corresponding user preference.

–CUrrent_RElative
This option may be used to cause relative filenames to be considered relative to the current directory. This is usually the default. See *aeuconf*(5) for the corresponding user preference.

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-Interactive
Specify that aegis should ask the user for confirmation before deleting each file. Answer the question *yes* to delete the file, or *no* to keep the file. You can also answer *all* to delete the file and all that follow, or *none* to keep the file and all that follow.

Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

If aegis is running in the background, the question will not be asked, and the files will be deleted.

-Keep
This option may be used to retain files and/or directories usually deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-No_Keep
This option may be used to ensure that the files and/or directories are deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-UNChanged
Examine the named files to see if they are unchanged. Only remove the unchanged files from the change, and leave the files which have changed. If no files are named on the command line all change files are checked.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be

added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aecpu 'aegis -cpu \!* -v'

sh$ aecpu(){aegis -cpu $* -v}

ERRORS

It is an error if the change is not in the *being developed* state.
It is an error if the change is not assigned to the current user.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aecp(1) copy files into a change

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -DIFFerence – find differences between a change and the baseline

SYNOPSIS

aegis -DIFFerence [*filename*...] [*option*...]
aegis -DIFFerence -List [*option*...]
aegis -DIFFerence -Help

DESCRIPTION

The *aegis -DIFFerence* command is used to generate difference listings between source files in the the development directory and the baseline. The purpose is to enable reviewers to find each and every edit performed on the source files. The difference listings will be placed into files named for the sources files but with an additional "`,D`" suffix.

The command used to perform the differences is specified in the *diff_command* field of the project *config* file (see *aepconf*(5) for more information).

If no files are named on the command line, all files in the change will be differenced.

You may name a directory on the command line, and all files in the change in that directory tree will be differenced.

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

THE BASELINE LOCK

The baseline lock is used to ensure that the baseline remains in a consistent state for the duration of commands which need to read the contents of files in the baseline.

The commands which require the baseline to be consistent (these include the *aeb*(1), *aecp*(1) and *aed*(1) commands) take a baseline *read* lock. This is a non-exclusive lock, so the concurrent development of changes is not hindered.

The command which modifies the baseline, *aeip*(1), takes a baseline *write* lock. This is an exclusive lock, forcing *aeip*(1) to block until there are no active baseline read locks.

It is possible that one of the above development commands will block until an in-progress *aegis -Integrate_PASS* completes. This is usually of short duration while the project history is updated. The delay is essential so that these commands receive a consistent view of the baseline. No other integration command will cause the above development commands to block.

When aegis' branch functionality is in use, a read (non-exclusive) lock is taken on the branch baseline and also each of the "parent" baselines. However, a baseline write (exclusive) lock is only taken on the branch baseline; the "parent" baselines are only read (non-exclusive) locked.

CONFLICT RESOLUTION

If the version of a file in the change is not the same as the version of the file in the baseline, it is out-of-date; some other change has altered the file while this change was being developed.

When a difference is requested for an out-of-date file, a merge is performed between the common ancestor, the version in the baseline, and the version in the development directory. The command used to perform the merge is specified by the *merge_command* field of the project *config* file (see *aepconf*(5) for more information).

After the merge is performed the version of the file will be changed to be the current version, marking the file as up to date, and a new build will be required.

The difference output in the *,D* files contains the result of the merge. You should edit them, to make sure the automatic merge has produced sensible results, and then rename them to be the corresponding source file.

This merge process works most of the time. Usually two changes to two logically separate areas of functionality will alter two logically separate parts of any files they may have in common. There are pathological cases where this merge process is spectacularly useless, but these are surprisingly rare in practice.

If any merges are required no differences will be performed. An error message and a non-zero exit status will also result. This is to ensure that developers notice that merges have been done, and that they reconcile the sources and the merged *,D* files before the next difference. See the **-No_Merge** and **-Only_Merge** options, below, for exact control of when merging is performed.

INTEGRATION

During integration, it is also necessary to difference a change. This provides the difference between the branch and its parent, for when development on a branch is completed and it is to be reviewed. The baseline of a branch is the development directory of the composite change it represents.

OPTIONS

The following options are understood:

-ANticipate *change-number*

This option is used to nominate a source for the reference files, rather than the baseline. This may be used to synchronize with a change without having to wait for it to arrive in the baseline. It is an error if the anticipated change is not in one of the *'being reviewed'* or *'awaiting integration'* or *'being integrated'* states. A merge is always performed, because the anticipated change is "about" to make any common file out-of-date. You will still have to perform a "real" merge later.

-BRanch *number*

This option may be used to specify a different branch for the origin file, rather than the baseline.

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Not_Logging
This option may be used to disable the automatic logging of output and errors to a file. This is often useful when several aegis commands are combined in a shell script.

-TRunk
This option may be used to specify the project trunk for the origin file, rather than the baseline.

-No_Merge
This option is used to cause only file differences to be generated, even when file versions are out-of-date. If not set, the default is to use the *diff_preference* field of the *aeuconf*(5) file.

-Only_Merge
This option is used to cause only file merges to be performed on files with out-of-date versions. Other source files are ignored. If not set, the default is to use the *diff_preference* field of the *aeuconf*(5) file.

-Automatic_Merge
This option is used to perform *-Only_Merge* is any source files have out-of-date versions, otherwise *-No_Merge is performed.* Only merges or differences will be performed, it will never use a mixture. If not set, the default is to use the *diff_preference* field of the *aeuconf*(5) file.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aed 'aegis -diff \!* -v'

sh$ aed(){aegis -diff $* -v}

For user's convenience, particularly when they have selected the ‘‘no merge’’ preference, there is also a merge alias:

csh% alias aem 'aegis -diff -only_merge \!* -v'

sh$ aem(){aegis -diff -only_merge $* -v}

ERRORS

It is an error if the change is not in the *being developed* or *being integrated* states.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeb(1) build also takes a baseline read lock (non-exclusive)

aecp(1) copy file also takes a baseline read lock (non-exclusive)

aedb(1) begin development of a change

aeip(1) integrate pass takes a baseline write lock (exclusive)

aepconf(5)
project configuration file format

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/˜millerp/

NAME

aegis -Develop_Begin – begin development of a change

SYNOPSIS

aegis -Develop_Begin *change-number* [*option*...]

aegis -Develop_Begin -List [*option*...]

aegis -Develop_Begin -Help

DESCRIPTION

The *aegis -Develop_Begin* command is used to commence development of a change.

The development directory for the change will be created automatically; below the directory specified in the default_development_directory field of *aeuconf*(5), or if not set below the directory specified in the default_development_directory field of *aepattr*(5), or if not set below the current user's home directory. It is rare to need to know the exact pathname of the development directory, as the *aecd*(1) command can take you there at any time.

Successful execution of this command will move the specified change from the *awaiting development* state to the *being developed* state.

The *develop_begin_command* in the project config file (see *aepconf*(5) for more information) will be run, if specified. This is run after the aegis locks are released, so additional aegis commands may be run from here, if used with care. The symbolic links (see below) have *not* yet been created.

Development Directory Location

Please Note: Aegis also consults the underlying file system, to determine its notion of maximum file size. Where the file system's maximum file size is less than *maximum_filename_length*, the filesystem wins. This can happen, for example, when you are using the Linux UMSDOS file system, or when you have an NFS mounted an ancient V7 filesystem. Setting *maximum_filename_length* to 255 in these cases does not alter the fact that the underlying file systems limits are far smaller (12 and 14, respectively).

If your development directories (or your whole project) is on filesystems with filename limitations, or a portion of the heterogeneous builds take place in such an environment, it helps to tell Aegis what they are (using the project *config* file's fields) so that you don't run into the situation where the project builds on the more permissive environments, but fails with mysterious errors in the more limited environments.

If your development directories are routinely on a Linux UMSDOS filesystem, you would probably be better off setting *dos_filename_required = true*, and also changing the *development_directory_template field. Heterogenous development with various Windows environments may also require this.*

ADMINISTRATOR OVERRIDE

It is possible for project administrators to use the **-User** option to force a developer to start developing a change. Some sites prefer to work this way. Note that developers still have the ability to use the *aedbu*(1) command.

Warning: capricious use of this command will rapidly alienate developers. The defaulting rules, particularly for the change number, depend on aegis and the developer agreeing on what the developer is currently working on.

The *forced_develop_begin_notify_command* project attribute (see *aepattr*(5) for more information) will be run when an administrator uses the **-User** option, in an attempt to minimize the surprises for developers. A suitable command is

```
forced_develop_begin_notify_command =
```

```
"$datadir/db_forced.sh $p $c $developer";
```

This command will send e-mail to the developer, informing her that the change has been assigned to her.

SYMBOLIC LINKS

Many dependency maintenance tools, and indeed some compilers, have little or no support for include file search paths, and thus for the concept of the two-level directory hierarchy employed by *aegis*. (It becomes multi-level when aegis' branching functionality is used.) To allow these tools to be used, *aegis* provides the ability to maintain a set of symbolic links between the development directory of a change and the baseline of a project, so it appears to these tools that all of the project's files are present in the development directory.

Project Configuration

Two boolean fields of the project *config* file control the behavior of this feature. They are the *create_-symlinks_before_build* and *remove_symlinks_after_build* fields. See *aepconf* (5) for more information.

If both fields are set to *true*, the user never sees the symbolic links, because they are added purely for the benefit of the dependency maintenance tool during the execution of the *aeb*(1) command.

If only the *create_symlinks_before_build* field is set to *true* (the other will default to false) the symbolic links will be created at develop begin time (see *aedb*(1) for more information) and also maintained by each *aeb*(1) invocation. Note that the symbolic links are only maintained at these times, so project integrations during the course of development may leave the symbolic links in an inconsistent state until the next build.

When files are copied from the baseline into a change, using the *aecp*(1) command, the symbolic link pointing into the baseline, if any, will be removed before the file is copied.

Note: Using this functionality in either form has implications for how the rules file of the dependency maintenance tool is written. Rules must *remove* their targets before creating them. (Usually with an *rm -f* command.) This is to avoid attempting to write the result on the symbolic link, which will point at a read-only file in the project baseline. This is similar to the same requirement for using the *link_integration_-directory* field of the project *config* file.

User Configuration

There is a *symbolic_link_preference* field in the user configuration file (see *aeuconf*(5) for more information). This controls whether *aeb*(1) will verify the symbolic links before the build (default) or whether it will assume they are up-to-date. (This field is only relevant if *create_symlinks_before_build* is true.)

For medium-to-large projects, verifying the symbolic links can take as long as the build itself. Assuming the symbolic links are up-to-date can be a large time-saving for these projects. It may be advisable to review your choice of DMT in such a situation.

The *aedb*(1) command **does not** consult this preference. Thus, in most situations, the symbolic links will be up-to-date when the build is performed. The only Aegis function which may result in the symbolic links becoming out-of-date is the integration of another change, as this may alter the presence or absence of files in the baseline. In this situation, the default *aeb*(1) action is to ignore the user preference and the verify symbolic links.

There are two command line options which modify *aeb*(1) behavior further: the **--Verify_Symbolic_Links** option says to verify the symbolic links; and the **--Assume_Symbolic_Links** option says to assume the symbolic links are up-to-date. In each case the option over-rides the default and the user preference.

OPTIONS

The following options are understood:

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-DIRectory *path*

This option may be used to specify which directory is to be used. It is an error if the current user does not have appropriate permissions to create the directory path given. This must be an absolute path.

Caution: If you are using an automounter do not use 'pwd' to make an absolute path, it usually gives the wrong answer.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*

This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse

This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-User *name*

This option is used to specify the user who is to develop the change. This option may only be used by a project administrator.

-Verbose

This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait

This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aedb 'aegis -db \!* -v'

sh$ aedb(){aegis -db $* -v}

ERRORS

It is an error if the change does not exist.
It is an error if the change is not in the *awaiting development* state.
It is an error if the current user is not a developer of the specified project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeb(1) build a change

aecd(1) change directory

aecp(1) copy files into a change

aed(1) find differences between a change and the baseline

aedbu(1)
undo the effects of aedb

aede(1) complete development of a change

aemv(1) rename a file as part of a change

aenc(1) add a new change to a project

aend(1) add a new developer to a project

aenf(1) add new files to a change

aent(1) add a new test to a change

aepa(1) modify the attributes of a project *aerm*(1) add files to be deleted to a change

aet(1) run tests

aepattr(5)
project attributes file format

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -Develop_Begin_Undo – undo the effects of aedb

SYNOPSIS

aegis -Develop_Begin_Undo *change-number* [*option*...]
aegis -Develop_Begin_Undo -List [*option*...]
aegis -Develop_Begin_Undo -Help

DESCRIPTION

The *aegis -Develop_Begin_Undo* command is used to reverse the effects of the 'aegis -Develop_Begin' command. The development directory is discarded, even if the change has files associated with it, and even if the development directory is not empty; all files in the development directory will be lost. The change is returned to the *awaiting development* state.

OPTIONS

The following options are understood:

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-Interactive
Specify that aegis should ask the user for confirmation before deleting each file. Answer the question *yes* to delete the file, or *no* to keep the file. You can also answer *all* to delete the file and all that follow, or *none* to keep the file and all that follow.

Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

If aegis is running in the background, the question will not be asked, and the files will be deleted.

-Keep
This option may be used to retain files and/or directories usually deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-No_Keep
This option may be used to ensure that the files and/or directories are deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's

$HOME/.aegisrc file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-User *name*
This option may be used by project administrators to undo the effects of using the same option with the *aedb*(1) command.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aedbu 'aegis -dbu \!* -v'
sh$     aedbu(){aegis -dbu $* -v}
```

ERRORS

It is an error if the change is no assigned to the current user.
It is an error if the change is not in the *being developed* state.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aedb(1) begin development of a change

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -Develop_End – complete development of a change

SYNOPSIS

aegis -Develop_End [*option...*]
aegis -Develop_End -List [*option...*]
aegis -Develop_End -Help

DESCRIPTION

The *aegis -Develop_End* command is used to notify aegis of the completion of the development of a change.

This command checks that you have successfully completed an 'aegis -Build' command since any change source file was edited. See *aeb*(1) for more information.

This command checks that you have successfully completed an 'aegis -DIFFerence' command since any change source file was edited. See *aed*(1) for more information.

This command checks that you have successfully completed an 'aegis -Test' command since the last successful build, unless the change has a *test_exempt* attribute. This command checks that you have successfully completed an 'aegis -Test -BaseLine' command, unless the change has a *test_baseline_exempt* attribute. This command checks that you have successfully completed an 'aegis -Test -REGression' command, unless the change has a *test_regression_exempt* attribute. See *aet*(1) and *aecattr*(5) for more information.

If the change includes the project *config* file, this command checks project file names, to make sure they conform to the *maximum_filename_length* and *posix_filename_charset* field settings. See *aepconf*(5) for more information.

Successful execution of the command advances the change from the *being developed* state to the *being reviewed* state.

Because branches may extend for many months or even years, it is common for the user who initiated the branch to be no longer with the project, or even the company. For this reason, project administrators may end the development of branches. For normal changes in this situation, use the *aechown*(1) command.

The change is no longer considered assigned to the developer.

OPTIONS

The following options are understood:

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aede 'aegis -de \!* -v'
sh$ aede(){aegis -de $* -v}

ERRORS

It is an error if the change is not assigned to the current user.
It is an error if The change is not in the *being developed* state.
It is an error if there has been no successful *'aegis -Build'* command since a change file was last edited.
It is an error if there has been no successful *'aegis -DIFFerence'* command since a change file was last edited.
It is an error if there has been no successful *'aegis -Test'* command since a change file was last edited.
It is an error if there has been no successful *'aegis -Test -BaseLine'* command since a change file was last

edited.

It is an error if an read-only file is still copied into the change. Read-only files are to insulate a change from the baseline during development; they must be removed before development may end.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeb(1) build a change

aeca(1) list or modify attributes of a change

aed(1) difference a change

aedb(1) begin development of a change

aedeu(1)
recall a change for further development

aerf(1) fail a change review

aerp(1) pass a change review

aet(1) test a change

aepconf(5)
project configuration file format

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -Develop_End_Undo – recall a change for further development

SYNOPSIS

aegis -Develop_End_Undo *change-number* [*option*...]

aegis -Develop_End_Undo -List [*option*...]

aegis -Develop_End_Undo -Help

DESCRIPTION

The *aegis -Develop_End_Undo* command is used to recall a change for further development.

Successful execution of this command returns the change to the *being developed* state.

The files are changed back to being owned by the current user, and cease to be read-only.

OPTIONS

The following options are understood:

-Change *number*
: This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
: This option may be used to obtain more information about how to use the *aegis* program.

-List
: This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
: This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
: This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
: This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait
: This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
: This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see

aeuconf(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aedeu 'aegis -deu \!* -v'
sh$	aedeu(){aegis -deu $* -v}

ERRORS

It is an error if the change is not in one of the *being reviewed* or *awaiting integration* states.
It is an error if the change was not developed by the current user.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

*aede*1 complete development of a change

*aerp*1 pass review of a change

*aerf*1 fail review of a change

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aedist – remotely distribute a change

SYNOPSIS

aedist -Send [*option*...]
aedist -Receive [*option*...]
aedist -List [*option*...]
aedist -Help
aedist -VERSion

DESCRIPTION

The *aedist* command is used to send and receive change sets to facilitate geographically distributed development. The expected transport mechanism is e-mail, however other mechanisms are equally possible.

The basic function is to reproduce a change, so a command like

```
aedist -send | aedist -receive
```

may be used to clone a change, though less efficiently than *aeclone*(1). The file format used is designed to withstand mail servers, so activities such as

```
aedist -send | e-mail | aedist -receive
```

(where *e-mail* represents sending, transporting and receiving your e-mail) will reproduce the change on a remote system. With suitable tools (such as PGP) is it possible to

```
aedist -send | encrypt | e-mail | decrypt | aedist -receive
```

The mechanism is also designed to allow web-based distribution such as

```
aedist -send | web-server → web-browser | aedist -receive
```

by the use of appropriate CGI scripts and mailcap entries.

It is possible to support both a “push” model and a “pull” model using this command. For suggestions and ideas for various ways to do this, see the Aegis Users Guide.

SEND

The send variant takes a specified change, or baseline, and constructs a distribution package containing all of the change attributes and source file attributes and source file contents. The result is compressed, and encoded into a text format which can be sent as e-mail without being corrupted by the mail transfer agents along the way.

Options

The following options are understood by the send variant:

–Ascii_Armor
This option may be used to specify that MIME base 64 encoding is to be used on the output. This is the default.

–No_Ascii_Armor
This option may be used to specify that no encoding is to be uses on the output. The output will be a binary file.

–BaseLine
This option may be used to specify the source of a project, rather than a change. Implies the *–Entire_Source* option, unless over-ridden.

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aedistrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

−COmpress

This option may be used to specify that the output is to be compressed. This is the default.

−No_COmpress

This option may be used to specify that the output is not to be compressed.

−Description_Header

This option may be used to add an RFC 822 style header to the change description being sent, with a From and Date line. This is the default.

−No_Description_Header

This option suppresses the description header.

−Entire_Source

This option may be used to send the entire source of the project, as well as the change source files.

−Not_Entire_Source

This option may be used to send only source files of a change. This is the default, except for the *−BaseLine* option.

−Output *filename*

This option may be used to direct the output into the specified file. Output is sent to the standard output by default.

−Trojan This option may be used to treat the change set as if it had a Trojan horse attack in it.

−No_Trojan

This option may be used to treat the change set as if it definitely does not have a Trojan horse attack in it. *Use with extreme care.* You need to have authenticated the message with something like PGP first **and** know the the author well.

-Project *name*

This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aedistrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

RECEIVE

The receive variant takes a change package created by the send variant and creates an Aegis change (see *aenc*(1)) to implement the change within. Files are added to the change (see *aenf*(1), *aecp*(1), *aerm*(1), *aent*(1)) and then the file contents are unpackaged into the development directory.

The change is then built (see *aeb*(1)), differenced (see *aed*(1)), and tested (see *aet*(1)). If all of this is successful, development of the change is ended (see *aed*(1)). The automatic process stops at this point, so that a local reviewer can confirm that the change is desired.

Options

The following options are understood by the receive variant:

–Change *number*
This option may be used to choose the change number to be used, otherwise one will be chosen automatically.

–DELta *number*
This option may be used to specify a particular delta in the project's history to copy the file from, just as for the *aecp*(1) command. You may also use a delta name instead of a delta number.

–File *filename*
Read the change set from the specified file. The default is to read it from the standard input.

–Project *name*
This option may be used to set the project name. If not specified, the project name in the input package will be used, rather than the usual project name defaulting mechanism.

Security

Receiving changes by e-mail, and automatically committing them to the baseline without checking them, would be a recipe for disaster. A number of safeguards are provided:

- The format of the package is confirmed to be correct, and the package verified for internal consistency, before it is unpacked and acted upon.
- The automatic portion of the process stops when development ends. This ensures that a local reviewer validates the change before it is committed, preventing accidental or malicious damage.
- If the change seeks to update the project *config* file, the automatic process terminates before the build or difference occurs. This is because this file could contain trojans for these operations, so a human must examine the file before the change proceeds any further.
- There is a *potential_trojan_horse = [string];* field in the project*config* file. Nominate build config files, shell scripts, code generators, *etc* here to specify files in addition to the config file which should cause the automatic processing to halt.
- The use of e-mail authentication and encryption systems, such as PGP and GPG, are encouraged. However, it is expected that this processing will occur after *aedist --send* has constructed the package and before *aedist --receive* examines and acts on the package. Verification of the sender is the surest defense against trojan horses.
- Automatic sending and receiving of packages is supported, but not implemented within the aedist command. It is expected that the aedist command will be used within shell scripts customized for your site and its unique security requirements. See the Aegis User Guide for several different ways to do this.
- The more you use Aegis' test management facilities (see *aent*(1) and *aet*(1)) the harder it is for an inadequate change to get into the baseline.

Duplicate Storms

In a distributed development environment, it is common for change sets to eventually be propagated back to the originator. There are situations (particularly in some star topologies) where several copies of the package will return to the originator.

If these change sets are not detected at the review stage, and are propagated out yet again, there is the possibility of an exponential explosion of redundant packages being distributed again and again.

To combat this, changes are checked after the files are unpacked, but before and build or difference or test is performed. The "*aecpu --unchanged*" command is used to exclude all files that the local repository already has in the desired form. If no change files remain after this, the change is dropped entirely (see *aedbu*(1) and *aencu*(1)).

LIST

The list variant can be used to list the contents of a package without actually unpacking it first. The output is reminiscent of the *aegis -list change-details* output.

Options

The following options are understood by the list variant:

−File *filename*
: Read the change set from the specified file. The default is to read it from the standard input.

−Output *filename*
: Write the listing to the specified file. Defaults to the standard output if not specified.

OPTIONS

The following options to this command haven't been mentioned yet:

-Help
: This option may be used to obtain more information about how to use the *aedist* program.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aedist* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

FILE FORMAT

The file format re-uses existing formats, rather than introduce anything new. This means it is possible to extract the contents of a package even when aedist is unavailable.

- The source files and other information is stored as a *cpio*(1) archive.
- The archive is compressed using the GNU gzip format. Typically primary source files are ASCII text, resulting in significant compression.
- The compressed result is encoded using the MIME base64 encoding. This makes the result approximately 33% larger than the compressed binary would be, but still smaller than the primary sources.

The *cpio* archive is used to store

`etc/project-name`
: This contains the project name to apply the package to, unless over-ridden by the --project command line option.

`etc/change-set`
: This contains the change attributes and the list of source files and usages, in *aecstate*(5) format.

`src/`*filename*
: Each source file in the package (named in *etc/change-set*) appears under the *src/* directory.

Extra files, or files out of order, are a fatal error.

EXIT STATUS

The *aedist* command will exit with a status of 1 on any error. The *aedist* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

CREDITS

This program evolved through discussion with a number of people. If I have forgotten anyone, it wasn't intentional.

Ralf Fassel	<ralf@akutech.de>	Catching trojan horses.
Florian Xhumari	<Florian.Xhumari@inria.fr>	On the need for pull interfaces.
Graham Wheeler	<gram@cdsec.com>	HTTP pull interfacing.

NAME

aegis -Delta_Name – assign a symbolic name to a project delta

SYNOPSIS

aegis -Delta_Name *delta_number* [*option...*] *name*
aegis -Help
aegis -VERSion

DESCRIPTION

The *aegis -Delta_Name* command is used to add a symbolic name to a project delta. This is so that this name may be used, rather than the number, when extracting previous versions of the file using the *aecp*(1) command.

The number on the command line is a delta number of the project. That is, it is the delta number assigned to an integration. Delta names may only be applied to project baselines. If no delta number is given on the command line, the current baseline is the default.

A name must be given on the command line. This is the name which will be assigned to the delta. If the name has already been used, you will be given a fatal error message. If you also specify the **-OverWriting** option the name will be removed from its previous delta and assigned to the requested delta.

OPTIONS

The following options are understood:

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-OverWriting
This option may be used to force overwriting of files. The default action is to give an error if an existing file would be overwritten.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aecp(1) copy a file into a change, particularly the **-DELta** option

aeib(1) start the integration of a change

ael(1) list interesting this, particularly the *project_history* listing

COPYRIGHT

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aefind – search for files in directory hierarchy

SYNOPSIS

aefind [*option*...] *path*... *expression*
aefind -Help
aefind -VERSion

DESCRIPTION

The *aefind* command is used to search the combined directory tree of a change and its project. It is intentionally similar to *find (1),* however it unifies the directory stack of a change and its branch baseline, and the branch's ancestors' baselines if any.

For each file found in the directory tree, the given expression is evaluated from left to right, according to the rules of precedence (see the section on OPERATORS, below), only until the outcome is known, at which point aefind moves on to the next file name.

OPTIONS

The following options are understood:

-BaseLine
This option may be used to specify that the project baseline is the subject of the command.

–BAse_RElative
This option may be used to cause relative filenames to be considered relative to the base of the source tree. See *aeuconf*(5) for the corresponding user preference.

–CUrrent_RElative
This option may be used to cause relative filenames to be considered relative to the current directory. This is usually the default. See *aeuconf*(5) for the corresponding user preference.

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aefindrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aefind* program.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aefindrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

–Resolve
This option may be used to request that filenames be absolute paths, referring to the fully resolved file name. By default, relative file names are used, relative to the root of the "stacked" directory tree.

−Verbose
This option may be used to request that the expression be printed again on the standard output. This is the expression as understood by *aefind*, to assist you in ensuring that you and the command agree. The expression is fully parenthesized, and all implicit operators made explicit. Where possible, constant expressions will have been folded.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aefind* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

EXPRESSIONS

The expression is made up of basic elements, tests (which return a true or false value), and actions (which have side effects and return a true or false value), all separated by operators.

BASIC ELEMENTS

{} The value of this expression is the full file name of the file currently being considered. The value is affected the the **-Resolve** option.

number Numbers may be specified directly, for use with other tests and operators. In the style of C, they may be hexadecimal with a ‘‘0x’’ prefix, octal with a ‘‘0’’ prefix, or decimal otherwise.

string Strings may be specified directly, for use with other tests and operators. If the string contains shell meta-characters, you may need to quote it.

−FAlse The value of this expression is always false.

−NOW The value of this expression is the current time, at the start of execution.

−TRue The value of this expression is always true.

OPERATORS

The **−and** operator is assumed where the operator is omitted. You will need to quote many of the operators, to protect them from interpretation by the shell. Each operator must be a separate command line argument.

(*expr* **)** Force precedence.

+ *expr* Unary plus. Is is an error if the argument cannot be coerced to a number.

- *expr* Unary minus. Result is the numeric negative of the argument. Is is an error if the argument cannot be coerced to a number.

! *expr* Logical negation of the sense of the expression. Is is an error if the argument cannot be coerced to a boolean.
Synonym: **−Not**

˜ *expr* Bitwise not of the argument. Is is an error if the argument cannot be coerced to an integer.

expr1 * *expr2*
This operation multiplies the two values. Is is an error if the arguments cannot be coerced to numbers.

expr1 / *expr2*
This operation divides the argument value by the second. Is is an error if the arguments cannot be coerced to numbers. Is is an error if the second argument is zero.

expr1 % *expr2*
This operation produces the remainder of the division of the first argument by the argument. Is is an error if the arguments cannot be coerced to numbers. Is is an error if the second argument is zero.

expr1 ~ *expr2*
Is is an error if the arguments cannot be coerced to strings. Is is an error if the first argument is not a valid pattern. The first argument is the pattern, and the second is the string The result is true if the pattern matches, and false if it does not. This operation performs a shell file pattern comparison. to be compared.

expr1 + *expr2*
This operation adds the two values. Is is an error if the values cannot be coerced to numbers.

expr1 - *expr2*
This operation subtracts the second values from the first. Is is an error if the values cannot be coerced to numbers.

expr1 ## *expr2*
This operation concatenates the arguments. Is is an error if the arguments cannot be coerced to strings. (Note: this is *not* the same as the **:** operator of the *expr*(1) command.)

expr1 << *expr2*
Shift the first argument left by the number of bits specified by the second argument. The left argument is treated as an *unsigned* number. Is is an error if the values cannot be coerced to numbers.

expr1 >> *expr2*
Shift the first argument right by the number of bits specified by the second argument. The left argument is treated as an *unsigned* number. Is is an error if the values cannot be coerced to numbers.

expr1 < *expr2*
Compare the values and produce true if the first value is less than the second value, false otherwise. If both values can be coerced to numbers, the comparison is numeric; if both values can be coerced to strings, the comparison is lexicographic; otherwise is it an error.

expr1 <= *expr2*
Compare the values and produce true if the first value is less than or equal to the second value, false otherwise. If both values can be coerced to numbers, the comparison is numeric; if both values can be coerced to strings, the comparison is lexicographic; otherwise is it an error.

expr1 > *expr2*
Compare the values and produce true if the first value is greater than the second value, false otherwise. If both values can be coerced to numbers, the comparison is numeric; if both values

can be coerced to strings, the comparison is lexicographic; otherwise is it an error.

expr1 **>=** *expr2*
: Compare the values and produce true if the first value is greater than or equal to the second value, false otherwise. If both values can be coerced to numbers, the comparison is numeric; if both values can be coerced to strings, the comparison is lexicographic; otherwise is it an error.

expr1 **==** *expr2*
: Compare the values and produce true if the first value is equal to the second value, false otherwise. If both values can be coerced to numbers, the comparison is numeric; if both values can be coerced to strings, the comparison is lexicographic; otherwise is it an error.

expr1 **!=** *expr2*
: Compare the values and produce true if the first value is not equal to the second value, false otherwise. If both values can be coerced to numbers, the comparison is numeric; if both values can be coerced to strings, the comparison is lexicographic; otherwise is it an error.

expr1 **&** *expr2*
: This operation produces the bitwise-and of the two values. Is is an error if the values cannot be coerced to numbers.

expr1 **|** *expr2*
: This operation produces the bitwise-or of the two values. Is is an error if the values cannot be coerced to numbers.

expr1 **&&** *expr2*
: Result is true if both expressions are true. Short circuit evaluation is used, and so *expr2* is not evaluated if *expr1* is false. Is is an error if the arguments cannot be coerced to booleans.
Synonym: **−And**

expr1 expr2
: Logical and (implied). Result is true if both expressions are true. Short circuit evaluation is used, and so *expr2* is not evaluated if *expr1* is false. Please note that implicit operator plays merry hell with operator precedence, because there is no operator. If you are getting odd results, use explicit operators.

expr1 **||** *expr2*
: Result is true if either expression is true. Short circuit evaluation is used, and so *expr2* is not evaluated if *expr1* is true. Is is an error if the arguments cannot be coerced to booleans.
Synonym: **−Or**

expr1 **?** *expr2* **:** *expr3*
: The value of this expression is *expr2* if *expr1* is true, and *expr3* otherwise. The *expr1* is always evaluated, but only one of *expr2* or *expr3* will be evaluated. It is an error if the value of *expr1* cannot be coerced to boolean.

expr1 **,** *expr2*
: Both *expr1* and *expr2* are always evaluated. The value of *expr1* is discarded; the value of the expression is the value of *expr2*.

Operators have precedence as described by the following table, highest to lowest:

Operator	Direction
(unary) + - ~ !	←
* / % ~	→
+ - :	→
<< >>	→
< <= > >=	→
== !=	→
&	→
^	→
|	→
&&	→
||	→
? :	→
,	→

FUNCTIONS

There are a number of built-in functions which may be used in the expression. Functions may be invoked using a syntax similar to C functions.

name **(** *arguments* **)**

You need to leave spaces around the parentheses so that they are separate command line arguments.

atime This function may be used to determine the last-accessed-time of a file. It takes one argument.

basename
This function returns the basename of the string argument passed to it. It takes one argument.

ctime This function may be used to determine the last-change-time of an inode. It takes one argument.

gid This function may be used to determine the gid of a file. It takes one argument.

inode This function may be used to determine the inode number of a file. It takes one argument.

mode This function may be used to determine the access mode (permissions) of a file. It takes one argument.

mtime This function may be used to determine the last-modified-time of a file. It takes one argument.

print This function may be used to print a value. It takes one argument. Always returns true.

size This function may be used to determine the size in bytes of a file. It takes one argument.

type This function may be used to determine the type of a file. It takes one argument. It returns a string: "block_special", "character_special", "directory", "file", "named_pipe", "socket" or "symbolic_link".

uid This function may be used to determine the uid of a file. It takes one argument.

TESTS

Most tests exist to provide compatibility with *find*(1).

−Access_Minutes [*relative-operator*] *number*
True if the current file was accessed exactly *number* minutes ago, false otherwise. If a relative operator is given (<, <=, ==, !=, > or >=) a relative comparison will be made, rather than the implicit equality test. This is *not* identical to the similar *find*(1) test. This is shorthand for the ‘‘(now - atime ({})) / 60 *relative-operator number*’’ expression.

−Access_Time [*relative-operator*] *number*
True if the current file was accessed exactly *number* days ago, false otherwise. If a relative operator is given (<, <=, ==, !=, > or >=) a relative comparison will be made, rather than the implicit equality test. This is *not* identical to the similar *find*(1) test. This is shorthand for the "(now - atime ({})) / 86400 *relative-operator number*" expression.

−Change_Minutes *number*
True if the current file's inode was changed exactly *number* minutes ago, false otherwise. If a relative operator is given (<, <=, ==, !=, > or >=) a relative comparison will be made, rather than the implicit equality test. This is *not* identical to the similar *find*(1) test. This is shorthand for the "(now - ctime ({})) / 60 *relative-operator number*" expression.

−Change_Time *number*
True if the current file's inode was changed exactly *number* days ago, false otherwise. If a relative operator is given (<, <=, ==, !=, > or >=) a relative comparison will be made, rather than the implicit equality test. This is *not* identical to the similar *find*(1) test. This is shorthand for the "(now - ctime ({})) / 86400 *relative-operator number*" expression.

−Modify_Minutes *number*
True if the current file was modified exactly *number* minutes ago, false otherwise. If a relative operator is given (<, <=, ==, !=, > or >=) a relative comparison will be made, rather than the implicit equality test. This is *not* identical to the similar *find*(1) test. This is shorthand for the "(now - mtime ({})) / 60 *relative-operator number*" expression.

−Modify_Time *number*
True if the current file was modified exactly *number* days ago, false otherwise. If a relative operator is given (<, <=, ==, !=, > or >=) a relative comparison will be made, rather than the implicit equality test. This is *not* identical to the similar *find*(1) test. This is shorthand for the "(now - mtime ({})) / 86400 *relative-operator number*" expression.

−Newer *filename*
True if the current file was modified after the given file. This is shorthand for the "mtime ({}) > mtime (*filename*)" expression.

-Name *pattern*
Base of file name (the path with the leading directories removed) matches shell pattern *pattern*. This is short-hand for the "*pattern* ~ basename ({})" expression.

-PAth *pattern*
File name matches shell pattern *pattern*. Note that the file name if affected by the **−resolve** option. This is short-hand for the "*pattern* ~ {}" expression.

−Type *string*
The file type matches the type given. This is shorthand for the "type ({}) == *string*" expression. Type names are matched similar to options:

Block	The file is a block special file.
Character	The file is a character special file.
Directory	The file is a directory.
File	The file is a normal file.
Link	The file is a symbolic link.
Pipe	The file is FIFO (a named pipe).

Socket — The file is a UNIX domain socket.

ACTIONS

–print This will print the full file name on the standard output, followed by a newline. The **-Resolve** option will affect what is printed. This is short-hand for the ‘‘print ({})’’ expression.

EXIT STATUS

The *aefind* command will exit with a status of 1 on any error. The *aefind* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/˜millerp/

NAME

aefp – calculate file fingerprint

SYNOPSIS

aefp [*option*...][*filename*...]
aefp -Help
aefp -VERSion

DESCRIPTION

The *aefp* program is used to calculate the fingerprints of files. A fingerprint is a hash of the contents of a file. The default fingerprint is cryptographically strong, so the probability of two different files having the same fingerprint is less than 1 in 2**200.

The fingerprint is based on Dan Berstien <djb@silverton.berkeley.edu> public domain fingerprint 0.50 beta package 930809, posted to the alt.sources newsgroup. This program produces identical results; the expected test results were generated using Dan's package.

The fingerprint is a base-64-sanely-encoded fingerprint of the input. Imagine this fingerprint as something universal and permanent. A fingerprint is 76 characters long, containing the following:

1. A Snefru-8 (version 2.5, 8 passes, 512->256) hash. (Derived from the Xerox Secure Hash Function.)
2. An MD5 hash, as per RFC 1321. (Derived from the RSADSI MD5 Message-Digest Algorithm.)
3. A CRC checksum, as in the new cksum utility.
4. Length modulo 2^40.

The output format is not expected to be compatible with anything. However, options are available to produce the purported output of Merkle's snefru program, the purported output of RSADSI's mddriver -x, or the purported output of the POSIX cksum program.

If no files are named as input, the standard input will be used. The special file name "-" is understood to mean the standard input.

OPTIONS

The following options are understood:

-Checksum
Print the CRC32 checksum and length of the named file(s).

-Identifier
Print a condensed form of the fingerprint (obtained by performing a CRC32 checksum on the full fingerprint described above - a definite overkill). This is an 8-digit hexadecimal number, useful for generating unique short identifiers out of long names. The first character is forced to be a letter (g-p), so there is no problem in using the output as a variable name.

-Help
Provide some help with using the *aefp* program.

-Message_Digest
Print the RSA Data Security, Inc. MD5 Message-Digest Algorithm hash of the named file(s).

-Snefru Print the Snefru hash of the named file(s), derived from the Xerox Secure Hash Function.

-VERSion
Print the version of the *aefp* program being executed.

All other options will produce a diagnostic error.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aefp* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

EXIT STATUS

The *aefp* command will exit with a status of 1 on any error. The *aefp* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

Portions of this program are derived from sources from other people, sometimes with liberal copyrights, and sometimes in the public domain. These include:

Dan Bernstien
See *common/fp/README* for details.

Gary S Brown.
See *common/fp/crc32.c* for details.

RSA Data Security, Inc.
See *common/fp/md5.c* for details.

Xerox Corporation
See *common/fp/snefru.c* for details.

In addition to the above copyright holders, there have been numerous authors and contributors, see the named files for details. Files names are relative to the root of the *aegis* distribution.

NAME

aegis -Integrate_Begin – begin integrating a change

SYNOPSIS

aegis -Integrate_Begin *change-number* [*option*...]
aegis -Integrate_Begin -List [*option*...]
aegis -Integrate_Begin -Help

DESCRIPTION

The *aegis -Integrate_Begin* command is used to begin the integration of a change into the baseline of a project.

The change will advance from the *awaiting integration* state to the *being integrated* state.

A (logical) copy of the baseline is created in an *integration directory* and the the files of the change are added to the integration directory. The time stamps of files copied from the baseline are preserved, time stamps on the files copied from the development directory are all set to the time of the beginning of the integration. The *'aegis -Change_Directory'* command may be used to locate the integration directory. The change will be assigned to the current user.

Please note that only regular files and symbolic links are copied (linked) from the baseline to the integration directory. This has some implications:

- Special files (devices, named pipes, *etc*) will not be reproduced in the integration directory; you will need to create these as part of the build.
- If the case of the **-mimimum** option (see below), only primary source files are copied (linked) across. Derived files (including symbolic links) are expected to be created as part of the build.
- If the case of the **-minimum** option, directories are only created when required to hold a file which satisfies the above criteria. If you need special empty directories, or directories which contain only special files, or only contain derived files, you need to create them as part of the build.

The *link_integration_directory* field of the project *config* file (see *aepconf*(5) for more information) controls whether the copy of the baseline is done by copying the files or by creating hard links to the files. The hard links are just one of the constraints on the location of the integration directory. The integrate begin will abort with an error if this copy operation fails, e.g. by running out of disk space. If this should happen, the change will remain in the *awaiting integration* state, and the integration directory will be removed.

The change will be assigned a delta number. Delta numbers are incremented once for each *aegis -Integrate_Begin* command for the project. If an integration is subsequently aborted with either the *aegis -Integrate_Begin_Undo* or *aegis -Integrate_FAIL* command, the delta number will not be re-used.

It is not possible to choose the integration directory, as there are many constraints upon it, including the fact that it must be on the same device as the baseline directory, and that many UNIX implementations don't allow renaming directories up and down the trees. The integration directory will be in the project directory, and named for the delta number.

OPTIONS

The following options are understood:

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-MAXimum
This option may be used to cause all files to be copied into the integration directory. This is the default, unless the change requires the deletion of a file.

-MINImum
This option may be used to cause only the source files to be copied into the integration directory. The default is to copy all files, unless the change requires the deletion of a file.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aeib 'aegis -ib \!* -v'

sh$ aeib(){aegis -ib $* -v}

ERRORS

It is an error if the change is not in the *awaiting integration* state.
It is an error if the current user is not an integrator of the project.
It is an error if there is an integration in progress for the project.
It is an error if the current user developed the change and the project is configured to disallow developers to integrate their own changes (default).
It is an error if the current user reviewed the change and the project is configured to disallow reviewers to integrate their such changes (default).

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeb(1) build a change

aecd(1) change directory

aeibu(1) reverse the aeib command

aeif(1) fail integration of a change

aeip(1) pass integration of a change

aeni(1) add new integrators to a project

aerp(1) pass review of a change

aet(1) run tests

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -Integrate_Begin_Undo – reverse the aeib command

SYNOPSIS

aegis -Integrate_Begin_Undo [*option*...]
aegis -Integrate_Begin_Undo -List [*option*...]
aegis -Integrate_Begin_Undo -Help

DESCRIPTION

The *aegis -Integrate_Begin_Undo* command is used to reverse the actions of the *'aegis -Integrate_Begin'* command.

Successful execution of this command will move the change from the *being integrated* state to the *awaiting integration* state. The integration directory will be deleted. The change will cease to be assigned to the current user.

OPTIONS

The following options are understood:

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-Keep
This option may be used to retain files and/or directories usually deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-No_Keep
This option may be used to ensure that the files and/or directories are deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aeibu 'aegis -ibu \!* -v'

sh$ aeibu(){aegis -ibu $* -v}

ERRORS

It is an error if the change is not in the *being_integrated* state.
It is an error if the change is not assigned to the current user.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeib(1) begin integration of a change

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -Integrate_Fail – fail a change integration

SYNOPSIS

aegis -Integrate_Fail -File *reason-file* [*option*...]
aegis -Integrate_Fail -REAson *'reason-text'* [*option*...]
aegis -Integrate_Fail -Edit [*option*...]
aegis -Integrate_Fail -List [*option*...]
aegis -Integrate_Fail -Help

DESCRIPTION

The *aegis -Integrate_Fail* command is used to inform aegis that a change has failed integration.

The change will be returned from the *being integrated* state to the *being developed* state. The change will cease to be assigned to the current user, and will be reassigned to the originating developer. The integration directory will be deleted.

The reviewer and the developer will be notified by mail. See the integrate_fail_notify_command in *aepconf*(5) for more information.

The *reason-file* will contain a description of why the change was failed. The file is in plain text. It is recommended that you only use newline to terminate paragraphs, (rather than to terminate lines) as with will result in better formatting in the various listings.

OPTIONS

The following options are understood:

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Edit

Edit the attributes with a text editor, this is usually more convenient than supplying a text file. The *VISUAL* and then *EDITOR* environment variables are consulted for the name of the editor to use; defaults to *vi*(1) if neither is set. Warning: tries to be well behaved when faced with errors, the temporary file is left in your home directory. This option may not be used in the background, or when the standard input is not a terminal.

-Edit_BackGround

Edit the attributes with a dumb text editor, this is most often desired when edit commands are being piped into the editor via the standard input. Only the *EDITOR* environment variable is consulted for the name of the editor to use; it is a fatal error if it is not set.

-File *filename*

Take the attributes from the specified file.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-Keep

This option may be used to retain files and/or directories usually deleted by the command.

Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-No_Keep
This option may be used to ensure that the files and/or directories are deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

−REAson *text*
This option may be used to provide the failure reason on the command line, rather than in a file. You will need to use quotes to insulate the spaces from the shell.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aeifail 'aegis -ifail \!* -v'
sh$	aeifail(){aegis -ifail $* -v}

ERRORS

It is an error if the change is not in the *being integrated* state.
It is an error if the change is not assigned to the current user.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeib(1) begin integration of a change

aeip(1) pass integration of a change

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/˜millerp/

NAME

aegis -Integrate_Pass – pass a change integration

SYNOPSIS

aegis -Integrate_Pass [*option*...]
aegis -Integrate_Pass -List [*option*...]
aegis -Integrate_Pass -Help

DESCRIPTION

The *aegis -Integrate_Pass* command is used to notify aegis that a change has passed integration. The change is advanced from the *being integrated* state to the *completed* state.

This command updates the file histories, so that future *aecp*(1) commands may extract previous file versions from history, and so that future *aed*(1) commands may merge out-of-date files. The history is updated using the *history_create_command* and *history_put_command* fields of the project *config* file (see *aepconf*(5) for more information). The integrate pass will abort with an error if one of these history commands should fail, *e.g.* by running out of disk space. If this should happen, the change will remain in the *being integrated* state, and the integration directory is unaltered.

Many history tools (*e.g.* RCS and SCCS) can modify the contents of the file when it is committed. This usually requies the use of specific ‘‘keyword’’ strings, and there are usually options to turn this behavior off, but users familiar with version control tools (as opposed to configuration management systems) will often use these features. The problem is that if the commit changes the file, the source file in the repository now no longer matches the object file in the repository. *I.e.* the history tool has compromised the referential integrity of the repository. By default, a fatal error is emitted if the file is changed by the check-in, however this can be modified to a be warning or even ignored completely; see the *history_put_trashes_-file* field of *aepconf*(5) for more information.

Once the history has been updated, the integration directory is renamed as the baseline directory, and the old baseline directory is deleted. The modification times of all files modified since the beginning of integration (see *aeib*(1) for more information) are updated to be since the beginning of integrate pass. The order of modification times will be preserved, however the time range will be compressed to the greatest extent possible. This ensures that subsequent development builds will notice that baseline files have changed.

Once integrate pass is complete the change is no longer assigned to the current user.

THE BASELINE LOCK

The baseline lock is used to ensure that the baseline remains in a consistent state for the duration of commands which need to read the contents of files in the baseline.

The commands which require the baseline to be consistent (these include the *aeb*(1), *aecp*(1) and *aed*(1) commands) take a baseline *read* lock. This is a non-exclusive lock, so the concurrent development of changes is not hindered.

The command which modifies the baseline, *aeip*(1), takes a baseline *write* lock. This is an exclusive lock, forcing *aeip*(1) to block until there are no active baseline read locks.

It is possible that one of the above development commands will block until an in-progress *aegis -Integrate_PASS* completes. This is usually of short duration while the project history is updated. The delay is essential so that these commands receive a consistent view of the baseline. No other integration command will cause the above development commands to block.

When aegis' branch functionality is in use, a read (non-exclusive) lock is taken on the branch baseline and also each of the "parent" baselines. However, a baseline write (exclusive) lock is only taken on the branch baseline; the "parent" baselines are only read (non-exclusive) locked.

The History Lock

Where a project has a number of branches active simultaneously, it is possible for independent integrate pass commands for different branches to be issued very close together. The is an exclusive *history lock* taken by integrate pass to ensure that only one branch is updating the file history at a time, thus preventing history file corruption.

TEST CORRELATIONS

The ‘‘aegis -Test -SUGgest’’ command may be used to have aegis suggest suitable regression tests for your change, based on the source files in your change. This automatically focuses testing effort to relevant tests, reducing the number of regression tests necessary to be confident that you have not introduced a bug.

The test correlations are generated by the ‘‘aegis -Integrate_Pass’’ command, which associates each test in the change with each source file in the change. Thus, each source file accumulates a list of tests which have been associated with it in the past. This is not as exact as code coverage analysis, but is a reasonable approximation in practice.

The *aecp*(1) and *aenf*(1) commands are used to associate files with a change. While they do not actively perform the association, these are the files used by *aeip*(1) and *aet*(1) to determine which source files are associated with which tests.

Test Correlation Accuracy

Assuming that the testing correlations are accurate and that the tests are evenly distributed across the function space, there will be a less than *1/number* chance that a relevant test has not been run by the ‘‘aegis -Test -SUGgest *number*’’ command. A small amount of noise is added to the test weighting, so that unexpected things are sometimes tested, and the same tests are not run every time.

Test correlation accuracy can be improved by ensuring that:

- Each change should be strongly focused, with no gratuitous file inclusions. This avoids spurious correlations.
- Each item of new functionality should be added in an individual change, rather than several together. This strongly correlates tests with functionality.
- Each bug should be fixed in an individual change, rather than several together. This strongly correlates tests with functionality.
- Test correlations will be lost if files are moved. This is because correlations are by name.

The best way for tests to correlate accurately with source files is when a change contains a test and exactly those files relating to the functionality under test. Too many spurious files will weaken the usefulness of the testing correlations.

METRICS

Aegis is capable of recording metrics as part of the file attributes of a change. This allows various properties of files to be recorded for later trend analysis, or other uses.

The specific metrics are not dictated by Aegis. It is expected that the integration build will create a metrics file for each of the source files the change. These metrics files must be in the format specified by *aemetrics*(5).

The name of the metrics file defaults to ‘‘*filename*,**S**’’, however it may be varied, by setting the *metrics_-filename_pattern* field of the project *config* file. See *aepconf*(5) for more information.

If such a metrics file exists, for each source file in a change, it will be read and remembered at integrate pass time. If it does not exist, Aegis assumes there are no relevant metrics for that file, and proceeds silently; it is not an error.

OPTIONS

The following options are understood:

-Change *number*
: This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
: This option may be used to obtain more information about how to use the *aegis* program.

-List
: This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Not_Logging
: This option may be used to disable the automatic logging of output and errors to a file. This is often useful when several aegis commands are combined in a shell script.

-Project *name*
: This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
: This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
: This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait
: This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
: This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aeipass 'aegis -ipass \!* -v'
sh$	aeipass(){aegis -ipass $* -v}

ERRORS

It is an error if the change is not assigned to the current user.
It is an error if The change is not in the *being integrated* state.
It is an error if there has been no successful *'aegis -Build'* command for the integration.
It is an error if there has been no successful *'aegis -Test'* command for the integration.
It is an error if there has been no successful *'aegis -Test -BaseLine'* command for the integration.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeib(1) begin integration of a change

aeif(1) fail integration of a change

aemetrics(5)
metrics values file format

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -List – list (possibly) interesting things

SYNOPSIS

aegis -List [*option*...] *list-name*
aegis -List -List [*option*...]
aegis -List -Help

DESCRIPTION

The *aegis -List* command is used to list information. There are a number of possible *list-name*s, as follows (abbreviations as for command line options):

Administrators
: List the administrators of a project.

Change_Details
: List full information about a change in large format.

Change_Files
: List all files in a change.

Change_History
: List the history of a change.

Changes
: List the changes of a project.

Default_Change
: List the default change for the current user.

Default_Project
: List the default project for the current user.

Developers
: List the developers of a project.

Integrators
: List the integrators of a project.

List_List
: List all lists available.

Locks
: List all currently active locks.

Outstanding_Changes
: List all changes not yet completed.

All_Outstanding_Changes
: List all changes not yet completed, for all projects.

Project_Files
: List all files in the baseline of a project.

Project_History
: List the integration history of a project.

Projects
: List all projects.

Reviewers
: List the reviewers of a project.

Users_Changes
: List of changes owned by the current user.

Version
: List version of a project or change. This includes the major and minor version number, and the previous version number if available. The list of copyright years is also printed.

Most of these lists are available from other aegis functions. Many aegis functions provide more specific lists.

OPTIONS

The following options are understood:

-Change *number*
: This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
: This option may be used to obtain more information about how to use the *aegis* program.

-List
: This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-PAGer The output of listings and help is piped through the pager command given in the PAGER environment variable (or *more* if not set). This is the default if the command is in the foreground. This option may be used to override an y preference specified in the *aeuconf*(5) file.

-No_PAGer
: This option may be used to ensure that the output of listings and help is not piped through a pager command. This option may be used to override an y preference specified in the *aeuconf*(5) file.

-Page_Length *number*
: This option may be used to set the page length of listings. The default, in order of preference, is obtained from the system, from the *LINES* environment variable, or set to 24 lines.

-Page_Width *number*
: This option may be used to set the page width of listings and error messages. The default, in order of preference, is obtained from the system, from the *COLS* environment variable, or set to 79 characters.

-Project *name*
: This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single

project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-UNFormatted
This option may be used with most listings to specify that the column formatting is not to be performed. This is useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias ael 'aegis -l \!* -v'

sh$ ael(){aegis -l $* -v}

ERRORS

It is an error if the list name given is unknown.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aedb(1) begin development of a change (listing option)

aeib(1) begin integration of a change (listing option)

aer(1) report generator

aerp(1) pass review of a change (listing option)

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> *one line to give the program's name and a brief idea of what it does.*
> Copyright (C) 19yy *name of author*
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) 19yy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> *signature of Ty Coon,* 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

NAME

aegis -MoVe_file – rename a file as part of a change

SYNOPSIS

aegis -MoVe_file [*option*...] *old-name new-name*
aegis -MoVe_file -List [*option*...]
aegis -MoVe_file -Help

DESCRIPTION

The *aegis -MoVe_file* command is used to copy a file into a change and change its name at the same time.

The named files will be copied from the baseline (*old-file*) into the development directory (*new-file*), and added to the list of files in the change.

Warning: If there is already files in the development directory of either the *old-name* or the *new-name* they will be overwritten.

The *old-file* in the development directory will contain 1KB of random text. The random text is sufficiently revolting that most compilers will give error messages, should the file be referenced accidentally. This is often very helpful when moving include files.

You may rename directories. All the files in the *old-name* directory tree will be renamed to be below the *new-name* directory tree.

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

Process Side Effects

This command will cancel any build or test registrations, because adding another file logically invalidates them.

When the change files are listed (*aegis -List Change_Files -TERse*) the new files (*new-name*) will appear in the listing, and the removed files (*old-name*) will **not** appear in the terse listing. Similarly, when the project files are listed with an explicit change number (*aegis -List Project_Files -TERse -Change* `N`) none of the change's files, including both the new and removed files, will appear in the terse listing. These two features are very helpful when calling aegis from within a DMT to generate the list of source files.

OPTIONS

The following options are understood:

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Not_Logging
This option may be used to disable the automatic logging of output and errors to a file. This is often useful when several aegis commands are combined in a shell script.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aemv 'aegis -mv \!* -v'
sh$	aemv(){aegis -mv $* -v}

ERRORS

It is an error if the change is not in the *being developed* state.
It is an error if the change is not assigned to the current user.
It is an error if either file is already in the change.
It is an error if you attempt to move the project *config* file.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aecp(1) copy files into a change

aedb(1) begin development of a change

aenf(1) add files to be created by a change

aenfu(1) remove files to be created by a change

aerm(1) add files to be deleted by a change

aermu(1)
remove files to be deleted by a change

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -New_Administrator – add a new administrator to a project

SYNOPSIS

aegis -New_Administrator *user-name*... [*option*...]
aegis -New_Administrator -List [*option*...]
aegis -New_Administrator -Help

DESCRIPTION

The *aegis -New_Administrator* command is used to add a new administrator to a project.

OPTIONS

The following options are understood:

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aena 'aegis -na \!* -v'
sh$	aena(){aegis -na $* -v}

ERRORS

It is an error if the current user is not an administrator of the project.
It is an error if any of the named users have a uid of less than 100.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aepa(1) modify the attributes of a project

aera(1) remove administrators from a project

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/˜millerp/

NAME

aegis -New_BRanch – create a new branch

SYNOPSIS

aegis -New_BRanch [*number*][*option...*]

aegis -New_BRanch -Help

DESCRIPTION

The *aegis -New_BRanch* command is used to create a new branch. A branch is very similar to a *change*, except that a branch may have changes (or branches) of its own, and a change may not.

You may choose your own branch number, if you want. Zero and positives are legal, but negatives are not. It is an error if that number has already been used for a change or another branch. If you do not specify a change number, the lowest available positive number (1 or more) will be used.

The new branch will be a special sort of change. It will be in the '*being developed*' state, but the usual commands in that stat (build, diff, etc) will not work. Instead, you must create changes on the branch, and when those changes are integrated into the branch, this is the equivalent of build, diff, etc, on the branch. Once the branch is completed, the *aede*(1) command may be used to advance it to the *being reviewed* state, and from then on it becomes a normal change. Should it be returned to the *being developed* state for any reason, it will once again require sub-changes to alter anything.

OPTIONS

The following options are understood:

-Help

This option may be used to obtain more information about how to use the *aegis* program.

–Output *filename*

This option may be used to specify a filename which is to be written with the automatically determined branch number. Useful for writing scripts.

–Wait

This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait

This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aenbr 'aegis -nbr \!* -v'

sh$ aenbr(){aegis -nbr $* -v}

SEE ALSO

aenbru(1)
remove a branch

aenc(1) create a new change

aede(1) develop end

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au

/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -New_BRanch_Undo – remove a branch

SYNOPSIS

aegis -New_Branch_Undo *number* [*option*...]

aegis -New_Branch_Undo -Help

DESCRIPTION

The *aegis -New_BRanch_Undo* command is used to remove a branch created with the *aenbr*(1) command.

OPTIONS

The following options are understood:

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-Keep
This option may be used to retain files and/or directories usually deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-No_Keep
This option may be used to ensure that the files and/or directories are deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aenbr(1)
    create a new branch

aencu(1)
    new change undo

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -New_Change – add a new change to a project

SYNOPSIS

aegis -New_Change [*number*] **-File** *attr-file* [*option*...]
aegis -New_Change [*number*] **-Edit** [*option*...]
aegis -New_Change -List [*option*...]
aegis -New_Change -Help

DESCRIPTION

The *aegis -New_Change* command is used to add a new change to a project. See *aecattr*(5) for information on the format of the *attr-file*.

The change is created in the *awaiting development* state. The change is not assigned to any user. The change has no development directory.

You may choose your own change number if you want, provided that it has not been used already. If you do not specify a change number, aegis will allocate the lowest unused change number. The first few change numbers are reserved for branches later in the project, and so automatically allocated change numbers will usually not start from 1. See *aepa*(1) and *aepattr*(5) for more information.

OPTIONS

The following options are understood:

-Edit
Edit the attributes with a text editor, this is usually more convenient than supplying a text file. The *VISUAL* and then *EDITOR* environment variables are consulted for the name of the editor to use; defaults to *vi*(1) if neither is set. Warning: tries to be well behaved when faced with errors, the temporary file is left in your home directory. This option may not be used in the background, or when the standard input is not a terminal.

-Edit_BackGround
Edit the attributes with a dumb text editor, this is most often desired when edit commands are being piped into the editor via the standard input. Only the *EDITOR* environment variable is consulted for the name of the editor to use; it is a fatal error if it is not set.

-File *filename*
Take the attributes from the specified file.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

–Output *filename*
This option may be used to specify a filename which is to be written with the automatically determined change number. Useful for writing scripts.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more

information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aenc 'aegis -nc \!* -v'

sh$ aenc(){aegis -nc $* -v}

ERRORS

It is an error if the current user is not an administrator of the project. (In some cases it is possible for developers of a project to create changes, see *aepattr*(5) for more information.)

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeca(1) modify the attributes of a change

aedb(1) begin development of a change

aena(1) add a new administrator to a project

aencu(1)
remove a new change from a project

aenpr(1)
create a new project

aepa(1) modify the attributes of a project

aecattr(5)
change attributes file format

aepattr(5)
project attributes file format

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -New_Change_Undo – remove a new change from a project

SYNOPSIS

aegis -New_Change_Undo [*option*...]

aegis -New_Change_Undo -List [*option*...]

aegis -New_Change_Undo -Help

DESCRIPTION

The *aegis -New_Change_Undo* command is used to remove a new change from a project.

It wasn't called *'aegis -Remove_Change'* in order to emphasize that fact the the change must be in the *awaiting development* state. In practice it is possible, with a combination of commands, to remove any change which has not reached the *completed* state.

In general, only project administrators may destroy changes. However, if the project *developers_may_-create_changes* attribute is true, and you are a developer and you created a particular change, you may also destroy it.

OPTIONS

The following options are understood:

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*

This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse

This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose

This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see

aeuconf(5) for more information.

−No_Wait

This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aencu 'aegis -ncu \!* -v'
sh$     aencu(){aegis -ncu $* -v}
```

ERRORS

It is an error if the change is not in the *awaiting development* state.
It is an error if the current user is not an administrator of the project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aenc(1) add a new change to a project

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -New_Developer – add new developers to a project

SYNOPSIS

aegis -New_Developer *user-name*... [*option*...]
aegis -New_Developer -List [*option*...]
aegis -New_Developer -Help

DESCRIPTION

The *aegis -New_Developer* command is used to add new developers to a project.

OPTIONS

The following options are understood:

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait
This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aend 'aegis -nd \!* -v'

sh$ aend(){aegis -nd $* -v}

ERRORS

It is an error if the current user is not an administrator of the project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aerd(1) remove developers from a project

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -New_File − add new files to be created by a change

SYNOPSIS

aegis -New_File *file-name*... [*option*...]
aegis -New_File -List [*option*...]
aegis -New_File -Help

DESCRIPTION

The *aegis -New_File* command is used to add new files to a change. The named files will be added to the list of files in the change.

For each file named, a new file is created in the development directory, if it does not exist already. If the file already exists, it will not be altered.

When a new file is created in the development directory the project *config* file is searched for a template for the new file. If a template is found, the new file will be initialized to the template, otherwise it will be created empty. See *aepconf*(5) for more information.

File Name Limitations

There are a number of controls available to limit the form of project file names. All of these controls may be found in the project *config* file, see *aepconf*(5) for more information. The most significant are briefly described here:

maximum_filename_length = integer;
: This field is used to limit the length of filenames. All new files may not have path components longer than this. Defaults to 14 if not set, for maximum portability.

posix_filename_charset = boolean;
: This field may be used to limit the characters allowed in filenames to only those explicitly allowed by POSIX. Defaults to *false* if not set, meaning whatever your operating system will tolerate, except white space and high-bit-on characters.

dos_filename_required = boolean;
: This field may be used to limit filenames so that they conform to the DOS 8+3 filename limits and to the DOS filename character set. Defaults to *false* if not set.

windows_filename_required = boolean;
: This field may be used to limit filenames so that they conform to the Windows98 and WindowsNT filename limits and character set. Defaults to *false* if not set.

shell_safe_filenames = boolean;
: This field may be used to limit filenames so that they do not contain shell special characters. Defaults to *true* if not set. If this field is set to *false*, you will need to use the *${quote}* substitution around filenames in commands, to ensure that filenames containing shell special characters do not have unintended side effects. Weird characters in filenames may also confuse your dependency maintenance tool.

filename_pattern_accept = [string];
: This field is used to specify a list of patterns of acceptable filenames. Defaults to "*" if not set.

filename_pattern_reject = [string];
: This field is used to specify a list of patterns of unacceptable filenames.

Please Note: Aegis also consults the underlying file system, to determine its notion of maximum file size.

Where the file system's maximum file size is less than *maximum_filename_length*, the filesystem wins. This can happen, for example, when you are using the Linux UMSDOS file system, or when you have an NFS mounted an ancient V7 filesystem. Setting *maximum_filename_length* to 255 in these cases does not alter the fact that the underlying file systems limits are far smaller (12 and 14, respectively).

If your development directories (or your whole project) is on filesystems with filename limitations, or a portion of the heterogeneous builds take place in such an environment, it helps to tell Aegis what they are (using the project *config* file's fields) so that you don't run into the situation where the project builds on the more permissive environments, but fails with mysterious errors in the more limited environments.

If your development directories are routinely on a Linux UMSDOS filesystem, you would probably be better off setting *dos_filename_required = true*, and also changing the *development_directory_template field. Heterogenous development with various Windows environments may also require this.*

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

TEST CORRELATIONS

The ‘‘aegis -Test -SUGgest’’ command may be used to have aegis suggest suitable regression tests for your change, based on the source files in your change. This automatically focuses testing effort to relevant tests, reducing the number of regression tests necessary to be confident that you have not introduced a bug.

The test correlations are generated by the ‘‘aegis -Integrate_Pass’’ command, which associates each test in the change with each source file in the change. Thus, each source file accumulates a list of tests which have been associated with it in the past. This is not as exact as code coverage analysis, but is a reasonable approximation in practice.

The *aecp*(1) and *aenf*(1) commands are used to associate files with a change. While they do not actively perform the association, these are the files used by *aeip*(1) and *aet*(1) to determine which source files are associated with which tests.

Test Correlation Accuracy

Assuming that the testing correlations are accurate and that the tests are evenly distributed across the function space, there will be a less than *1/number* chance that a relevant test has not been run by the ‘‘aegis -Test -SUGgest *number*’’ command. A small amount of noise is added to the test weighting, so that unexpected things are sometimes tested, and the same tests are not run every time.

Test correlation accuracy can be improved by ensuring that:

- Each change should be strongly focused, with no gratuitous file inclusions. This avoids spurious correlations.
- Each item of new functionality should be added in an individual change, rather than several together. This strongly correlates tests with functionality.
- Each bug should be fixed in an individual change, rather than several together. This strongly correlates tests with functionality.
- Test correlations will be lost if files are moved. This is because correlations are by name.

The best way for tests to correlate accurately with source files is when a change contains a test and exactly those files relating to the functionality under test. Too many spurious files will weaken the usefulness of the testing correlations.

OPTIONS

The following options are understood

-Build
This option may be used to specify that the file is constructed during a build (often only an integrate build), so that history of it may be kept. This is useful for generating patch files, where a history of generated files is important. Files created in this way may not be copied into a change, though they may be deleted. Avoid using files of this type, if at all possible.

−BAse_RElative
This option may be used to cause relative filenames to be considered relative to the base of the source tree. See *aeuconf*(5) for the corresponding user preference.

−CUrrent_RElative
This option may be used to cause relative filenames to be considered relative to the current directory. This is usually the default. See *aeuconf*(5) for the corresponding user preference.

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Not_Logging
This option may be used to disable the automatic logging of output and errors to a file. This is often useful when several aegis commands are combined in a shell script.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single

project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aenf 'aegis -nf \!* -v'
sh$     aenf(){aegis -nf $* -v}
```

ERRORS

It is an error if the change is not in the *being developed* state.
It is an error if the change is not assigned to the current user.
It is an error if the file is already part of the change.
It is an error if the file is already part of the baseline.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aecp(1) copy files into a change

aedb(1) begin development of a change

aemv(1) rename a file as part of a change

aenfu(1) remove new files from a change

aent(1) add new tests to a change

aerm(1) add files to be deleted by a change

aepconf(5)
project configuration file format

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -New_File_Undo – remove new files from a change

SYNOPSIS

aegis -New_File_Undo *file-name*... [*option*...]
aegis -New_File_Undo -List [*option*...]
aegis -New_File_Undo -Help

DESCRIPTION

The *aegis -New_File_Undo* command is used to remove new files from a change (reverse the actions of the 'aegis -New_File' command). The file is removed from the list of files in the change.

The file is removed from the development directory unless the **-Keep** option is used. The **-Keep** option should be used with great care, as you can confuse tools such as *make*(1) by leaving these files in place.

You may specify a directory name to remove all new files in the named directory tree, other files in the tree will be ignored. It is an error if there are no relevant files.

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

Process Side Effects

This command will cancel any build or test registrations, because deleting a file logically invalidates them.

The difference file (`,D`) will also be removed, however any DMT derived files (e.g a `.o` file from a `.c` file) will not be removed. This is because aegis is decoupled from the DMT, and cannot know what these derived file may be called. You may need to delete derived files manually.

OPTIONS

The following options are understood:

–BAse_RElative
This option may be used to cause relative filenames to be considered relative to the base of the source tree. See *aeuconf*(5) for the corresponding user preference.

–CUrrent_RElative
This option may be used to cause relative filenames to be considered relative to the current directory. This is usually the default. See *aeuconf*(5) for the corresponding user preference.

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not

exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-Interactive
Specify that aegis should ask the user for confirmation before deleting each file. Answer the question *yes* to delete the file, or *no* to keep the file. You can also answer *all* to delete the file and all that follow, or *none* to keep the file and all that follow.

Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

If aegis is running in the background, the question will not be asked, and the files will be deleted.

-Keep
This option may be used to retain files and/or directories usually deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-No_Keep
This option may be used to ensure that the files and/or directories are deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see

aeuconf(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aenfu 'aegis -nfu \!$ -v'
sh$ aenfu(){aegis -nfu $* -v}

ERRORS

It is an error if the change is not in the *being developed* state.
It is an error if the change is not assigned to the current user.
It is an error if the file is not in the change.
It is an error if the file was not added to the change with the 'aegis -New_File' command.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aenf(1) add new files to a change

aepconf(5)
project configuration file format

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -New_Integrator – add new integrators to a project

SYNOPSIS

aegis -New_Integrator *user-name*... [*option*...]

aegis -New_Integrator -List [*option*...]

aegis -New_Integrator -Help

DESCRIPTION

The *aegis -New_Integrator* command is used to add new integrators to a project.

OPTIONS

The following options are understood:

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait
This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aeni 'aegis -ni \!* -v'
sh$	aeni(){aegis -ni $* -v}

ERRORS

It is an error if the current user is not an administrator of the project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeri(1) remove integrators from a project

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -New_PRoject – create a new project

SYNOPSIS

aegis -New_PRoject *project-name* [*option*...]
aegis -New_PRoject -List [*option*...]
aegis -New_PRoject -Help

DESCRIPTION

The *aegis -New_PRoject* command is used to create a new project. The project is created as an empty directory structure with no staff except the administrator, no changes, and branches to implement the version specified.

Directory

The project directory, under which the project baseline and history and state and change data are kept, will be created at this time. If the **-DIRectory** option is not given, the project directory will be created in the directory specified by the default_project_directory field of *aeuconf*(5), or if not set in current user's home directory; in either case with the same name as the project.

Staff

The project is created with the current user and group as the owning user and group. The current user is an administrator for the project. The project has no developers, reviewers, integrators or other administrators. The project's umask is derived from the current user's umask, but guaranteeing that group members will have access and that only the project owner will have write access.

Pointer

The project pointer will be added to the first element of the search path, or */usr/local/com/aegis* if no path is set. If this is inappropriate, use the **-LIBrary** option to explicitly set the desired location. See the **-LIBrary** option for more information.

Version

You may specify the project version in two ways:

1. The version number may be implicit in the project name, in which case the version numbers will be stripped off. For example, ‘‘aenpr example.1.2’’ will create a project called ‘‘example’’ with branch number 1 created, and sub-branch 2 of branch 1 created.
2. The version number may be stated explicitly, in which case it will be subdivided for branch numbers. For example, ‘‘aenpr example -version 1.2’’ will create a project called ‘‘example’’ with branch number 1 created, and sub-branch 2 of branch 1 created.

In each case, these branches may be named wherever a project name may be given, such as ‘‘-p example.1’’ and ‘‘-p example-1.2’’. The actual punctuation character is unimportant.

You may have any depth of version numbers you like. Both methods of specifying version numbers may be used, and they will be combined. If you want no version numbers at all, use -version with the empty string as an argument, as in ‘‘-version ""’’

If no version number is given, either explicitly or implicitly, version 1.0 is used.

Project Directory Location

Please Note: Aegis also consults the underlying file system, to determine its notion of maximum file size. Where the file system's maximum file size is less than *maximum_filename_length*, the filesystem wins. This can happen, for example, when you are using the Linux UMSDOS file system, or when you have an

NFS mounted an ancient V7 filesystem. Setting *maximum_filename_length* to 255 in these cases does not alter the fact that the underlying file systems limits are far smaller (12 and 14, respectively).

If your development directories (or your whole project) is on filesystems with filename limitations, or a portion of the heterogeneous builds take place in such an environment, it helps to tell Aegis what they are (using the project *config* file's fields) so that you don't run into the situation where the project builds on the more permissive environments, but fails with mysterious errors in the more limited environments.

If your development directories are routinely on a Linux UMSDOS filesystem, you would probably be better off setting *dos_filename_required = true*, and also changing the *development_directory_template field. Heterogenous development with various Windows environments may also require this.*

OPTIONS

The following options are understood:

-DIRectory *path*

This option may be used to specify which directory is to be used. It is an error if the current user does not have appropriate permissions to create the directory path given. This must be an absolute path.

Caution: If you are using an automounter do not use 'pwd' to make an absolute path, it usually gives the wrong answer.

-Edit

Edit the attributes with a text editor, this is usually more convenient than supplying a text file. The *VISUAL* and then *EDITOR* environment variables are consulted for the name of the editor to use; defaults to *vi*(1) if neither is set. Warning: tries to be well behaved when faced with errors, the temporary file is left in your home directory. This option may not be used in the background, or when the standard input is not a terminal.

-Edit_BackGround

Edit the attributes with a dumb text editor, this is most often desired when edit commands are being piped into the editor via the standard input. Only the *EDITOR* environment variable is consulted for the name of the editor to use; it is a fatal error if it is not set.

-File *filename*

Take the attributes from the specified file.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-Keep

This option may be used to re-attach a project detached using *aermpr −keep* and possibly moved by the system administrator.

-LIBrary *abspath*

This option may be used to specify a directory to be searched for global state files and user state files. (See *aegstate*(5) and *aeustate*(5) for more information.) Several library options may be present on the command line, and are search in the order given. Appended to this explicit search path are the directories specified by the *AEGIS_PATH* environment variable (colon separated), and finally, */usr/local/lib/aegis* is always searched. All paths specified, either on the command line or in the *AEGIS_PATH* environment variable, must be absolute.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

-VERSion *number*
This option may be used to specify the version number for the project. Version number are implemented as branches. Use the empty string as the argument if you want no version branches created.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aenpr 'aegis -npr \!* -v'

sh$ aenpr(){aegis -npr $* -v}

ERRORS

It is an error if the project name already exists.
It is an error if the project directory already exists.
It is an error if the current user does not have sufficient permissions to create the directory specified with the **-DIRectory** option.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aena(1) add a new administrator to a project

aenc(1) add a new change to a project

aend(1) add a new developer to a project

aenrls(1)
create a new project from an existing project

aenrv(1) add a new reviewer to a project

aermpr(1)
remove project

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -New_ReLeaSe – create a new project from an old-style project.

SYNOPSIS

aegis -New_ReLeaSe project-name [*new-project-name*][*option*...]
aegis -New_ReLeaSe -List [*option*...]
aegis -New_ReLeaSe -Help

DESCRIPTION

The *aegis -New_ReLeaSe* command is used to create a new project from an existing project. *It creates a new post-3.0 project from an old pre-3.0 project.*

This command was essential before the introduction of branches into the Aegis model. It is more useful to create a new release of a project by ending development on the branch of the previous release and starting development of a new branch numbered for the desired release.

Once you have a new-style project, use the *aenbr*(1) command to create new branches on this project. This provides more efficient release management, and allows historical versions to be reproduced more simply.

If no *new-project-name* is specified, it will be derived from the project given as follows: any minor version dot suffix will be removed from the name, then any major version dot suffix will be removed from the name. A major version dot suffix will be appended, and then a minor version dot suffix will be appended. As an example, "foo.1.0" would become "foo.1.1" assuming the default minor version increment, and "foo" would become "foo.1.1" assuming the same minor version increment.

The entire project baseline will be copied. The project state will be as if change 1 had already been integrated, naming every file (in the old project) as a new file. The history files will reflect this. No build will be necessary; it is assumed that the old baseline was built successfully. Change numbers will commence at 2, as will build numbers. Test numbers will commence where the old project left off (because all the earlier test numbers were used by the old project).

The default is for the minor version number to be incremented. If the major version number is incremented or set, the minor version number will be set to zero if it is not explicitly given.

The pointer to the new project will be added to the first element of the search path, or */usr/local/com/aegis* if none is set. If this is inappropriate, use the **-LIBrary** option to explicitly set the desired location. See the **-LIBrary** option for more information.

The project directory, under which the project baseline and history and state and change data are kept, will be created at this time. If the **-DIRectory** option is not given, the project directory will be created in the directory specified by the default_project_directory field of the project user's *aeuconf*(5), or if not set in project user's home directory; in either case with the same name as the project.

All staff will be copied from the old project to the new project without change, as will all of the project attributes.

THE BASELINE LOCK

The baseline lock is used to ensure that the baseline remains in a consistent state for the duration of commands which need to read the contents of files in the baseline.

The commands which require the baseline to be consistent (these include the *aeb*(1), *aecp*(1) and *aed*(1) commands) take a baseline *read* lock. This is a non-exclusive lock, so the concurrent development of changes is not hindered.

The command which modifies the baseline, *aeip*(1), takes a baseline *write* lock. This is an exclusive lock, forcing *aeip*(1) to block until there are no active baseline read locks.

It is possible that one of the above development commands will block until an in-progress *aegis -Integrate_PASS* completes. This is usually of short duration while the project history is updated. The delay is essential so that these commands receive a consistent view of the baseline. No other integration command will cause the above development commands to block.

When aegis' branch functionality is in use, a read (non-exclusive) lock is taken on the branch baseline and also each of the "parent" baselines. However, a baseline write (exclusive) lock is only taken on the branch baseline; the "parent" baselines are only read (non-exclusive) locked.

Project Directory Location

Please Note: Aegis also consults the underlying file system, to determine its notion of maximum file size. Where the file system's maximum file size is less than *maximum_filename_length*, the filesystem wins. This can happen, for example, when you are using the Linux UMSDOS file system, or when you have an NFS mounted an ancient V7 filesystem. Setting *maximum_filename_length* to 255 in these cases does not alter the fact that the underlying file systems limits are far smaller (12 and 14, respectively).

If your development directories (or your whole project) is on filesystems with filename limitations, or a portion of the heterogeneous builds take place in such an environment, it helps to tell Aegis what they are (using the project *config* file's fields) so that you don't run into the situation where the project builds on the more permissive environments, but fails with mysterious errors in the more limited environments.

If your development directories are routinely on a Linux UMSDOS filesystem, you would probably be better off setting *dos_filename_required = true*, and also changing the *development_directory_template field. Heterogenous development with various Windows environments may also require this.*

OPTIONS

The following options are understood:

-DIRectory *path*

This option may be used to specify which directory is to be used. It is an error if the current user does not have appropriate permissions to create the directory path given. This must be an absolute path.

Caution: If you are using an automounter do not use 'pwd' to make an absolute path, it usually gives the wrong answer.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-LIBrary *abspath*

This option may be used to specify a directory to be searched for global state files and user state files. (See *aegstate*(5) and *aeustate*(5) for more information.) Several library options may be present on the command line, and are search in the order given. Appended to this explicit search path are the directories specified by the *AEGIS_PATH* environment variable (colon separated), and finally, */usr/local/lib/aegis* is always searched. All paths specified, either on the command line or in the *AEGIS_PATH* environment variable, must be absolute.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Not_Logging
This option may be used to disable the automatic logging of output and errors to a file. This is often useful when several aegis commands are combined in a shell script.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

-VERSion *number*
This option may be used to specify the version number for the project. Version number are implemented as branches. Use the empty string as the argument if you want no version branches created.

−Wait
This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user’s *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user’s *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading ’-’. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aenrls ’aegis -nrls \!* -v’
sh$     aenrls(){aegis -nrls $* -v}
```

ERRORS

It is an error if the old project named does not exist.

It is an error if the old project named has not yet had any changes integrated.

It is an error if the old project named has any changes not in the *completed* state.

It is an error if the current user is not an administrator of the old project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aenpr(1)
: create a new project

aermpr(1)
: remove project

aeuconf(5)
: user configuration file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

Aegis User Guide
: The chapter on *Branching* has useful information about releases and branching.

NAME

aegis -New_ReViewer – add new reviewers to a project

SYNOPSIS

aegis -New_ReViewer *user-name*... [*option*...]

aegis -New_ReViewer -List [*option*...]

aegis -New_ReViewer -Help

DESCRIPTION

The *aegis -New_ReViewer* command is used to add new reviewers to a project.

OPTIONS

The following options are understood:

-Help
: This option may be used to obtain more information about how to use the *aegis* program.

-List
: This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
: This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
: This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
: This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait
: This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
: This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aenrv 'aegis -nrv \!* -v'

sh$ aenrv(){aegis -nrv $* -v}

ERRORS

It is an error if the current user is not an administrator of the project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aerrv(1) remove reviewers from a project

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au

/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -New_Test – add a new test to a change

SYNOPSIS

aegis -New_Test [*option*...][*filename*...]
aegis -New_Test -List [*option*...]
aegis -New_Test -Help

DESCRIPTION

The *aegis -New_Test* command is used to add a new test to a change. A new file is created in the development directory. The *config* file will be searched for a template for the new file. If a template is found, the new file will be initialized to the template, otherwise it will be created empty. See *aepconf*(5) for more information.

New tests default to ‘‘automatic’’ unless otherwise specified.

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

Test Filename Generation

You may choose your own filename for a test, by specifying it on the command line.

If no filename is specified on the command line, a test filename is automatically generated. This is controlled by the *new_test_filename* field of the project *config* file (see *aepconf*(5) for more information. All automatically generated test filenames within a project are numbered uniquely. The default pattern for new test filenames is "*test/XX/tXXXX[am].sh*", where *XX* is the first 2 digits of the test number, *XXXX* is the whole test number, and *[am]* is `a` for automatic tests and `m` for manual tests.

Modifying Tests

Tests may be modified in future by adding them to a change with the *aecp*(1) command. Tests are treated just like any other source file, and are subject to the same process.

File Name Limitations

There are a number of controls available to limit the form of project file names. All of these controls may be found in the project *config* file, see *aepconf*(5) for more information. The most significant are briefly described here:

maximum_filename_length = integer;

This field is used to limit the length of filenames. All new files may not have path components longer than this. Defaults to 14 if not set, for maximum portability.

posix_filename_charset = boolean;

This field may be used to limit the characters allowed in filenames to only those explicitly allowed by POSIX. Defaults to *false* if not set, meaning whatever your operating system will tolerate, except white space and high-bit-on characters.

dos_filename_required = boolean;

This field may be used to limit filenames so that they conform to the DOS 8+3 filename limits and to the DOS filename character set. Defaults to *false* if not set.

windows_filename_required = boolean;

This field may be used to limit filenames so that they conform to the Windows98 and WindowsNT filename limits and character set. Defaults to *false* if not set.

shell_safe_filenames = boolean;

This field may be used to limit filenames so that they do not contain shell special characters. Defaults to *true* if not set. If this field is set to *false*, you will need to use the *${quote}* substitution around filenames in commands, to ensure that filenames containing shell special characters do not have unintended side effects. Weird characters in filenames may also confuse your dependency maintenance tool.

filename_pattern_accept = [string];

This field is used to specify a list of patterns of acceptable filenames. Defaults to "*" if not set.

filename_pattern_reject = [string];

This field is used to specify a list of patterns of unacceptable filenames.

Please Note: Aegis also consults the underlying file system, to determine its notion of maximum file size. Where the file system's maximum file size is less than *maximum_filename_length*, the filesystem wins. This can happen, for example, when you are using the Linux UMSDOS file system, or when you have an NFS mounted an ancient V7 filesystem. Setting *maximum_filename_length* to 255 in these cases does not alter the fact that the underlying file systems limits are far smaller (12 and 14, respectively).

If your development directories (or your whole project) is on filesystems with filename limitations, or a portion of the heterogeneous builds take place in such an environment, it helps to tell Aegis what they are (using the project *config* file's fields) so that you don't run into the situation where the project builds on the more permissive environments, but fails with mysterious errors in the more limited environments.

If your development directories are routinely on a Linux UMSDOS filesystem, you would probably be better off setting *dos_filename_required = true*, and also changing the *development_directory_template field. Heterogenous development with various Windows environments may also require this.*

TEST PROCESS

Each change is required to be accompanied by tests, and those tests are required to be run against the built development directory, and they must pass. This ensures that new functionality is accompanied by tests to verify its correctness, and bug fixes are accompanied by tests which confirm that the bug has been fixed.

Regression Tests

Tests are treated as any other source file, and are maintained in the baseline and history with all other source files. The tests which must accompany every change accumulate in the project baseline, providing a definition of correct function for the baseline. These accumulated tests may be executed using an ‘‘aegis -REGression’’ command, to verify that the project will not ‘‘regress’’ as a result of a change.

Baseline Tests

Bug fixes are required to have their tests *fail* against the project baseline (in contrast to the development directory). This ensures that the test actually demonstrates the bug in the baseline, as well as demonstrating that it is fixed by the change. New functionality trivially fails against the baseline, and so aegis does not attempt to guess if a test is a bug fix test or new functionality test, it simply requires tests to fail against the baseline.

This requirement applies both to new tests being created by a change and also to tests which have been copied into a change for modification.

Reviewing Tests

Reviewers may be confident that aegis has enforced the test requirements; that a change must have tests, that the change must build, that the tests pass against the development directory, and that the tests fail against the baseline. These conditions are enforced by *aede*(1) and the change will not be advanced to the *being reviewed* state until these conditions are met. Reviewers should thus review tests for *completeness* of coverage of the code in the change, and insensitivity to changes in the execution environment (e.g. not date sensitive). Reviewers should also use ‘‘aegis -list change_details’’ to verify that a change does or does not have testing exemptions.

Exemptions

Various test exemptions may be granted by project administrators, see *aepa*(1) and *aepattr*(5) for more information. Copying tests into a change, or adding new tests to a change, may cancel those exemptions.

TEST CORRELATIONS

The ‘‘aegis -Test -SUGgest’’ command may be used to have aegis suggest suitable regression tests for your change, based on the source files in your change. This automatically focuses testing effort to relevant tests, reducing the number of regression tests necessary to be confident that you have not introduced a bug.

The test correlations are generated by the ‘‘aegis -Integrate_Pass’’ command, which associates each test in the change with each source file in the change. Thus, each source file accumulates a list of tests which have been associated with it in the past. This is not as exact as code coverage analysis, but is a reasonable approximation in practice.

The *aecp*(1) and *aenf*(1) commands are used to associate files with a change. While they do not actively perform the association, these are the files used by *aeip*(1) and *aet*(1) to determine which source files are associated with which tests.

Test Correlation Accuracy

Assuming that the testing correlations are accurate and that the tests are evenly distributed across the function space, there will be a less than *1/number* chance that a relevant test has not been run by the ‘‘aegis -Test -SUGgest *number*’’ command. A small amount of noise is added to the test weighting, so that unexpected things are sometimes tested, and the same tests are not run every time.

Test correlation accuracy can be improved by ensuring that:

- Each change should be strongly focused, with no gratuitous file inclusions. This avoids spurious correlations.
- Each item of new functionality should be added in an individual change, rather than several together. This strongly correlates tests with functionality.
- Each bug should be fixed in an individual change, rather than several together. This strongly correlates tests with functionality.

- Test correlations will be lost if files are moved. This is because correlations are by name.

The best way for tests to correlate accurately with source files is when a change contains a test and exactly those files relating to the functionality under test. Too many spurious files will weaken the usefulness of the testing correlations.

OPTIONS

The following options are understood;

-AUTOmatic
This option may be used to specify manual tests. Automatic tests require no human assistance.

−BAse_RElative
This option may be used to cause relative filenames to be considered relative to the base of the source tree. See *aeuconf*(5) for the corresponding user preference.

−CUrrent_RElative
This option may be used to cause relative filenames to be considered relative to the current directory. This is usually the default. See *aeuconf*(5) for the corresponding user preference.

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-MANual
This option may be used to specify manual tests. Manual tests require some human intervention, e.g.: confirmation of some screen behavior (X11, for instance), or some user action, "unplug ethernet cable now".

-Not_Logging
This option may be used to disable the automatic logging of output and errors to a file. This is often useful when several aegis commands are combined in a shell script.

−Output *filename*
This option may be used to specify a filename which is to be written with the automatically determined test file name. Useful for writing scripts.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aent 'aegis -nt \!* -v'

sh$ aent(){aegis -nt $* -v}

ERRORS

It is an error if the change is not in the *being developed* state.
It is an error if the change is not assigned to the current user.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aecp(1) copy an existing test into a change

aedb(1) begin development of a change

aentu(1) remove a new test from a change *aerm*(1) remove an existing test as part of a change

aet(1) run tests

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/˜millerp/

NAME

aegis -New_Test_Undo – remove new tests from a change

SYNOPSIS

aegis -New_Test_Undo *file-name*... [*option*...]
aegis -New_Test_Undo -List [*option*...]
aegis -New_Test_Undo -Help

DESCRIPTION

The *aegis -New_Test_Undo* command is used to remove new tests from a change (reverse the actions of the 'aegis -New_Test' command). The file is removed from the development directory.

You may specify a directory name to remove all new tests in the named directory tree, other files in the tree will be ignored. It is an error if there are no relevant files.

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

Process Side Effects

This command will cancel any build or test registrations, because deleting a file logically invalidates them.

OPTIONS

The following options are understood:

–BAse_RElative

This option may be used to cause relative filenames to be considered relative to the base of the source tree. See *aeuconf*(5) for the corresponding user preference.

–CUrrent_RElative

This option may be used to cause relative filenames to be considered relative to the current directory. This is usually the default. See *aeuconf*(5) for the corresponding user preference.

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-Interactive
Specify that aegis should ask the user for confirmation before deleting each file. Answer the question *yes* to delete the file, or *no* to keep the file. You can also answer *all* to delete the file and all that follow, or *none* to keep the file and all that follow.

Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

If aegis is running in the background, the question will not be asked, and the files will be deleted.

-Keep
This option may be used to retain files and/or directories usually deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-No_Keep
This option may be used to ensure that the files and/or directories are deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Not_Logging
This option may be used to disable the automatic logging of output and errors to a file. This is often useful when several aegis commands are combined in a shell script.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait
This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aentu 'aegis -ntu \!$ -v'

sh$ aentu(){aegis -ntu $* -v}

ERRORS

It is an error if the change is not in the *being developed* state.
It is an error if the change is not assigned to the current user.
It is an error if the file is not in the change.
It is an error if the file was not added to the change with the 'aegis -New_Test' command.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aent(5) add a new test to a change

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -Project_Attributes – modify the attributes of a project

SYNOPSIS

aegis -Project_Attributes *attr-file* [*option*...]

aegis -Project_Attributes -Edit [*option*...]

aegis -Project_Attributes -List [*option*...]

aegis -Project_Attributes -Help

DESCRIPTION

The *aegis -Project_Attributes* command is used to set, edit or list the attributes of a project.

The output of the **-List** variant is suitable for use as input at a later time.

See *aepattr*(5) for a description of the file format.

OPTIONS

The following options are understood:

-Edit

Edit the attributes with a text editor, this is usually more convenient than supplying a text file. The *VISUAL* and then *EDITOR* environment variables are consulted for the name of the editor to use; defaults to *vi*(1) if neither is set. Warning: tries to be well behaved when faced with errors, the temporary file is left in your home directory. This option may not be used in the background, or when the standard input is not a terminal.

-Edit_BackGround

Edit the attributes with a dumb text editor, this is most often desired when edit commands are being piped into the editor via the standard input. Only the *EDITOR* environment variable is consulted for the name of the editor to use; it is a fatal error if it is not set.

-File *filename*

Take the attributes from the specified file.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*

This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse

This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose

This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be

added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aepa 'aegis -pa \!* -v'
sh$     aepa(){aegis -pa $* -v}
```

ERRORS

It is an error if the current user is not an administrator of the specified project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeca(1) modify the attributes of a change

aepattr(5)
project attribute file format

aepstate(5)
project state file format

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -RePorT – report generator

SYNOPSIS

aegis -RePorT [*option*...] *report-name*
aegis -RePorT [*option*...] **-File** *filename*
aegis -RePorT -List
aegis -Help
aegis -VERSion

DESCRIPTION

The *aegis -RePorT* command is used to generate reports from aegis' database. Reports are specified in a C-like language described in the *aer*(5) manual entry.

For a list of the reports available on your system, use the '*aer -list*' command. These reports live in the */usr/local/share/aegis/report* directory, and it initially contains the reports distributed with aegis, however sites are free to add their own here.

OPTIONS

The following options are understood:

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Output *filename*
This option may be used to specify an output file for the report. The default is to use the standard output.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be

added.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aer 'aegis -rpt \!* -v'
sh$     aer(){aegis -rpt $* -v}
```

SEE ALSO

ael(1) list (possibly) interesting things

aer(5) report script language definition

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/˜millerp/

NAME

aegis -Remove_Administrator – remove administrators from a project

SYNOPSIS

aegis -Remove_Administrator *user-name* ... [*option*...]

aegis -Remove_Administrator -List [*option*...]

aegis -Remove_Administrator -Help

DESCRIPTION

The *aegis -Remove_Administrator* command is used to remove administrators from a project.

OPTIONS

The following options are understood:

-Help
: This option may be used to obtain more information about how to use the *aegis* program.

-List
: This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
: This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
: This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
: This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait
: This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
: This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aera 'aegis -ra \!* -v'
sh$	aera(){aegis -ra $* -v}

ERRORS

It is an error if the current user is not an administrator of the project.

It is an error if an attempt is made to remove the last administrator from the project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aena(1) add new administrators to a project

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -Remove_Developer – remove developers from a project

SYNOPSIS

aegis -Remove_Developer *user-name*... [*option*...]

aegis -Remove_Developer -List [*option*...]

aegis -Remove_Developer -Help

DESCRIPTION

The *aegis -Remove_Developer* command is used to remove developers from a project.

OPTIONS

The following options are understood:

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait
This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aerd 'aegis -rd \!* -v'
sh$	aerd(){aegis -rd $* -v}

ERRORS

It is an error if the current user is not an administrator of the project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aend(1) add a new developer to a project

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/˜millerp/

NAME

aerect – draw a rectangle

SYNOPSIS

aerect [*option*...] *width height*

aegis -Help

DESCRIPTION

The *aerect* command is used to draw rectangles for use with the intranet interface.

OPTIONS

The following options are understood:

–Bevel *size*

This option may be used to specify the bevel size. A size of 0 may be use to specify no bevel. Defaults to 3 if not specified.

–Color *red green blue*

This option may be used to specify the color of the rectangle. The components are specified in a range from 0 to 255. If not specified, the color will be based on the size of the rectangle.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

–Output *filename*

This option may be used to specify the output file. The output is sent to the standard output by default.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -Review_FAIL – fail a change review

SYNOPSIS

aegis -Review_FAIL -File *reason-file* [*option*...]
aegis -Review_FAIL -REAson *'reason-text'* [*option*...]
aegis -Review_FAIL -Edit [*option*...]
aegis -Review_FAIL -List [*option*...]
aegis -Review_FAIL -Help

DESCRIPTION

The *aegis -Review_FAIL* command is used to inform aegis that a change has failed review.

The change will be returned from the *being reviewed* state to the *being developed* state. The change will cease to be assigned to the current user, and will be reassigned to the originating developer.

The developer will be notified by mail. See the review_fail_notify_command in *aepattr*(5) for more information.

The *reason-file* will contain a description of why the change was failed. The file is in plain text. It is recommended that you only use newline to terminate paragraphs, (rather than to terminate lines) as with will result in better formatting in the various listings.

OPTIONS

The following options are understood:

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Edit
Edit the attributes with a text editor, this is usually more convenient than supplying a text file. The *VISUAL* and then *EDITOR* environment variables are consulted for the name of the editor to use; defaults to *vi*(1) if neither is set. Warning: tries to be well behaved when faced with errors, the temporary file is left in your home directory. This option may not be used in the background, or when the standard input is not a terminal.

-Edit_BackGround
Edit the attributes with a dumb text editor, this is most often desired when edit commands are being piped into the editor via the standard input. Only the *EDITOR* environment variable is consulted for the name of the editor to use; it is a fatal error if it is not set.

-File *filename*
Take the attributes from the specified file.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

–REAson *text*
This option may be used to provide the failure reason on the command line, rather than in a file. You will need to use quotes to insulate the spaces from the shell.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aerfail 'aegis -rfail \!* -v'
sh$     aerfail(){aegis -rfail $* -v}
```

ERRORS

It is an error if the change is not in the *being reviewed* state.
It is an error if the current user is not a reviewer for the project.
It is an error if the current user developed the change and the project is configured to disallow developers to review their own changes (default).

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aecd(1) change directory

aede(1) complete development of a change

aedeu(1)
recall a change for further development

aenrv(1) add a reviewer to a project

aerp(1) pass review of a change

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -Remove_Integrator – remove integrators from a project

SYNOPSIS

aegis -Remove_Integrator *user-name*... [*option*...]

aegis -Remove_Integrator -List [*option*...]

aegis -Remove_Integrator -Help

DESCRIPTION

The *aegis -Remove_Integrator* command is used to remove integrators from a project.

OPTIONS

The following options are understood:

-Help
: This option may be used to obtain more information about how to use the *aegis* program.

-List
: This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
: This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
: This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
: This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait
: This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
: This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aeri 'aegis -ri \!* -v'
sh$	aeri(){aegis -ri $* -v}

ERRORS

It is an error if the current user is not an administrator of the project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeni(1) add a new administrator to a project

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -ReMove_file – add files to be deleted to a change

SYNOPSIS

aegis -ReMove_file *file-name*... [*option*...]
aegis -ReMove_file -List [*option*...]
aegis -ReMove_file -Help

DESCRIPTION

The *aegis -ReMove_file* command is used to add files to be deleted to a change. The file will be added to the list of files in the change, and will be removed from the baseline at integration time.

This command may be used to remove tests, not just source files. Tests are treated just like any other source file, and are subject to the same process.

A file will be created in the development directory containing 1KB of random text. The random text is sufficiently revolting that most compilers will give error messages, should the file be referenced accidentally. This is often very helpful when removing include files.

You may specify a directory name to remove all files in the named directory tree. It is an error if there are no relevant files.

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

Process Side Effects

This command will cancel any build or test registrations, because adding a file logically invalidates them.

When the change files are listed (*aegis -List Change_Files -TERse*) the removed files will not appear in the terse listing. Similarly, when the project files are listed with an explicit change number (*aegis -List Project_Files -TERse -Change* N) none of the change's files, including the the removed files, will not appear in the terse listing. These two features are very helpful when calling aegis from within a DMT to generate the list of source files.

OPTIONS

The following options are understood:

–BAse_RElative
This option may be used to cause relative filenames to be considered relative to the base of the source tree. See *aeuconf*(5) for the corresponding user preference.

−CUrrent_RElative
This option may be used to cause relative filenames to be considered relative to the current directory. This is usually the default. See *aeuconf*(5) for the corresponding user preference.

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Not_Logging
This option may be used to disable the automatic logging of output and errors to a file. This is often useful when several aegis commands are combined in a shell script.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option.

The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aerm 'aegis -rm \!* -v'
sh$	aerm(){aegis -rm $* -v}

ERRORS

It is an error if the change is not in the *being developed* state.
It is an error if the change is not assigned to the current user.
It is an error if the file does not exist in the baseline.
It is an error if the file is already part of the change.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aecp(1) copy files into a change

aedb(1) begin development of a change

aemv(1) rename a file as part of a change

aenf(1) add files to be created to a change

aermu(1)
remove files to be deleted from a change

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -ReMove_PRoject − remove project

SYNOPSIS

aegis -ReMove_Project *project-name* [*option*...]

aegis -ReMove_Project -List [*option*...]

aegis -ReMove_Project -Help

DESCRIPTION

The *aegis -ReMove_PRoject* command is used to remove a project, either entirely, or just from aegis' supervision.

OPTIONS

The following options are understood:

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-Interactive

Specify that aegis should ask the user for confirmation before deleting each file. Answer the question *yes* to delete the file, or *no* to keep the file. You can also answer *all* to delete the file and all that follow, or *none* to keep the file and all that follow.

Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

If aegis is running in the background, the question will not be asked, and the files will be deleted.

-Keep

This option may be used to retain files and/or directories usually deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-No_Keep

This option may be used to ensure that the files and/or directories are deleted by the command. Defaults to the user's *delete_file_preference* if not specified, see *aeuconf*(5) for more information.

-LIBrary *abspath*

This option may be used to specify a directory to be searched for global state files and user state files. (See *aegstate*(5) and *aeustate*(5) for more information.) Several library options may be present on the command line, and are search in the order given. Appended to this explicit search path are the directories specified by the *AEGIS_PATH* environment variable (colon separated), and finally, */usr/local/lib/aegis* is always searched. All paths specified, either on the command line or in the *AEGIS_PATH* environment variable, must be absolute.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*

This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aermpr 'aegis -rmpr \!* -v'
sh$     aermpr(){aegis -rmpr $* -v}
```

ERRORS

It is an error if the project has any changes between the *being developed* and *being integrated* states, inclusive.
It is an error if the current user is not an administrator.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aenpr(1)
: create a new project

aenrls(1)
: create a new project from an existing project

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -ReMove_file_Undo – remove files to be deleted from a change

SYNOPSIS

aegis -ReMove_file_Undo *file-name*... [*option*...]
aegis -ReMove_file_Undo -List [*option*...]
aegis -ReMove_file_Undo -Help

DESCRIPTION

The *aegis -ReMove_file_Undo* command is used to remove files to be deleted from a change. The files is removed from the list of files in the change.

You may specify a directory name to delete from the change all files being removed in the named directory tree, other files in the tree will be ignored. It is an error if there are no relevant files.

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

Process Side Effects

This command will cancel any build or test registrations, because deleting a file logically invalidates them.

OPTIONS

The following options are understood:

–BAse_RElative
This option may be used to cause relative filenames to be considered relative to the base of the source tree. See *aeuconf*(5) for the corresponding user preference.

–CUrrent_RElative
This option may be used to cause relative filenames to be considered relative to the current directory. This is usually the default. See *aeuconf*(5) for the corresponding user preference.

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aermu 'aegis -rmu \!* -v'
sh$     aermu(){aegis -rmu $* -v}
```

ERRORS

It is an error if the change is not in the *being developed* state.
It is an error if the change is not assigned to the current user.
It is an error if the file is not in the change.
It is an error if the was not added to the change using the *aegis -ReMove_file* command.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aerm(1) add files to be deleted to a change

aeuconf(5)
user configuration file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/˜millerp/

NAME

aegis -Review_PASS – pass a change review

SYNOPSIS

aegis -Review_PASS [*option*...]

aegis -Review_PASS -List [*option*...]

aegis -Review_PASS -Help

DESCRIPTION

The *aegis -Review_PASS* command is used to notify aegis that a change has passed review.

The change will be advanced from the *being reviewed* state to the *awaiting integration* state.

OPTIONS

The following options are understood:

-Change *number*

This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help

This option may be used to obtain more information about how to use the *aegis* program.

-List

This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*

This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse

This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose

This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait

This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aerpass 'aegis -rpass \!* -v'
sh$     aerpass(){aegis -rpass $* -v}
```

ERRORS

It is an error if the change is not in the *being reviewed* state.
It is an error if the current user is not a reviewer of the project.
Its is an error if the current user developed the change and the project is configured to disallow developers to review their own changes (default).

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aecd(1) change directory

aede(1) complete development of a change

aedeu(1)
recall a change for further development

aeib(1) begin integrating a change

aenrv(1) add a reviewer to a project

aerf(1) fail review of a change

aerpu(1)
rescind a change review pass

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis -Review_Pass_Undo – rescind a change review pass

SYNOPSIS

aegis -Review_Pass_Undo [*option*...]

aegis -Review_Pass_Undo -List [*option*...]

aegis -Review_Pass_Undo -Help

DESCRIPTION

The *aegis -Review_Pass_Undo* command is used to notify aegis that a change review pass has been rescinded.

The change will be moved from the *awaiting integration* state to the *being reviewed* state.

OPTIONS

The following options are understood:

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait
This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see

aeuconf(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh% alias aerpu 'aegis -rp \!* -v'
sh$ aerpu(){aegis -rp $* -v}

ERRORS

It is an error if the change is not in the *awaiting integration* state.
It is an error if the current user is not the reviewer of the change.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aerp(1) pass review of a change

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aegis -Remove_ReViewer – remove reviewers from a project

SYNOPSIS

aegis -Remove_ReViewer *user-name...* [*option...*]

aegis -Remove_ReViewer -List [*option...*]

aegis -Remove_ReViewer -Help

DESCRIPTION

The *aegis -Remove_ReViewer* command is used to remove reviewers from a project.

OPTIONS

The following options are understood:

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

–Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

–No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

csh%	alias aerrv 'aegis -rrv \!* -v'
sh$	aerrv(){aegis -rrv $* -v}

ERRORS

It is an error if the current user is not an administrator of the project.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aenrv(1) add a new reviewer to a project

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/˜millerp/

NAME

aegis -Test – run tests

SYNOPSIS

aegis -Test [*file-name*...][*option*...]
aegis -Test -INDependent [*file-name*...][*option*...]
aegis -Test -List [*option*...]
aegis -Test -Help

DESCRIPTION

The *aegis -Test* command is used to run tests. If no files are named, all relevant tests are run. By default both automatic and manual tests are run.

Each architecture must be tested separately. This is because there may be subtle problems that are only revealed on some architectures. Some projects may also have different code for different architectures.

The status of the last test run is remembered so that tests are not run if there is no need. (This does not apply to *-REGression* tests, unfortunately.) Tests must be re-run if the test previously failed, if the test file has changed, if there has been a build, and for each architecture.

The command used to execute tests is defined by the *test_command* field in the project *config* file (see *aepconf*(5) for more information), this defaults to using the Bourne shell if not set. The current directory will be the top of the appropriate directory tree. If tests require temporary files, they should create them in */tmp*, as a test cannot expect to have write permission in the current directory.

You may name directories on the command line, and all relevant tests in that directory tree in the change will be run. It is an error if there are no relevant tests.

File Name Interpretation

The aegis program will attempt to determine the project file names from the file names given on the command line. All file names are stored within aegis projects as relative to the root of the baseline directory tree. The development directory and the integration directory are shadows of this baseline directory, and so these relative names apply here, too. Files named on the command line are first converted to absolute paths if necessary. They are then compared with the baseline path, the development directory path, and the integration directory path, to determine a baseline-relative name. It is an error if the file named is outside one of these directory trees.

The **-BAse_RElative** option may be used to cause relative filenames to be interpreted as relative to the baseline path; absolute filenames will still be compared with the various paths in order to determine a baseline-relative name.

The *relative_filename_preference* in the user configuration file may be used to modify this default behavior. See *aeuconf*(5) for more information.

TEST PROCESS

Each change is required to be accompanied by tests, and those tests are required to be run against the built development directory, and they must pass. This ensures that new functionality is accompanied by tests to verify its correctness, and bug fixes are accompanied by tests which confirm that the bug has been fixed.

Regression Tests

Tests are treated as any other source file, and are maintained in the baseline and history with all other source files. The tests which must accompany every change accumulate in the project baseline, providing a definition of correct function for the baseline. These accumulated tests may be executed using an ‘‘aegis

-REGression'' command, to verify that the project will not ''regress'' as a result of a change.

Baseline Tests

Bug fixes are required to have their tests *fail* against the project baseline (in contrast to the development directory). This ensures that the test actually demonstrates the bug in the baseline, as well as demonstrating that it is fixed by the change. New functionality trivially fails against the baseline, and so aegis does not attempt to guess if a test is a bug fix test or new functionality test, it simply requires tests to fail against the baseline.

This requirement applies both to new tests being created by a change and also to tests which have been copied into a change for modification.

Reviewing Tests

Reviewers may be confident that aegis has enforced the test requirements; that a change must have tests, that the change must build, that the tests pass against the development directory, and that the tests fail against the baseline. These conditions are enforced by *aede*(1) and the change will not be advanced to the *being reviewed* state until these conditions are met. Reviewers should thus review tests for *completeness* of coverage of the code in the change, and insensitivity to changes in the execution environment (e.g. not date sensitive). Reviewers should also use ''aegis -list change_details'' to verify that a change does or does not have testing exemptions.

Exemptions

Various test exemptions may be granted by project administrators, see *aepa*(1) and *aepattr*(5) for more information. Copying tests into a change, or adding new tests to a change, may cancel those exemptions.

TEST RESULT CODES

As each test is run (via the *test_command* field in the project *config* file), aegis determines whether the test succeeded or failed by looking at its exit status. This exit status is mostly as expected for UNIX commands.

Success

A test should exit 0 to indicate success, i.e. that the specific function under test worked as expected.

Failure

A test should exit 1 to indicate failure, i.e. that the specific function under test did not work as expected.

No Result

A test should exit 2 to indicate no result, i.e. that the specific function under test could not be exersize because something else went wrong. For example, running out of disk space when creating the test input files in the */tmp* directory.

Actually, any exit code other than 0 or 1 will be interpreted as ''no result''. However, always using 0, 1 or 2 means that if a new result code is required by a later release of aegis, your existing tests will continue to work.

If you want to use a more sophisticated test engine, rather than a simple shell script, but this test engine does not return result codes suitable for use with aegis, you could wrap it in a shell script which re-writes the exit status into the values aegis expects. You could also achieve the same results by writing a more complex *test_command* in the project *config* file.

TEST CORRELATIONS

The ‘‘aegis -Test -SUGgest’’ command may be used to have aegis suggest suitable regression tests for your change, based on the source files in your change. This automatically focuses testing effort to relevant tests, reducing the number of regression tests necessary to be confident that you have not introduced a bug.

The test correlations are generated by the ‘‘aegis -Integrate_Pass’’ command, which associates each test in the change with each source file in the change. Thus, each source file accumulates a list of tests which have been associated with it in the past. This is not as exact as code coverage analysis, but is a reasonable approximation in practice.

The *aecp*(1) and *aenf*(1) commands are used to associate files with a change. While they do not actively perform the association, these are the files used by *aeip*(1) and *aet*(1) to determine which source files are associated with which tests.

Test Correlation Accuracy

Assuming that the testing correlations are accurate and that the tests are evenly distributed across the function space, there will be a less than *1/number* chance that a relevant test has not been run by the ‘‘aegis -Test -SUGgest *number*’’ command. A small amount of noise is added to the test weighting, so that unexpected things are sometimes tested, and the same tests are not run every time.

Test correlation accuracy can be improved by ensuring that:

- Each change should be strongly focused, with no gratuitous file inclusions. This avoids spurious correlations.
- Each item of new functionality should be added in an individual change, rather than several together. This strongly correlates tests with functionality.
- Each bug should be fixed in an individual change, rather than several together. This strongly correlates tests with functionality.
- Test correlations will be lost if files are moved. This is because correlations are by name.

The best way for tests to correlate accurately with source files is when a change contains a test and exactly those files relating to the functionality under test. Too many spurious files will weaken the usefulness of the testing correlations.

OPTIONS

The following options are understood:

-AUTOmatic

This option may be used to specify manual tests. Automatic tests require no human assistance.

-BaseLine

This option may be used to specify that the project baseline is the subject of the command.

−BAse_RElative

This option may be used to cause relative filenames to be considered relative to the base of the source tree. See *aeuconf*(5) for the corresponding user preference.

−CUrrent_RElative

This option may be used to cause relative filenames to be considered relative to the current directory. This is usually the default. See *aeuconf*(5) for the corresponding user preference.

-Change *number*
This option may be used to specify a particular change within a project. When no **-Change** option is specified, the *AEGIS_CHANGE* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default change field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on one change within a project, that is the default change number. Otherwise, it is an error.

-Help
This option may be used to obtain more information about how to use the *aegis* program.

-INDependent
This option is used to specify that the test is to be run independent of any particular change. If no tests are named, all tests in the baseline will be run.

-List
This option may be used to obtain a list of suitable subjects for this command. The list may be more general than expected.

-MANual
This option may be used to specify manual tests. Manual tests require some human intervention, e.g.: confirmation of some screen behavior (X11, for instance), or some user action, "unplug ethernet cable now".

-PErsevere
This option may be used to specify that all tests should be run, even if some fail. Defaults to the user's *persevere_preference* if not specified, see *aeuconf*(5) for more information.

-No_PErsevere
This option may be used to specify that the test run should stop after the first failure. Defaults to the user's *persevere_preference* if not specified, see *aeuconf*(5) for more information.

-Project *name*
This option may be used to select the project of interest. When no **-Project** option is specified, the *AEGIS_PROJECT* environment variable is consulted. If that does not exist, the user's *$HOME/.aegisrc* file is examined for a default project field (see *aeuconf*(5) for more information). If that does not exist, when the user is only working on changes within a single project, the project name defaults to that project. Otherwise, it is an error.

-REGression
This option is used to specify that the regression test suite is to be run. The regression test suite consists of all tests in the baseline which do not appear in the change. It is an error if there are no regression tests. You may not name tests on the command line when using the -REGression option. You may name individual tests to be run on the command line, without using the -REGression option; if they are not part of the change, the tests of the same name in the baseline will be run.

-SUGgest [*number*]
The "*aegis -Integrate_Pass*" command collects test correlation statistics when changes are integrated. This option may be used to request that aegis suggest which tests should be run, using these testing correlations. If no number is specified, 10 tests will be suggested. This option implies the **-REGression** option.

-SUGgest_Noise *number*
This option may be used to control the amount of noise injected into the test selection performed by the **-SUGgest** option. The number is a percentage of noise to be injected. Defaults to 10 if not specified. The injection of noise ensures that a variety of tests are run on subsequent runs, and also some from left-field as a sanity check.

-FOrce This option may be used to specify that all tests should be run, even if the status of the last test run indicates that there is no need to run a specific test.

-TERse
This option may be used to cause listings to produce the bare minimum of information. It is usually useful for shell scripts.

-Verbose
This option may be used to cause aegis to produce more output. By default aegis only produces output on errors. When used with the **-List** option this option causes column headings to be added.

−Wait This option may be used to require Aegis commands to wait for access locks, if they cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

−No_Wait
This option may be used to require Aegis commands to emit a fatal error if access locks cannot be obtained immediately. Defaults to the user's *lock_wait_preference* if not specified, see *aeuconf*(5) for more information.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aet 'aegis -t \!* -v'
sh$     aet(){aegis -t $* -v}
```

ERRORS

It is an error if the change is not in one of the *being developed* or *being integrated* states.
It is an error if the change is not assigned to the current user.
It is an error if your have no relevant tests and no relevant exemption.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

SEE ALSO

aeb(1) build a change

aeca(1) modify the attributes of a change

aedb(1) begin development of a change

aeib(1) begin integration of a change

aent(1) add a new test to a change

aecp(1) copy an existing test into a change

aepconf(5)
project configuration file format

aeuconf(5)
user configuration file format

COPYRIGHT

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/˜millerp/

NAME

aegis -VERSion – give version information

SYNOPSIS

aegis -VERSion [*info-name*]

aegis -VERSion -Help

DESCRIPTION

The *aegis -VERSion* command is used to give version information and conditions of use.

There are a number of possible *info-name*s, as follow (abbreviations as for command line options):

Copyright
: The copyright notice for the aegis program. Version information will also be printed. This is the default.

Redistribution
: Print the conditions of use and redistribution.

Warranty
: Print the limited warranty.

OPTIONS

The following options are understood:

-Help
: This option may be used to obtain more information about how to use the *aegis* program.

All options may be abbreviated; the abbreviation is documented as the upper case letters, all lower case letters and underscores (_) are optional. You must use consecutive sequences of optional letters.

All options are case insensitive, you may type them in upper case or lower case or a combination of both, case is not important.

For example: the arguments "-project, "-PROJ" and "-p" are all interpreted to mean the **-Project** option. The argument "-prj" will not be understood, because consecutive optional characters were not supplied.

Options and other command line arguments may be mixed arbitrarily on the command line, after the function selectors.

The GNU long option names are understood. Since all option names for *aegis* are long, this means ignoring the extra leading '-'. The "**--***option***=***value*" convention is also understood.

RECOMMENDED ALIAS

The recommended alias for this command is

```
csh%    alias aev 'aegis -vers \!*'
sh$     aev(){aegis -vers $*}
```

ERRORS

It is an error if the *info-name* given is unknown.

EXIT STATUS

The *aegis* command will exit with a status of 1 on any error. The *aegis* command will only exit with a status of 0 if there are no errors.

ENVIRONMENT VARIABLES

See *aegis*(1) for a list of environment variables which may affect this command.

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

tkaegis − GUI interface for Aegis, using TCL/TK

SYNOPSIS

tkaegis

DESCRIPTION

The *tkaegis* command is used to provide a GUI interface to Aegis. Its use should be self-evident to anyone familiar with Aegis.

There are some areas where tkaegis is still missing functionality; these are primarily related to project and change attributes that are not yet included in the dialogs, and also issues such as configuring the change and history tools, the architectures, and so on.

INSTALLATION

First, you will need Tcl/Tk installed, and will need to modify the path in the first line of tkaegis to reflect the path of your Tk wish interpreter. Hopefully, the *./configure* script took care of this.

Next, you may need to modify some important variables that occur immediately below these comments, to specify the architecture, project base directory, and the editor you are using (if you leave that blank, tkaegis will try to determine the editor to use from the EDITOR environment variable; if that fails, it will fall back to emacs or vi).

NAVIGATION

When you run tkaegis, a window will appear with a menu at the top. The window is used to display the output of aegis commands and some other feedback. The menu will initially have only two items, Project and Help. At this stage the Help menu only has an About dialog box.

The Project menu will allow you to create new projects, select from your existing projects, clear the contents of the feedback window, or exit the program. If you create a new project, a dialog box will appear allowing you to enter the project name, directory, and initial branch number. When you press OK the project will be created and should then appear in the Project menu.

If you select an existing project, a new option will be added to the Project menu, allowing you to delete the project. A Branch menu will also appear. This is similar to the Project menu, but allows you to create, delete, or select project branches.

If you select a branch in the branch menu, a Role menu will appear. This will allow you to choose the role that you will be playing, namely one of administrator, developer, reviewer, or integrator. tkaegis uses your UNIX login name and the names of the roleplayers associated with the project and branch, to determine which of the roles it will allow you to choose. If you create a new project, only the administrator role will appear.

Selecting a role will put you in a ‘mode’, which will determine what other menus appear and what you can do next. Each mode will now be described in turn, by giving a brief description of the role-specific menu hierarchy. Following the name of each menu item is the corresponding aegis command, where applicable.

ADMINISTRATOR MODE

In this mode, you can modify the staff and roles associated with the branch, and create, remove, and change the attributes of change requests, and view all the change requests.

Admin:

Edit Branch Attributes (aepa) -
Change the attributes for the branch

Staff:

Administrators:

Add (aena) -
Add an administrator for the branch

View (ael a) -
View the administrators for the branch

Remove (aera) -
Remove an administrator for the branch

Developers:

Add (aend) -
Add a developer for the branch

View (ael d) -
View the developers for the branch

Remove (aerd) -
Remove a developer for the branch

Reviewers:

Add (aenrv) -
Add a reviewer for the branch

View (ael r) -
View the reviewers for the branch

Remove (aerrv) -
Remove a reviewer for the branch

Integrators:

Add (aeni) -
Add an integrator for the branch

View (ael i) -
View the integrators for the branch

Remove (aeri) -
Remove an integrator for the branch

Change:

Add New Change (aenc) -
Add a new change request

New Change Undo (aencu) -
Undo the addition of a change request

Edit Change Attributes (aeca) -
Modify the attributes of a change request

View Changes (ael c) -
View the set of changes

DEVELOPER MODE

This mode is used by developers. When entering this mode, the Develop menu will appear, but no others. A change must be selected after which the other menus will appear. If there is only one change awaiting development, this will be auto-selected.

Develop:

View Changes (ael c) -
View all the change requests

Begin Change (aedb) -
Start work on a new change

Continue Change -
Continue work on a change in development

View Differences (aediff) -
Show all the diffs for this change

Abort Change (aedbu) -
Abort working on the change

End Change (aede) -
(Attempt to) end working on the change

Resume Change (aedeu) -
Resume work on a change awaiting review

File:

Edit Files -
Allow files to be loaded into an editor

Add New File (aenf) -
Add a new file to the project

Discard New File (aenfu) -
Discard a newly added file

Remove Existing File (aerm) -
Discard a previously existing file

Restore Existing File (aermu) -
Undo discard of a previously existing file

Change Existing File (aecp) -
Allow an existing file to be edited

Undo Changes to Existing File (aecpu) -
Lose changes to an existing file

Build:

Build Project (aeb) -
Attempt to build the project

Test:

Add New Test Script (aent) -
Add a new test script to the project

Discard New Test Script (aentu) -
Remove a new test script

Run New Tests (aet) -
Run the new tests

Run Regression Tests (aet -reg) -
Run the old tests

Run Baseline Test (aet -bl) -
Run the baseline test

REVIEWER MODE

In this mode you are able to review changes.

Review:

View Changes (ael c) -
View all the changes

Begin Review (aerb) -
Start reviewing a change

Abort Review (aerbu) -
Abort reviewing a change

Pass (aerpass) -
Pass a change review

Fail (aerfail) -
Fail a change review

Undo Pass (aerpu) -
Undo a previously passed review

INTEGRATOR MODE

In this mode you can perform integration activities.

Integrate:

View Changes (ael c) -
View all the changes

Start Integration (aeib) -
Start integrating a change

Resume Integration -
Resume an integration in progress

Cancel Integration (aeibu) -
: Cancel an integration

View Differences (aediff) -
: Show the file differences for the change

Build (aeb) -
: Build the project

New Tests (aet) -
: Run the new tests

Baseline Test (aet -bl) -
: Run the baseline test

Regression Test (aet -reg) -
: Run the regression tests

Pass (aeipass) -
: Pass the integration

Fail (aeifail) -
: Fail the integration

COPYRIGHT

AUTHOR

Graham Wheeler <gram@cdsec.com>
Citadel Data Security

NAME

aecattr - aegis change attributes file

DESCRIPTION

A change attributes file is used to describe the modifiable portion of a change.

CONTENTS

A change attributes file contains the following fields:

description = string;
: This field contains a detailed description of the change.

brief_description = string;
: This field contains a brief description of the change.

cause = (...);
: This field describes the cause which motivated the change.

external_bug
: The change was created in response to a bug report from outside the development team. This repairs existing functionality.

external_enhancement
: The change was created in response to an enhancement request from outside the development team. This adds new functionality.

external_improvement
: The change was created in response to an improvement request from outside the development team. This improves existing functionality.

internal_bug
: The change was created in response to a bug report from inside the development team. This repairs existing functionality.

internal_enhancement
: The change was created in response to an enhancement request from inside the development team. This adds new functionality.

internal_improvement
: The change was created in response to an improvement request from inside the development team. This improves existing functionality.

chain
: This cause is where you have a fix to fix a fix; tracking these is an interesting quality metric.

test_exempt = boolean;
: This field is true if it is not necessary to test the change. It is, in general, desirable to test all changes, whether new functionality or a bug fix. This is, however, a project management issue.

test_baseline_exempt = boolean;
: This field is true if it is not necessary to test the change against the baseline before it is changed. The test of the baseline is required to fail; this is to establish that the test has isolated the bug, and that the change has fixed that isolated bug.

regression_test_exempt = boolean;
This field is true if it is not necessary to perform a full regression test on the change. If absent, defaults to true for all causes except improvements.

architecture = [string];
This field is a list of names of system and machine architectures on which the change must successfully build and test.

copyright_years = [integer];
This field details the years in which the change was worked on. This field is present in trunk, branch and leaf nodes.

As a change is edited, years in which the chnage was worked on accumulate in this field automatically. Branches accumulate years as integrations occur. You may need to manually edit this, though it should be rare.

version_previous = string;
This field records the "previous" version, mostly to simplify patch generation. It is only meaningful for trunks and branches. It is set automatically when a branch is started or integrated.

SEE ALSO

aenc(1) create a new change

aeca(1) modify the attributes of a change

aegis(5) aegis file format syntax

aecstate(5)
change state file format

aepattr(5)
project attributes file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/˜millerp/

NAME

aecstate – aegis change state file

SYNOPSIS

project/`info/change/`*[0-9]*/*[0-9][0-9][0-9]*

DESCRIPTION

A change state file is used to store information about a change. These files are created and maintained by aegis. These files should not be edited by humans. These files is owned by the project owner and group.

The change number is at least 3 digits, zero padded if necessary. (More digits will be used if a project has a thousand or more changes in any one release, although this is rare.) The files are spread across a directory tree, 100 per subdirectory, to improve the directory search times, and to avoid various systems' directory length limitations.

CONTENTS

description = string;

This field contains a detailed description of the change.

brief_description = string;

This field contains a brief description of the change.

cause = (...);

This field describes the cause which motivated the change.

external_bug

The change was created in response to a bug report from outside the development team. This repairs existing functionality.

external_enhancement

The change was created in response to an enhancement request from outside the development team. This adds new functionality.

external_improvement

The change was created in response to an improvement request from outside the development team. This improves existing functionality.

internal_bug

The change was created in response to a bug report from inside the development team. This repairs existing functionality.

internal_enhancement

The change was created in response to an enhancement request from inside the development team. This adds new functionality.

internal_improvement

The change was created in response to an improvement request from inside the development team. This improves existing functionality.

chain

This cause is where you have a fix to fix a fix; tracking these is an interesting quality metric.

test_exempt = boolean;
This field is true if it is not necessary to test the change. It is, in general, desirable to test all changes, whether new functionality or a bug fix. This is, however, a project management issue.

test_baseline_exempt = boolean;
This field is true if it is not necessary to test the change against the baseline before it is changed. The test of the baseline is required to fail; this is to establish that the test has isolated the bug, and that the change has fixed that isolated bug.

regression_test_exempt = boolean;
This field is true if it is not necessary to perform a full regression test on the change. If absent, defaults to true for all causes except improvements.

architecture = [string];
This field is a list of names of system and machine architectures on which the change must successfully build and test.

copyright_years = [integer];
This field details the years in which the change was worked on. This field is present in trunk, branch and leaf nodes.

As a change is edited, years in which the chnage was worked on accumulate in this field automatically. Branches accumulate years as integrations occur. You may need to manually edit this, though it should be rare.

version_previous = string;
This field records the "previous" version, mostly to simplify patch generation. It is only meaningful for trunks and branches. It is set automatically when a branch is started or integrated.

state = (...);
This field is used to describe what state the change is in. The state determines what operations may be performed on the change.

awaiting_development
The change has been created, but has yet to be worked on.

being_developed
The change is being developed.

being_reviewed
The change has been developed, and is being reviewed. (There is no 'awaiting_review' state.)

awaiting_integration
The change has passed review, and is queued ready for integration.

being_integrated
The change is being integrated.

completed
The change has been completed and is now part of the baseline. Changes in this state can not be reversed.

delta_number = integer;

This field records the delta number for this change. It is only present if the change is in one of the *being_integrated* or *completed* states.

project_file_command_sync = integer;

This field records the last change integrated into the project. If it disagrees with the project, a 'project_file_command' (from pconf) needs to be executed at the next build.

test_time = time;

This field records the time the last successful *aegis -Test* command was run for all architectures. It is only present in the *being_developed* and *being_integrated* states.

test_baseline_time = time;

This field records the time the last successful *aegis -Test -BaseLine* command was run for all architectures. It is only present in the *being_developed* and *being_integrated* states.

regression_test_time = time;

This field records the time the last successful *aegis -Test -REGression* command was run for all architectures. It is only present in the *being_developed* and *being_integrated* states.

build_time = time;

This field records the last time the last successful *aegis -Build* command was run for all architectures. It is only present in the *being_developed* and *being_integrated* states.

architecture_times = [{ ... }];

This field records the time of various operations for each variant named in the *architecture* field. It is only present in the *being_developed* and *being_integrated* states.

variant = string;

This field is one of the patterns named in the *architecture* field.

node = string;

This field is the computer on which the command was run which last changed this structure.

test_time = time;

This field records the last time the last successful *aegis -Test* command was run for this specific pattern instance.

test_baseline_time = time;

This field records the last time the last successful *aegis -Test -BaseLine* command was run for this specific pattern instance.

regression_test_time = time;

This field records the last time the last successful *aegis -Test -REGression* command was run for this specific pattern instance.

build_time = time;

This field records the last time the last successful *aegis -Build* command was run for this specific pattern instance.

development_directory = string;

This field is the absolute path of the change's development directory. It is only present of the change is in a state between *being_developed* and *being_integrated* inclusive.

integration_directory = string;

This field is the absolute path of the change's integration directory. It is only present of the change is in the *being_integrated* state.

history = [{ ... }, ...];

This field records the history of the change, in the form of state transitions. The history records have the form

when = time;

This field records the time the state transition occurred.

what = (...);

This field records what happened. Valid value names echo the various aegis functions.

who = string;

This field records the user name of the user who caused the state transition.

why = string;

This field is optional. It is a comment of some sort. In the cases of *review_fail* and *integrate_fail*, this field will contain why the change failed.

branch = { ... };

This field is only present for branches (long transactions).

umask = integer;

File permission mode mask. See *umask*(2) for more information. This value will always be OR'ed with 022, because *aegis* is paranoid.

developer_may_review = boolean;

If this field is true, then a developer may review her own change. This is probably only a good idea for projects of less than 3 people. The idea is for as many people as possible to critically examine a change.

developer_may_integrate = boolean;

If this field is true, then a developer may integrate her own change. This is probably only a good idea for projects of less than 3 people. The idea is for as many people as possible to critically examine a change.

reviewer_may_integrate = boolean;

If this field is true, then a reviewer may integrate a change she reviewed. This is probably only a good idea for projects of less than 3 people. The idea is for as many people as possible to critically examine a change.

developers_may_create_changes = boolean;

This field is true if developers may created changes, in addition to administrators. This tends to be a very useful thing, since developers find most of the bugs.

forced_develop_begin_notify_command = string;

This command is used to notify a developer that a change requires developing; it is issued when a project administrator uses an *aedb -User* command to force development of a change by a specific user. All of the substitutions described in *aesub*(5) are available. This field is optional.

develop_end_notify_command = string;

This command is used to notify that a change is ready for review. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

develop_end_undo_notify_command = string;

This command is used to notify that a change had been withdrawn from review for further development. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

review_pass_notify_command = string;

This command is used to notify that a review has passed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

review_pass_undo_notify_command = string;

This command is used to notify that a review has passed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. Defaults to the same action as the *develop_end_notify_command* field. All of the substitutions described by *aesub*(5) are available.

review_fail_notify_command = string;

This command is used to notify that a review has failed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

integrate_pass_notify_command = string;

This command is used to notify that an integration has passed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

integrate_fail_notify_command = string;

This command is used to notify that an integration has failed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

default_test_exemption = boolean;
: This field contains what to do when a change is created with no test exemption specified.

history = [{ ... }];
: This field contains a history of integrations for the project. Updated by each successful 'aegis -Integrate_Pass' command.

  delta_number = integer;
  : The delta number of the integration.

  change_number = integer;
  : The number of the change which was integrated.

  name = [string];
  : The names by which this delta is known.

change = [integer];
: The list of changes which have been created on this branch to date.

sub_branch = [integer];
: The list of branches which have been created on this branch to date. This will be a subset of the above (possibly empty, possibly complete, never larger).

administrator = [string];
: The list of administrators of the branch.

developer = [string];
: The list of developers of the branch.

reviewer = [string];
: The list of reviewers of the branch.

integrator = [string];
: The list of integrators of the branch.

currently_integrating_change = integer;
: The change currently being integrated. Only one change (within a branch) may be integrated at a time. Only set when an integration is in progress.

default_development_directory = string;
: The pathname of where to place new development directories. The pathname must be absolute. This field is only consulted if the field of the same name in the user configuration file is not set.

minimum_change_number = integer;
: The minimum change number for *aenc(1),* if no change number is specified. This allows the low-numbered change numbers to be used for branches later in the project. Defaults to 10 if not set, may not be less than 1.

reuse_change_numbers = boolean;
: This controls whether the automatically selected *aenc*(1) change numbers ‘‘fill in’’ any gaps. Defaults to true if not set.

minimum_branch_number = integer;

The minimum branch number for *aenbr(1),* if no branch number is specified. Defaults to 1 if not set.

skip_unlucky = boolean;

This field may be set to true if you want to skip various unlucky numbers for changes, branches and tests. Various traditions are avoided, both Eastern and Western. Defaults to false if not set.

compress_database = boolean;

This field may be set to true if you want to compress the database on writing. (It is always uncompress on reading if necessary.) Defaults to false if not set.

Unless you have an exceptionally large project, coupled with fast CPUs and high network latency, there is probably very little benefit in using this feature. (The database is usually less than 5% of the size of the repository.) On slow networks, however, this can improve the preformance of file-related commands.

Obsolete Fields

The following fields are only present is old projects. They will be moved to an appropritate file state when the change is next modified. See *aefstate*(5) for more information.

src = [{ ... }, ...];

This field is a list of all the files in the change. The records have the form

file_name = string;

This file names the file. The name is relative to the root of the baseline directory tree.

action = (create, modify, remove);

This field describes what is being done with the file.

edit_number = string;

This field records the edit number of the file when it was added to the change (or updated using the *aegis -DIFFerence* command). This field is not present for new files.

usage = (source, test, manual_test);

This field describes what function the file serves.

diff_time = time;

This field records the last time modified of the change file when the last *aegis -DIFFerence* command was run. It is only present between the *being_developed* and *being_integrated* states, inclusive. It is not present for files which are being deleted. This field is used to determine if a difference has been done, and if the file has been tampered with before state transitions.

diff_file_time = time;

This field records the last time modified of the difference file when the last *aegis -DIFFerence* command was run. It is only present between the *being_developed* and *being_integrated* states, inclusive. This field is used to determine if a difference has been done, and if the difference file has been tampered with before state transitions.

move = string;

To change the name of a file, a combination of deleting the old name and creating the new name is used. With deleted files, this field is used to say where it went. With new files, this field is used to say where it came from.

SEE ALSO

aenc(1) create a new change

aegis(5) aegis file format syntax

aecattr(5)
change attributes file format

aefstate(5)
file state file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aedir - aegis directory structures

DESCRIPTION

The project directory structure is dictated by *aegis* at the top level, but is completely under the project's control from various points below the top level.

The project directory has the following contents

```
project/
        baseline/
                config
                ...project specific...
                test/
                        [0-9][0-9]/
                                t[0-9][0-9][0-9][0-9]a.sh
                                t[0-9][0-9][0-9][0-9]m.sh
        history/
                ...echo of baseline...
        delta.[0-9][0-9][0-9]/
                ...echo of baseline...
        info/
                state
                change/
                        [0-9]/
                                [0-9][0-9][0-9]
```

The directory is structured in this way so that it is possible to pick an entire project up off the disk, and be confident that you got it all.

The location of the root of this tree is configurable, and may even be changed during the life of a project.

The contents of the *baseline* subdirectory, other than those given, are defined by the project, and not dictated by aegis.

The contents of the *delta.NNN* directory, when it exists, are an image of the *baseline* directory. It is frequently linked with the baseline, rather than a copy of it; see the *link_integration_directory* field description in *aepconf*(5) for more information.

The contents of the *history* contains the edit histories of the *baseline* directory, and is in all other ways an image of it. Note that *baseline* always contains the latest source; the *history* directory is just history. The actual files in the history directory tree will not always have names the same as those in the baseline; compare the methods used by SCCS and RCS.

The contents of the *baseline/test* directory are the tests which are created by changes. Test histories are also stored in the *history* subdirectory. Tests are treated as project source.

The edit histories are separated out to simplify the task of taking a "snapshot" of the source of a project, without airing all the dirty laundry.

The *baseline* directory always contains the latest source, and so the *history* directory need not be readily accessible, because the build mechanism (something like *make*(1), but preferably better) does not need to know anything about it. Similarly for tests.

The *baseline/config* file is used to tell aegis everything else it needs to know about a project. See *aepconf*(5) for more information. This file is a source file of the project, and is treated in the same way as all source files.

SEE ALSO

aenc(1) create a new change

aenpr(1)
create a new project

aegis(5) aegis file format syntax

aepconf(5)
project configuration file format

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aefstate – aegis file state file

SYNOPSIS

project/`info/change/`*[0-9]*/*[0-9][0-9][0-9]*.fs

DESCRIPTION

A file state file is used to store information about the files in a transaction. These files are created and maintained by aegis. These files should not be edited by humans. These files is owned by the project owner and group.

CONTENTS

src = [{ ... }, ...];

This field is a list of all the files in the change. The records have the form

file_name = string;

This file names the file. The name is relative to the root of the baseline directory tree.

action = (create, modify, remove);

This field describes what is being done with the file.

edit_number = string;

For a project or an active branch, this field records the head revision of the file. For a completed change or branch, this field records the revision number after integrate pass.

edit_number_origin = string;

This field records the edit number of the file when it was added to the change or branch. In changes, this field is not present for new files. (A change file is out of date if it's edit number_origin field does not equal the edit_number field in the project.)

edit_number_origin_new = string;

This field records the edit number of the file to replace the edit_number_ofigin field in the branch at integrate pass time. This is used to perform cross branch merging. This field cleared at integrate pass time.

usage = (source, test, manual_test);

This field describes what function the file serves.

file_time = fingerprint;

This field records the last time modified of the source file. It is only present between the *being_developed* and *being_integrated* states, inclusive (for both changes and branches). It is not present for files which are being deleted. This field is used to determine if a difference has been done, or a test has been done if the source file is a test, and if the file has been tampered with before state transitions.

The fingerprint consists of the following fields:

youngest = time;

The youngest time see for this file with this fingerprint.

oldest = time;

The oldest time see for this file with this fingerprint.

crypto = string;

This field records a cryptographically strong fingerprint for the file. There is no known method of constructing a file to match a given fingerprint, and there is less than 1 in 2**200 chance that two files will have the same fingerprint. Thus if the fingerprint is the same, the file can reliably assumed to be the same.

diff_file_fp = fingerprint;

This field records the last time modified of the difference file when the last *aegis -DIFFerence* command was run. It is only present between the *being_developed* and *being_integrated* states, inclusive (for both changes and branches). This field is used to determine if a difference has been done, and if the difference file has been tampered with before state transitions.

idiff_file_fp = fingerprint;

This field records the last time modified of the integration difference file when the last *aegis -DIFFerence* command was run. It is only present in the *being_integrated* state. This field is used to determine if a difference has been done.

architecture_times = [{ ... }];

This field records the time of various operations for each variant named in the *architecture* field. It is only present in the *being_developed* and *being_integrated* states. This field is used to determine if a test has been done, and thus optimize test runs.

variant = string;

This field is one of the patterns named in the *architecture* field.

test_time = time;

This field records the last time the last successful *aegis -Test* command was run for this specific pattern instance.

test_baseline_time = time;

This field records the last time the last successful *aegis -Test -BaseLine* command was run for this specific pattern instance.

move = string;

To change the name of a file, a combination of deleting the old name and creating the new name is used. With deleted files, this field is used to say where it went. With new files, this field is used to say where it came from.

locked_by = integer;

The change which locked this file.
Caveat: this field is redundant, you can figure it out by scanning all of he change files.
Having it here is very convenient, even though it means multiple updates.

about_to_be_created_by = integer;

The change which is about to create this file for the first time. Same caveat as above.

about_to_be_copied_by = integer;

For each change file that is acting on a project file from a deeper baseline than the immediate parent project's baseline, the file needs to be added to the immediate parent project. Note that this field says that this file record is a placemarker, so that it can be deleted again should the change not be integrated for some reason.

deleted_by = integer;

The change which last deleted this file. We never throw them away, because (a) it may be created again, and more important (b) we need it to recreate earlier deltas.

test = [string];

This field is used to remember test correlations for source files. This is used by *aet*(1) to suggest suitable tests.

metrics = [{ ... }];

This field is used to describe various file metrics. It is committed during *aeip*(1), when the file is added to the history. The name must be given, and exactly one value.

name = string;

This is the name of the metric. This field must be set.

value = real;

This is the value of the metric. This field must be set. (If you have an integer-valued metric, just use integers, Aegis will cope. If you have a string-valued metric, assign integers to the enumerands.)

SEE ALSO

aegis(5) aegis file format syntax

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegis - file format

DESCRIPTION

The files used by the *aegis* program all have the same format. Some of the files used by aegis are created and maintained by humans, and some are created an maintained by aegis itself. The various manual entries say which is which.

LEXICAL CONSIDERATIONS

Names are any C identifier. Comments are C-style comments. Numbers are decimal, octal or hexadecimal, as for C constants. Strings are C-style strings. Whitespace (spaces, tabs and newlines) are ignored except in strings or as they serve to separate tokens.

GRAMMAR

The format of all aegis files is described by a *yacc (1)* grammar.

```
%%
file
        : field_list
        ;
```

A file contains a field list.

```
field_list
        : /* empty */
        | field_list field
        ;
```

A field list is zero or more fields.

```
field
        : NAME '=' value ';'
        ;
```

A field is set by giving a name and a value.

```
value
        : NAME
        | INTEGER
        | STRING
        | structure
        | list
        ;
```

A value may be a member of an enumeration (NAME), or an integer constant, or a literal string. More complex values may be constructed from these simple values.

```
structure
        : '{' field_list '}'
        ;
```

A structure is a grouped list of fields.

```
list
        : '[' list_body ']'
        ;
list_body
        : /* empty */
        | value_list
        | value_list ','
```

```
                ;
        value_list
                : value
                | value_list ',' value
```

A list is a sequential list of values separated by commas. It may be empty, or it may have a trailing comma.

SEMANTICS

The types of the values must match those in the definition of the file. See the relevant man pages for more information.

Files which are rewritten by **aegis** will lose any comments placed in them. When time fields are emitted by **aegis** they are usually followed by a human readable date in a comment.

SEE ALSO

aegis(1) a project change supervisor

aecattr(5)
change attribute file format

aecstate(5)
change state file format

aedir(5) directory structures

aegstate(5)
aegis state file

aepattr(5)
project attribute file format

aepconf(5)
project configuration file format

aepstate(5)
project state file format

aesub(5)
available command substitutions

aeuconf(5)
user configuration file format

aeustate(5)
user state file format

COPYRIGHT

aegis version 3.15

The aegis program comes with ABSOLUTELY NO WARRANTY; for details use the '*aegis -VERSion License*' command. This is free software and you are welcome to redistribute it under certain conditions; for details use the '*aegis -VERSion License*' command.

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aegstate – aegis state file

SYNOPSIS

`/usr/local/com/aegis/state`

DESCRIPTION

The aegis state file is used to store the pointers to project directories.

CONTENTS

where = [{ ... }];

This field is a table relating project name to project directory. The structure is as follows:

project_name = string;

The name of a project.

directory = string;

Absolute path of the project's directory.

SEE ALSO

aegis(5) aegis file format syntax

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aemetrics − metrics values file format

SYNOPSIS

filename`,S`

DESCRIPTION

Metrics files are created at integration build time, and recorded into the file attributes at integration pass time. This allows trend analysis and other statistics to be calculated.

CONTENTS

metrics = [{ ... }];

This field is used to describe various file metrics. It is committed during *aeip*(1), when the file is added to the history. The name must be given, and exactly one value.

name = string;

This is the name of the metric. This field must be set.

value = real;

This is the value of the metric. This field must be set. (If you have an integer-valued metric, just use integers, Aegis will cope. If you have a string-valued metric, assign integers to the enumerands.)

SEE ALSO

aegis(5) aegis file format syntax

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aepattr - project attribute file

DESCRIPTION

The project attribute file is used to store modifiable information about a project.

CONTENTS

description = string;

This field contains a description of the project. Large amounts of prose are not required; a single line is sufficient.

developer_may_review = boolean;

If this field is true, then a developer may review her own change. This is probably only a good idea for projects of less than 3 people. The idea is for as many people as possible to critically examine a change.

developer_may_integrate = boolean;

If this field is true, then a developer may integrate her own change. This is probably only a good idea for projects of less than 3 people. The idea is for as many people as possible to critically examine a change.

reviewer_may_integrate = boolean;

If this field is true, then a reviewer may integrate a change she reviewed. This is probably only a good idea for projects of less than 3 people. The idea is for as many people as possible to critically examine a change.

developers_may_create_changes = boolean;

This field is true if developers may created changes, in addition to administrators. This tends to be a very useful thing, since developers find most of the bugs.

forced_develop_begin_notify_command = string;

This command is used to notify a developer that a change requires developing; it is issued when a project administrator uses an *aedb -User* command to force development of a change by a specific user. All of the substitutions described in *aesub*(5) are available. This field is optional.

develop_end_notify_command = string;

This command is used to notify that a change is ready for review. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

develop_end_undo_notify_command = string;

This command is used to notify that a change had been withdrawn from review for further development. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

review_pass_notify_command = string;

This command is used to notify that a review has passed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

review_pass_undo_notify_command = string;
This command is used to notify that a review has passed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. Defaults to the same action as the *develop_end_notify_command* field. All of the substitutions described by *aesub*(5) are available.

review_fail_notify_command = string;
This command is used to notify that a review has failed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

integrate_pass_notify_command = string;
This command is used to notify that an integration has passed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

integrate_fail_notify_command = string;
This command is used to notify that an integration has failed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

default_development_directory = string;
The pathname of where to place new development directories. The pathname must be absolute. This field is only consulted if the field of the same name in the user configuration file is not set.

umask = integer;
File permission mode mask. See *umask*(2) for more information. This value will always be OR'ed with 022, because *aegis* is paranoid.

default_test_exemption = boolean;
This field contains what to do when a change is created with no test exemption specified.

minimum_change_number = integer;
The minimum change number for *aenc(1),* if no change number is specified. This allows the low-numbered change numbers to be used for branches later in the project.

reuse_change_numbers = boolean;
This controls whether the automatically selected *aenc*(1) change numbers "fill in" any gaps. Defaults to true if not set.

minimum_branch_number = integer;
The minimum branch number for *aenbr(1),* if no branch number is specified. Defaults to 1 if not set.

skip_unlucky = boolean;
This field may be set to true if you want to skip various unlucky numbers for changes, branches and tests. Various traditions are avoided, both Eastern and Western. Defaults to false if not set.

compress_database = boolean;

This field may be set to true if you want to compress the database on writing. (It is always uncompressed on reading if necessary.) Defaults to false if not set.

Unless you have an exceptionally large project, coupled with fast CPUs and high network latency, there is probably very little benefit in using this feature. (The database is usually less than 5% of the size of the repository.) On slow networks, however, this can improve the preformance of file-related commands.

SEE ALSO

aepa(1) modify the attributes of a project

aegis(5) aegis file format syntax

aecattr(5)
change attributes file format

aecstate(5)
change state file format, particularly as branches are used to remember most project state

aepstate(5)
project state file format

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aepconf - aegis project configuration file

SYNOPSIS

project/`baseline/config`

DESCRIPTION

A project configuration file is used to store information about a project. This file is under source control, and is one of the project's source files. Developers may thus modify this file as part of a change.

This file contains a number of commands to be executed by Aegis. There are times when the substitutions in these commands may contain shell special characters, which would change the meaning of the commands in unintended ways. There are two main sources of these problems: filenames and architecture names. In order to have shell special characters in filenames, you must set the *shell_safe_filenames* field (see below) to *false*. If you do this, you will need to use the `quote` substitution (see *aesub*(5)) to quote them, so that the shell does not abuse them. Other things which may need quoting include architecture names if you get creative, and edit numbers if unusual ones are generated by your history tool.

CONTENTS

This file contains the following fields:

build_command = string;

This field describes how to build the project (actually, how to do an integration build). This field is mandatory. Used by the *aeb*(1) command. All of the substitutions described by *aesub*(5) are available.

development_build_command = string;

This field describes how to do a development build. If this field is absent, it defaults to the above. Used by the *aeb*(1) command. All of the substitutions described by *aesub*(5) are available.

create_symlinks_before_build = boolean;

This flag is true if Aegis should create symlinks from the development directory to the baseline for all files in the baseline not in the development directory immediately before a development_build_command is issued. Usually used to trick dumb DMTs into believing the development directory contains an entire copy of the project, though sometimes the DMT is smart enough, the tools it must work with are not. Symlinks in the development directory which point to nonexistent files will be removed. Defaults to false if not set.

remove_symlinks_after_build = boolean;

This flag is true if Aegis should remove symlinks which point from the development directory to the baseline directory immediately after a development_build_command is issued. Only consulted if the create_symlinks_before_build field is true, for the purpose of reversing the actions of the create_symlinks_before_build field. Defaults to false if not set.

symlink_exceptions = [string];

This field is used to list filename patterns for which symbolic links must not be made between the development directory and the baseline. These are usually state files for various tools.

change_file_command = string;

This field contains a command to be executed whenever a 'aegis -CoPy_file', 'aegis -CoPy_file_Undo', 'aegis -New_File', or 'aegis -ReMove_file_Undo' command is successful. If this field is absent, nothing is done. Used by the *aecp*(1), *aecpu*(1), *aenf*(1), *aenfu*(1), *aerm*(1), *aermu*(1), and *aemv*(1) commands. All of the substitutions described by *aesub*(5) are available;

in addition,

${File_List}
Space separated list of files named.

project_file_command = string;
This field contains a command to be executed during a development build before the *development build command* above, when (a) it is the first build after a develop begin, or (b) some other change has been integrated into the baseline since the last build. If this field is absent, nothing is done. Used by the *aeb*(1) command. All of the substitutions described by *aesub*(5) are available.

develop_begin_command = string;
This field contains a command to be executed whenever a 'aegis -Develop_Begin' command is successful. If this field is absent, nothing is done. Used by the *aedb*(1) command. All of the substitutions described by *aesub*(5) are available.

integrate_begin_command = string;
This field contains a command to be executed whenever a 'aegis -Integrate_Begin' command is successful. If this field is absent, nothing is done. Used by the *aeib*(1) command. All of the substitutions described by *aesub*(5) are available.

link_integration_directory = boolean;
This flag is true if Aegis should link the files from the baseline into the integration directory, rather than copy them (the default). This has risks, as the build script (e.g. *Howto.cook* or *Makefile*, etc) must unlink targets before rebuilding them; if this is not done the baseline will be corrupted. Used by the *aeib*(1) command.

integrate_begin_exceptions = [string];
This field may be used to specify a list of file names (and file name patterns) which are to be omitted from the copy (link) of the baseline when creating the integration directory. Used by the *aeib*(1) command.

history_create_command = string;
This field is used to create a new history. The command is always executed as the project owner. Used by the *aeip*(1) command. All of the substitutions described by *aesub*(5) are available; in addition,

${Input}
Absolute path of the source file.

${History}
Absolute path of the history file. This may need to be reworked with the *Dirname* and *Basename* substitutions to yield a string suitable for the history tool in question.

See also the *history_put_trashes_file* field, below.

history_get_command = string;
This field is used to get a file from history. The command may be executed by developers. Used by the *aeip*(1) and *aecp*(1) commands. All of the substitutions described by *aesub*(5) are available; in addition,

${History}
The absolute path of the history file. This may need to be reworked with the *Dirname* and *Basename* substitutions to yield a string suitable for the history tool in question.

${Edit}

The edit number to be extracted. It may be an arbitrary string, varying on the particular history tool.

${Output}

The absolute path of the destination file.

history_put_command = string;

This field is used to add a new change to the history. The command is always executed as the project owner. Used by the *aeip*(1) command. All of the substitutions described by *aesub*(5) are available; in addition,

${Input}

The absolute path of the source file.

${History}

The absolute path of the history file. This may need to be reworked with the *Dirname* and *Basename* substitutions to yield a string suitable for the history tool in question.

See also the *history_put_trashes_file* field, below.

history_query_command = string;

This field is used to query the topmost edit of a history file. Result to be printed on the standard output. This command may be executed by developers. Used by the *aeip*(1) and *aecp*(1) commands. All of the substitutions described by *aesub*(5) are available; in addition,

${History}

The absolute path of the history file. This may need to be reworked with the *Dirname* and *Basename* substitutions to yield a string suitable for the history tool in question.

history_put_trashes_file = (fatal, warn, ignore);

Many history tools (e.g. RCS) can modify the contents of the file when it is committed. While there are usually options to turn this off, they are seldom used. The problem is: if the commit changes the file, the source in the repository now no longer matches the object file in the repository - i.e. the history tool has compromised the referential integrity of the repository.

fatal

Emit a fatal error if one or more source files are modified by a *history_put_command* or *history_create_command*. This is the default.

warn

Emit a warning if a source file is modified.

ignore

Ignore a source file changing. You sure better hope it was only in a comment!

diff_command = string;

This field is used to difference of 2 files. The command is always executed by developers. Used by the *aed*(1) command. All of the substitutions described by *aesub*(5) are available; in addition,

${ORiginal}

The absolute path of the original file copied into the change. Usually in the baseline, but not always.

${Input}

The absolute path of the file in the development directory.

${Output}

The absolute path of the file in which to write the difference listing.

merge_command = string;

This field is used to merge two competing edits to a file. The command is always executed by developers. The current directory will be the development directory. This field is used by the *aed*(1) command. All of the substitutions described by *aesub*(5) are available; in addition,

${ORiginal}

The absolute path of the original file copied into the change. Usually not in the baseline, often a temporary file.

${Most_Recent}

The absolute path of the competing edit, usually in the baseline.

${Input}

The absolute path of the file in the development directory. This is the ‘‘preferred’’ edit, if the tool has this concept when highlighting conflicting edits.

${Output}

The absolute path of the file in which to write the merged result. This will usually be the name if a change source file in the development directory.

It is important that this command does not move files around. (See the obsolete *diff3_command* field, below, for some history.)

test_command = string;

This field is used to set the command to be executed by the *aet*(1) command. Defaults to "$shell $file_name" if not set.

All of the substitutions described in *aesub*(5) are available. In addition:

${File_Name}

The absolute path of the test to be executed.

Note that tests are source files, and thus never have the execute bit set.

development_test_command = string;

This field is used to set the command to be executed by the *aet*(1) command when a change is in the *being developed* state. Defaults to be the same as the *test_command* field if not set.

Note: It is a significantly bad idea to make tests behave differently in *being development* and *being integrated* states; avoid this at all costs.

All of the substitutions described in *aesub*(5) are available. In addition:

${File_Name}

The absolute path of the test to be executed.

Note that tests are source files, and thus never have the execute bit set.

architecture = [{ ... }];

This field is a list of system and machine architectures on which each change must successfully build and test. The structures listed have fields as follows:

name = string;

The name of the architecture. This name is available in the *${ARCHitecture}* substitution (see *aesub*(5) for more information), as well as being used internally by Aegis. You may use almost any name for your architecture, but it is best to avoid shell special characters and white space, because it may be substituted into commands to be executed by Aegis.

pattern = string;

The system and machine architecture are determined by using the *uname*(2) system call. The *uname*(2) return value is assembled into a string of the form "*sysname*-*release*-*version*-*machine*".

The pattern field must match this uname result string. The first match found is used. The pattern is a shell file name pattern, see *sh*(1) for more information.

For example, the pattern *SunOS-4.1*-*-sun4** matches a machine the author commonly uses, which returns *SunOS-4.1.3-8-sun4m* from the *uname*(2) system call.

If the architecture field is not set, it defaults to

```
architecture =
[
        {
                name = "unspecified";
                pattern = "*";
        }
];
```

file_template = [{ ... }];

The file template is consulted whenever a new file is created, by one of the *aenf*(1) or commands. Each list item has the form:

pattern = [string];

The name of the file, relative to the development directory. Each string is a shell file name pattern; see *sh*(1) for more information.

body = string;

What to initialize the body of the file to. There are no substitutions.

All of the substitutions described in *aesub*(5) are available for the body string. In addition:

${File_Name}

will be replaced by the name of the new file.

maximum_filename_length = integer;

This field is used to limit the length of filenames. All new files may not have path components longer than this. Existing files are not affected. The last component must also allow for the ",D" suffix of difference files. Where this value is larger than the file system allows, the file system limit will be imposed. Defaults to 14 if not set. Legal values range from 9 to 255.

The file name lengths of project files will be checked at develop end if the project *config* file is in the change. See *aede (1)* for more information.

posix_filename_charset = boolean;

This field may be used to limit the characters allowed in filenames to only those explicitly allowed by POSIX. Defaults to false if not set.

For a filename to be portable across conforming implementations of IEEE Std 1003.1-1988, it shall consist only of alphanumeric characters, dot, hyphen or underscore. Hyphen shall not be used as the first character of a portable filename.

If this field is false, all characters are allowed except non-printing characters, space characters and leading hyphens.

dos_filename_required = boolean;

This field may be used to limit filenames so that they conform to the DOS 8+3 filename limits and to the DOS filename character set. Also denies filenames which look like devices (AUX, *etc*). Defaults to false if not set. This field is used in combination with the other filename fields, it does not replace them.

windows_filename_required = boolean;

This field may be used to limit filenames so that they conform to the Windows98 and WindowsNT filename limits and character set. Also denies filenames which look like devices (AUX, *etc*). Defaults to false if not set. This field is used in combination with the other filename fields, it does not replace them.

shell_safe_filenames = boolean;

This field may be used to limit filenames so that they may not contain shell special characters. If you do not set this to true, you will need to use the ${quote} substitution around filenames in commands, or risk unexpected errors. This field defaults to true if not set.

filename_pattern_accept = [string];

This field is used to specify a list of patterns of acceptable filenames. The patterns are matched against each filename path element. The patterns are constructed from the usual shell filename wildcards. Defaults to "*" if not set.

filename_pattern_reject = [string];

This field is used to specify a list of patterns of unacceptable filenames. The patterns are matched against each filename path element. The patterns are constructed from the usual shell filename wildcards. Defaults to "*,D" if not set. The pattern "*,D" is always appended. Where the *filename_pattern_accept* and filename_pattern_reject fields conflict, the reject takes precedence.

new_test_filename = string;

This field is used to form the filename of new tests, where the filename is not specified on the aent command line. Defaults to "test/${zpad $hundred 2}/t${zpad $number 4}${left $type 1}.sh" if not set.

All of the substitutions defined in *aesub*(5) are available. The following three substitutions are also available:

$Hundred

The test number divided by 100, optional

$Number

The test number, mandatory

$Type The test type: "automatic" or "manual", optional

development_directory_template = string;

This field is used to determine the name of the development directory at develop begin. All of the substitutions defined in aesub(5) are available. The following substitutions is also available:

Default_Development_Directory

The directory within which the development directory is to be created.

Magic A single letter, starting from ‘‘C’’, which can be inserted. This must be used, as it allows Aegis to try different names should there be a conflict.

If not set, defaults to "`$ddd/${left $p ${expr ${namemax $ddd} - ${length .$magic$c}}}.$magic$c`".

For DOS compatibility (8+3 filenames), a useful setting is "`$ddd/${downcase ${left ${id $p} 8}.$magic${right 0$c 2}}`". This ensures that the filename is always a valid 8.3 filename, that it is always lowercase, and it translates any punctuation in the project name into underscores.

metrics_filename_pattern = string;

This field is used to form the name of the metrics file, given a source file. All of the substitutions defined in aesub(5) are available. The following substitutions is also available:

File_Name

The absolute pathname of the source file.

Defaults to "$filename,S" if not set.

trojan_horse_suspect = [string];

This list of filename patterns is consulted by aedist --receive when it is checking for files which could be used to host Trojan horse attacks. This will be different for different projects, so you will need to update this youself.

OBSOLETE FIELDS

There are some obsolete fields in the file. They are provided for backwards compatibility only, and should not be used.

diff3_command = string;

This field is used to difference 3 files. The command is always executed by developers. Used by the *aed*(1) command. All of the substitutions described by *aesub*(5) are available; in addition,

${ORiginal}

The absolute path of the original file copied into the change. Usually not in the baseline.

${Most_Recent}

The absolute path of the competing edit, usually in the baseline.

${Input}

The absolute path of the file in the development directory.

${Output}

The absolute path of the file in which to write the difference listing.

The problem with this field was that the default usage placed the merged source in a strange place. And subsequent *aed*(1) commands would over-write it. This meant that merges would be lost, causing a number of nasty problems. Some sites overcame this by adding ‘‘mv’’ commands to put the output back where the input came from, but this meant that Aegis’ commentary was misleading. Use the ‘‘merge_command’’ field instead. It is almost identical, but Aegis will move the files around for you - so you get the good behavior by default (no lost merges) and the error message is consistent.

SEE ALSO

aeb(1) build a change

aecp(1) copy a file into a change

aecpu(1)
reverse action of aecp

aed(1) find differences between a change and the baseline

aede(1) end development of a change

aeib(1) begin integration of a change

aeip(1) pass integration of a change

aemv(1) rename a file as part of a change

aenf(1) add new files to be created by a change

aenfu(1) remove new files from a change

aent(1) add a new test to be created by a change

aentu(1) remove new tests from a change

aet(1) run tests

aegis(5) aegis file format syntax

aesub(5)
available command substitutions

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aepstate - project state file

SYNOPSIS

project`/info/state`

DESCRIPTION

The *project*`/info/state` file is used to store state information about a project.

This file is maintained by **aegis** and thus should not be edited by humans.

CONTENTS

next_test_number = integer;

Each test is numbered uniquely acrosss all branches of the project. The name is of the form *t[0-9][0-9][0-9][0-9][am].sh* ('a' for automatic and 'm' for manual.)

Almost Obsolete Fields

The following fields are obsolete. They will persist until the next *aenrls*(1), and the new project so generated will use them to define its default branching.

version_major = integer;

The major version number of this release of the project. Always one or more.

version_minor = integer;

The minor version number of this release of the project. Always zero or more.

Obsolete Fields

The following fields are obsolete. They are only present in projects which have yet to be converted to the new branch format. When *Aegis* sees them, they will be moved into the "trunk" transaction.

description = string;

This field contains a description of the project. Large amounts of prose are not required; a single line is sufficient.

owner_name = string;

This field is ignored.

group_name = string;

This field is ignored.

developer_may_review = boolean;

If this field is true, then a developer may review her own change. This is probably only a good idea for projects of less than 3 people. The idea is for as many people as possible to critically examine a change.

developer_may_integrate = boolean;

If this field is true, then a developer may integrate her own change. This is probably only a good idea for projects of less than 3 people. The idea is for as many people as possible to critically examine a change.

reviewer_may_integrate = boolean;

If this field is true, then a reviewer may integrate a change she reviewed. This is probably only a good idea for projects of less than 3 people. The idea is for as many people as possible to critically examine a change.

developers_may_create_changes = boolean;

This field is true if developers may created changes, in addition to administrators. This tends to be a very useful thing, since developers find most of the bugs.

forced_develop_begin_notify_command = string;

This command is used to notify a developer that a change requires developing; it is issued when a project administrator uses an *aedb -User* command to force development of a change by a specific user. All of the substitutions described in *aesub*(5) are available. This field is optional.

develop_end_notify_command = string;

This command is used to notify that a change is ready for review. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

develop_end_undo_notify_command = string;

This command is used to notify that a change had been withdrawn from review for further development. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

review_pass_notify_command = string;

This command is used to notify that a review has passed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

review_pass_undo_notify_command = string;

This command is used to notify that a review has passed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. Defaults to the same action as the *develop_end_notify_command* field. All of the substitutions described by *aesub*(5) are available.

review_fail_notify_command = string;

This command is used to notify that a review has failed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

integrate_pass_notify_command = string;

This command is used to notify that an integration has passed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

integrate_fail_notify_command = string;

This command is used to notify that an integration has failed. It will probably use mail, or it could be an in-house bulletin board. This field is optional, if not present no notification will be given. This command could also be used to notify other management systems, such as progress and defect tracking. All of the substitutions described by *aesub*(5) are available.

default_development_directory = string;
: The pathname of where to place new development directories. The pathname must be absolute. This field is only consulted if the field of the same name in the user configuration file is not set.

umask = integer;
: File permission mode mask. See *umask*(2) for more information. This value will always be OR'ed with 022, because *aegis* is paranoid.

default_test_exemption = boolean;
: This field contains what to do when a change is created with no test exemption specified.

copyright_years = [integer];
: This field contains a list of copyright years, for use in copyright notices, etc. It is updated each integrate_begin, if necessary, to include the current year. Available as the ${Copyright_Years} substitution, and included in the version listing.

next_change_number = integer;
: Changes are numbered sequentially from one. This field records the next unused change number.

next_delta_number = integer;
: Deltas are numbered sequentially from one. This field records the next unused delta number.

src = [{ ... }];
: This field is a list of files in the project. Each list item has the form:

  file_name = string;
  : The name of the file, relative to the baseline.

  usage = file_usage;
  : What the file is for.

  edit_number = string;
  : The edit number of the file.

  locked_by = integer;
  : The change which locked this file.
    Caveat: this field is redundant, you can figure it out by scanning all of he change files. Having it here is very convenient, even though it means multiple updates.

  about_to_be_created_by = integer;
  : The change which is about to create this file for the first time. Same caveat as above.

  deleted_by = integer;
  : The change which last deleted this file. We never throw them away, because (a) it may be created again, and more important (b) we need it to recreate earlier deltas.

history = [{ ... }];
: This field contains a history of integrations for the project. Updated by each successful 'aegis -Integrate_Pass' command.

  delta_number = integer;
  : The delta number of the integration.

  change_number = integer;
  : The number of the change which was integrated.

name = [string];
The names by which this delta is known.

change = [integer];
The list of changes which have been created to date.

administrator = [string];
The list of administrators of the project.

developer = [string];
The list of developers of the project.

reviewer = [string];
The list of reviewers of the project.

integrator = [string];
The list of integrators of the project.

currently_integrating_change = integer;
The change currently being integrated. Only one change (within a project) may be integrated at a time. Only set when an integration is in progress.

version_major = integer;
The major version number of this release of the project. Always one or more.

version_minor = integer;
The minor version number of this release of the project. Always zero or more.

version_previous = string;
The version number this project was derived from. This is of most use when producing "patch" files.

SEE ALSO

aenpr(1)
create a new project

aegis(5) aegis file format syntax

aepattr(5)
project attributes file format

aecstate(5)
change state file

aefstate(5)
file state file

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aer – report script language definition

DESCRIPTION

This manual entry describes the report generator script language used by the *aer*(1) command. The language resembles C, with a touch of *awk* and *perl* for flavour. It also closely resembles the appearance of aegis' database files.

This language grew out of the need to have a general purpose programming language to describe reports, and yet be as familiar as possible to the people who will be using it.

WORDS AND SYMBOLS

This section describes the various words and symbols understood by the language.

Names

A name is a contiguous set of alphanumeric characters, including underscore (_). It must not start with a digit. Names may be of any length. Names are case sensitive, so uppercase and lowercase letters are unique.

Here are some examples of names

print	sqrt	if
how_long	UpperCase	dig57

Some words are *reserved* as keywords. These are the words which appear in **bold** in the statement descriptions, below.

Integer Constants

An integer constant may be decimal, any sequence of digits. Constants may be octal, any sequence of octal digits starting with a zero. Constant may be hexadecimal, any sequence of hexadecimal digits, starting with a `0x` prefix. These are represented by the internal `long` type, so significance is limited.

Here are some examples of integer constants:

43	015	0xbeEf
2147483647	017777777777	0x7FFFFFFF

Floating Point Constants

A floating point constant has an integer part, a fraction part and an exponent part.

Here are some examples of floating point constants:

1.2e3	4.2e+1	1.628e-94
0.567	5e6	.67

String Constants

A string constant is represented as characters within double quotes ("). All characters in the script file are required to be printable, so special characters are represented by *escape sequences.* These escape sequences are:

`\"`	the `"` character
`\\`	the \ character
`\n`	Newline
`\f`	Form Feed
`\r`	Carriage Return
`\b`	Backspace
`\t`	Horizontal Tab
nnn	octal character value

Here are some examples of string constants:

"Hello, World!"	"Go away"	""
"The End0	"slosh is \\\\"	"Say \\"Please\\""

Symbols

The non-alphanumeric characters are used to represent symbols, usually expression operators or statement terminators. The symbols used include:

`!`	`!=`	`!~`	`##`	`##=`
`%`	`%=`	`&`	`&&`	`&=`
`(`	`)`	`*`	`**`	`**=`
`*=`	`+`	`++`	`+=`	`,`
`-`	`--`	`-=`	`.`	`/`
`/=`	`:`	`;`	`<`	`<<`
`<<=`	`<=`	`=`	`==`	`>`
`>=`	`>>`	`>>=`	`?`	`[`
`]`	`^`	`^=`	`{`	`|`
`|=`	`||`	`}`	`~`	`~~`

White Space

White space serves to separate words and symbols, and has no other significance. The language is free-form. White space includes the SPACE, TAB, FF, and NEWLINE characters.

Comments

Comments are delimited by `/*` and `*/` pairs, and are treated as a single white space character.

STATEMENTS

Statement serve to control the flow of execution of the program, or the existence of variables.

The Expression Statement

The commonest statement consists of an expression terminated by a semicolon. The expression is evaluated, and any result is discarded.

Examples of this statement include

```
x = 42;
print("Hello, World!0);
```

The If Statement

The *if* statement is used to conditionally execute portions of code. Examples if the *if* statement include:

```
if (x == 42)
      x = 1;
if (x * x < 1)
      print("no");
```

```
else
        print("yes");
```

The For Statement

The *for* statement has two forms. The first form is described as

```
for (expr1; expr2; expr3)
        stmt
```

The *expr1* is done before the loop begins. The *expr2* controls, the loop; if it does not evaluate to `true` the loop terminates. The loop body is the *stmt*. The loop increment is done by the *expr3*, and the the test is performed again.

Each of the expressions is optional; any or all may be omitted.

Here is an example of a *for* loop:

```
for (j = 0; j < 10; ++j)
        print(j);
```

The second form of the *for* statement looks like this:

```
for (name in keys(passwd))
        print(name, passwd[name].pw_comment);
```

The Break Statement

The *break* statement is used to break out of a loop.

Here is an example of a *break* statement:

```
for (j = 0; ; j = 2 * j + 4)
{
        print(j);
        if (j >= 0x800)
                break;
}
```

The *break* statement works within all loop statements.

The Continue Statement

The *continue* statement is used to terminate the loop body and start another repetition.

Here is an example of a *continue* statement:

```
for (j = 0; j < 1000; j = 2 * j + 4)
{
        if (j < 42)
                continue;
        print(j);
}
```

The *continue* statement works within all loop statements.

The While Statement

The *while* statement is another loop construct. The condition is evaluated before the loop body.

```
line = 0;
while (line < 7)
{
        print("");
        ++line;
```

```
}
```

The Do Statement

The *do* statement is another loop construct. The condition is evaluate after the loop body.

```
do
        print("yuck");
while
        (line++ < 7);
```

The Compound Statement

The *compound* statement is a way of grouping other statements together. It is enclosed in curly braces.

```
if ( lines < 7)
{
        print("This\n");;
        print("could\n");;
        print("have\n");;
        print("been\n");;
        print("seven\n");;
        print("blank\n");;
        print("lines.\n");;
}
```

The Local Statement

The *local* statement is used to declare variables and initialize them to be nul.

```
local x, y, z;
x = 42;
```

All user-defined variables must be declared before they are used.

The Null Statement

The *null* statement does nothing. It consists of a single semicolon. It is most often seen as a loop body.

```
for (n = 0, bit = 1; n < bit_num; ++n, bit <<= 1)
        ;
```

EXPRESSIONS

Expressions are much the same as in C, using the same operators. The following table describes operator precedence and associativity:

[ ]	subscripting	*value* [*expr*]
( )	function call	*expr* (*expr_list*)
( )	grouping	(*expr*)
++	post increment	*lvalue* ++
++	pre increment	++*lvalue*
−−	post decrement	*lvalue* −−
−−	pre decrement	−−*lvalue*
˜	compliment	˜ *expr*
!	not	! *expr*
−	unary minus	− *expr*
+	unary plus	+ *expr*

`**`	exponentiation	*expr* `**` *expr*
`*`	multiply	*expr* `*` *expr*
`/`	divide	*expr* `/` *expr*
`%`	modulo (remainder)	*expr* `%` *expr*
`~~`	matches	*expr* `~~` *expr*
`!~`	does not match	*expr* `!~` *expr*
`in`	list member	*expr* `in` *expr*
`+`	addition (plus)	*expr* `+` *expr*
`-`	subtraction (minus)	*expr* `-` *expr*
`##`	list and string join	*expr* `##` *expr*
`<<`	shift left	*expr* `<<` *expr*
`>>`	shift right	*expr* `>>` *expr*
`<`	less than	*expr* `<` *expr*
`<=`	less than or equal	*expr* `<=` *expr*
`>`	greater than	*expr* `>` *expr*
`>=`	greater than or equal	*expr* `>=` *expr*
`==`	equal	*expr* `==` *expr*
`!=`	not equal	*expr* `!=` *expr*
`&`	bitwise AND	*expr* `&` *expr*
`^`	bitwise exclusive OR	*expr* `^` *expr*
`|`	bitwise inclusive OR	*expr* `|` *expr*
`? :`	arithmetic if	*expr* `?` *expr* `:` *expr*
`=`	simple assignment	*expr* `=` *expr*
`*=`	multiply and assign	*expr* `*=` *expr*
`/=`	divide and assign	*expr* `/=` *expr*
`%=`	modulo and assign	*expr* `%=` *expr*
`+=`	add and assign	*expr* `+=` *expr*
`-=`	subtract and assign	*expr* `-=` *expr*
`<<=`	shift left and assign	*expr* `<<=` *expr*
`>>=`	shift right and assign	*expr* `>>=` *expr*
`&=`	AND and assign	*expr* `&=` *expr*
`^=`	exclusive OR and assign	*expr* `^=` *expr*
`|=`	inclusive OR and assign	*expr* `|=` *expr*
`,`	comma (sequencing)	*expr* `,` *expr*

Most of these operators behave as they do in C, but some of these operators will require some explanation.

Exponentiation

The `**` operator raises the left argument to the right'th power. It is right associative.

Match

The `~~` operator compares two strings. It returns a number between 0.0 and 1.0. Zero means completely different, one means identical. Case is significant.

Not Match

The `!~` is used to compare two strings, and returns the opposite of the `~~` operator; one if completely different, and zero if identical.

String Join

The `##` operator is used to join two strings together.

TYPES

There are several types used within the report language.

array
: Values of this type contain other values, indexed by a string. If you attempt to index by an arithmetic type, it will be silently converted to a string. Use the *keys* function to determine all of the keys; use the *count* function to determine how many entries an array has. The type of an array element is not restricted, only the index must be a string.

boolean
: This type has two values: `true` and `false`. These value arise from the boolean operators described earlier.

integer
: This type is represented by the *long* C type. It has a limited range of values (usually -2e9 to 2e9 approximately). If used in a string context, it will be silently converted to a string. For exact control of the format, used the *sprintf* function.

list
: Values of this type contain a list of other values. The type of these values is not restricted. The array index operator (e[e]) may be used to access list elements; indexes start at zero (0).

string
: Values of this type are an arbitrary string of C characters, except the NUL character ( ). Strings may be of any length.

struct
: Values of this type contain additional values. These values are accessed using the "dot" operator. These values may also be treated as if they were arrays.

real
: This type is represented the the *double* C type. If used in a string context, it will be silently converted to a string. For exact control of the format, used the *sprintf* function.

FUNCTIONS

There are a number of built-in functions.

basename
: This function is used to extract the last element from a file path.

ceil
: This function is used to round a number to an integer, towards positive infinity.

change_number
: This function is used to determine the change number. It may be set by the **-Change** command line option, or it may default. The return value is an integer.

change_number_set
: This function maybe used to determine if the change number was set by the **-Change** command line option. The return value is a boolean.

columns This function is used to define the report columns. Each argument is a structure containing some or all of the following fields:

left	the left margin, counting characters from 0 on the left
right	the right margin, plus one
width	the width in characters, defaults to 7 if right not specified
padding	white space between columns, defaults to 1 if not set
title	the title for this column, separate multiple lines with \n

The columns must be defined before the *print* function is used.

count This function is used to count the number of elements in a list or array.

dirname This function is used to extract all but the last element from a file path.

eject This function is used to start a new page of output.

floor This function is used to round a number to an integer, towards negative infinity.

getenv This function is used to get the value of an environment variable. Will return the empty string if not set.

gettime This function is used to parse a string to produce a time. It understands a variety of different date formats.

keys This function may be given an array or a list as argument. It returns a list of keys which may be used to index the argument. Most often seen in for loops.

length This function is used to find the length of a string.

mktime This a synonym for the *gettime* function.

need This function is used to insert a page break into the report if the required number of lines is not available before the end of page. If sufficient lines are available, only a single blank line will be inserted. The return value is void.

now This function takes not arguments, and returns the current time.

page_length
This function may be used to determine the length of the output page in lines. The return value is an integer.

page_width
This function may be used to determine the width of the output page in columns. The return value is an integer.

print This function is used to print into the defined columns. Columns will wrap around.

project_name
This function is used to determine the project name. It may be set by the **-Project** command line option, or it may default. The return value is a string.

project_name_set
This function maybe used to determine if the project name was set by the **-Project** command line option. The return value is a boolean.

quote_html
This function quotes its argument string to insulate HTML special characters; these include ‘‘less than’’ (<), ‘‘ampersand’’ (&) and non-printing characters. This is most often used to generate

suitable text for web pages.

quote_url
: This function quotes its argument string to insulate URL special characters; these include "?+#:&=" and non-printing characters. This is most often used to generate suitable text for web pages.

ceil
: This function is used to round a number to an integer, towards the closest integer.

sort
: This function must be given a list as argument. The values are sorted into ascending order. A new list is returned.

split
: This function is used to split a string into a list of strings. The first argument is the string to split, the second argument is the character to split around.

sprintf
: This function is used to build strings. It is similar to the *sprintf*(3) function.

strftime
: This function is used to format times as strings. The first argument is the format string, the second argument is a time. See the *strftime*(3) man page for more the format specifiers.

subst
: This function is used to substitute strings by regular expression. The first argument is the pattern to match, the second argument is the substitution pattern, the third argument is the input string to be substituted. The option fourth argument is the number of substitutions to perform; the default is as many as possible.

substr
: This function is used to extract substrings from strings. The first argument is a string, the second argument is the starting position, starting from 0, and the third argument is the length.

terse
: This function may be used to determine of the **-TERse** command line option was used. The return type is a boolean.

title
: This function is used to set the title of the report. It takes at most two arguments, one for each available title line.

trunc
: This function is used to round a number to an integer, towards zero.

typeof
: This function is used to determine the type of a value. The return type is a string containing the name of the type, as described in the

unquote_url
: This function will remove URL quoting from the argument string. URL quoting takes the form of a percent sign (%) followed by two hex digits. This is replaced by a single character with the value represented by the hex digits.

working_days
: This function is used to determine the number of working days between two times.

wrap
: This function is used to wrap a string into a list of strings. The first argument is the wring to wrap, the second argument is the maxmium width of the output strings.

wrap_html
: This function is used to wrap a string into a list of strings. The first argument is the wring to wrap, the second argument is the maxmium width of the output strings. This is very similar to the *wrap* functions, except thatit inserts HTML paragraph breaks <p> or line breaks
 to reflect the newlines within the string (2 or 1, respectively). *TYPES* section.

VARIABLES

There are a number of built-in variables.

arg — This variable is a list containing the arguments passed on the *aer*(1) command line.

group — This variable is an array containing all of the entries in the */etc/group* file. Each entry is a structure with fields as documented in the *group*(5) manual entry. The *gr_mem* element is a list of strings. This array may be indexed by either a string, treated as a group name, or by an integer, treated as a GID.

passwd — This variable is an array containing all of the entries in the */etc/passwd* file. Each entry is a structure with fields as documented in the *passwd*(5) manual entry. This array may be indexed by either a string, treated as a user name, or by an integer, treated as a uid.

project — This variable is an array containing one entry for each aegis project, indexed by name. Each array element is a structure, containing

name	the project name
directory	the root of the project directory tree
state	the project state

The project state contains the fields documented in the *aepstate*(5) manual entry. Except: the *change* field is not a list of change numbers, it is an array indexed by change number of change states, as documented in the *aecstate*(5) manual entry.

FILES

The reports are kept in the */usr/local/share/aegis/report* directory. The reports are associated with a name by the */usr/local/share/aegis/report.index* file. Their names use the command line argument abbreviation scheme, so that report names may be abbreviated.

SEE ALSO

aer(1) — report generator

aecstate(5) — change state description

aepstate(5) — project state description

aerptidx(5) — report index file format

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
/\/*	WWW:	http://www.canb.auug.org.au/˜millerp/

NAME

aerptidx – report index file format

SYNOPSIS

/usr/local/share/aegis/report.index
/usr/local/share/aegis/report.local

DESCRIPTION

The report index file is used to store pointers to report scripts, and descriptions of the reports.

When searching for a report, the *aer*(1) command searches down the *AEGIS_PATH* looking for *report.index* files, and searching them for the report named.

CONTENTS

where = [{ ... }];

This field is a table relating report name to file name. The structure is as follows:

name = string;

The name of a report. The command line argument naming scheme is used, to provide abbreviatable names.

description = string;

A brief description of the report. It should not be very long, one or two lines at most.

filename = string;

The name of the file containing the report script. If a relative path is given, it will be interpreted to be relative to the directory containing the *report.index* file.

SEE ALSO

aer(1) report generator

aegis(5) aegis file format syntax

COPYRIGHT

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au
/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aesub − available command substitutions

DESCRIPTION

When other programs are invoked by the *aegis* program, it is usually via a command string in a configuration file. This section describes the format of these command strings.

GENERAL FORM

The command strings are very similar to shell variables. An example will illustrate this:

```
build_command =
        "cook project=${project} change=${change}";
```

In this command definition, the "${project}" part is a substitution: the name of the project will be substituted in the command at this point.

Substitutions may take several forms:

$name

This is the same as saying "${name}". The name must start with an alphabetic, and be followed by zero or more alphanumerics.

${name}

The name in this form may contain any non-blank characters, and it may be subject to substitution.

${name *arg*...}

The name and the arguments in this form may contain any non-blank characters, and it may be subject to further substitution. Within the braces (**{** and **}**) pairs of single quote characters (**'***blah blah***'**) may be used to insulate spaces and other special characters, or you may use the back quote (****) to escape a single character.

$$

This is replaced by a single $ character. It avoid RCS expansions, you can also use ${$}.

%%

This is replaced by a single % character. Percent (**%**) followed by anything else is illegal.

$#...\n

This is a comment, usually found in template files read in using the ${read_file} substitution. It consumes all characters up to and including the next newline. (See also ${comment}, below.)

As another example, the *dirname* substitution is replaced by the directory name of the argument, rather like the *dirname*(1) command. In the command

```
history_query_command =
        "get -t -g ${Dirname $History}/s.${Basename $History}";
```

the *Dirname* and *Basename* substitutions are used to construct a suitable path to the SCCS file in the history directory.

ABBREVIATIONS

The names of the various substitutions may be abbreviated. In the above examples, and in the lists which follow, the minimum abbreviation is the uppercase letters. All substitution name are case insensitive.

The above example could be abbreviated to

```
history_query_command =
        "get -t -g ${d $h}/s.${b $h}";
```

Ambiguous abbreviations will result in a fatal error message.

SUBSTITUTIONS

There are many substitutions which are always understood, and some which are specific to the command being substituted. Specific entries will be defined in the relevant manual section.

The following lists contains those substitutions which are always understood:

Add_Path_Suffix

This substitution may be used to add a suffix to each element of a colon-separated path list. The first argument is the suffix to use, the second and subsequent arguments are the colon-separated paths to work on. The result is a single colon separated path. Often used in combination with the ${search_path} substitution, below.

Administrator_List

Space separated list of the project's administrators.

ARCHitecture

This substitution is replaced by the architecture name appropriate for the current execution environment. Requires no arguments. See the *architecture* field of *aepconf*(5) for more information. When used in commands, you may need to surround this substitution with the *quote* substitution (see below), if any of your architecture names contain shell special characters.

BaseLine

Absolute path of the the project's baseline.

Basename

This substitution takes one argument, a pathname. The value of the substitution will be the last element of the pathname. This is similar to the *basename*(1) command.

BINary_DIRectory

The absolute path of Aegis' architecture-specific binary (executables) directory. This correspondes to the "*./configure -bindir*" option when Aegis was built. This is where most of the Aegis executable programs are installed.

Change

The change number.

COMment

Inserts exactly nothing; any and all arguments are ignored. Another form of comment is "$#" which extends to the end of the current line.

Copyright_Years

Inserts a comma separated list of copyright years from the project attributes. This list of years is maintained by *aegis* at integrate begin, and so is only guaranteed to be up-to-date in the'*being integrated*' state. Do not use this substitution in new file templates, it is not guaranteed to be up-to-date in the '*being developed*' state; use the ${date %Y} substitution in new file templates.

This list contains spaces, so if you use it to build commands, you will probably need to quote, it as well.

DATa_DIRectory

The absolute path of Aegis' architecture-neutral library directory. This correspondes to the "*./configure -datadir*" option when Aegis was built. This is where most of the scripts included with Aegis are installed.

DAte
: With no arguments, the output is the current date. If there are arguments, they form a format string. This is similar to the *date*(1) command on many UNIX systems. For a description of the date formats, see the *DATE* section, below.

DELta
: The delta number of the change. This is only available when the change is in the *being integrated* state or the *completed* state.

DEVeloper
: The name of the developer.

DEVeloper_List
: Space separated list of the project's developers.

Development_Directory
: The absolute path of the change's development directory. Only available when the change is between the *being developed* state and the *being integrated* state.

Dirname
: This substitution takes one argument, a pathname. The value of the substitution will be everything but the last element of the pathname. This is similar to the *dirname*(1) command.

DownCase
: This substitution takes one argument. The value of the substitution will be the argument with any upper case letters mapped to lower case.

ERrno
: This substitution takes no arguments. The value of the substitution will be the value if the *errno* variable provided by the system, as mapped through the *strerror* function. Thus you may give the users informative system error messages.

EXpression
: This substitution evaluates simple arithmetic expressions. The arguments must constitute a valid expression, white space and word boundaries are ignored.

IDentifier
: This substitution takes one argument. The value of the substitution will be the argument with all characters but alpha numerics mapped into an underscore (_), so as to form a legal C identifier.

INTegration_Directory
: The absolute path of the change's integration directory. Only available when the change is in the *being integrated* state.

INTegrator
: The name of the change's integrator. Only available when the change is in the *being integrated* state or the *completed* state.

INTegrator_List
: Space separated list of the project's integrators.

LEFt
: This substitution extracts the left hand side of strings. It takes two arguments: the first is the string, the second is the number of characters.

LENgth This substitution determines the length of strings, the result is a number. It takes one argument: the string to be measured.

LIBrary
The absolute path of Aegis' library directory. This correspondes to the "*./configure -datadir*" option when Aegis was built. This substitution is deprecated – please use ${datadir} instead.

LIBrary_DIRectory
The absolute path of Aegis' architecture-specific library directory. This correspondes to the "*./configure -libdir*" option when Aegis was built.

Name_Maximum
This substitution is used to get the maximum file name length within a file system. It takes one argument: the name of a directory within the file system. Frequently used with ${left} to crop filenames to the file system maximum.

PLural
This function requires 2 or 3 arguments. The first argument is evaluated as a number, if it is plural (not equal to 1) the second argument is the result, otherwise the thrid argument is the result (or empty if not given). This is mostly used to pluralize sentences in error messages.

Project
This substitution provides various information about the project, based on the argument it is given.

name This returns the name of the project. (This is the default if no argument is given.)

description
This returns the description of the project (the one which appears in the project listing).

trunk_name
This returns the name of the trunk of the project (i.e. no branch numbers included).

trunk_description
This returns the description of the trunk of the project.

QUote
This substitution may be used to quote shell special characters. If no quoting is required, not quotes will be inserted. This is used to insulate shell special characters in filenames when forming commands.

Read_File
Read a file and substitute the contents of the file. Requires exactly one argument, the pathname of the file to be read. If the pathname is a project source file, you will need to use the *source* substitution to resolve the path. It is a fatal error if the file does not exist, or is not readable. It is a fatal error if the pathname is not absolute (because the current directory is undefined).

Reviewer
The name of the change's reviewer. Only available when the change is between the *awaiting integration* state and the *completed* state.

Reviewer_List
Space separated list of the project's reviewers.

RIght
This substitution extracts the right hand side of strings. It takes two arguments: the first is the string, the second is the number of characters.

Search_Path
The Search_Path substitution is replaced by a colon separated list of absolute paths to search when building a change, it will point from a change to its branch and so on up to the project trunk.

SHell
The absolute path of a Bourne shell which understands functions. Requires exactly zero arguments.

Source
Resolve the argument filename into a pathname. It is an error if the file is not a source file. An optional second argument may be "Absolute" or "Relative", and may be abbreviated. Relative will attempt to provide a development-directory-relative pathname whenever possible, absolute will always result in an absolute path. The default is "Relative".

STate
The state the current change is in. It is an error if the substitution does not refer to a change.

Trim_DIRectory
This substitution takes one or two arguments. If given one argument, one directory component (if present) is removed from the argument, which is assumed to be a file name. If two arguments are present, the first is a directory count; at most this many directory components (if present) will be removed. The base file name is always left.

Trim_EXTension
This substitution takes one argument. Any file name extension (a dot characters and the characters following) will be removed from the final filename section of the argument.

UpCase
This substitution takes one argument. The value of the substitution will be the argument with any lower case letters mapped to upper case.

USer
This substitution provides various information about the user who executed the command, based on the argument it is given.

- login The login name of the user. (This is the default if no argument is given.)
- name The full name of the user.
- email The email address of the user.
- home The home directory of the user.

Version
The version of the change. If the change is in the *being integrated* state or the *completed* state, the version will be of the form "***a*.*b*.D***ddd*", where "a" is the project's major version number, "b" is the project's minor version number, and "ddd" is the change's delta number. If the change is in any other state, the version will be of the form "***a*.*b*.C***ccc*", where "ccc" is the change number.

Zero_Pad

This substitution is used to zero pad a string on the left. It takes two arguments: the first is the string to be padded, the second is the minimum string width.

DATE

This section describes the format specifiers accepted by the date substitution. These are the same specifiers as defined by the ANSI C standard for the strftime function.

%%	The percent character (%)
%a	the abbreviated weekday name
%A	the full weekday name
%b	the abbreviated month name
%B	the full month name
%c	the date and time
%d	the day of the month, zero padded
%H	the hour of the 24-hour day
%I	the hour of the 12-hour day
%j	the day number of year, zero padded, one based
%m	the month of the year, zero padded, one based
%M	the minute of the hour, zero padded
%p	meridian indicator, AM or PM as appropriate
%S	the second of the minute
%U	the Sunday week of the year
%w	the day of the week, Sunday is 0
%W	the Monday week of the year
%x	the date, as *mmm dd yyyy*
%X	the time, as *hh:mm:ss*
%y	the year of the century
%Y	the year including the century
%Z	time zone abbreviation

Using an undefined format specifier will produce random results, depending on the version of UNIX you are on.

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/~millerp/

NAME

aeuconf - user configuration file

SYNOPSIS

$HOME/.aegisrc

DESCRIPTION

A user configuration file is used to hold user defaults. This file is created and edited by the user. This file is only ever read by aegis, it is never written.

AEGIS_FLAGS

This environment variable has the same format. It is read first, and over-rides the *.aegisrc* file contents. This is intended to be used within the tests distributed with aegis, but can also be of use within some shell scripts.

CONTENTS

The file contains the following fields:

default_development_directory = string;

The pathname of where to place new development directories. The pathname may be relative, in which case it is relative to *$HOME*. The default is the field of the same name in the project attributes, or *$HOME* neither is set.

default_project_directory = string;

The pathname of where to place new project directories. The pathname may be relative. If this path is relative, it is relative to *$HOME*. The default is *$HOME*.

delete_file_preference = (no_keep, interactive, keep);

All of the commands which delete files will consult this field to determine if the file should be deleted. Defaults to *no_keep* if not set.

default_project_name = string;

The name of a project.

default_change_number = integer;

The number of a change.

Please note that the *default_project_name* field and the *default_change_number* field are unrelated. Specifying both does not mean that single change within that single project, they have nothing to do with each other.

diff_preference = (automatic_merge, no_merge, only_merge);

The *aed(1)* command will consult this field to determine what to do:

no_merge

means only diff the files, even if some have out of date versions.

only_merge

means merge those files with out of date versions, and do not do anything else, even if they need to be diffed.

automatic_merge

means to do *only_merge* if any source files require merging, otherwise do *no_merge*. It never combines merges and differences in the same pass.

The corresponding command line options to the *aed*(1) command take precedence, this field is only consulted if you do not give a corresponding command line argument. Defaults to *automatic_merge* if not set.

pager_preference = (foreground, never);
: This field is consulted for listings and help. The standard output is only piped to a pager if the command is run in the foreground and the standard output is directed at a terminal.

foreground
: The standard output will be piped through the command given in the `PAGER` environment variable (or *more* if not set).

never
: The standard output will not be redirected.

This field defaults to *foreground* if not set.

persevere_preference = (all, stop);
: This field is consulted by the *aet*(1) command, to determine if it should run all tests, or stop after the first failure. This field defaults to *all* if not set.

log_file_preference = (snuggle, append, replace, never);
: This field controls the behavior of the log file. It usually defaults to *snuggle* if not set, although some commands may default it to *append*. When the log file is in use, the output continues to be sent to the screen if the process is in the foreground and the standard output is a terminal.

never
: Do not redirect the output to a log file.

replace
: Replace any log file that is present, create a new one if none already exists.

append
: Append the log to the end of any existing log file, create a new one if none already exists.

snuggle
: Append the log to the end of any existing log file if that log file was last modified less than 30 seconds ago, otherwise replace any existing log file; create a new one if none already exists. This option allows runs of aegis commands to produce a meaningful log file.

lock_wait_preference = (always, background, never);
: This field is consulted by all commands which wait for locks.

always
: The "always" setting says that all commands should always wait for locks. This is the default.

background
: The "background" setting says that background commands should always wait for locks, and foreground commands will not.

never
: The "never" setting says that no command should ever wait for locks. If the command would wait, it will exit with status 1.

This user preference can be over-ridden by the **−wait** and **−nowait** command line options.

symbolic_link_preference = (verify, assume);
: This field is consulted by *aeb*(1) when the project config file specifies *create_symbolic_links_before_build* as true. The verification of the links can be quite time consuming; if you are confident that they are already correct (say, from a recent build run) you may wish to assume they are correct and not verify them repeatedly.

verify This setting says to always verify the symbolic links to the baseline. This is the default.

assume This setting says to always assume the links are correct, unless there has been a recent integration.

This user preference can be over-ridden by the **-Verify_Symbolic_Links** and **-Assume_Symbolic_Links** command line options.

relative_filename_preference = (current, base);

This field is consulted by most commands which accept filenames on the command line. It controls wether relative filenames are relative to the current directory (this is the default), or relative to the base of the project source tree.

current This setting says to interpret relative filenames against the current directory.

base This setting says to interpret relative filenames against the base of the source tree.

This user preference can be over-ridden by the **-BAse_RElative** and **-CUrrent_RElative** command line options.

email_address = string;

This field may be used to set the preferred email address. Defaults to *‘whoami‘@‘hostname‘* if not set, which is usually not what is required, particularly if you are behind a firewall.

SEE ALSO

aegis(5) aegis file format syntax

aed(1) difference and merge files

COPYRIGHT

aegis version 3.15

AUTHOR

Peter Miller E-Mail: millerp@canb.auug.org.au

/\/* WWW: http://www.canb.auug.org.au/~millerp/

NAME

aeustate – user state file

SYNOPSIS

`/usr/local/com/aegis/user/`*user-name*

DESCRIPTION

A user state file is used to store information about a user. These file are created and maintained by aegis. These file should not be edited by humans.

CONTENTS

own = [{ ... }];

This field is a list of change the user is currently working on, within project. The changes are in either the *being_developed* or *being_integrated* state. The structure of this field is as follows:

project = string;

The name of a project.

change = [integer];

The changes this user is working on in the project.

SEE ALSO

aegis(5) aegis file format syntax

COPYRIGHT

AUTHOR

Peter Miller	E-Mail:	millerp@canb.auug.org.au
`/\/\*`	WWW:	http://www.canb.auug.org.au/˜millerp/

	The README File	1
	Release Notes	5
	How to Build the Sources	16
aegis(1)	project change supervisor	26
ae_c(1)	set change number	34
ae_p(1)	set project name	35
aeb(1)	build a change	36
aeca(1)	modify the attributes of a change	42
aecd(1)	change directory	45
aechown(1)	set change owner	48
aecp(1)	copy a file into a change	58
aecpu(1)	reverse action of aecp	63
aed(1)	difference a change	67
aedb(1)	begin development of a change	72
aedbu(1)	undo the effects of aedb	77
aede(1)	complete development of a change	80
aedeu(1)	recall a change for further development	83
aedist(1)	remotely distribute a change	85
aedn(1)	assign a symbolic name to a project delta	90
aefind	search for files in directory hierarchy	92
aefp(1)	calculate file fingerprint	99
aeib(1)	begin integrating a change	101
aeibu(1)	reverse the aeib command	105
aeif(1)	fail a change integration	108
aeip(1)	pass a change integration	111
ael(1)	list interesting things	116
aelic(1)	GNU General Public License	120
aemv(1)	rename a file as part of a change	126
aena(1)	add a new administrator to a project	129
aenbr(1)	create a new branch	131
aenbru(1)	remove a branch	133
aenc(1)	add a new change to a project	135
aencu(1)	remove a change	138
aend(1)	add new developers to a project	141
aenf(1)	add new files to be created by a change	143
aenfu(1)	remove new files from a change	148
aeni(1)	add new integrators to a project	152
aenpr(1)	create a new project	154
aenrls(1)	create a new project from an old-style project	158
aenrv(1)	add new reviewers to a project	162
aent(1)	add a new test to a change	164
aentu(1)	remove new tests from a change	170
aepa(1)	modify the attributes of a project	173
aer(1)	report generator	176
aera(1)	remove administrators from a project	178
aerd(1)	remove developers from a project	180
aerect	draw a rectangle	182
aerf(1)	fail a change review	184
aeri(1)	remove integrators from a project	187
aerm(1)	add files to be deleted to a change	189
aermpr(1)	remove project	192
aermu(1)	remove files to be deleted from a change	195
aerp(1)	pass a change review	198
aerpu(1)	rescind a change review pass	201

aerrv(1)	remove reviewers from a project	203
aet(1)	run tests	205
aev(1)	version information	211
aecattr(5)	change attributes file format	218
aecstate(5)	change state file format	220
aedir(5)	directory structures	228
aefstate(5)	file state file format	230
aegis(5)	file format	233
aegstate(5)	global state file format	236
aemetrics(5)	metrics values file format	237
aepattr(5)	project attribute file format	238
aepconf(5)	project configuration file format	241
aepstate(5)	project state file format	249
aer(5)	report script language definition	254
aerptidx(5)	report index file format	263
aesub(5)	available command substitutions	264
aeuconf(5)	user configuration file format	271
aeustate(5)	user state file	274

aecpu(1)	63	aegis -CoPy file Undo - reverse	action of aecp
aena(1)	129	aegis -New Administrator -	add a new administrator to a project
aenc(1)	135	aegis -New Change -	add a new change to a project
aent(1)	164	aegis -New Test -	add a new test to a change
aerm(1)	189	aegis -ReMove file -	add files to be deleted to a change
aend(1)	141	aegis -New Developer -	add new developers to a project
aenf(1)	143	aegis -New File -	add new files to be created by a change
aeni(1)	152	aegis -New Integrator -	add new integrators to a project
aenrv(1)	162	aegis -New ReViewer -	add new reviewers to a project
aena(1)	129	aegis -New	Administrator - add a new administrator to a project
aera(1)	178	aegis -Remove	Administrator - remove administrators from a project
aena(1)	129	aegis -New Administrator - add a new	administrator to a project
aera(1)	178	aegis -Remove Administrator - remove	administrators from a project
ae_c(1)	34		ae c - set change number
ae_p(1)	35		ae p - set project name
aecattr(5)	218		aecattr - aegis change attributes file
aecpu(1)	63	aegis -CoPy file Undo - reverse action of	aecp
aecstate(5)	220		aecstate - aegis change state file
aedbu(1)	77	aegis -Develop Begin Undo - undo the effects of	aedb
aedir(5)	228		aedir - aegis directory structures
aedist(1)	85		aedist - remotely distribute a change
aefstate(5)	230		aefstate - aegis file state file
tkaegis(1)		aegis - GUI interface for	Aegis, using TCL/TK
aegstate(5)	236		aegstate - aegis state file
aeibu(1)	105	aegis -Integrate Begin Undo - reverse the	aeib command
aemetrics(5)	237		aemetrics - metrics values file format
aepattr(5)	238		aepattr - project attribute file
aepconf(5)	241		aepconf - aegis project configuration file
aepstate(5)	249		aepstate - project state file
aer(5)	254		aer - report script language definition
aerect(1)			aerect - draw a rectangle
aerptidx(5)	263		aerptidx - report index file format
aesub(5)	264		aesub - available command substitutions
aeuconf(5)	271		aeuconf - user configuration file
aeustate(5)	274		aeustate - user state file
aeclone(1)		aegis -clone - make	an exact copy of a change
aenrls(1)	158	aegis -New ReLeaSe - create a new project from	an old-style project.
aedn(1)	90	aegis -Delta Name -	assign a symbolic name to a project delta
aepattr(5)	238	aepattr - project	attribute file
aeca(1)	42	aegis -Change	Attributes - modify the attributes of a change
aepa(1)	173	aegis -Project	Attributes - modify the attributes of a project
aecattr(5)	218	aecattr - aegis change	attributes file

aeca(1)	42	aegis -Change Attributes - modify the	attributes of a change
aepa(1)	173	aegis -Project Attributes - modify the	attributes of a project
aesub(5)	264	aesub -	available command substitutions
aed(1)	67	aegis -DIFFerence - find differences between a change and the	baseline
aedb(1)	72	aegis -Develop	Begin - begin development of a change
aeib(1)	101	aegis -Integrate	Begin - begin integrating a change
aedb(1)	72	aegis -Develop Begin -	begin development of a change
aeib(1)	101	aegis -Integrate Begin -	begin integrating a change
aeibu(1)	105	aegis -Integrate	Begin Undo - reverse the aeib command
aedbu(1)	77	aegis -Develop	Begin Undo - undo the effects of aedb
aed(1)	67	aegis -DIFFerence - find differences	between a change and the baseline
aenbr(1)	131	aegis -New BRanch - create a new	branch
aenbru(1)	133	aegis -New BRanch Undo - remove a	branch
aenbr(1)	131	aegis -New	BRanch - create a new branch
aenbru(1)	133	aegis -New	BRanch Undo - remove a branch
aeb(1)	36	aegis -	Build - build a change
aeb(1)	36	aegis -Build -	build a change
ae_c(1)	34	ae	c - set change number
aefp(1)	99	aegis -	calculate file fingerprint
aedist(1)	85	aedist - remotely distribute a	change
aeb(1)	36	aegis -Build - build a	change
aeca(1)	42	aegis -Change Attributes - modify the attributes of a	change
aecp(1)	58	aegis -CoPy file - copy a file into a	change
aedb(1)	72	aegis -Develop Begin - begin development of a	change
aede(1)	80	aegis -Develop End - complete development of a	change
aeib(1)	101	aegis -Integrate Begin - begin integrating a	change
aemv(1)	126	aegis -MoVe file - rename a file as part of a	change
aenf(1)	143	aegis -New File - add new files to be created by a	change
aenfu(1)	148	aegis -New File Undo - remove new files from a	change
aent(1)	164	aegis -New Test - add a new test to a	change
aentu(1)	170	aegis -New Test Undo - remove new tests from a	change
aerm(1)	189	aegis -ReMove file - add files to be deleted to a	change
aermu(1)	195	aegis -ReMove file Undo - remove files to be deleted from a	change
aeclone(1)		aegis -clone - make an exact copy of a	change
aenc(1)	135	aegis -New	Change - add a new change to a project
aed(1)	67	aegis -DIFFerence - find differences between a	change and the baseline

aeca(1)	42	aegis -	Change Attributes - modify the attributes of a change
aecattr(5)	218	aecattr - aegis	change attributes file
aecd(1)	45	aegis -Change Directory -	change directory
aecd(1)	45	aegis -	Change Directory - change directory
aedeu(1)	83	aegis -Develop End Undo - recall a	change for further development
aencu(1)	138	aegis -New Change Undo - remove a new	change from a project
aeif(1)	108	aegis -Integrate Fail - fail a	change integration
aeip(1)	111	aegis -Integrate Pass - pass a	change integration
ae_c(1)	34	ae c - set	change number
aechown(1)	48	aegis -Change Owner - set	change owner
aechown(1)	48	aegis -	Change Owner - set change owner
aerf(1)	184	aegis -Review FAIL - fail a	change review
aerp(1)	198	aegis -Review PASS - pass a	change review
aerpu(1)	201	aegis -Review Pass Undo - rescind a	change review pass
aecstate(5)	220	aecstate - aegis	change state file
aegis(1)	26	aegis - project	change supervisor
aenc(1)	135	aegis -New Change - add a new	change to a project
aencu(1)	138	aegis -New	Change Undo - remove a new change from a project
aeclean(1)		aegis -	CLEan - clean files from development directory
aeclean(1)		aegis -CLEan -	clean files from development directory
aeclone(1)		aegis -	clone - make an exact copy of a change
aeibu(1)	105	aegis -Integrate Begin Undo - reverse the aeib	command
aesub(5)	264	aesub - available	command substitutions
aede(1)	80	aegis -Develop End -	complete development of a change
aepconf(5)	241	aepconf - aegis project	configuration file
aeuconf(5)	271	aeuconf - user	configuration file
aecp(1)	58	aegis -CoPy file -	copy a file into a change
aecp(1)	58	aegis -	CoPy file - copy a file into a change
aecpu(1)	63	aegis -	CoPy file Undo - reverse action of aecp
aeclone(1)		aegis -clone - make an exact	copy of a change
aenbr(1)	131	aegis -New BRanch -	create a new branch
aenpr(1)	154	aegis -New PRoject -	create a new project
aenrls(1)	158	aegis -New ReLeaSe -	create a new project from an old-style project.
aenf(1)	143	aegis -New File - add new files to be	created by a change
aer(5)	254	aer - report script language	definition
aermu(1)	195	aegis -ReMove file Undo - remove files to be	deleted from a change
aerm(1)	189	aegis -ReMove file - add files to be	deleted to a change
aedn(1)	90	aegis -Delta Name - assign a symbolic name to a project	delta
aedn(1)	90	aegis -	Delta Name - assign a symbolic name to a project delta

aedb(1)	72	aegis -	Develop Begin - begin development of a change
aedbu(1)	77	aegis -	Develop Begin Undo - undo the effects of aedb
aede(1)	80	aegis -	Develop End - complete development of a change
aedeu(1)	83	aegis -	Develop End Undo - recall a change for further development
aend(1)	141	aegis -New	Developer - add new developers to a project
aerd(1)	180	aegis -Remove	Developer - remove developers from a project
aerd(1)	180	aegis -Remove Developer - remove	developers from a project
aend(1)	141	aegis -New Developer - add new	developers to a project
aedeu(1)	83	aegis -Develop End Undo - recall a change for further	development
aeclean(1)		aegis -CLEan - clean files from	development directory
aedb(1)	72	aegis -Develop Begin - begin	development of a change
aede(1)	80	aegis -Develop End - complete	development of a change
aed(1)	67	aegis -	DIFFerence - find differences between a change and the baseline
aed(1)	67	aegis -DIFFerence - find	differences between a change and the baseline
aeclean(1)		aegis -CLEan - clean files from development	directory
aecd(1)	45	aegis -Change Directory - change	directory
aecd(1)	45	aegis -Change	Directory - change directory
aefind(1)		aegis - search for files in	directory hierarchy
aedir(5)	228	aedir - aegis	directory structures
aedist(1)	85	aedist - remotely	distribute a change
aerect(1)		aerect -	draw a rectangle
aedbu(1)	77	aegis -Develop Begin Undo - undo the	effects of aedb
aede(1)	80	aegis -Develop	End - complete development of a change
aedeu(1)	83	aegis -Develop	End Undo - recall a change for further development
aeclone(1)		aegis -clone - make an	exact copy of a change
aeif(1)	108	aegis -Integrate	Fail - fail a change integration
aerf(1)	184	aegis -Review	FAIL - fail a change review
aeif(1)	108	aegis -Integrate Fail -	fail a change integration
aerf(1)	184	aegis -Review FAIL -	fail a change review
aecattr(5)	218	aecattr - aegis change attributes	file
aecstate(5)	220	aecstate - aegis change state	file
aefstate(5)	230	aefstate - aegis file state	file
aegstate(5)	236	aegstate - aegis state	file
aepattr(5)	238	aepattr - project attribute	file
aepconf(5)	241	aepconf - aegis project configuration	file
aepstate(5)	249	aepstate - project state	file
aeuconf(5)	271	aeuconf - user configuration	file

aeustate(5)	274	aeustate - user state	file
aerm(1)	189	aegis -ReMove	file - add files to be deleted to a change
aenf(1)	143	aegis -New	File - add new files to be created by a change
aecp(1)	58	aegis -CoPy	file - copy a file into a change
aemv(1)	126	aegis -MoVe	file - rename a file as part of a change
aemv(1)	126	aegis -MoVe file - rename a	file as part of a change
aefp(1)	99	aegis - calculate	file fingerprint
aegis(5)	233	aegis -	file format
aemetrics(5)	237	aemetrics - metrics values	file format
aerptidx(5)	263	aerptidx - report index	file format
aecp(1)	58	aegis -CoPy file - copy a	file into a change
aefstate(5)	230	aefstate - aegis	file state file
aermu(1)	195	aegis -ReMove	file Undo - remove files to be deleted from a change
aenfu(1)	148	aegis -New	File Undo - remove new files from a change
aecpu(1)	63	aegis -CoPy	file Undo - reverse action of aecp
aenfu(1)	148	aegis -New File Undo - remove new	files from a change
aeclean(1)		aegis -CLEan - clean	files from development directory
aefind(1)		aegis - search for	files in directory hierarchy
aenf(1)	143	aegis -New File - add new	files to be created by a change
aermu(1)	195	aegis -ReMove file Undo - remove	files to be deleted from a change
aerm(1)	189	aegis -ReMove file - add	files to be deleted to a change
aed(1)	67	aegis -DIFFerence -	find differences between a change and the baseline
aefp(1)	99	aegis - calculate file	fingerprint
tkaegis(1)		aegis - GUI interface	for Aegis, using TCL/TK
aefind(1)		aegis - search	for files in directory hierarchy
aedeu(1)	83	aegis -Develop End Undo - recall a change	for further development
aegis(5)	233	aegis - file	format
aemetrics(5)	237	aemetrics - metrics values file	format
aerptidx(5)	263	aerptidx - report index file	format
aenfu(1)	148	aegis -New File Undo - remove new files	from a change
aentu(1)	170	aegis -New Test Undo - remove new tests	from a change
aermu(1)	195	aegis -ReMove file Undo - remove files to be deleted	from a change
aencu(1)	138	aegis -New Change Undo - remove a new change	from a project
aera(1)	178	aegis -Remove Administrator - remove administrators	from a project
aerd(1)	180	aegis -Remove Developer - remove developers	from a project
aeri(1)	187	aegis -Remove Integrator - remove integrators	from a project
aerrv(1)	203	aegis -Remove ReViewer - remove reviewers	from a project

aenrls(1)	158	aegis -New ReLeaSe - create a new project	from an old-style project.
aeclean(1)		aegis -CLEan - clean files	from development directory
aedeu(1)	83	aegis -Develop End Undo - recall a change for	further development
aer(1)	176	aegis -RePorT - report	generator
aev(1)	211	aegis -VERSion -	give version information
tkaegis(1)		aegis -	GUI interface for Aegis, using TCL/TK
aefind(1)		aegis - search for files in directory	hierarchy
aefind(1)		aegis - search for files	in directory hierarchy
aerptidx(5)	263	aerptidx - report	index file format
aev(1)	211	aegis -VERSion - give version	information
aeib(1)	101	aegis -	Integrate Begin - begin integrating a change
aeibu(1)	105	aegis -	Integrate Begin Undo - reverse the aeib command
aeif(1)	108	aegis -	Integrate Fail - fail a change integration
aeip(1)	111	aegis -	Integrate Pass - pass a change integration
aeib(1)	101	aegis -Integrate Begin - begin	integrating a change
aeif(1)	108	aegis -Integrate Fail - fail a change	integration
aeip(1)	111	aegis -Integrate Pass - pass a change	integration
aeni(1)	152	aegis -New	Integrator - add new integrators to a project
aeri(1)	187	aegis -Remove	Integrator - remove integrators from a project
aeri(1)	187	aegis -Remove Integrator - remove	integrators from a project
aeni(1)	152	aegis -New Integrator - add new	integrators to a project
ael(1)	116	aegis -List - list (possibly)	interesting things
tkaegis(1)		aegis - GUI	interface for Aegis, using TCL/TK
aecp(1)	58	aegis -CoPy file - copy a file	into a change
aer(5)	254	aer - report script	language definition
ael(1)	116	aegis -	List - list (possibly) interesting things
ael(1)	116	aegis -List -	list (possibly) interesting things
aeclone(1)		aegis -clone -	make an exact copy of a change
aemetrics(5)	237	aemetrics -	metrics values file format
aeca(1)	42	aegis -Change Attributes -	modify the attributes of a change
aepa(1)	173	aegis -Project Attributes -	modify the attributes of a project
aemv(1)	126	aegis -	MoVe file - rename a file as part of a change
ae_p(1)	35	ae p - set project	name
aedn(1)	90	aegis -Delta	Name - assign a symbolic name to a project delta
aedn(1)	90	aegis -Delta Name - assign a symbolic	name to a project delta
aena(1)	129	aegis -	New Administrator - add a new administrator to a project
aena(1)	129	aegis -New Administrator - add a	new administrator to a project
aenbr(1)	131	aegis -New BRanch - create a	new branch
aenbr(1)	131	aegis -	New BRanch - create a new branch
aenbru(1)	133	aegis -	New BRanch Undo - remove a branch
aenc(1)	135	aegis -	New Change - add a new change to a project

aencu(1)	138	aegis -New Change Undo - remove a	new change from a project
aenc(1)	135	aegis -New Change - add a	new change to a project
aencu(1)	138	aegis -	New Change Undo - remove a new change from a project
aend(1)	141	aegis -	New Developer - add new developers to a project
aend(1)	141	aegis -New Developer - add	new developers to a project
aenf(1)	143	aegis -	New File - add new files to be created by a change
aenfu(1)	148	aegis -	New File Undo - remove new files from a change
aenfu(1)	148	aegis -New File Undo - remove	new files from a change
aenf(1)	143	aegis -New File - add	new files to be created by a change
aeni(1)	152	aegis -	New Integrator - add new integrators to a project
aeni(1)	152	aegis -New Integrator - add	new integrators to a project
aenpr(1)	154	aegis -New PRoject - create a	new project
aenpr(1)	154	aegis -	New PRoject - create a new project
aenrls(1)	158	aegis -New ReLeaSe - create a	new project from an old-style project.
aenrls(1)	158	aegis -	New ReLeaSe - create a new project from an old-style project.
aenrv(1)	162	aegis -	New ReViewer - add new reviewers to a project
aenrv(1)	162	aegis -New ReViewer - add	new reviewers to a project
aent(1)	164	aegis -	New Test - add a new test to a change
aent(1)	164	aegis -New Test - add a	new test to a change
aentu(1)	170	aegis -	New Test Undo - remove new tests from a change
aentu(1)	170	aegis -New Test Undo - remove	new tests from a change
ae_c(1)	34	ae c - set change	number
aenrls(1)	158	aegis -New ReLeaSe - create a new project from an	old-style project.
aechown(1)	48	aegis -Change Owner - set change	owner
aechown(1)	48	aegis -Change	Owner - set change owner
ae_p(1)	35	ae	p - set project name
aemv(1)	126	aegis -MoVe file - rename a file as	part of a change
aerpu(1)	201	aegis -Review Pass Undo - rescind a change review	pass
aeip(1)	111	aegis -Integrate	Pass - pass a change integration
aerp(1)	198	aegis -Review	PASS - pass a change review
aeip(1)	111	aegis -Integrate Pass -	pass a change integration
aerp(1)	198	aegis -Review PASS -	pass a change review
aerpu(1)	201	aegis -Review	Pass Undo - rescind a change review pass
ael(1)	116	aegis -List - list (	possibly) interesting things
aena(1)	129	aegis -New Administrator - add a new administrator to a	project

aenc(1)	135	aegis -New Change - add a new change to a	project
aencu(1)	138	aegis -New Change Undo - remove a new change from a	project
aend(1)	141	aegis -New Developer - add new developers to a	project
aeni(1)	152	aegis -New Integrator - add new integrators to a	project
aenpr(1)	154	aegis -New PRoject - create a new	project
aenrls(1)	158	aegis -New ReLeaSe - create a new project from an old-style	project.
aenrv(1)	162	aegis -New ReViewer - add new reviewers to a	project
aepa(1)	173	aegis -Project Attributes - modify the attributes of a	project
aermpr(1)	192	aegis -ReMove PRoject - remove	project
aera(1)	178	aegis -Remove Administrator - remove administrators from a	project
aerd(1)	180	aegis -Remove Developer - remove developers from a	project
aeri(1)	187	aegis -Remove Integrator - remove integrators from a	project
aerrv(1)	203	aegis -Remove ReViewer - remove reviewers from a	project
aenpr(1)	154	aegis -New	PRoject - create a new project
aermpr(1)	192	aegis -ReMove	PRoject - remove project
aepattr(5)	238	aepattr -	project attribute file
aepa(1)	173	aegis -	Project Attributes - modify the attributes of a project
aegis(1)	26	aegis -	project change supervisor
aepconf(5)	241	aepconf - aegis	project configuration file
aedn(1)	90	aegis -Delta Name - assign a symbolic name to a	project delta
aenrls(1)	158	aegis -New ReLeaSe - create a new	project from an old-style project.
ae_p(1)	35	ae p - set	project name
aepstate(5)	249	aepstate -	project state file
aedeu(1)	83	aegis -Develop End Undo -	recall a change for further development
aerect(1)		aerect - draw a	rectangle
aenrls(1)	158	aegis -New	ReLeaSe - create a new project from an old-style project.
aedist(1)	85	aedist -	remotely distribute a change
aenbru(1)	133	aegis -New BRanch Undo -	remove a branch
aencu(1)	138	aegis -New Change Undo -	remove a new change from a project
aera(1)	178	aegis -	Remove Administrator - remove administrators from a project
aera(1)	178	aegis -Remove Administrator -	remove administrators from a project

aerd(1)	180	aegis -	Remove Developer - remove developers from a project
aerd(1)	180	aegis -Remove Developer -	remove developers from a project
aerm(1)	189	aegis -	ReMove file - add files to be deleted to a change
aermu(1)	195	aegis -	ReMove file Undo - remove files to be deleted from a change
aermu(1)	195	aegis -ReMove file Undo -	remove files to be deleted from a change
aeri(1)	187	aegis -	Remove Integrator - remove integrators from a project
aeri(1)	187	aegis -Remove Integrator -	remove integrators from a project
aenfu(1)	148	aegis -New File Undo -	remove new files from a change
aentu(1)	170	aegis -New Test Undo -	remove new tests from a change
aermpr(1)	192	aegis -ReMove PRoject -	remove project
aermpr(1)	192	aegis -	ReMove PRoject - remove project
aerrv(1)	203	aegis -	Remove ReViewer - remove reviewers from a project
aerrv(1)	203	aegis -Remove ReViewer -	remove reviewers from a project
aemv(1)	126	aegis -MoVe file -	rename a file as part of a change
aer(1)	176	aegis -	RePorT - report generator
aer(1)	176	aegis -RePorT -	report generator
aerptidx(5)	263	aerptidx -	report index file format
aer(5)	254	aer -	report script language definition
aerpu(1)	201	aegis -Review Pass Undo -	rescind a change review pass
aecpu(1)	63	aegis -CoPy file Undo -	reverse action of aecp
aeibu(1)	105	aegis -Integrate Begin Undo -	reverse the aeib command
aerf(1)	184	aegis -Review FAIL - fail a change	review
aerp(1)	198	aegis -Review PASS - pass a change	review
aerf(1)	184	aegis -	Review FAIL - fail a change review
aerpu(1)	201	aegis -Review Pass Undo - rescind a change	review pass
aerp(1)	198	aegis -	Review PASS - pass a change review
aerpu(1)	201	aegis -	Review Pass Undo - rescind a change review pass
aenrv(1)	162	aegis -New	ReViewer - add new reviewers to a project
aerrv(1)	203	aegis -Remove	ReViewer - remove reviewers from a project
aerrv(1)	203	aegis -Remove ReViewer - remove	reviewers from a project
aenrv(1)	162	aegis -New ReViewer - add new	reviewers to a project
aet(1)	205	aegis -Test -	run tests
aer(5)	254	aer - report	script language definition
aefind(1)		aegis -	search for files in directory hierarchy
ae_c(1)	34	ae c -	set change number
aechown(1)	48	aegis -Change Owner -	set change owner
ae_p(1)	35	ae p -	set project name
aecstate(5)	220	aecstate - aegis change	state file
aefstate(5)	230	aefstate - aegis file	state file
aegstate(5)	236	aegstate - aegis	state file

aepstate(5)	249	aepstate - project	state file
aeustate(5)	274	aeustate - user	state file
aedir(5)	228	aedir - aegis directory	structures
aenrls(1)	158	aegis -New ReLeaSe - create a new project from an old-	style project.
aesub(5)	264	aesub - available command	substitutions
aegis(1)	26	aegis - project change	supervisor
aedn(1)	90	aegis -Delta Name - assign a	symbolic name to a project delta
tkaegis(1)		aegis - GUI interface for Aegis, using	TCL/TK
aent(1)	164	aegis -New	Test - add a new test to a change
aet(1)	205	aegis -	Test - run tests
aent(1)	164	aegis -New Test - add a new	test to a change
aentu(1)	170	aegis -New	Test Undo - remove new tests from a change
aet(1)	205	aegis -Test - run	tests
aentu(1)	170	aegis -New Test Undo - remove new	tests from a change
ael(1)	116	aegis -List - list (possibly) interesting	things
tkaegis(1)		aegis - GUI interface for Aegis, using TCL/	TK
aedeu(1)	83	aegis -Develop End	Undo - recall a change for further development
aenbru(1)	133	aegis -New BRanch	Undo - remove a branch
aencu(1)	138	aegis -New Change	Undo - remove a new change from a project
aermu(1)	195	aegis -ReMove file	Undo - remove files to be deleted from a change
aenfu(1)	148	aegis -New File	Undo - remove new files from a change
aentu(1)	170	aegis -New Test	Undo - remove new tests from a change
aerpu(1)	201	aegis -Review Pass	Undo - rescind a change review pass
aecpu(1)	63	aegis -CoPy file	Undo - reverse action of aecp
aeibu(1)	105	aegis -Integrate Begin	Undo - reverse the aeib command
aedbu(1)	77	aegis -Develop Begin	Undo - undo the effects of aedb
aedbu(1)	77	aegis -Develop Begin Undo -	undo the effects of aedb
aeuconf(5)	271	aeuconf -	user configuration file
aeustate(5)	274	aeustate -	user state file
tkaegis(1)		aegis - GUI interface for Aegis,	using TCL/TK
aemetrics(5)	237	aemetrics - metrics	values file format
aev(1)	211	aegis -	VERSion - give version information
aev(1)	211	aegis -VERSion - give	version information