Reporting Problems

Using `send-pr`, version 3.97-96q1
October 1993

Jeffrey M. Osier
Cygnus Support

Table of Contents

Overview

This manual documents `send-pr`, which uses electronic mail to submit support questions and problem reports to a central Support Site. No body of work is perfect, and support organizations understand this; `send-pr` is designed to allow users who have problems to submit reports of these problems to sites responsible for supporting the products in question, in a defined form which can be read by an electronically managed database.

`send-pr` is part of a suite of programs known collectively as PRMS, the GNU Problem Report Management System. PRMS consists of several programs which, used in concert, formulate and partially administer a database of *Problem Reports*, or *PRs*, at a central Support Site. A PR goes through several states in its lifetime; PRMS tracks the PR and all information associated with it through each state and finally acts as an archive for PRs which have been *closed*.

Because `send-pr` exists as a shell (‘`/bin/sh`’) script and as an Elisp file for use with GNU Emacs, it can be used from any machine on your network which can run a shell script and/or Emacs.

In general, you can use any editor and mailer to submit valid Problem Reports, as long as the format required by PRMS is preserved. `send-pr` automates the process, however, and ensures that certain fields necessary for automatic processing are present. `send-pr` is strongly recommended for all initial problem-oriented correspondence with your Support Site. The organization you submit Problem Reports to supplies an address to which further information can be sent; the person responsible for the category of the problem you report contacts you directly.

1 Details about send-pr and PRMS

A *Problem Report* is a message that describes a problem you are having with a body of work. send-pr organizes this message into a form which can be understood and automatically processed by PRMS, the GNU Problem Report Management System. A Problem Report is organized into *fields* which contain data describing you, your organization, and the problem you are announcing (see Section 1.2 "Problem Report format," page 3). Problem Reports go through several defined states in their lifetimes, from *open* to *closed* (see Section 1.1 "States of Problem Reports," page 3).

1.1 States of Problem Reports

Each PR goes through a defined series of states between origination and closure. The originator of a PR receives notification automatically of any state changes.

open The initial state of a Problem Report. This means the PR has been filed and the responsible person(s) notified.

analyzed The responsible person has analyzed the problem. The analysis should contain a preliminary evaluation of the problem and an estimate of the amount of time and resources necessary to solve the problem. It should also suggest possible workarounds.

feedback The problem has been solved, and the originator has been given a patch or other fix. The PR remains in this state until the originator acknowledges that the solution works.

closed A Problem Report is closed ("the bug stops here") only when any changes have been integrated, documented, and tested, and the submitter has confirmed the solution.

suspended
Work on the problem has been postponed. This happens if a timely solution is not possible or is not cost-effective at the present time. The PR continues to exist, though a solution is not being actively sought. If the problem cannot be solved at all, it should be closed rather than suspended.

1.2 Problem Report format

The format of a PR is designed to reflect the nature of PRMS as a database. Information is arranged into *fields*, and kept in individual records (Problem Reports).

Problem Report fields are denoted by a keyword which begins with '>' and ends with ':', as in '`>Confidential:`'. Fields belong to one of three data types:

ENUMERATED
: One of a specific set of values, which vary according to the field. The value for each keyword must be on the same line as the keyword. These values are not configurable (yet).

  For each ENUMERATED keyword, the possible choices are listed in the `send-pr` template as a comment. The following fields are ENUMERATED format; see the descriptions of fields below for explanations of each field in detail:

```
>Confidential:   >Severity:       >Priority:
>Class:          >State:          >Number:
```

TEXT
: One single line of text which must begin and end on the same line (i.e., before a newline) as the keyword. See the descriptions of fields below for explanations of each field in detail. The following fields are TEXT format:

```
>Submitter-Id:   >Originator:     >Synopsis:
>Category:       >Release:        >Responsible:
>Arrival-Date:
```

MULTITEXT
: Text of any length may occur in this field. MULTITEXT may span multiple lines and may also include blank lines. A MULTITEXT field ends only when another keyword appears. See the descriptions of fields below for explanations of each field in detail.

  The following fields are MULTITEXT format:

```
>Organization:   >Environment:    >Description:
>How-To-Repeat:  >Fix:            >Audit-Trail:
>Unformatted:
```

A Problem Report contains two different types of fields: *Mail Header* fields, which are used by the mail handler for delivery, and *Problem Report* fields, which contain information relevant to the Problem Report and its submitter. A Problem Report is essentially a specially formatted electronic mail message.

The following is an example Problem Report. Mail headers are at the top, followed by PRMS fields, which begin with '>' and end with ':'. The '`Subject:`' line in the mail header and the '`>Synopsis:`' field are usually duplicates of each other.

```
Message-Id:  message-id
Date:        date
From:        address
Reply-To:    address
To:          bug-address
Subject:     subject

>Number:        prms-id
>Category:      category
>Synopsis:      synopsis
>Confidential: yes or no
>Severity:      critical, serious, or non-critical
>Priority:      high, medium or low
>Responsible:   responsible
>State:         open, analyzed, suspended, feedback, or closed
>Class:         sw-bug, doc-bug, change-request, support,
or duplicate
>Submitter-Id: submitter-id
>Arrival-Date: date
>Originator:   name
>Organization: organization
>Release:      release
>Environment:
   environment
>Description:
   description
>How-To-Repeat:
   how-to-repeat
>Fix:
   fix
>Audit-Trail:
appended-messages. . .
State-Changed-From-To: from-to
State-Changed-When: date
State-Changed-Why:
   reason
Responsible-Changed-From-To: from-to
Responsible-Changed-When: date
Responsible-Changed-Why:
   reason
>Unformatted:
   miscellaneous
```

1.2.1 Mail header fields

A Problem Report may contain any mail header field described in the Internet standard RFC-822. However, only the fields which identify the sender and the subject are required by `send-pr`:

`To:` The preconfigured mail address for the Support Site where the PR is to be sent, automatically supplied by `send-pr`.

`Subject:` A terse description of the problem. This field normally contains the same information as the '`>Synopsis:`' field.

`From:` Usually supplied automatically by the originator's mailer; should contain the originator's electronic mail address.

`Reply-To:`
A return address to which electronic replies can be sent; in most cases, the same address as the `From:` field.

1.2.2 Problem Report fields

Field descriptions

The following fields are present whenever a PR is submitted via the program `send-pr`. PRMS adds additional fields when the PR arrives at the Support Site; explanations of these follow this list.

`>Submitter-Id:`
(TEXT) A unique identification code assigned by the Support Site. It is used to identify all Problem Reports coming from a particular site. (Submitters without a value for this field can invoke `send-pr` with the '`--request-id`' option to apply for one from the support organization. Problem Reports from those not affiliated with the support organization should use the default value of '`net`' for this field.)

`>Originator:`
(TEXT) Originator's real name. The default is the value of the originator's environment variable `NAME`.

`>Organization:`
(MULTITEXT) The originator's organization. The default value is set with the variable `DEFAULT_ORGANIZATION` in the `send-pr` shell script.

`>Confidential:`
(ENUMERATED) Use of this field depends on the originator's relationship with the support organization; contractual

agreements often have provisions for preserving confidentiality. Conversely, a lack of a contract often means that any data provided will not be considered confidential. Submitters should be advised to contact the support organization directly if this is an issue.

If the originator's relationship to the support organization provides for confidentiality, then if the value of this field is 'yes' the support organization treats the PR as confidential; any code samples provided are not made publicly available (e.g., in regression test suites). The default value is 'yes'.

`>Synopsis:`
(TEXT) One-line summary of the problem. `send-pr` copies this information to the '`Subject:`' line when you submit a Problem Report.

`>Severity:`
(ENUMERATED) The severity of the problem. Accepted values include:

`critical` The product, component or concept is completely non-operational or some essential functionality is missing. No workaround is known.

`serious` The product, component or concept is not working properly or significant functionality is missing. Problems that would otherwise be considered '`critical`' are rated '`serious`' when a workaround is known.

`non-critical`
The product, component or concept is working in general, but lacks features, has irritating behavior, does something wrong, or doesn't match its documentation.

The default value is '`serious`'.

`>Priority:`
(ENUMERATED) How soon the originator requires a solution. Accepted values include:

`high` A solution is needed as soon as possible.

`medium` The problem should be solved in the next release.

`low` The problem should be solved in a future release.

The default value is 'medium'.

>Category:
(TEXT) The name of the product, component or concept where the problem lies. The values for this field are defined by the Support Site.

>Class: (ENUMERATED) The class of a problem can be one of the following:

- sw-bug — A general product problem. ('sw' stands for "software".)
- doc-bug — A problem with the documentation.
- change-request — A request for a change in behavior, etc.
- support — A support problem or question.
- duplicate (*pr-number*) — Duplicate PR. *pr-number* should be the number of the original PR.

The default is 'sw-bug'.

>Release:
(TEXT) Release or version number of the product, component or concept.

>Environment:
(MULTITEXT) Description of the environment where the problem occured: machine architecture, operating system, host and target types, libraries, pathnames, etc.

>Description:
(MULTITEXT) Precise description of the problem.

>How-To-Repeat:
(MULTITEXT) Example code, input, or activities to reproduce the problem. The support organization uses example code both to reproduce the problem and to test whether the problem is fixed. Include all preconditions, inputs, outputs, conditions after the problem, and symptoms. Any additional important information should be included. Include all the details that would be necessary for someone else to recreate the problem reported, however obvious. Sometimes seemingly arbitrary or obvious information can point the way toward a solution. See also Section 2.4 "Helpful hints," page 17.

`>Fix:` (MULTITEXT) A description of a solution to the problem, or a patch which solves the problem. (This field is most often filled in at the Support Site; we provide it to the submitter in case she has solved the problem.)

PRMS adds the following fields when the PR arrives at the Support Site:

`>Number:` (ENUMERATED) The incremental identification number for this PR.

The '`>Number:`' field is often paired with the '`>Category:`' field as

`category/number`

in subsequent email messages. This is for historical reasons, as well as because Problem Reports are stored in subdirectories which are named by category.

`>State:` (ENUMERATED) The current state of the PR. Accepted values are:

`open` The PR has been filed and the responsible person notified.

`analyzed` The responsible person has analyzed the problem.

`feedback` The problem has been solved, and the originator has been given a patch or other fix.

`closed` The changes have been integrated, documented, and tested, and the originator has confirmed that the solution works.

`suspended`
Work on the problem has been postponed.

The initial state of a PR is 'open'. See Section 1.1 "States of Problem Reports," page 3.

`>Responsible:`
(TEXT) The person responsible for this category.

`>Arrival-Date:`
(TEXT) The time that this PR was received by PRMS. The date is provided automatically by PRMS.

`>Audit-Trail:`
(MULTITEXT) Tracks related electronic mail as well as changes in the '`>State:`' and '`>Responsible:`' fields with the sub-fields:

`State-Changed-<From>-<To>:` *`oldstate`*`->-<`*`newstate`*
The old and new '`>State:`' field values.

`Responsible-Changed-<From>-<To>:` *`oldresp`*`->-<`*`newresp`*
The old and new '`>Responsible:`' field values.

`State-Changed-By:` *`name`*
`Responsible-Changed-By:` *`name`*
The name of the maintainer who effected the change.

`State-Changed-When:` *`timestamp`*
`Responsible-Changed-When:` *`timestamp`*
The time the change was made.

`State-Changed-Why:` *`reason...`*
`Responsible-Changed-Why:` *`reason...`*
The reason for the change.

The '`>Audit-Trail:`' field also contains any mail messages received by PRMS related to this PR, in the order received.

`>Unformatted:` (MULTITEXT) Any random text found outside the fields in the original Problem Report.

2 Editing and sending PRs

You can invoke `send-pr` from a shell prompt or from within GNU Emacs using '`M-x send-pr`'.

2.1 Creating new Problem Reports

Invoking `send-pr` presents a PR *template* with a number of fields already filled in. Complete the template as thoroughly as possible to make a useful bug report. Submit only one bug with each PR.

A template consists of three sections:

Comments
: The top several lines of a blank template consist of a series of comments that provide some basic instructions for completing the Problem Report, as well as a list of valid entries for the '`>Category:`' field. These comments are all preceded by the string '`SEND-PR:`' and are erased automatically when the PR is submitted. The instructional comments within '`<`' and '`>`' are also removed. (Only these comments are removed; lines you provide that happen to have those characters in them, such as examples of shell-level redirection, are not affected.)

Mail Header
: `send-pr` creates a standard mail header. `send-pr` completes all fields except the '`Subject:`' line with default values. (See Section 1.2 "Problem Report format," page 3.)

PRMS *fields*
: These are the informational fields that PRMS uses to route your Problem Report to the responsible party for further action. They should be filled out as completely as possible. (See Section 1.2 "Problem Report format," page 3. Also see Section 2.4 "Helpful hints," page 17.)

For examples of a Problem Report template and complete Problem Report, see Chapter 3 "An Example," page 19.

The default template contains your preconfigured '`>Submitter-Id:`'. `send-pr` attempts to determine values for the '`>Originator:`' and '`>Organization:`' fields (see Section 1.2 "Problem Report format," page 3). `send-pr` will set the '`>Originator:`' field to the value of the `NAME` environment variable if it has been set; similarly, '`>Organization:`' will be set to the value of `ORGANIZATION`. `send-pr` also attempts to find out some information about your system and architecture, and places this information in the '`>Environment:`' field if it finds any.

You may submit problem reports to different Support Sites from the default site by specifying the alternate site when you invoke `send-pr`. Each `site` has its own list of categories for which it accepts Problem Reports. (See Section A.2 “Setting a default *site*,” page 28.)

`send-pr` also provides the mail header section of the template with default values in the ‘`To:`’, ‘`From:`’, and ‘`Reply-To:`’ fields. The ‘`Subject:`’ field is empty.

The template begins with a comment section:

```
SEND-PR: -*- send-pr  -*-
SEND-PR: Lines starting with ‘SEND-PR’ will be removed
SEND-PR: automatically as well as all comments (the text
SEND-PR: below enclosed in ‘<’ and ‘>’).
SEND-PR:
SEND-PR: Please consult the document ‘Reporting Problems
SEND-PR: Using send-pr’ if you are not sure how to fill out
SEND-PR: a problem report.
SEND-PR:
SEND-PR: Choose from the following categories:
```

and also contains a list of valid `>Category:` values for the Support Site to whom you are submitting this Problem Report. One (and only one) of these values should be placed in the `>Category:` field. A complete sample bug report, from template to completed PR, is shown in Chapter 3 “An Example,” page 19. For a complete list of valid categories, type ‘`send-pr -L`’ at your prompt. See “Valid Categories,” page 24, for a sample list of categories, .

The mail header is just below the comment section. Fill out the ‘`Subject:`’ field, if it is not already completed using the value of ‘`>Synopsis:`’. The other mail header fields contain default values.

```
To: support-site
Subject: complete this field
From: your-login@your-site
Reply-To: your-login@your-site
X-send-pr-version: send-pr 3.97-96q1
```

where *`support-site`* is an alias for the Support Site you wish to submit this PR to.

The rest of the template contains PRMS fields. Each field is either automatically completed with valid information (such as your ‘`>Submitter-Id:`’) or contains a one-line instruction specifying the information that field requires in order to be correct. For example, the ‘`>Confidential:`’ field expects a value of ‘yes’ or ‘no’, and the answer must fit on one line; similarly, the ‘`>Synopsis:`’ field expects a short synopsis of the problem, which must also fit on one line. Fill out the fields

as completely as possible. See Section 2.4 “Helpful hints,” page 17, for suggestions as to what kinds of information to include.

In this example, words in *italics* are filled in with pre-configured information:

```
>Submitter-Id: your submitter-id
>Originator:   your name here
>Organization:
    your organization
>Confidential:<[ yes | no ] (one line)>
>Synopsis:    <synopsis of the problem (one line)>
>Severity:    <[non-critical | serious | critical](one line)>
>Priority:    <[ low | medium | high ] (one line)>
>Category:    <name of the product (one line)>
>Class:       <[sw-bug | doc-bug | change-request | support]>
>Release:     <release number (one line)>
>Environment:
         <machine, os, target, libraries (multiple lines)>

>Description:
       <precise description of the problem (multiple lines)>
>How-To-Repeat:
       <code/input/activities to reproduce (multiple lines)>
>Fix:
       <how to correct or work around the problem, if known
        (multiple lines)>
```

When you finish editing the Problem Report, `send-pr` mails it to the address named in the ‘`To:`’ field in the mail header. `send-pr` checks that the complete form contains a valid ‘`>Category:`’.

Your copy of `send-pr` should have already been customized on installation to reflect your ‘`>Submitter-Id:`’. (See Appendix A “Installing `send-pr` on your system,” page 27.) If you don’t know your ‘`>Submitter-Id:`’, you can request it using ‘`send-pr --request-id`’. If your organization is not affiliated with the site you send Problem Reports to, a good generic ‘`>Submitter-Id:`’ to use is ‘`net`’.

If your PR has an invalid value in one of the ENUMERATED fields (see Section 1.2 “Problem Report format,” page 3), `send-pr` places the PR in a temporary file named ‘`/tmp/pbad`*nnnn*’ on your machine. *nnnn* is the process identification number given to your current `send-pr` session. If you are running `send-pr` from the shell, you are prompted as to whether or not you wish to try editing the same Problem Report again. If you are running `send-pr` from Emacs, the Problem Report is placed in the buffer ‘`*send-pr-error*`’; you can edit this file and then submit it with

```
M-x prms-submit-pr
```

Any further mail concerning this Problem Report should be carbon-copied to the PRMS mailing address as well, with the category and identification number in the 'Subject:' line of the message.

```
Subject: Re: PR category/prms-id: original message subject
```

Messages which arrive with 'Subject:' lines of this form are automatically appended to the Problem Report in the '>Audit-Trail:' field in the order received.

2.2 Using send-pr from within Emacs

You can use an interactive send-pr interface from within GNU Emacs to fill out your Problem Report. We recommend that you familiarize yourself with Emacs before using this feature (see section "Introduction" in *GNU Emacs*).

Call send-pr with 'M-x send-pr'.[1] send-pr responds with a Problem Report template preconfigured for the Support Site from which you received send-pr. (If you use send-pr locally, the default Support Site is probably your local site.)

You may also submit problem reports to different Support Sites from the default site. To use this feature, invoke send-pr with

```
C-u M-x send-pr
```

send-pr prompts you for the name of a *site*. *site* is an alias on your local machine which points to an alternate Support Site.

send-pr displays the template and prompts you in the minibuffer with the line:

```
>Category: other
```

Delete the default value 'other' *in the minibuffer* and replace it with the keyword corresponding to your problem (the list of valid categories is in the topmost section of the PR template). For example, if the problem you wish to report has to do with the GNU C compiler, and your support organization accepts bugs submitted for this program under the category 'gcc', delete 'other' and then type 'gcc[RET]'. send-pr replaces the line

```
>Category:       <name of the product (one line)>
```

in the template with

```
>Category:       gcc
```

and moves on to another field.

send-pr provides name completion in the minibuffer. For instance, you can also type 'gc[TAB]', and send-pr attempts to complete the entry

[1] If typing 'M-x send-pr' doesn't work, see your system administrator for help loading send-pr into Emacs.

for you. Typing 'g[TAB]' may not have the same effect if several possible entries begin with 'g'. In that case send-pr cannot complete the entry because it cannot determine whether you mean 'gcc' or, for example, 'gdb', if both of those are possible categories. send-pr continues to prompt you for a valid entry until you enter one.

send-pr prompts you interactively to enter each field for which there is a range of specific choices. If you attempt to enter a value which is not in the range of acceptable entries, send-pr responds with '[No match]' and allows you to change the entry until it contains an acceptable value. This avoids unusable information (at least in these fields) and also avoids typographical errors which could cause problems later.

send-pr prompts you for the following fields:

```
>Category:
>Confidential: (default: no)
>Severity:     (default: serious)
>Priority:     (default: medium)
>Class:        (default: sw-bug)
>Release:
>Synopsis:     (this value is copied to Subject:)
```

After you complete these fields, send-pr places the cursor in the '>Description:' field and displays the message

```
To send the problem report use: C-c C-c
```

in the minibuffer. At this point, edit the file in the main buffer to reflect your specific problem, putting relevant information in the proper fields. See Chapter 3 "An Example," page 19, for a sample Problem Report.

'send-pr' provides a few key bindings to make moving around in a template buffer more simple:

C-c C-f
M-x change-field
: Changes the field under the cursor. edit-pr prompts you for a new value.

M-C-b
M-x prms-backward-field
: Moves the cursor to the beginning of the value of the current field.

M-C-f
M-x prms-forward-field
: Moves the cursor to the end of the value of the current field.

M-p

`M-x prms-previous-field`
: Moves the cursor back one field to the beginning of the value of the previous field.

`M-n`
`M-x prms-next-field`
: Moves the cursor forward one field to the beginning of the value of the next field.

`send-pr` takes over again when you type '`C-c C-c`' to send the message. `send-pr` reports any errors in a separate buffer, which remains in existence until you send the PR properly (or, of course, until you explicitly kill the buffer).

For detailed instructions on using Emacs, see section "Introduction" in *GNU Emacs*.

2.3 Invoking send-pr from the shell

```
send-pr [ site ]
        [ -f problem-report | --file problem-report ]
        [ -t mail-address | --to mail-address ]
        [ --request-id ]
        [ -L | --list ] [ -P | --print ]
        [ -V | --version] [ -h | --help ]
```

site is an alias on your local machine which points to an address used by a Support Site. If this argument is not present, the default *site* is usually the site which you received `send-pr` from, or your local site if you use PRMS locally. (See Section A.2 "Setting a default *site*," page 28.)

Invoking `send-pr` with no options calls the editor named in your environment variable `EDITOR` on a default PR template. If the environment variable `PR_FORM` is set, its value is used as a file name which contains a valid template. If `PR_FORM` points to a missing or unreadable file, or if the file is empty, `send-pr` generates an error message and opens the editor on a default template.

`-f` *problem-report*
`--file` *problem-report*
: Specifies a file, *problem-report*, where a completed Problem Report already exists. `send-pr` sends the contents of the file without invoking an editor. If *problem-report* is '`-`', `send-pr` reads from standard input.

`-t` *mail-address*
`--to` *mail-address*
: Sends the PR to *mail-address*. The default is preset when `send-pr` is configured. *This option is not recommended*; instead, use the argument *site* on the command line.

`-c` *`mail-address`*
`--cc` *`mail-address`*
: Places *`mail-address`* in the `Cc:` header field of the message to be sent.

`--request-id`
: Sends a request for a `>Submitter-Id:` to the Support Site.

`-L`
`--list`
: Prints the list of valid `>Category:` values on standard output. No mail is sent.

`-s` *`severity`*
`--severity` *`severity`*
: Sets the initial value of the `>Severity:` field to *`severity`*.

`-P`
`--print`
: Displays the PR template. If the variable `PR_FORM` is set in your environment, the file it specifies is printed. If `PR_FORM` is not set, `send-pr` prints the standard blank form. If the file specified by `PR_FORM` doesn't exist, `send-pr` displays an error message. No mail is sent.

`-V`
`--version`
: Displays the `send-pr` version number and a usage summary. No mail is sent.

`-h`
`--help`
: Displays a usage summary for `send-pr`. No mail is sent.

2.4 Helpful hints

There is no orthodox standard for submitting effective bug reports, though you might do well to consult the section on submitting bugs for GNU `gcc` in section "Reporting Bugs" in *Using and Porting GNU CC*, by Richard Stallman. This section contains instructions on what kinds of information to include and what kinds of mistakes to avoid.

In general, common sense (assuming such an animal exists) dictates the kind of information that would be most helpful in tracking down and resolving problems in software.

- Include anything *you* would want to know if you were looking at the report from the other end. There's no need to include every minute detail about your environment, although anything that might be different from someone else's environment should be included (your path, for instance).

- Narratives are often useful, given a certain degree of restraint. If a person responsible for a bug can see that A was executed, and then B and then C, knowing that sequence of events might trigger the realization of an intermediate step that was missing, or an extra step that might have changed the environment enough to cause a visible problem. Again, restraint is always in order ("I set the build running, went to get a cup of coffee (Columbian, cream but no sugar), talked to Sheila on the phone, and then THIS happened. . .") but be sure to include anything relevant.
- Richard Stallman writes, "The fundamental principle of reporting bugs usefully is this: **report all the facts**. If you are not sure whether to state a fact or leave it out, state it!" This holds true across all problem reporting systems, for computer software or social injustice or motorcycle maintenance. It is especially important in the software field due to the major differences seemingly insignificant changes can make (a changed variable, a missing semicolon, etc.).
- Submit only *one* problem with each Problem Report. If you have multiple problems, use multiple PRs. This aids in tracking each problem and also in analyzing the problems associated with a given program.
- It never hurts to do a little research to find out if the bug you've found has already been reported. Most software releases contain lists of known bugs in the Release Notes which come with the software; see your system administrator if you don't have a copy of these.
- The more closely a PR adheres to the standard format, the less interaction is required by a database administrator to route the information to the proper place. Keep in mind that anything that requires human interaction also requires time that might be better spent in actually fixing the problem. It is therefore in everyone's best interest that the information contained in a PR be as correct as possible (in both format and content) at the time of submission.

3 An Example

Cygnus Support in Mountain View, CA, uses PRMS and `send-pr` extensively for their support activities. As a support company, Cygnus finds problem tracking to be a crucial part of everyday business. Cygnus supports the GNU compiling tools (including PRMS and `send-pr`) over several many platforms

With each shipment of the Cygnus Support Developer's Kit, customers receive the latest version of `send-pr`, which contains an up-to-date listing of valid categories (values for the `>Category:` field). Using these tools, Cygnus' customers can communicate their problems to Cygnus effectively and receive automatic confirmation of receipt as well as notification of changes in the status of their reported problems. Much of Cygnus' support mechanism relies on electronic mail.

As an example, let's pretend we're a customer of Cygnus Support, and that we're having a problem compiling some of our software using the GNU C compiler, which Cygnus supports.

Assume that we're getting an error in our `bifrabulator` program wherein the '`prestidigitation`' routines don't match with the '`whatsitsname`'. We've made sure we're following the rules of the program and checked the Release Notes from Cygnus and found that the bug isn't already known. In other words, we're pretty sure we've found a bug.

Our first step is to call `send-pr`. It really doesn't matter whether we use `send-pr` from the shell or from within Emacs. Indeed, if we use Emacs as a primary editor, calling `send-pr` from the shell is likely to start `send-pr` in an Emacs buffer anyway. So, since our company, *Imaginary Software, Ltd.*, uses GNU software extensively, we're pretty familiar with Emacs, so from within Emacs we type

```
M-x send-pr
```

and we're greeted with the following screen:

```
SEND-PR: -*- text  -*-
SEND-PR: Lines starting with 'SEND-PR' will be removed
SEND-PR: automatically as well as all comments (the text
SEND-PR: below enclosed in '<' and '>').
SEND-PR: Please consult the manual if you are not sure
SEND-PR: how to fill out a problem report.
SEND-PR:
SEND-PR: Choose from the following categories:
SEND-PR:
SEND-PR:           bfd       binutils  bison
SEND-PR: byacc     clib      config    cvs         diff
SEND-PR: doc       emacs     flex      g++         gas
SEND-PR: gcc       gdb       glob      gprof       grep
SEND-PR: info      ispell    kerberos  ld          libg++
SEND-PR: libiberty make      makeinfo  mas         newlib
SEND-PR: other     patch     rcs       readline    send-pr
SEND-PR: test      texindex  texinfo   texinfo.tex
SEND-PR: bifrabulator  <---note: this one is fake
SEND-PR:
To: cygnus-bugs@cygnus.com
Subject:
From: jeffrey@imaginary.com
Reply-To: jeffrey@imaginary.com
X-send-pr-version: send-pr 3.97-96q1

>Submitter-Id:  imaginary
>Originator:    Jeffrey Osier
>Organization:
Imaginary Software, Ltd.
>Confidential:  <[ yes | no ] (one line)>
>Synopsis:      <synopsis of the problem (one line)>
>Severity:      <[ non-critical | serious | critical ] (one line)>
>Priority:      <[ low | medium | high ] (one line)>
>Category:      <name of the product (one line)>
>Class:         <[sw-bug|doc-bug|change-request|support](oneline)>
>Release:       <release number or tag (one line)>
>Environment:
         <machine, os, target, libraries (multiple lines)>
System: SunOS imaginary.com 4.1.1 1 sun4
Architecture: sun4

>Description:
       <precise description of the problem (multiple lines)>
>How-To-Repeat:
       <code/input/activities to reproduce (multiple lines)>
>Fix:
-----Emacs: *send-pr*   (send-pr Fill)----All------------------
>Category: other[]
```

We know from past experience that we need to set certain information into each field, so we compile all the information we know about our problem. We have some sample code which we know should work, even though it doesn't, so we'll include that. Below is the completed PR; we send this using `C-c C-c`. (The comments have been truncated).

```
SEND-PR: Lines starting with 'SEND-PR' will be removed
SEND-PR: automatically as well as all comments (the text
SEND-PR: ...
SEND-PR:
To: cygnus-bugs@cygnus.com
Subject: bifrabulator routines don't match
From: jeffrey@imaginary.com
Reply-To: jeffrey@imaginary.com
X-send-pr-version: send-pr 3.97-96q1

>Submitter-Id:  imaginary
>Originator:    Jeffrey Osier
>Organization:
Imaginary Software, Ltd.
>Confidential:  no
>Synopsis:      bifrabulator routines don't match
>Severity:      serious
>Priority:      medium
>Category:      bifrabulator
>Class:         sw-bug
>Release:       progressive-930101
>Environment:
System: SunOS imaginary.com 4.1.1 1 sun4
Architecture: sun4 (SPARC)

>Description:
   the following code I fed into the bifrabulator came back
   with a strange error.  apparently, the prestidigitation
   routine doesn't match with the whatsitsname in all cases.

>How-To-Repeat:
   call the bifrabulator on the following code.
   code sample...

>Fix:

-----Emacs: *send-pr*   (send-pr Fill)----All-----------------

To send the problem report use: C-c C-c
```

We type `C-c C-c`, and off it goes. Now, we depend on Cygnus Support to figure out the answer to our problem.

Soon afterward, we get the following message from Cygnus:

```
From: prms (PRMS management)
Sender: prms-admin
Reply-To: hacker@cygnus.com
To: jeffrey@imaginary.com
Subject: Re: bifrabulator/1425: routines don't match

Thank you very much for your problem report.
It has the internal identification: g++/1425.
The individual assigned to look at your bug is:  hacker
(F.B. Hacker)

Category: bifrabulator
Responsible: hacker
Synopsis: bifrabulator routines don't match
Arrival-Date: Sat Feb 30 03:12:55 1993
```

This is our receipt that the bug has been accepted and forwarded to the responsible party.

A while later, we get the analysis:

```
To:  jeffrey@imaginary.com
From:  hacker@cygnus.com
Subject:  Re: bifrabulator/1425: routines don't match
Reply-To: hacker@cygnus.com

Got your message, Jeff.  It seems that the bifrabulator was
confusing the prestidigitation routines with the realitychecker
when lexically parsing the whatsitsname.

I'm working on robustisizing the bifrabulator now.

How about lunch next week?
--
F.B. Hacker
Cygnus Support, Mountain View, CA  415 903 1400
#include <std-disclaimer.h>
```

About the same time, we get another message from Cygnus.

```
From: hacker@cygnus.com
To:  jeffrey@imaginary.com
Subject:  Re: bifrabulator/1425: doesn't match prestidig
Reply-To:  hacker@cygnus.com


             'F.B. Hacker' changed the state to 'analyzed'.

State-Changed-From-To: open-analyzed
State-Changed-By: hacker
State-Changed-When: Fri Feb 31 1993 08:59:16 1993
State-Changed-Why:
    figured out the problem, working on a patch this afternoon
--
F.B. Hacker
Cygnus Support, Mountain View, CA  415 903 1400
#include <std-disclaimer.h>
```

The bug has now been analyzed, and Cygnus is working on a solution.

Sometime later, we get more mail from F.B.:

```
To:  jeffrey@imaginary.com
From:  hacker@cygnus.com
Subject:  Re: bifrabulator/1425: routines don't match
Reply-To: hacker@cygnus.com

There's a patch now that you can ftp over and check out.

Hey, that joke you sent me was great!  The one about the
strings walking into a bar...  my boss laughed for an hour!
--
F.B. Hacker
Cygnus Support, Mountain View, CA  415 903 1400
#include <std-disclaimer.h>
```

```
From: hacker@cygnus.com
To:  jeffrey@imaginary.com
Subject:  Re: bifrabulator/1425: doesn't match prestidig
Reply-To:  hacker@cygnus.com


             'F.B. Hacker' changed the state to 'feedback'.

State-Changed-From-To: analyzed-feedback
State-Changed-By: hacker
State-Changed-When: Fri Feb 31 1993 23:43:16 1993
State-Changed-Why:
    got the patch finished, notified Jeff at Imaginary Software
--
F.B. Hacker
Cygnus Support, Mountain View, CA  415 903 1400
#include <std-disclaimer.h>
```

The bug has gone into *feedback* status now, until we get the patch, install it and test it. When everything tests well, we can mail F.B. back and tell him the bug's been fixed, and he can change the state of the PR from *feedback* to *closed*.

Following is a list of valid '`>Category:`' entries that are supported by Cygnus.

Valid Categories

`bfd` GNU binary file descriptor library.

`bifrabulator`
This one doesn't actually exist.

`binutils` GNU utilities for binary files (`ar`, `nm`, `size`...).

`bison` GNU parser generator.

`byacc` Free parser generator.

`config` Cygnus Support Software configuration and installation.

`cvs` Concurrent Version System.

`diff` GNU `diff` program.

`doc` Documentation and manuals.

`emacs` GNU Emacs editor and related functions.

`flex` GNU lexical analyzer.

`g++`	GNU C++ compiler.
`gas`	GNU assembler.
`gcc`	GNU C compiler.
`gdb`	GNU source code debugger.
`glob`	The filename globbing functions.
`gprof`	GNU profiler.
`grep`	GNU `grep` program.
`info`	GNU `info` hypertext reader.
`ispell`	GNU spelling checker.
`kerberos`	Kerberos authentication system.
`ld`	GNU linker.
`libc`	Cygnus Support C Support Library.
`libg++`	GNU C++ class library.
`libiberty`	GNU '`libiberty`' library.
`libm`	Cygnus Support C Math Library.
`make`	GNU `make` program.
`makeinfo`	GNU utility to build Info files from Texinfo documents.
`mas`	GNU Motorola syntax assembler.
`newlib`	Cygnus Support C Support and Math Libraries.
`patch`	GNU bug patch program.
`prms`	GNU Problem Report Management System.
`rcs`	Revision Control System.
`readline`	GNU `readline` library.
`send-pr`	GNU Problem Report submitting program.
`test`	Category to use when testing `send-pr`.
`texindex`	GNU documentation indexing utility.
`texinfo`	GNU documentation macros.
`other`	Anything which is not covered by the above categories.

Appendix A Installing send-pr on your system

If you receive send-pr as part of a larger software distribution, it probably gets installed when the full distribution is installed. If you are using PRMS at your site as well, you must decide where send-pr sends Problem Reports by default; see Section A.2 “Setting a default *site*,” page 28.

A.1 Installing send-pr by itself

Install send-pr by following these steps (you may need root access in order to change the ‘aliases’ file and to install send-pr):

- Unpack the distribution into a directory which we refer to as *srcdir*.
- Edit the file ‘Makefile’ to reflect local conventions. Specifically, you should edit the variable ‘prefix’ to alter the installation location. The default is ‘/usr/local’. All files are installed under ‘prefix’ (see below).
- *Run*

  ```
  make all install [ info ] [ install-info ] [ clean ]
  ```

  The targets mean the following:

  `all` Builds send-pr and install-sid

  `install` Installs the following:

  install-sid
  send-pr into ‘*prefix*/bin’

  send-pr.1
  into ‘*prefix*/man/man1’

  site the list of valid *categories* for the Support Site from which you received send-pr, installed as ‘*prefix*/lib/prms/*site*’

  send-pr.el
  into ‘*prefix*/lib/emacs/lisp’[1]

  `info` (*optional*)
  Builds ‘send-pr.info’ from ‘send-pr.texi’ (‘send-pr.info’ is included with this distribution)

[1] If your main Emacs lisp repository is in a different directory from this, substitute that directory for ‘*prefix*/lib/emacs/lisp’.

`install-info` (*optional*)
: Installs '`send-pr.info`' into '*`prefix`*`/info`'

`clean` (*optional*)
: Removes all intermediary build files that can be rebuilt from source code

- Run

  ```
  install-sid your-sid
  ```

  where *`your-sid`* is the identification code you received with `send-pr`. `send-pr` automatically inserts this value into the template field '`>Submitter-Id:`'. If you've downloaded `send-pr` from the Net, use '`net`' for this value.
- Place the following line in '*`prefix`*`/lib/emacs/lisp/default.el`', or instruct your users to place the following line in their '`.emacs`' files:

  ```
  (autoload 'send-pr "send-pr" "Submit a Problem Report." t)
  ```
- Create a mail alias for the Support Site from which you received `send-pr`, and for every site with which you wish to use `send-pr` to communicate. Each alias must have a suffix of '`-prms`'. The Support Site(s) will provide the correct addresses where these aliases should point. For instance, edit your mail aliases file to contain something like:

  ```
  # support sites; for use with send-pr
  cygnus-prms:     bugs@cygnus.com            # Cygnus Support
  bumblebee-prms:  bumblebugs@bumblebee.com   # Bumblebee Inc.
  mycompany-prms:  bugs@my.company.com (if you use prms locally)
  ```

  `send-pr` automatically searches for these aliases when you type

  ```
  send-pr cygnus
  send-pr bumblebee
  send-pr site...
  ```

  `send-pr` also uses *`site`* to determine the categories of problems accepted by the site in question by looking in

  ```
  prefix/lib/prms/site
  ```

A.2 Setting a default *`site`*

`send-pr` is capable of sending Problem Reports to any number of Support Sites, using mail aliases which have '`-prms`' appended them. `send-pr` automatically appends the suffix, so that when you type

```
send-pr site
```

the Problem Report goes to the address noted in the '`aliases`' file as '*`site`*`-prms`'. You can do this in the Emacs version of `send-pr` by invoking the program with

```
C-u M-x send-pr
```

You are prompted for *site*.

site is also used to error-check the '>Category:' field, as a precaution against sending mistaken information (and against sending information to the wrong site).

You may also simply type

```
send-pr
```

from the shell (or 'M-x send-pr' in Emacs), and the Problem Report you generate will be sent to the *site*, which is usually the site from which you received your distribution of send-pr. If you use PRMS at your own organization, the default is usually your local address for reporting problems.

To change this, simply edit the file 'Makefile' before installing and change the line

```
PRMS_SITE = site
```

to reflect the site where you wish to send PRs by default.

Index

G

H

I

K

L

M

N

O

P

R

S

T

U

V