mkpkg

Release 4.8.0

Gene C

Nov 17, 2023

CONTENTS:

CHAPTER

ONE

MKPKG

1.1 Overview

Tool to rebuild Arch packages based on dependency triggers.

1.2 New / Interesting

- Version comparisons now use pyalpm instead of packaging packaging.version barfs on systemd package version 255rc2.1

CHAPTER

TWO

MKPKG APPLICATION

2.1 Overview of mkpkg

Building an Arch package requires invoking *makepkg* with a *PKGBUILD* file. PKGBUILD file contains a *depends* variable which lists those packages that are needed to use tool provided by the package.

It also has a 'makedepends' variable which is a list of other packages that are needed to build the package. makepkg also assumes that any package listed in the *depends* variable must also be present to build the package.

However, once a package has been built, then the only thing which causes a rebuild is a change to the actual package version itself. This can be either because the version of the tool itself changed or because the packager manually changed the release version, thereby forcing a rebuild.

If you have ever needed to rebuild a package by manually bumping the release version, then something is not ideal. If something requires a rebuild, other than the package itself having an update, it would be far better if this is done automatically rather than by hand.

This is what mkpkg does. It automates rebuilds when they are needed for some reason other than the tool / package version itself being newer. As a simple example, if something depends on openssl and the last build was against 3.0.0, then it can be set to rebuild if openssl has been installed more recently version than when the last package build was done. It could also be set to rebuild if openssl has a minor version update like 3.1.x.

Triggers are discussed in detail below, *mkpkg-triggers*, but we'll provide a short summary here.

2.2 Triggering Rebuilds Summary

To accomplish this mkpkg allows you to define a set of *triggers* that will cause a rebuild. These are packages, or files, that trigger a rebuild whenever they change in a specified way. Straightforward concept.

The packager is responsible for providing the list of appropriate triggers.

The way to provide the these triggers is by adding a PKGBUILD array variable to provide the list of conditions that should trigger a rebuild.

2.2.1 Package Trigger

_mkpkg_depends variable is this list of such triggers

There are 2 ways a package can trigger a rebuild.

1. **A package name**

   Rebuilds if the install date is more recent than the last build time e.g.

   ```
   _mkpkg_depends=('openssl' 'systemd')
   ```

2. **A package name plus a version condition**

   It can be an explicit version or a key word such as *major* which would then only trigger a rebuild when the major version of that package was greater than that at the last build. More details and the different options are detailed below. e.g.

   ```
   _mkpkg_depends=('openssl>3.0.6' 'systemd')
   ```

2.2.2 File Trigger

It can also use any file to trigger a build using *_mkpkg_depends_file*. When a file in this list is newer than the lsat build, it triggers a rebuild.

A typical use case for these file triggers are files provided by the packager, rather than the source, and include things such as systemd unit files or pacman hook files or other package related items. e.g.:

```
_mkpkg_depends_file=('xxx.service')
```

This is useful to ensure packages build and work when conditions are met by other packages being updated.

It is certainly helpful for packages which statically link in libraries, or when core build tools change and it's important to rebuild with the newer versions. Do we really need to rebuild a package when tool chain changes? Sometimes yes; for example, whenever the compiler toolchain is updated, I always rebuild my kernel packages and test.

The majority of compiled packages are built against shared libraries and this can be helpful in this case too; there are additional comments on this topic below.

As another example, I rebuild my python applications when python's major.minor is larger than what was used for previous build.

An additional little benefit, if packages are up to date then running mkpkg is significantly faster than makepkg; can be something like 10x faster or even more.

2.3 Background Motivation

mkpkg has one run-time dependency, python.

It uses makepkg to perform the actual package builds in the usual way. That said, makepkg is a part of pacman which is always installed and thus not a *dependency* as far as PKGBUILD is concerned.

When a tool chain used to build a package is updated, it's good practice, IMHO, to rebuild packages which use that tool chain. For example, when gcc, cargo, binutils et al are updated packages using those tools should also be updated. As mentioned above, whenever compiler/binutils tool chain changes, I always rebuild and test my kernel packages. This not only ensures that things compile and work properly with the new build tools but can also be key to reducing the attack surface. One recent (as of time of writing) little example, not to pick on cargo, is CVE-2022-36113

Of course this would require a case where cargo is actually downloading something which should never be permitted; still, it's a conceivable danger.

While static linked libraries surely don't demand a rebuild to function, obviously, because the older library is part of the binary itself, it's still a good idea to rebuild it. This will pick up bug fixes, including security related ones, as well as improvements. Of course, it's always sensible to confirm that an application properly builds and works with the newer tool or library as well.

Here's an example. The *refind* boot manager statically links against gnu-efi. So when gnu-efi is updated, refind should be rebuilt as well even though the previous one will continue to work just fine.

Recently, arch started switching many packages to be compiled with lto. The gnu-efi package was subsequently compiled with * -flto -ffat-lto-object*. The refind boot manager statically links gnu-efi. At this point, refind itself had not changed and so it's up to date as far standard approach is concerned.

However, I would like to know as early as possible that refind builds and runs with the the new gnu-efi library that was updated. In fact, unfortunately perhaps, this build failed and refind not longer builds with the updated gnu-efi library due to lto changes. Good to know.

You could of course have waited until refind itself gets an update and then discover - oh no it no longer builds. But, by doing this early and in this case knowing refind itself has not changed, I know with certainty that this problem stems from the gnu-efi rebuild and not from a refind change - without even looking at any refind source changes.

Given the large number of packages I build I doubt I'd remember what trigger packages are approprate for every package anyway. Computers are good at automating repetitive tasks after all and are much quicker at identifying the trigger packages.

mkpkg was created to address this need. It automates this for you and rebuilds packages when needed. This allows for early detection of problems or confirmation that things are actually fine.

A small comment on shared libraries. While these are generally not a problem, there is an assumption that the library itself still functions the same for whatever part of it the tool is using.

The majority of providers are careful with *sonames* as well, so most of the time that's likely true, however, the cautious among us may want to run regression tests even in this case.

Certainly for mission critical tools. Bugs happen, and it's good to learn of any issues as soon as possible.

But there are indeed some shared library packages, some with dynamically loaded libraries (plugins) that may also be trigger packages. One symptom of that need are those packages that are manually rebuilt by forcing a release version bump typically with a comment such as *rebuilt with latest ...* - we certainly see plenty of that happening.

CHAPTER

THREE

USING MKPKG

3.1 Getting Started

Edit the PKGBUILD and add a *_mkpkg_depends* variable with a list of triggers that should cause a rebuild when the condition is met. Triggers are discussed in in detail (*mkpkg-triggers*) below, but a simple example is:

```
_mkpkg_depends=('python>major', 'python-foo')
```

This would trigger a package rebuild if a version of *python-foo* is installed more recently than the last package build or if *python* has a major version which is larger than that used when package was last built.

With the trigger conditions in the PKGBUID, then simply call mkpkg instead of makepkg. Couldn't be simpler. Options for mkpkg are those before any double dash –. Any options following – are passed through to *makepkg*[3].

3.2 Options

The options currently supported by mkpkg are:

- (**-v, –verb**)

  Show (stdout) output of makepkg. Default is not to show it.

- (**-f, –force**)

  Force a makepkg run even if not needed. Bump the package release and rebuild

- (**-r, –refresh**)

  Attempts to update saved metadata files. Faster, if imperfect, alternative to rebuild. If there is no saved metadata, and build is up to date, will try refresh the build info. Files updated are *.mkp_dep_vers* and *.mkp_dep_soname*. The soname data can only be updated if the .PKGINFO file is still in the *pkg* directory. Forcing a rebuild is the slower alternative but is guaranteed to have information needed.

- (–)

  All options following this are passed to makepkg

Config file

Configs are looked for in first in /etc/mkpkg/config and then in ~/.config/mkpkg/config. Config files are in TOML format. e.g. to change the default soname rebuild option:

[3] The older style options using *–mkp-* are deprecated but are supported to ensure backward compatibility. They will be removed at some point in the future.

```
soname_build = "newer"
```

3.3 How mkpkg works

Outline of what it does

- If PKGBUILD has a pkgver() function, check if the pkgver variable matches its output
- If the 2 pkgver match or if there is no pkgver() function then check if a matching package exists
- If package not up to date, then run makepkg build.
- If package seems otherwise up to date, then check if any of the conditions given by *mkpkg_depends* or *mkpkg_depends_files* triggers a build. If a build is called for, then bump the pkgrel and rebuild.
- If the package is out of date, as there is newer version then reset pkgrel back to "1" and build.

So, if a package builds and gets larger package release number, it was because of some trigger package dependency; absent manual modification. If package release is "1" - then you know its a fresh package version.

I use separate tool to run all my package builds so I prefer the output to be easily parseable and provide simple and clear information to feed the builder too.

mkpkg thus prints a line of the form:

```
*mkp-status: <status> <package-version>*
```

Where status is one of :

- **current** -> package is up to date
- **success** -> package was built successfully
- **error** -> problem occurred.

Obviously, package-version is what is sounds like.

It is possible for mkpkg itself to fail for some reason, in which case the *mkp-status:* line could be absent. This is also simple to detect programatically.

3.4 Triggering Rebuilds Details

3.4.1 _mkpkg_depends

There are 2 kinds of triggers. A trigger based on package and a trigger based on file changed. Each is set using the PKGBUILD variable with a an array of triggers. The variables used are:

- **_mkpkg_depends**

This variable provides a list of packages to trigger a rebuild. Each item in the list can be in one of 2 forms:

1. *name*

   The item is the name of the package then this will trigger a rebuild if the install time of a listed package is newer than the time of the last build.

2. *package_name compare-op vers_trigger*

   This provides semantic version triggers. Package versions are taken to be of the form 'major.minor.patch' or more generally 'elem1.elem2.elem3...' White space around the comparison operator is optional.

- *compare-op*

  is one of : **>**, **>=** or **<**

- *vers_trigger*

  Based on comparing the first [N] elems of the version or the entire version.

  - First_[N] : rebuild if first [N] elems of package version greater than when last built
  - major : alias for First_1 (rebuild if major > last_build)
  - minor : alias for First_2 (rebuild if major.minor > last_build)
  - patch : alias for First_3 (if major.minor.patch > last_build)
    - micro : another name for patch
  - extra : alias for First_4 (major.minor.patch.extra)
    - releaselevel : alias for extra
  - serial : alias for First_5 (major.minor.patch.extra.serial)
  - last : rebuild if package version > last_build version.

last is very similar to a time based trigger but based on version instead of time.

For example if the expression is

```
'pkg_name>First_2'
```

or equivalently:

```
'pkg_name>minor'
```

and the current package version is 1.2.3, while the version when last built was 1.2.0 then the versions being compared would be

```
'1.2' > '1.2' which is false.
```

Whereas if the expression was:

```
'pkg_name>First_3'
```

then the comparison would be

```
'1.2.3' > '1.2.0'
```

which is true

N.B. The package must be built at least once using mkpkg so it can save the dependent package versions used. So if a version trigger is added, then this triggers a rebuild as it treats this as if the dependent package version is greater than last used (which is not known at this point). On subsequent builds the last built version of each dependent package is then known.

Unlike the standard *makedepends* variable, this allows one to not include things that are required to build the package but don't have any affect on the tool function. For example 'git' - which while required to build will not generally change the tool.

Another example, if python was version 3.10 when the package was last built and we have::

```
_mkpkg_depends=('python>minor' 'python-dnspython')
```

Then a rebuild will be done if python is greater than or equal to 3.11.x or if python-dnspython was installed more recently than the last build. This will not trigger a rebuild if python is updated from 3.10.7 to 3.10.8, since this is a patch update not a minor or major update.

Why support '<' you may ask. The only sensible use for less than operator would be to provide a mechanism to trigger a rebuild when a package gets downgraded. This would be accomplished using

```
pkg_name < last
```

3.4.2 _mkpkg_depends_files

- *_mkpkg_depends_files*

  This variable can be used to provide a list of files that should trigger a rebuild. The files are relative to the directory containing PKGBUILD.

This might be useful, for example, if the source for some daemon doesn't provide a systemd service file, and the packager adds the file. Adding the file to this list would now trigger rebuilds should there be changes to the service file. An alternative would be to put these files into a git repo and just using the git version. For a small number of files this may be more convenient/simpler.

These variables offer considerable control over what can be used to trigger rebuilds.

3.5 Discussion and Next Steps

3.5.1 Possible future enhancement

While mkpkg works for all the packages I build, I am more than happy to take enhancement requests - and, of course, to fix bugs!

As mentioned earlier, it's pretty useful to run regression tests after run-time dependencies change. For example shared libraries or other programs used by the tool. To handle this case we might consider adding a separate variable - such as *mkpkg_test_depends* which lists these kind of dependencies.

We note that *checkdepends* vartiable is quite different in intent, as it is used to identify those packages needed to do testing but NOT for things which could impact the outcome of running the tool.

CHAPTER

FOUR

APPENDIX

4.1 mkpkg Source

The source is kept in the github repository Github-mkpkg.

4.2 Installation

Available on

- Github-mkpkg
- Archlinux AUR

On Arch you can build using the provided PKGBUILD in the packaging directory or from the AUR. To build manually, clone the repo and :

```
rm -f dist/*
/usr/bin/python -m build --wheel --no-isolation
root_dest="/"
./scripts/do-install $root_dest
```

When running as non-root then set root_dest a user writable directory

4.3 Dependencies

- Run Time: - python (3.9 or later) - pyalpm
- Building Package : - git - build aka python-build - intaller aka python-installer - wheel aka python-wheel - poetry aka python-poetry - rsync
- Optional for building docs:
  - sphinx
  - texlive-latexextra (archlinux packaguing of texlive tools)

4.4 Philosophy

We follow the *live at head commit* philosophy. This means we recommend using the latest commit on git master branch. We also provide git tags.

This approach is also taken by Google[1][2].

4.5 License

Created by Gene C. and licensed under the terms of the MIT license.

- SPDX-License-Identifier: MIT
- Copyright (c) 2022-2023 Gene C

4.6 Some history

4.6.1 Version 4.1.0

- Arguments

  Change in argument handling. Arguments to be passed to *makepkg* must now follow –. Arguments before the double dash are used by mkpkg itself. To keep backward compatibility the older *–mkp-* style arguments are honored, but the newer simpler ones are preferred. e.g. *-v, –verb* for verbose. Help availble via *-h*.

- New argument for how soname changes are treated : *-so-bld, –soname-build.*

  Can be *missing*, *newer* or *never*. Default is missing - rebuild if soname no longer available.

- Config file now available.

  Configs are looked for in /etc/mkpkg/config then ~/.config/mkpkg/config. It should be in TOML format. e.g. to change the default soname rebuild option:

```
soname_build = "newer"
```

4.6.2 Version 4.0.0

- Soname drive rebuilds.

  Adds support for detecting missing soname libraries, and triggering rebuild. If soname is found then no rebuild is done. Typically happens when older soname is deprecated.

- Adds new option *–mkp-refresh*.

  Attempts to update saved metadata files. Faster, if imperfect, alternative to rebuild.

[1] https://github.com/google/googletest
[2] https://abseil.io/about/philosophy#upgrade-support

4.6.3 Older

Adds support for epoch.

Version 2.x.y brings fine grain control by allowing package dependences to trigger builds using semantic version. For example 'python>minor' will rebuild only if a new python package has it's major.minor greater than what it was when package was last built. See *_mkpkg_depends* below for more detail.

The source has been reorganized and packaged using poetry which simplifies installation. The installer script, callable from package() function in PKGBUILD has been updated accordingly. Ther build() function uses python build module to generate the wheel package, as outlined above.

Changed the PKGBUILD variables to have underscore prefix to follow Arch Package Guidelines. Variables are now: *_mkpkg_depends* and *_mkpkg_depends_files*. The code is backward compatible and supports the previous variable names without the leading "_" as well as the ones with the "_".

To fall back to *makedepends* when there are no *_mkpkg_depends* variables now requires using the option *–mkp-use_makedepends* to turn it on.

Now also available on aur.

CHAPTER

FIVE

CHANGELOG

[4.8.0] —— 2023-11-17

- Change to using pyalpm to compare package versions instead of packaging.
- ing.version() barfs on systemd version 255rc2.1 for some reason
- update Docs/Changelog.rst for 4.7.0

[4.7.0] —— 2023-10-03

- Bug fix semantic version comparisons
- Stop treating Arch pkgrel as part of the last version element - its separate additional element
- update Docs/Changelog.rst for 4.6.0

[4.6.0] —— 2023-09-28

- Reorganize the tree and documents.
- Switch from markdown to restructured text.
- Now easy to build html and pdf docs using sphinx
- update CHANGELOG.md for 4.5.5

[4.5.5] —— 2023-06-05

- Small tweak to README
- update CHANGELOG.md for 4.5.4

[4.5.4] —— 2023-05-18

- Change PKGBUILD makedepnds from pip to installer
- update CHANGELOG.md for 4.5.3

[4.5.3] —— 2023-05-18

- install: switch from pip to python installer package. This adds optimized bytecode
- update CHANGELOG.md for 4.5.2

[4.5.2] —— 2023-05-18

- PKGBUILD: build wheel back to using python -m build instead of poetry
- update CHANGELOG.md for 4.5.1

[4.5.1] —— 2023-05-17

- Simplify Arch PKGBUILD and more closely follow arch guidelines

- update CHANGELOG.md for 4.5.0

[4.5.0] —— 2023-02-19

- Fix bug when soname dependency drives rebuild by ensuring pkgrel is bumped
- update CHANGELOG.md for 4.4.0

[4.4.0] —— 2023-02-18

- Bug fix extracting PKGBUILD info for certain cases
- update CHANGELOG.md for 4.3.0

[4.3.0] —— 2023-01-31

- Force now bumps the package release and rebuilds
- update CHANGELOG.md for 4.2.1

[4.2.1] —— 2023-01-06

- Add SPDX licensing lines
- Lint and tidy
- update CHANGELOG.md for 4.2.0

[4.2.0] —— 2023-01-03

- Fix for potential color name match bug - not with current color sets
- update CHANGELOG.md for 4.1.1

[4.1.1] —— 2022-12-16

- Add toml dependency to PKGBUILD
- update CHANGELOG.md for 4.1.0

[4.1.0] —— 2022-12-16

- Add config file support.
- Change option handling. Options to be passed to makepkg must now be placed after –
- Improveed soname treatment via option –soname-build (missing (default), newer or never)
- update CHANGELOG.md

[4.0.0] —— 2022-12-15

- Add –mkp-refresh
- Attempts to update saved metadata files. Faster, if imperfect, alternative to rebuild.
- refactor some code
- pull out pacman queries to more easily share
- Add suport for missing soname library driving rebuild
- suggestion thanks to Alberto Novella Archlinux subredit.
- update CHANGELOG.md

[3.5.4] —— 2022-11-29

- Small change to README.
- Change variable check in installer (no functional change)

- update CHANGELOG.md

[3.5.3] —— 2022-11-05

- tweak readme
- installer script change list to bash array for apps being installed. zero impact
- update CHANGELOG.md

[3.5.2] —— 2022-11-04

- PKGBUILD - duh - put back makedepends on poetry
- update CHANGELOG.md

[3.5.1] —— 2022-11-04

- Add package name to screen message
- update CHANGELOG.md

[3.5.0] —— 2022-11-03

- bug fix incorrectly handling triggers pkg>xxx
- update CHANGELOG.md

[3.4.0] —— 2022-11-03

- Better handling of PKGBUILD syntax errors
- update CHANGELOG.md

[3.3.1] —— 2022-11-03

- unwind prev error check - needs more work
- update CHANGELOG.md

[3.3.0] —— 2022-11-03

- Additional check for errors when sourcing PKGBUILD
- update CHANGELOG.md

[3.2.0] —— 2022-10-31

- typo - so sorry
- update CHANGELOG.md

[3.1.0] —— 2022-10-31

- Add more aliases of First_N for version comparisons (micro, serial)
- Change build from poetry/pip to python -m build/installer
- update CHANGELOG.md

[3.0.0] —— 2022-10-30

- update CHANGELOG.md
- Add epoch support - needs wider testing
- update changelog

[2.5.0] —— 2022-10-26

- bug fix for _mkpkg_depends_files - silly typo

- CHANGELOG.md

[2.4.1] —— 2022-10-24

- update pyproject.toml vers
- update changelog

[2.4.0] —— 2022-10-24

- oops - accidently left debugger on!
- update changelog

[2.3.6] —— 2022-10-24

- Fix bug parsion <package> >= xxx. Greater than is fine.
- update changelog

[2.3.5] —— 2022-10-23

- avoid all but tag in pkgver()
- update pyproject.toml vers
- update changelog

[2.3.4] —— 2022-10-23

- PKGBUILD - remove tag= now that pgkver() is getting latest tag

[2.3.3] —— 2022-10-23

- PKGBUILD now builds latest release tag
- update changelog
- Add comment about being fast
- update changelog

[2.3.2] —— 2022-10-14

- Improve PKGBUILD for aur as per comments
- update pyproject.toml version
- Clean the dist directory before doing poetry build
- fix python depends version > 3.9
- Add makedepends packages in aur PKGBUILD
- fix comment
- add aur comment
- update changelog

[2.3.1] —— 2022-10-13

- Update readme with link to AUR for mkpkg
- Change PKGBUILD for AUR
- little word smithing on readme
- Clean up some comments
- readme word smithing

- update changelog

[2.3.0] —— 2022-10-13

- In the event mkpkg_depends / mkpkg_depends_files are absent,
- no longer fall back to use makedepends unless turned on with the –mkp-use_makedepends option
- update changelog

[2.2.1] —— 2022-10-13

- Bug fix for _mkpkg_depends_files
- better packge description in PKGBUILD
- readme markdown missed 2 spaces for newline
- Readme - markdown requires escape for underscore
- update CHANGELOG.md

[2.2.0] —— 2022-10-13

- Change PKGBUILD variables to have leading "_" to follow arch packaging guidelines
- Code is backward compatible and will work with or without the _
- New names are: _mkpkg_depends and _mkpkg_depends_files
- update changelog
- more readme tweaks
- update changelog

[2.1.1] —— 2022-10-13

- Provide sample PKGBUILD to build mkpkg
- update changelog
- typo in readme
- update changelog
- README tweak to explain "patch" being same as "First_3" for version triggers
- update CHANGELOG.md

[2.1.0] —— 2022-10-13

- Enhance version triggers to handle version with more than 3 elements
- update changelog
- readme tweaks
- update CHANGELOG

[2.0.1] —— 2022-10-12

- update changelog
- remove unused from do-install
- update CHANGELOG
- tweak readme
- update changelog

[2.0.0] —— 2022-10-12

- Reorganize directory structure and use poetry for packaging.
- Add support for triggers now based on semantic versions.
- e.g python>3.12 or python>minor - where minor triggers build if
- major.minor version of dependency package is greater than that used when
- it was last built.
- Reorganize source tree
- Update changelog
- tweak readme little more
- update Changelog
- Tweak README
- tweak README

[1.3.1] —— 2022-09-22

- Update Changelog
- Add CVE-2022-36113 as example of build tool danger
- Update Changelog
- Add Changelog

[1.3.0] —— 2022-09-07

- fix out of date comment in mkpkg.py
- fix little markdown issue
- tweak readme format

[1.2.0] —— 2022-09-06

- Add support for trigger files : mkpkg_depends_files
- add README discssion comment
- lint picking
- Add comment in README
- few more README tweaks

[1.1.1] —— 2022-09-04

- tidy message output
- typo
- Little tidy on README

[1.1.0] —— 2022-09-04

- Handle edge case when PKGBUILD hand edited
- Bug fix for case when override mkpkg_depends set to empty set

[1.0.5] —— 2022-09-03

- Now that we implemented mkpkg_depends, remove some readme comments

- typo
- minor README tweak
- Fix typo (resolves issue #1) and tweak README

[1.0.4] —— 2022-09-03

- fix section numbers in README

[1.0.3] —— 2022-09-03

- Support mkpkg_depends overriding makepends - gives full control to user

[1.0.2] —— 2022-09-03

- README use lower case for mkpkg

[1.0.1] —— 2022-09-03

- Tidy couple comments

[1.0.0] —— 2022-09-03

- Initial Revision of mkpkg.
- mkpkg builds Arch packages and rebuilds them whenever a make dependency is more recent than the last package

CHAPTER

SIX

MIT LICENSE

CHAPTER

SEVEN

HOW TO HELP WITH THIS PROJECT

Thank you for your interest in improving this project. This project is open-source under the MIT license.

7.1 Important resources

- Git Repo

7.2 Reporting Bugs or feature requests

Please report bugs on the issue tracker in the git repo. To make the report as useful as possible, please include

- operating system used
- version of python
- explanation of the problem or enhancement request.

7.3 Code Changes

If you make code changes, please update the documentation if it's appropriate.

CHAPTER

EIGHT

CONTRIBUTOR COVENANT CODE OF CONDUCT

8.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

8.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

8.3 Our Responsibilities

Maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

8.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

8.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at <arch@sapience.com>. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The Code of Conduct Committee is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

8.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

8.7 Interpretation

The interpretation of this document is at the discretion of the project team.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search