Problem Report Handling Guidelines With Bugzilla

Abstract

These guidelines describe recommended handling practices for FreeBSD Problem Reports (PRs) as submitted via Bugzilla. Whilst developed for the FreeBSD Bugzilla Database Maintenance Team freebsd-bugbusters@FreeBSD.org, these guidelines should be followed by anyone working with FreeBSD Problem Reports.

Table of Contents

1. Introduction . . . 1
2. Problem Report Life-cycle . . . 1
3. Problem Report State . . . 2
4. Types of Problem Reports . . . 3
5. Unassigned PRs . . . 3
6. Assigned PRs . . . 8
7. Duplicate PRs . . . 8
8. Stale PRs . . . 8
9. Non-Bug PRs . . . 9
10. Further Reading . . . 9

1. Introduction

Bugzilla is an issue management system used by the FreeBSD Project. As accurate tracking of outstanding software defects is important to FreeBSD's quality, the correct use of the software is essential to the forward progress of the Project.

Access to Bugzilla is available to the entire FreeBSD community. In order to maintain consistency within the database and provide a consistent user experience, guidelines have been established covering common aspects of bug management such as presenting followup, handling close requests, and so forth.

Note: in this Article, the term "PR" means "Bugzilla Problem Report".

2. Problem Report Life-cycle

- The Reporter submits a bug report on the website. The bug is in the `Needs Triage` state.
- Jane Random BugBuster confirms that the bug report has sufficient information to be

reproducible. If not, she goes back and forth with the reporter to obtain the needed information. At this point the bug is set to the `Open` state.

- Joe Random Committer takes interest in the report and assigns it to himself, or Jane Random BugBuster decides that Joe is best suited to handle it and assigns it to him. The bug should be set to the `In Discussion` state.
- Joe has a brief exchange with the originator (making sure it all goes into the audit trail) and determines the cause of the problem.
- Joe pulls an all-nighter and whips up a patch that he thinks fixes the problem, and submits it in a follow-up, asking the originator to test it. He then sets the report's state to `Patch Ready`.
- A couple of iterations later, both Joe and the originator are satisfied with the patch, and Joe commits it to `-CURRENT` (or directly to `-STABLE` if the problem does not exist in `-CURRENT`), making sure to reference the Problem Report in his commit log (and credit the originator if they submitted all or part of the patch) and, if appropriate, start an MFC countdown. The bug is set to the `Needs MFC` state.
- If the patch does not need MFCing, Joe then closes the report as `Issue Resolved`.

Many reports are submitted with very little information about the problem, and some are either very complex to solve, or just scratch the surface of a larger problem; in these cases, it is very important to obtain all the necessary information needed to solve the problem. If the problem contained within cannot be solved, or has occurred again, it is necessary to re-open the report.

3. Problem Report State

It is important to update the state of a Problem Report when certain actions are taken. The state should accurately reflect the current state of work on the issue.

Example 1. A small example on when to change PR state

When a PR has been worked on and the developer(s) responsible feel comfortable about the fix, they will submit a followup to the PR and change its state to "feedback". At this point, the originator should evaluate the fix in their context and respond indicating whether the defect has indeed been remedied.

A Problem Report may be in one of the following states:

open

Initial state; the problem has been pointed out and it needs reviewing.

analyzed

The problem has been reviewed and a solution is being sought.

feedback

Further work requires additional information from the originator or the community; possibly information regarding the proposed solution.

patched

A patch has been committed, but something (MFC, or maybe confirmation from originator) is still pending.

suspended

The problem is not being worked on, due to lack of information or resources. This is a prime candidate for somebody who is looking for a project to take on. If the problem cannot be solved at all, it will be closed, rather than suspended. The documentation project uses suspended for wish-list items that entail a significant amount of work which no one currently has time for.

closed

A problem report is closed when any changes have been integrated, documented, and tested, or when fixing the problem is abandoned.

The "patched" state is directly related to feedback, so you may go directly to "closed" state if the originator cannot test the patch, and it works in your own testing.

4. Types of Problem Reports

While handling problem reports, either as a developer who has direct access to the Problem Reports database or as a contributor who browses the database and submits followups with patches, comments, suggestions or change requests, you will come across several different types of PRs.

- Unassigned PRs
- Assigned PRs
- Duplicate PRs
- Stale PRs
- Non-Bug PRs

The following sections describe what each different type of PRs is used for, when a PR belongs to one of these types, and what treatment each different type receives.

5. Unassigned PRs

When PRs arrive, they are initially assigned to a generic (placeholder) assignee. These are always prepended with `freebsd-`. The exact value for this default depends on the category; in most cases, it corresponds to a specific FreeBSD mailing list. Here is the current list, with the most common ones listed first:

Table 1. Default Assignees - most common

Type	Categories	Default Assignee
base system	bin, conf, gnu, kern, misc	freebsd-bugs

Type	Categories	Default Assignee
architecture-specific	arm, amd64, i386, mips	freebsd-*arch*
architecture-specific: powerpc	powerpc64	freebsd-ppc
architecture-specific: riscv64	riscv64	freebsd-risc
ports collection	ports	freebsd-ports-bugs
documentation shipped with the system	docs	freebsd-doc
FreeBSD web pages (not including docs)	Website	freebsd-www

Table 2. Default Assignees - other

Type	Categories	Default Assignee
advocacy efforts	advocacy	freebsd-advocacy
Java Virtual Machine™ problems	java	freebsd-java
standards compliance	standards	freebsd-standards
threading libraries	threads	freebsd-threads
usb(4) subsystem	usb	freebsd-usb

Do not be surprised to find that the submitter of the PR has assigned it to the wrong category. If you fix the category, do not forget to fix the assignment as well. (In particular, our submitters seem to have a hard time understanding that just because their problem manifested on an i386 system, that it might be generic to all of FreeBSD, and thus be more appropriate for `kern`. The converse is also true, of course.)

Certain PRs may be reassigned away from these generic assignees by anyone. There are several types of assignees: specialized mailing lists; mail aliases (used for certain limited-interest items); and individuals.

For assignees which are mailing lists, please use the long form when making the assignment (e.g., `freebsd-foo` instead of `foo`); this will avoid duplicate emails sent to the mailing list.

Since the list of individuals who have volunteered to be the default assignee for certain types of PRs changes so often, it is much more suitable for the FreeBSD wiki.

Here is a sample list of such entities; it is probably not complete.

Table 3. Common Assignees - base system

Type	Suggested Category	Suggested Assignee	Assignee Type
problem specific to the ARM® architecture	arm	freebsd-arm	mailing list

Type	Suggested Category	Suggested Assignee	Assignee Type
problem specific to the MIPS® architecture	kern	freebsd-mips	mailing list
problem specific to the PowerPC® architecture	kern	freebsd-ppc	mailing list
problem with Advanced Configuration and Power Management (acpi(4))	kern	freebsd-acpi	mailing list
problem with embedded or small-footprint FreeBSD systems (e.g., NanoBSD/PicoBSD/FreeBSD-arm)	kern	freebsd-embedded	mailing list
problem with FireWire® drivers	kern	freebsd-firewire	mailing list
problem with the filesystem code	kern	freebsd-fs	mailing list
problem with the geom(4) subsystem	kern	freebsd-geom	mailing list
problem with the ipfw(4) subsystem	kern	freebsd-ipfw	mailing list
jail(8) subsystem	kern	freebsd-jail	mailing list
problem with Linux® or SVR4 emulation	kern	freebsd-emulation	mailing list
problem with the networking stack	kern	freebsd-net	mailing list
problem with the pf(4) subsystem	kern	freebsd-pf	mailing list
problem with the scsi(4) subsystem	kern	freebsd-scsi	mailing list
problem with the sound(4) subsystem	kern	freebsd-multimedia	mailing list
problems with the wlan(4) subsystem and wireless drivers	kern	freebsd-wireless	mailing list
problem with sysinstall(8) or bsdinstall(8)	bin	freebsd-sysinstall	mailing list

Type	Suggested Category	Suggested Assignee	Assignee Type
problem with the system startup scripts (rc(8))	kern	freebsd-rc	mailing list
problem with VIMAGE or VNET functionality and related code	kern	freebsd-virtualization	mailing list
problem with Xen emulation	kern	freebsd-xen	mailing list

Table 4. Common Assignees - Ports Collection

Type	Suggested Category	Suggested Assignee	Assignee Type
problem with the ports framework (*not* with an individual port!)	ports	portmgr	alias
port which is maintained by apache@FreeBSD.org	ports	apache	mailing list
port which is maintained by autotools@FreeBSD.org	ports	autotools	alias
port which is maintained by doceng@FreeBSD.org	ports	doceng	alias
port which is maintained by eclipse@FreeBSD.org	ports	freebsd-eclipse	mailing list
port which is maintained by gecko@FreeBSD.org	ports	gecko	mailing list
port which is maintained by gnome@FreeBSD.org	ports	gnome	mailing list
port which is maintained by hamradio@FreeBSD.org	ports	hamradio	alias
port which is maintained by haskell@FreeBSD.org	ports	haskell	alias

Type	Suggested Category	Suggested Assignee	Assignee Type
port which is maintained by java@FreeBSD.org	ports	freebsd-java	mailing list
port which is maintained by kde@FreeBSD.org	ports	kde	mailing list
port which is maintained by mono@FreeBSD.org	ports	mono	mailing list
port which is maintained by office@FreeBSD.org	ports	freebsd-office	mailing list
port which is maintained by perl@FreeBSD.org	ports	perl	mailing list
port which is maintained by python@FreeBSD.org	ports	freebsd-python	mailing list
port which is maintained by ruby@FreeBSD.org	ports	freebsd-ruby	mailing list
port which is maintained by secteam@FreeBSD.org	ports	secteam	alias
port which is maintained by vbox@FreeBSD.org	ports	vbox	alias
port which is maintained by x11@FreeBSD.org	ports	freebsd-x11	mailing list

Ports PRs which have a maintainer who is a ports committer may be reassigned by anyone (but note that not every FreeBSD committer is necessarily a ports committer, so you cannot simply go by the email address alone.)

For other PRs, please do not reassign them to individuals (other than yourself) unless you are certain that the assignee really wants to track the PR. This will help to avoid the case where no one looks at fixing a particular problem because everyone assumes that the assignee is already working on it.

Table 5. Common Assignees - Other

Type	Suggested Category	Suggested Assignee	Assignee Type
problem with Problem Report database	bin	bugmeister	alias
problem with Bugzilla web form.	doc	bugmeister	alias

6. Assigned PRs

If a PR has the `responsible` field set to the username of a FreeBSD developer, it means that the PR has been handed over to that particular person for further work.

Assigned PRs should not be touched by anyone but the assignee or bugmeister. If you have comments, submit a followup. If for some reason you think the PR should change state or be reassigned, send a message to the assignee. If the assignee does not respond within two weeks, unassign the PR and do as you please.

7. Duplicate PRs

If you find more than one PR that describe the same issue, choose the one that contains the largest amount of useful information and close the others, stating clearly the number of the superseding PR. If several PRs contain non-overlapping useful information, submit all the missing information to one in a followup, including references to the others; then close the other PRs (which are now completely superseded).

8. Stale PRs

A PR is considered stale if it has not been modified in more than six months. Apply the following procedure to deal with stale PRs:

- If the PR contains sufficient detail, try to reproduce the problem in `-CURRENT` and `-STABLE`. If you succeed, submit a followup detailing your findings and try to find someone to assign it to. Set the state to "analyzed" if appropriate.
- If the PR describes an issue which you know is the result of a usage error (incorrect configuration or otherwise), submit a followup explaining what the originator did wrong, then close the PR with the reason "User error" or "Configuration error".
- If the PR describes an error which you know has been corrected in both `-CURRENT` and `-STABLE`, close it with a message stating when it was fixed in each branch.
- If the PR describes an error which you know has been corrected in `-CURRENT`, but not in `-STABLE`, try to find out when the person who corrected it is planning to MFC it, or try to find someone else (maybe yourself?) to do it. Set the state to "patched" and assign it to whomever will do the MFC.
- In other cases, ask the originator to confirm if the problem still exists in newer versions. If the originator does not reply within a month, close the PR with the notation "Feedback timeout".

9. Non-Bug PRs

Developers that come across PRs that look like they should have been posted to FreeBSD problem reports mailing list or some other list should close the PR, informing the submitter in a comment why this is not really a PR and where the message should be posted.

The email addresses that Bugzilla listens to for incoming PRs have been published as part of the FreeBSD documentation, have been announced and listed on the web-site. This means that spammers found them.

Whenever you close one of these PRs, please do the following:

- Set the component to `junk` (under `Supporting Services`).
- Set Responsible to `nobody@FreeBSD.org`.
- Set State to `Issue Resolved`.

Setting the category to `junk` makes it obvious that there is no useful content within the PR, and helps to reduce the clutter within the main categories.

10. Further Reading

This is a list of resources relevant to the proper writing and processing of problem reports. It is by no means complete.

- How to Write FreeBSD Problem Reports-guidelines for Problem Report originators.