The Swallowtail Bugtracking System

David A. Holland

September 3, 2012

1 Introduction

Swallowtail is a bugtracking system meant to serve as a migration path for large GNATS installations. It is intended to address various shortcomings of GNATS that appear with a large database, and to provide a migration path for that database that does not lose information.

As in GNATS, individual bug reports are called PRs ("problem reports") and are given unique numbers from an increasing sequence.

The database is kept in PostgreSQL.

2 Design Criteria

Swallowtail is intended for use by large projects that take their bug tracking seriously. It assumes the existence of a project server that can host the database, and that project developers have shell access on this machine. Unless special measures are taken, everyone with shell access to the host machine is assumed to be a developer and has developer-level access to the database. Developers are assumed to be grownups, at least mentally, and to not require either baby-sitting or padded cells. Developers are also assumed to be generally clueful and able to take advantage of direct query access to the database.

A web interface is also provided but it allows only end-user-level access. It also does not attempt to offer the full range of query and annotation support. (At least to begin with.)

It is assumed that most bug report traffic will be handled by email, and that all public traffic will be reflected onto one or more mailing lists for general distribution. However, both new bug reports and followup traffic can be submitted via the web interface. It is not necessary to sign up to file a bug report.

Swallowtail assumes that the database is large (as bug databases go, not large in DBMS terms), both in terms of the total number of bugs on file and the number of issues currently open. This means that hand-editing is not feasible. Manual generation of semantic indexes and tags is presumed possible but expensive. It is assumed that multiple types of semantic indexing and tagging will be required. Supporting this is the primary motivation for creating Swallowtail at all.

Swallowtail is also intended to replace GNATS, so it explicitly allows importing an existing GNATS database without losing information. (One of the major problems with migrating away from GNATS, or migrating any bug database, is that most bug databases have a hardcoded schema with fixed semantics, and no two are equivalent.)

It also is written to allow incoming bug reports to be sent with the GNATS send-pr tool, as this tool is deployed in the field and can be updated in the field only gradually.

We are explicitly assuming that everything that talks directly to GNATS can be updated, rewritten, or replaced for Swallowtail, and that there are relatively few such things.

Swallowtail is being written specifically to manage NetBSD's bug database, but it is intended to remain general enough to be useful to other groups who may be in a similar position.

3 Submitting PRs

New PRs can be submitted either by email or via a web form. The email interface accepts only structured mail in the form sent by the GNATS send-pr tool. Malformed emails are returned. (Sending handwritten mail to the PR submission address is not supported.)

The web form requires a valid email address, as most transactions are handled by email. In the future it might be desirable to set up a scheme that allows entirely web-based access for end users.

XXX: need to decide whether/how much to validate email addresses in the web form, and also what measures to take to prevent spam. Maybe an email address already in the system can be used to submit new PRs without any kind of further validation step.

XXX: Also I wonder if we ought to have a blacklist for emails that aren't allowed to submit PRs.

Submitting a PR generates an email to the submitter that reports the PR number and includes brief instructions for further actions. If the PR is confidential, a notice to this effect is included in the response mail, as well as instructions and/or a link for making the PR non-confidential. (This is because most confidential PR submissions are confidential by mistake.) The mail also includes a generated password for logging in to the web interface as a submitter. (Currently this does nothing; in the future it will allow e.g. unsubscribing from the PR.)

The submitter of a PR is automatically subscribed to the PR as a respondent.

Note that currently Swallowtail does not provide its own send-pr script, as this is not required for the initial deployment. There will probably be one in the future.

NetBSD: The submission address for NetBSD will be swallowtail@NetBSD.org (unless we decide on something else) although the old gnats-bugs address will be kept for the indefinite future to allow old send-pr scripts to continue to work.

GNATS: For some reason GNATS accepts a wide range of malformed incoming emails. This almost invariably makes a mess when it occurs.

4 Retrieving PRs

The web interface includes a scheme for retrieving any PR (except PRs marked confidential) by a well-known address. The search page also allows retrieving a single PR by number.

NetBSD: The address will probably be `http://bugs.netbsd.org/12345`.

In the web interface, text within the administrative or message logs that can be identified as a PR number is rendered as a clickable link.

Email addresses are not displayed in the web interface at all. (Or maybe, in the future when this is possible, only when logged in.)

The `query-pr` command line tool can retrieve any PR, including confidential PRs. This is done by passing it the PR number. As in GNATS the option `--full` prints the message and administrative logs as well as the base PR data. Use `--attach N` to retrieve an attachment.

`query-pr` can also be used for searching. See below.

5 Commenting on PRs

Comments on PRs can be submitted either by email or via the PR's web page. In either case, files can be provided (as MIME attachments or via upload) that are filed as attachments to the PR. Comments on PRs can also be submitted by running `comment-pr`, or from `browse-pr`. All comments go to the PR's message log.

MIME multipart messages that include both text and HTML will be converted to plain text. MIME mail that is HTML-only will be rejected. The MIME content-type of mails is retained, so if non-ASCII mail appears it can be displayed correctly, and so remailed text is marked correctly.

The comment address is the same as the submission address. Comments are identified by the Subject: line, which should begin with `Re: category/12345` like in GNATS. (In the future we might switch to a tidier tagging scheme, such as `[NetBSD 12345]` anywhere in the Subject: line, but the old style will have to be supported indefinitely regardless.) The category need not be the right category for the PR; if it is wrong, the right one will be substituted before the comment is filed or remailed. Also, if the Subject: line of a comment is empty except for the tag, the PR's synopsis will be inserted.

Source control commit messages routed to the bug database are treated as comments.

Comments are remailed to all email addresses subscribed to the PR, and also (if not confidential) to the mailing list associated with the PR's category.

Remailed comments have a References: header that allows them to thread together. (The contents of it need to be site-configurable.)

Comments filed on a locked PR go to the administrator queue.

GNATS: GNATS has no web-based way of filing comments, which occasionally annoys people.

GNATS: GNATS has no MIME support at all and corrupts mails with attachments when filing them in the database (not fatally but annoyingly), which aggravates everyone regularly.

GNATS: GNATS cannot lock PRs, and old PRs often receive spam or misdirected mail that has to be cleaned up by hand.

6 Updating/Editing PRs

PRs are edited with `edit-pr`. (This is not accessible to end users. An end user wanting to adjust the PR state should file a comment requesting the change.)

Changes made with `edit-pr` are recorded in the PR's administrative log. (XXX: some of them should be mailed out and not others, but it isn't clear which are which yet.)

Each PR has the following base properties:

- Its number. (This cannot be changed.)
- The *synopsis*. This is a short (one-line) description of the PR provided by the submitter. It can (and should) be adjusted as necessary during the lifetime of the PR.

  If in a new submission or conversion from GNATS the Subject: line of the email is not the same as the synopsis field, the effect will be that the synopsis is first set to the provided Subject: and then to the provided synopsis, generating the corresponding change entry in the PR's administrative log.
- The confidential flag. PRs marked confidential can be accessed only by developers, and traffic about them is not reflected to mailing lists.
- The *state*. This is the basic state the PR is in. (Open, closed, pending-pullups, etc.) The list of states is configurable. States have a name (which should be an identifier), a description, and flags that identify if the state is a closed state, whether reminder and/or feedback mail should be sent. There is also an obsolete flag that prevents new uses of the state.

  NetBSD: we will start out with the same set of states as we have in GNATS, plus "stuck". (A PR is stuck if a respondent is needed and there are none.)

  XXX: when you put a PR into feedback state you should provide a timeout period and whether the PR should, if the timeout expires, be changed to "stuck" or "closed".
- The *locked* flag. A locked PR cannot be edited or commented on until it is first unlocked. Attempting to run `edit-pr` or `comment-pr` on a locked PR will result in a message that the PR is locked and the option to unlock it or abort. (Filing a comment on a locked PR by email or via the web interface results in the comment going to the administrator queue.) PRs are automatically locked by the system a month after being closed.

  The database conversion from GNATS will mark all PRs locked that have been closed for more than a month.

  GNATS: GNATS does not support locked PRs.

- The database schema version that was current when the PR arrived. This is useful for knowing what conversions were applied to the data. The initial schema version is 1. Version 0 means "converted from GNATS".
- The date (and time) that the PR arrived.
- The date (and time) that the PR was closed, if appropriate.
- The original submitter, who remains the original submitter even if later unsubscribed.
- The *release*, which is the Release field from GNATS.
- The *environment*, which is the Environment field from GNATS.
- The administrative log, which is a list of messages generated when the PR is updated.
- The message log, which is a list of comments filed on the PR and their attachments.
- The subscription lists. As described below, a user can be subscribed in any of three ways (and more than once). Each subscription also has a "batch" flag to send out update mails in bundles. (XXX: it is not clear if this needs to be implemented up front or not.)
- Classification entries. There are, in general, multiple classification schemes, which come in different flavors. Each PR has a value for each classification scheme (which might be missing or null in some cases) whose allowable values and meaning are defined by the classification scheme. The GNATS Severity, Priority, Class, and Category fields are handled as classification schemes. (The category scheme has some special-case handling as described elsewhere.)

The mail from-address and message-ID, which are retained by GNATS, are in Swallowtail just handled as entries in the administrative log.

The Description, How-To-Repeat, and Fix fields of a GNATS PR, along with any "unformatted" text that confused GNATS, are in Swallowtail handled as the initial entries in the PR's message log.

Idea for the future: provide up/down vote buttons (both in the web page and in edit-pr) to help rank the importance of individual bugs. If you are also seeing a particular problem, you vote it up; if you are using the affected software but not seeing it, you vote it down. Or something like that.

The data associated with users is described in another section below. Decoupling user data from individual PRs makes it feasible to update your email address in the database, which is tedious to the point of impossible with GNATS.

7 Tracking PRs

The simple way to track PRs is to subscribe to the mailing list(s) that Swallowtail traffic is reflected to. However, since those lists are likely to be high volume, and also do not carry traffic about confidential PRs, it is also possible to subscribe directly to individual PRs.

In the web interface, there is a link for this. XXX: there should be a way to do this by email, but it isn't yet clear what it should be.

Everyone subscribed to a PR receives a copy of all comments made on the PR, and notices of significant administrative changes. There are two additional modes:

- A *respondent* is someone who is experiencing the problem and can provide feedback if needed. The original submitter is always subscribed as a respondent, but especially for old PRs, they may disapper and others may appear.
- A *responsible party* is someone who is fixing the problem, or at least taking charge of seeing that it gets fixed. This may be a specific developer, or it may be a role account that looks after PRs nobody specific is handling. For new PRs a default responsible party can be subscribed based on the PR category. Ordinarily only one developer will be subscribed as responsible.

If the last respondent with `mailto` permission unsubscribes (or is unsubscribed) from a PR that is in feedback state, it is automatically switched to the stuck state. (XXX: but do we want to make "stuck" a known special case like this?)

XXX: there should be a way to bring a PR to the attention of another developer short of subscribing them to it.

In the future there should be an RSS feed for each PR, but this is not expected to be implemented in the first rollout. (There should also be individual RSS feeds for categories, and based on classification schemes, and whatever else.)

GNATS: GNATS has no ability to allow anyone but the original submitter to function as a respondent.

8 Reminder Mails

The system periodically sends out two kinds of mail: *reminder mail*, that goes to developers responsible for open PRs, and *feedback mail*, that goes to the submitters (typically) of PRs that have been placed in feedback state.

One reminder mail is sent to each developer subscribed as "responsible" to at least one PR, listing the PR by number and synopsis. (XXX: sorted how?) Reminders are not sent for PRs that are in feedback.

NetBSD: reminders are also not sent for PRs that are in pending-pullups state.

Feedback mail is sent to each user subscribed as "respondent" to at least one PR. If the user is being nagged about three or fewer PRs, one email is sent for each PR, with the headers adjusted so replies get filed in that PR. This makes it easier for casual users to respond correctly. If the user is being nagged about more PRs than that, only one mail is sent, in the same form as reminder mail. In the future the feedback mails will repeat the message requesting feedback; this is not expected to be implemented for the initial deployment.

9 Statistical Reporting

Various scripts to gather statistics are provided, both as part of the web interface and to generate periodic report mail.

These include:

- The summary statistics, originally written by Erik E. Fair for the NetBSD GNATS;
- A daily report, based on the one GNATS sends to the netbsd-bugs mailing list;
- A monthly report, based on the one I wrote to send to the developers list;
- Whatever other stuff may come up or looks interesting.

10 Searching for PRs

The web interface provides a simple search form, which can search on various fields or on text in the log. In the future this may be made more powerful; for the time being it is expected to remain largely the same as the interface to GNATS.

Command-line searching can be done with `query-pr`. Any value in any field can be matched; text search in the administrative and message logs (or anywhere in the PR) is also supported. XXX: this should take advantage of postgres's full text search tools, and this section should be rewritten once it's clear what that provides.

There is a flag for also searching closed PRs, which are skipped by default. There is also a flag to retrieve a randomly selected PR.

Note that the weird GNATS query-pr query language is not supported. (Complex queries are best issued using the SQL interface.)

11 Browsing the Database

The tool `browse-pr` is provided for examining the database. Its user interface is reminiscent of a newsreader, e.g. slrn. And, it will probably only be implemented in the future. (It is something I would really like to have, but it's not critical for the first deployment.)

12 Direct SQL Interface

XXX: document this, including particularly the views that developers are meant to use to look at things.

13 Database Users

Unlike GNATS, the Swallowtail database tracks users explicitly. Each one has a username and a real name, and one or more email addresses. When more than one email address is on file, one of the addresses is selected for sending mail. The Organization field that GNATS stores and send-pr provides is filed with the email address that sent it. The database also tracks the last time mail was received from each address, as this is often useful when sorting out problems.

Each address also has a "web password" attached to it. This is generated randomly when the address is first encountered in a PR submission and sent back to the user in the response to the submission. In the future it is meant to allow logging into the web site as a submitter; for now it does nothing.

Each user also has the following permission bits:

- `mailto` – if this is false, Swallowtail will not send any email to this user. If the only users subscribed to a PR as respondents have mail disabled (or there are no users subscribed as respondents) setting the PR to feedback state will fail. This flag can be used for users whose mail bounces, or who are otherwise no longer available.
- `responsible` – if this is true, the user is allowed to be responsible for open PRs. This should normally be set for all developers, and only developers, and should be cleared when a developer quits. (Any open PRs they are responsible for should be reassigned.) It should also be set for role accounts that are responsible by default when nobody else is.
- `oldresponsible` – if this is true, the user is allowed to be responsible only for closed PRs. This should be set if `responsible` is set, and should also be left set when a developer quits.
- `editpr` – if this is true, the user is allowed to run `edit-pr` and make changes to the database. This should be set for all developers, and only developers, and should be cleared when a developer quits.
- `admin` – if this is true, the user is allowed to perform administrative tasks.

Note that while all developers are assumed to have a username, end users will in general not have one.

XXX: what's the mechanism for updating user information?

14 Administrator Queue

The database contains an explicit queue, or in fact several explicit queues, of issues requiring administrator attention. The following subsections describe the components of the administrator queue. The administrator queue is accessible only to users with the `admin` permission bit set.

14.1 Comments filed on locked PRs.

When a comment is made on a PR that is locked, instead of being applied to the PR it is placed in the administrator queue for manual attention. The possible actions are:

- defer: put it off for tomorrow;
- accept: add it to the PR, for valid/intended comments;
- new: create a new PR instead;
- reroute: send to another PR instead, for mistyped PR numbers;
- bounce: return to sender, for non-spam with no clear other choice;
- discard: erase it, as comments on locked PRs are often spam.

Accepting a comment on a locked PR automatically unlocks it, and by default also reopens it.

14.2 Feedback mail bounces.

When bounced mail arrives, Swallowtail examines it to try to determine which PR and which user it applies to. If it appears to be a bounce generated from feedback mail (or from reminder mail) it goes into this queue. The full mail is retained for inspection.

The possible actions are:

- defer: put it off for tomorrow;
- count: just add one to the number of bounces recorded;
- changeaddr: change to a different address for the same user;
- nomail: inhibit sending further mail to the user;
- discard: delete the mail from the queue.

14.3 Other mail bounces.

Bounced mail that can be associated with a PR and/or user but is not feedback goes into this queue. The actions are the same as for feedback bounces.

14.4 Junk mail.

All other unrecognized mail goes into this queue. The possible actions are:

- defer: put it off for tomorrow;
- forward: forward it somewhere (normally to yourself);
- discard: erase it.

(XXX: maybe this stuff should just be forwarded on or dumped out into a mailbox somewhere.)

15 Classification Schemes

Here is a quick discussion of the classification schemes that we expect to use, or might use.

There are four kinds of classifications: enumerated (the value must be one from a fixed set of constants), tags (the value is zero or more constants from a fixed pool), text (the value can be any string), and hierarchical (like enumerated, but with structure).

In the long run some of the things that are currently PR states may want to be tags instead.

15.1 Enumerated classifications

Category. The part of the system affected, in broad terms. The same as the GNATS "Category" field. This classification is special-cased in some places, at least for now; in the future it might make sense to allow those places to do matching on arbitrary classifications.

Class. The GNATS "Class" field: `sw-bug`, `doc-bug`, `change-request`, etc.

Severity. The GNATS "Severity" field.

Priority. The GNATS "Priority" field.

15.2 Tag sets.

Release-goals. Tags like `6-CRITICAL`, to be applied and managed by releng when planning for a release.

Branch-notes. Tags like `5-ONLY`, to mark bugs that do not affect HEAD and that can therefore be retired along with branches.

15.3 Text classifications

Manpage. The name of the nearest man page to where the problem appears to be. FreeBSD has been using this as a tagging scheme for some time.

Package. The identity (category/directory form) of the package affected. (For pkgsrc bugs only.)

15.4 Hierarchical taxonomies

Subsystem. The hierarchical taxonomy I've been using for the buglists web page.

Consequences. A hierarchical taxonomy of the results of the problem (annoyance, system crash, remote root exploit, etc.)

16 Other Future Features

Some other things that aren't likely to be deployed immediately but are worth considering later:

- Have a way to note dependencies among PRs, as in 12345 is stalled until 24680 is fixed. (This does not occur often enough in practice to be a priority.)
- Have a way to massedit a group of PRs; for example, put all PRs against a particular driver or filesystem or whatever into feedback after major fixes.
- There should be administrator buttons for moving comments to different PRs, for e.g. misfiled comment notices.
- It should be possible for users to set personal tags/priorities for annotating and/or sorting nag mail. (E.g. sort by priority, or age, or time since last change, etc.)
- It would be good for each nag mail (at least reminder mail, maybe not feedback mail) to report the changes in the list since the last one that went out.
- It should be possible to blacklist Cc: of certain addresses (particularly internal lists) in bug traffic.
- Any new send-pr script should be configurable by the user with return addresses and default values for organization and so forth.
- It would be nice to be able to track PR status separately (or semi-separately) for release branches. However, it isn't clear how to do this without making a cumbersome mess.
- It might be useful to be able to explicitly merge duplicate PRs, without either losing information or creating a confused mess of interleaved comments.
- It might be worth adding threading for comments.
- We should figure out how to make Google see and index the pages for each PR. Or maybe just the open ones.
- It would be nice if the web page for each PR had browse features like links for "show all PRs of this priority" or "show all PRs from this submitter" and whatnot.
- Lists of PRs in the web interface should have a sidebar to let you narrow the list, and a list of the filters currently applied (like e.g. NewEgg's product searches). Also, the sort order should be defined (unlike GNATS) and selectable.

- It might be worthwhile to allow PR subscriptions in a brief mode where all you get is "Joe commented on PR 12345".
- We will eventually want a way to turn MIME blobs already in the database into attachments.
- Should have per-user tags that are more or less private.
- Should have per-user tags that are public.
- In the long run it should be possible to log into the web interface as a developer and do all developer things there instead of from the command line.