Problem Report Handling Guidelines

Dag-Erling Smørgrav
Hiten Pandya
Revision: 43184

FreeBSD is a registered trademark of the FreeBSD Foundation.

Motif, OSF/1, and UNIX are registered trademarks and IT DialTone and The Open Group are trademarks of The Open Group in the United States and other countries.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this document, and the FreeBSD Project was aware of the trademark claim, the designations have been followed by the "™" or the "®" symbol.

2013-11-13 by hrs.

Abstract

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

Table of Contents

1. Introduction 2
2. Problem Report Life-cycle 2
3. Problem Report State 3
4. Types of Problem Reports 4
5. Further Reading 14

1. Introduction

GNATS is a defect management (bug reporting) system used by the FreeBSD Project. As accurate tracking of outstanding software defects is important to FreeBSD's quality, the correct use of GNATS is essential to the forward progress of the Project.

Access to GNATS is available to FreeBSD developers, as well as to the wider 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.

2. Problem Report Life-cycle

- The Reporter submits a PR with send-pr(1) and receives a confirmation message.
- Joe Random Committer takes interest in the PR and assigns it to himself, or Jane Random BugBuster decides that Joe is best suited to handle it and assigns it to him.
- Joe has a brief exchange with the originator (making sure it all goes into the audit trail) and determines the cause of the problem. He then makes sure the cause is documented in the audit trail, and sets the PRs state to “analyzed”.
- 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 PRs state to “feedback”.
- 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 he submitted all or part of the patch) and, if appropriate, start an MFC countdown.
- If the patch does not need MFCing, Joe then closes the PR.
- If the patch needs MFCing, Joe leaves the Problem Report in “patched” state until the patch has been MFCed, then closes it.

Note

Many PRs 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 PR.

Note

The “email address” used on the PR might not be able to receive mail. In this case, followup to the PR as usual and ask the originator (in the followup) to provide a working email address. This is normally the case when send-pr(1) is used from a system with the mail system disabled or not installed.

3. Problem Report State

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

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.
repocopy (obsolete)	The resolution of the problem report is dependent on a repository copy, or repocopy, operation within the CVS repository which is awaiting completion. Given that all repositories now use Subversion, there is no need for this state anymore. Subversion has native support for copying and moving files.
closed	A problem report is closed when any changes have been integrated, documented, and tested, or when fixing the problem is abandoned.

Note

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 GNATS 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.

- PRs not yet assigned to anyone.
- PRs already assigned to someone.
- Duplicates of existing PRs.
- Stale PRs
- Misfiled 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.

4.1. 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
architecture-specific	alpha, amd64, arm, i386, ia64, powerpc, sparc64	freebsd-*arch*
ports collection	ports	freebsd-ports-bugs
documentation shipped with the system	docs	freebsd-doc
FreeBSD web pages (not including docs)	www	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.

Note

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
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 Asynchronous Transfer Mode (ATM) drivers	kern	freebsd-atm	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

Type	Suggested Category	Suggested Assignee	Assignee Type
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
problem with Integrated Services Digital Network (ISDN) drivers	kern	freebsd-isdn	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
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
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

Type	Suggested Category	Suggested Assignee	Assignee Type
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 GNATS itself (send-pr(1))	bin	bugmeister	alias
problem with GNATS web form	www	bugmeister	alias

4.2. 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.

4.3. 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).

4.4. 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".

4.5. Misfiled PRs

GNATS is picky about the format of a submitted bug report. This is why a lot of PRs end up being "misfiled" if the submitter forgets to fill in a field or puts the wrong sort of data

in some of the PR fields. This section aims to provide most of the necessary details for FreeBSD developers that can help them to close or refile these PRs.

When GNATS cannot deduce what to do with a problem report that reaches the database, it sets the responsible of the PR to `gnats-admin` and files it under the `pending` category. This is now a "misfiled" PR and will not appear in bug report listings, unless someone explicitly asks for a list of all the misfiled PRs. If you have access to the FreeBSD cluster machines, you can use `query-pr` to view a listing of PRs that have been misfiled:

```
% query-pr --x --q --r gnats-admin
   52458 gnats-ad   open      serious   medium    Re: declaration ↺
clash f
   52510 gnats-ad   open      serious   medium    Re: lots of ↺
sockets in
   52557 gnats-ad   open      serious   medium
   52570 gnats-ad   open      serious   medium    Jigdo maintainer ↺
update
```

Commonly PRs like the ones shown above are misfiled for one of the following reasons:

- A followup to an existing PR, sent through email, has the wrong format on its `Subject:` header.
- A submitter sent a Cc: to a mailing list and someone followed up to that post instead of the email issued by GNATS after processing. The email to the list will not have the category/PRnumber tracking tag. (This is why we discourage submitters from doing this exact thing.)
- When completing the send-pr(1) template, the submitter forgot to set the category or class of the PR to a proper value.
- When completing the send-pr(1) template, the submitter set Confidential to `yes`. (Since we allow anyone to mirror GNATS via rsync, our PRs are public information. Security alerts should therefore not be sent via GNATS but instead via email to the Security Team.)
- It is not a real PR, but some random message sent to <`bug-followup@FreeBSD.org`> or <`freebsd-gnats-submit@FreeBSD.org`>.

4.5.1. Followups misfiled as new PRs

The first category of misfiled PRs, the one with the wrong subject header, is actually the one that requires the greatest amount of work from developers. These are not real PRs, describing separate problem reports. When a reply is received for an existing PR at one of the addresses that GNATS "listens" to for incoming messages, the subject of the reply should always be of the form:

```
Subject: Re: category/number: old synopsis text
```

Most mailers will add the "`Re: `" part when you reply to the original mail message of a PR. The "`category/number: `" part is a GNATS-specific convention that you have to manually insert to the subject of your followup reports.

Any FreeBSD developer, who has direct access to the GNATS database, can periodically check for PRs of this sort and move interesting bits of the misfiled PR into the audit trail of the original PR (by posting a proper followup to a bug report to the address <bug-followup@FreeBSD.org >). Then the misfiled PR can be closed with a message similar to:

```
Your problem report was misfiled.  Please use the format
"Subject: category/number: original text" when following
up to older, existing PRs.  I've added the relevant bits
from the body of this PR to kern/12345
```

Searching with `query-pr` for the original PR, of which a misfiled followup is a reply, is as easy as running:

```
% query-pr --q --y -"some text"
```

After you locate the original PR and the misfiled followups, use the `-F` option of `query-pr` to save the full text of all the relevant PRs in a UNIX® mailbox file, i.e.:

```
% query-pr --F 52458 52474 > mbox
```

Now you can use any mail user agent to view all the PRs you saved in `mbox`. Copy the text of all the misfiled PRs in a followup to the original PR and make sure you include the proper `Subject:` header. Then close the misfiled PRs. When you close the misfiled PRs remember that the submitter receives a mail notification that his PR changed state to "closed". Make sure you provide enough details in the log about the reason of this state change. Typically something like the following is ok:

```
Followup to ports/45364 misfiled as a new PR.
This was misfiled because the subject did not have the format:

 Re: ports/45364: -...
```

This way the submitter of the misfiled PR will know what to avoid the next time a followup to an existing PR is sent.

4.5.2. PRs misfiled because of missing fields

The second type of misfiled PRs is usually the result of a submitter forgetting to fill all the necessary fields when writing the original PR.

Missing or bogus "category" or "class" fields can result in a misfiled report. Developers can use edit-pr(1) to change the category or class of these misfiled PRs to a more appropriate value and save the PR.

Another common cause of misfiled PRs because of formatting issues is quoting, changes or removal of the `send-pr` template, either by the user who edits the template or by mailers which do strange things to plain text messages. This does not happen a lot of the time, but it can be fixed with `edit-pr` too; it does require a bit of work from the developer who refiles the PR, but it is relatively easy to do most of the time.

4.5.3. Misfiled PRs that are not really problem reports

Sometimes a user wants to submit a report for a problem and sends a simple email message to GNATS. The GNATS scripts will recognize bug reports that are formatted using the send-pr(1) template. They cannot parse any sort of email though. This is why submissions of bug reports that are sent to <`freebsd-gnats-submit@FreeBSD.org`> have to follow the template of `send-pr`, but email reports can be sent to FreeBSD problem reports mailing list.

Developers that come across PRs that look like they should have been posted to freebsd-bugs or some other list should close the PR, informing the submitter in their state-change log why this is not really a PR and where the message should be posted.

The email addresses that GNATS 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. Spam messages that reach GNATS are promptly filed under the "pending" category until someone looks at them. Closing one of these with edit-pr(1) is very annoying though, because GNATS replies to the submitter and the sender's address of spam mail is never valid these days. Bounces will come back for each PR that is closed.

Currently, with the installation of some antispam filters that check all submissions to the GNATS database, the amount of spam that reaches the "pending" state is very small.

All developers who have access to the FreeBSD.org cluster machines are encouraged to check for misfiled PRs and immediately close those that are spam mail. Whenever you close one of these PRs, please do the following:

- Set Category to `junk`.
- Set Confidential to `no`.
- Set Responsible to `gnats-admin`.
- Set State to `closed`.

Junk PRs are not backed up, so filing spam mail under this category makes it obvious that we do not care to keep it around or waste disk space for it. If you merely close them without changing the category, they remain both in the master database and in any copies of the database mirrored through cvsup.

5. 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 PR originators.