

# Recommendations from the Gentoo Working Group on improving the state of the stable tree

Gentoo Linux

August 18, 2016

## Contents

<b>1</b>	<b>Mandate</b>	<b>1</b>
1.1	Participants . . . . .	1
<b>2</b>	<b>Definitions</b>	<b>1</b>
<b>3</b>	<b>How do we want the stable tree to function</b>	<b>1</b>
<b>4</b>	<b>Current state of the stable tree</b>	<b>1</b>
<b>5</b>	<b>What methods can we use to migrate between current and wanted stable states</b>	<b>2</b>
5.1	Targeted architectures . . . . .	2
5.2	Stabilisation process . . . . .	2
5.3	Stable tree freshness . . . . .	2
<b>6</b>	<b>Recommendations</b>	<b>2</b>

## 1 Mandate

This is the final report of the Gentoo Working Group established up to come up with recommendations to the Gentoo Council on improving the state of the stable tree

### 1.1 Participants

- k.f (chair)
- blueness
- dilfridge
- rich0
- pacho
- mjo
- kentnl
- kensington

## 2 Definitions

**Architecture** Supported Gentoo platform, as described in the first field of profiles.desc

**Stable architecture** Architecture that has at least one stable profile, and a maintaining project that determines stable keywords are to be used for that architecture.

**Stable profile** Profile with a “stable” status in profiles.desc. This implies ensure dependency graph consistency, and not necessarily any stable keywords.

## 3 How do we want the stable tree to function

Maintaining a stable tree is important for providing a known-good option for those who wish to use Gentoo in a production environment or simply to avoid the potential volatility of always using the “latest and greatest”.

There is a careful balance to maintain between length of time in testing and ebuild freshness – too little time in testing and we risk bugs in stable, too much and we risk losing relevance due to excessively outdated ebuilds.

## 4 Current state of the stable tree

There are currently forty-six supported architectures, with nine of those architectures considered stable. Each architecture is maintained by a team consisting of a varying number of developers, but in practice between zero and two developers are actively working on stabilisation requests on any given architecture.

Architecture	Outstanding stable requests
alpha	41
amd64	265
arm	45
hppa	149
ia64	209
ppc	194
ppc64	180
sparc	251
x86	310

Table 1: Outstanding stable requests per arch as of 2016-08-17

A common developer complaint is the delay between filing a stabilisation request and it being actioned by an arch team. This can block the maintainer from removing old versions that contain bugs, security issues, or that are simply no longer maintained.

A common user complaint is that the latest stable version of package is old. Many developers run full  $\sim arch$  systems and forget to or simply do not bother filing stabilisation requests.

## 5 What methods can we use to migrate between current and wanted stable states

### 5.1 Targeted architectures

We need to evaluate the architectures we currently target for stabilisation and consider their ongoing relevance, hardware availability, general interest, and targeted audience. Options for any given architecture may include:

- Switching the profile to dev/exp and leaving keywords intact (this option has previously been used as the result of a Council decision)
- Removing all stable keywords while retaining a stable profile (to keep dependency graph consistency)
- Remove stable keywords from all non-core packages
- Review the suitability of stable keywords with each new stable request, dropping as necessary (this option has previously been self-initiated by arch teams)

### 5.2 Stabilisation process

Once we have determined which architectures we wish to target for stabilisation, we must decide how we want to handle the stabilisation process. Options to consider may include:

- Waiting period in testing (30 days)
- Formalising the ALLARCHES policy – better document suitable types of packages (dictionaries, man pages, pure non-compiled language packages...)
- Encourage stabilisation to be performed by maintainers on architectures they have access to – better document exactly what should be checked
- Once a runtime test has been performed on one architecture, stabilise on other architectures with only a build test
- Perform build-only testing, assuming that the 30 day waiting period will smoke out any runtime issues

Note that while these options may seem relaxed compared to the current “official” policy, in reality many stabilisations are currently performed with build-only testing or trivial runtime testing. We also assume the package maintainer is the best qualified person to determine if a package is suitable to be stabilised or not.

### 5.3 Stable tree freshness

Many packages have old stable versions simply because nobody every filed a stabilisation request. We should encourage maintainers to regularly review their packages and make considered decisions about what should be stabilised. Again, it is important to maintain the balance between stability and freshness – a package that sees weekly releases likely doesn’t need every version stabilised, but having a 5-year-old package in stable with bugs fixed in *~arch* isn’t helpful either. To assist, we could consider:

- Reinstating automated stabilisation requests/reminders
- Producing per-maintainer stabilisation candidate lists (eg. imlate output, with a link to open a pre-filled stabilisation bug)
- Destabilising packages that don’t make sense in stable

## 6 Recommendations

lorem ipsum...

## References

- [1] Announcement of the Gentoo Working Group: <https://archives.gentoo.org/gentoo-dev/message/9cc27b25e7e1742653ffeaf77ada5c18>