This manual is for GNU GNATS (version 4.2.0, 8 August 2006), the GNU Bug Tracking System.


Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, with the Front-Cover Texts being “A GNU Manual,” and with the Back-Cover Texts as in (a) below. A copy of the license is included in the section entitled “GNU Free Documentation License.”

(a) The FSF’s Back-Cover Text is: “You have freedom to copy and modify this GNU Manual, like GNU software. Copies published by the Free Software Foundation raise funds for GNU development.”
Table of Contents

Overview .......................................................... 1

1 Introducing GNATS ................................. 3
   1.1 The database paradigm .......................... 3
   1.2 Flowchart of GNATS activities .............. 5
   1.3 States of Problem Reports .................... 5
   1.4 Problem Report format ....................... 6
       1.4.1 Field datatypes reference .......... 7
       1.4.2 Mail header fields .................. 8
       1.4.3 Problem Report fields ............... 8

2 The GNATS User Tools ....................... 13
   2.1 Environment variables and GNATS tools ...... 13
   2.2 Submitting Problem Reports ................ 13
       2.2.1 Invoking send-pr from the shell .... 13
       2.2.2 Using send-pr from within Emacs .... 14
       2.2.3 The Problem Report template ......... 14
       2.2.4 Submitting a Problem Report via direct e-mail .... 16
       2.2.5 Helpful hints ........................ 17
   2.3 Editing existing Problem Reports ........... 17
       2.3.1 Invoking edit-pr from the shell ...... 18
       2.3.2 Following up via direct email ......... 19
   2.4 Querying the database ..................... 20
       2.4.1 Invoking query-pr .................... 20
       2.4.2 Formatting query-pr output .......... 24
       2.4.3 Query expressions ................... 25
       2.4.4 Example queries .................... 27
   2.5 The Emacs interface to GNATS ............. 27
       2.5.1 Viewing Problem Reports .............. 27
       2.5.2 Querying Problem Reports .......... 28
       2.5.3 Submitting new Problem Reports ...... 28
       2.5.4 Editing Problem Reports ............. 29
       2.5.5 The Problem Report editing buffer ... 29
       2.5.6 Changing the database ............... 30
       2.5.7 dbconfig mode ........................ 30
       2.5.8 Other commands ...................... 30
       2.5.9 Customization ......................... 30
3 Installing GNATS ......................................................... 31
  3.1 Configuring and compiling the software ...................... 31
  3.2 Installing the utilities ........................................ 34
  3.3 Installing the default database ............................... 34
  3.4 Setting up periodic jobs ...................................... 35
  3.5 Setting up mail aliases ....................................... 36
  3.6 Installing the daemon ......................................... 36
  3.7 Installing the user tools ..................................... 37
    3.7.1 User tools on a local network ............................ 38
    3.7.2 User tools for remote users .............................. 38
  3.8 Upgrading from older versions ............................... 39
    3.8.1 Overview ................................................ 39
    3.8.2 Upgrading .............................................. 40

4 GNATS Administration .............................................. 43
  4.1 Overview of GNATS configuration .............................. 44
  4.2 The databases file ........................................... 45
  4.3 The dbconfig file ............................................ 46
    4.3.1 Overall database configuration ............................ 46
    4.3.2 Individual field configuration ............................ 48
    4.3.3 Field datatypes ......................................... 49
    4.3.4 Edit controls ........................................... 52
    4.3.5 Named query definitions ................................ 53
    4.3.6 Audit-trail formats .................................... 53
    4.3.7 Outgoing email formats ................................ 54
    4.3.8 Index file description .................................. 55
    4.3.9 Initial PR input fields .................................. 57
  4.4 Other database-specific config files ........................ 58
    4.4.1 The categories file ...................................... 58
    4.4.2 The responsible file .................................... 59
    4.4.3 The submitters file ..................................... 59
    4.4.4 The states file .......................................... 60
    4.4.5 The addresses file ...................................... 61
    4.4.6 The classes file ........................................ 62
  4.5 The send-pr.conf file ......................................... 62
  4.6 Administrative data files ..................................... 63
    4.6.1 The index file .......................................... 63
    4.6.2 The current file ........................................ 63
  4.7 Administrative utilities ...................................... 64
    4.7.1 Adding another database ................................ 64
    4.7.2 Adding a problem category ............................... 64
    4.7.3 Removing a problem category ............................. 64
    4.7.4 Regenerating the index .................................. 64
    4.7.5 Checking database health ................................. 65
    4.7.6 Managing user passwords ................................ 66
  4.8 Internal utilities ............................................. 66
    4.8.1 Handling incoming traffic ................................ 66
    4.8.2 Processing incoming traffic .............................. 67
Overview

This manual documents GNATS, the GNU Problem Report Management System, version 4.2.0. GNATS is a bug-tracking tool designed for use at a central Support Site. Users who experience problems use electronic mail, web-based or other clients communicating with the GNATS network daemon running at the support site or direct database submissions to communicate these problems to maintainers at that Support Site. GNATS partially automates the tracking of these Problem Reports (PRs) by:

- organizing problem reports into a database and notifying responsible parties of suspected bugs;
- allowing support personnel and their managers to edit and query accumulated bugs; and
- providing a reliable archive of problems (and their subsequent solutions) with a given program.

GNATS offers many of the same features offered by more generalized databases, including editing, querying, and basic reporting. The GNATS database itself is an ordered repository for problem reports; each PR receives a unique, incremental PR number which identifies it throughout its lifetime. For a discussion on the working system adopted by GNATS, see Section 1.1 [The database paradigm], page 3.

You can access the submitting, editing, and querying functions of GNATS from within GNU Emacs. See Chapter 2 [The GNATS user tools], page 13.
1 Introducing GNATS

Any support organization realizes that a large amount of data flows back and forth between the maintainers and the users of their products. This data often takes the form of problem reports and communication via electronic mail. GNATS addresses the problem of organizing this communication by defining a database made up of archived and indexed problem reports.

GNATS was designed as a tool for software maintainers. It consists of several utilities which, when used in concert, formulate and administer a database of Problem Reports grouped by site-defined problem categories. It allows a support organization to keep track of problems (hence the term Problem Report) in an organized fashion. Essentially, GNATS acts as an active archive for field-separated textual data.

1.1 The database paradigm

It is in your best interest as the maintainer of a body of work to organize the feedback you receive on that work, and to make it easy for users of your work to report problems and suggestions.

GNATS makes this easy by automatically filing incoming problem reports into appropriate places, by notifying responsible parties of the existence of the problem and (optionally) sending an acknowledgment to the submitter that the report was received, and by making these Problem Reports accessible to queries and easily editable. GNATS is a database specialized for a specific task.

GNATS was designed for use at a Support Site that handles a high level of problem-related traffic. It maintains Problem Reports in the form of text files with defined fields (see Section 1.4 [GNATS data fields], page 6). The location of the database, as well as the categories it accepts as valid, the maintainers for whom it provides service, and the submitters from whom it accepts Problem Reports, are all definable by the Support Site. See Chapter 4 [GNATS administration], page 43.

Each PR is a separate file within a main repository (see Appendix A [Where GNATS lives], page 73). Editing access to the database is regulated to maintain consistency. To make queries on the database faster, an index is kept automatically (see Section 4.6.1 [The index file], page 63).

We provide several software tools so that users may submit new Problem Reports, edit existing Problem Reports, and query the database.

- **send-pr** is used by both product maintainers and the end users of the products they support to submit PRs to the database.
- **edit-pr** is used by maintainers when editing problem reports in the database.
- Maintainers, managers and administrators can use **query-pr** to make inquiries about individual PRs or groups of PRs.

Other interfaces to GNATS include Gnsatsweb, a web-based tool which provides features for submitting and editing PRs and querying the database, and TkGnats, a Tcl/Tk-based frontend. These tools are distributed together with GNATS.

At the Support Site, a GNATS administrator is charged with the duty of maintaining GNATS. These duties are discussed in detail in Chapter 4 [GNATS Administration], page 43,
and generally include configuring gnats for the Support Site, editing PRs that gnats cannot process, pruning log files, setting up new problem categories, backing up the database, and distributing send-pr so that others may submit Problem Reports.

Responsibility for a given Problem Report initially depends on the category of the problem. Optionally, an automated reminder can be sent to the responsible person if a problem report is neglected for a requisite time period. See Section 4.1 [Overview of GNATS configuration], page 44.

GNATS does not have the ability to decipher random text. If there is no default category set, any problem reports which arrive in a format gnats does not recognize are placed in a separate directory pending investigation by the gnats administrator (see Chapter 4 [GNATS Administration], page 43).

Once a problem is recorded in the database, work can begin toward a solution. A problem’s initial state type is open (see Section 1.3 [States of Problem Reports], page 5). An acknowledgment may be sent to the originator of the bug report (depending on whether this feature has been switched on in the gnats configuration). GNATS forwards copies of the report to the party responsible for that problem category and to the person responsible for problems arriving from that submitter.

When a problem has been identified, the maintainer can change its state to analyzed, and then to feedback when a solution is found. GNATS may be configured so that each time the state of a PR changes, the submitter of the problem report is notified of the reason for the change. If the party responsible for the PR changes, the previous responsible party and the new responsible party receive notice of the change. The change and reason are also recorded in the Audit-Trail field of the Problem Report. (See Section 2.3 [Editing existing Problem Reports], page 17. For information on the Audit-Trail field, see Section 1.4 [Problem Report format], page 6.)

When the originator of the Problem Report confirms that the solution works, the maintainer can change the state to closed. If the PR cannot be closed, the maintainer can change its state to suspended as a last resort. (For a more detailed description of the standard states, see Section 1.3 [States of Problem Reports], page 5.)

It should be emphasized that what we describe here is the default way that GNATS works, but as of version 4, GNATS is extremely customizable, allowing sites to tailor almost every aspect of its behavior. See for instance Section 4.3 [The dbconfig file], page 46 and See Section 1.3 [States of Problem Reports], page 5.)
1.2 Flowchart of GNATS activities

This informal flowchart shows the relationships of the GNATS tools to each other and to the files with which they interoperate.

1.3 States of Problem Reports

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

Unless your site has customized states (see Section 4.4.4 [states file], page 60), GNATS uses these states:

- **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.

### 1.4 Problem Report format

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

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.

Problem Report fields are denoted by a keyword which begins with ‘>’ and ends with ‘:’, as in ‘>Confidential:’. Fields belong to one of eight data types as listed in Section 1.4.1 [Field datatypes reference], page 7. As of version 4 of GNATS all characteristics of fields, such as field name, data type, allowed values, permitted operations, on-change actions etc. are configurable.

For details, see see Section 4.3 [The dbconfig file], page 46.

### Example Problem Report

The following is an example Problem Report with the fields that would be present in a standard GNATS configuration. Mail headers are at the top, followed by GNATS fields, which begin with ‘>’ and end with ‘:’. The ‘Subject:’ line in the mail header and the `Synopsis` field are usually duplicates of each other.
1.4.1 Field datatypes reference

The following is a short reference to the characteristics of the different field types.

For details, see Section 4.3.3 [Field datatypes], page 49.

**text**  A one-line text string.

**multitext**  Multiple lines of text.

**enum**  The value in this field is required to be from a list of specified values, defined at the Support Site.

(See Section 4.3 [The dbconfig file], page 46, for details.

**multienum**  Similar to the enum datatype, except that the field can contain multiple values.
enumerated-in-file
Similar to enum, but the allowed field values are listed in a separate file on the GNATS server.

multi-enumerated-in-file
Similar to enumerated-in-file, except that the field can contain multiple values.

date
Used to hold dates.

integer
Used to hold integer numbers.

1.4.2 Mail header fields
A Problem Report may contain any mail header field described in the Internet standard RFC-822. The send-pr tool can be configured either to submit PRs to the support site by e-mail or by talking directly to the gnatsd network daemon on the GNATS server. This is also true for other client tools such as Gnatsweb. Even when these tools are set up submit PRs directly to gnatsd, they will still include mail header fields that identify the sender and the subject in the submitted PR:

To: The mail address for the Support Site, automatically supplied by the tool used to submit the PR or by the originator if plain e-mail was used.

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

From: Supplied automatically when PRs are submitted by plain e-mail and when well-behaved tools such as send-pr are used; should always contain the originator’s e-mail address.

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

One of the configurable options for GNATS is whether or not to retain Received-By headers, which often consume a lot of space and are not often used. See Section 4.3 [The dbconfig file], page 46.

1.4.3 Problem Report fields
In a standard GNATS installation, certain fields will always be present in a Problem Report. If a PR arrives without one or more of these fields, GNATS will add them, and if they have default values set by the configuration at the Support Site, they will be filled in with these values. Certain tools such as send-pr are set up to provide sensible defaults for most fields (see Section 4.5 [The send-pr.conf configuration file], page 62.)

In the list below, the field type (text, multitext, enumerated, etc.) is supplied in parantheses. The different field types are explained briefly in Section 1.4.1 [Field datatypes reference], page 7.

Submitter-Id
(enumerated-in-file) 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 \texttt{send-pr} with the \texttt{--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.

See Section 4.4.3 [The submitters file], page 59, for details.

\textbf{Notify-List}

\texttt{(text)} Comma-separated list of e-mail-addresses of people to notify when the PR changes significantly, i.e. when the Audit-Tail changes. This list is independent from the Notify subfield of entries in the \texttt{categories} file of the GNATS database.

\textbf{Originator}

\texttt{(text)} Originator’s real name. Note that the Submitter and Originator might not be the same person/entity in all cases.

\textbf{Organization}

\texttt{(multitext)} The originator’s organization.

\textbf{Confidential}

\texttt{(enum)} 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.

\textbf{Synopsis}

\texttt{(text)} One-line summary of the problem. \texttt{send-pr} copies this information to the \texttt{Subject} line when you submit a Problem Report.

\textbf{Severity}

\texttt{(enum)} The severity of the problem. By default, accepted values include:

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

\texttt{serious} The product, component or concept is not working properly or significant functionality is missing. Problems that would otherwise be considered ‘critical’ are usually rated ‘serious’ when a workaround is known.

\texttt{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.

\textbf{Priority}

\texttt{(enumerated)} How soon the originator requires a solution. Accepted values include:

\texttt{high} A solution is needed as soon as possible.

\texttt{medium} The problem should be solved in the next release.
The problem should be solved in a future release.

Category (enumerated-in-file) The name of the product, component or concept where the problem lies. The values for this field are defined by the Support Site. See Section 4.4.1 [The categories file], page 58, for details.

Class (enumerated-in-file) The class of a problem in a default GNATS installation 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.
- **mistaken** No problem, user error or misunderstanding. This value can only be set by tools at the Support Site, since it has no meaning for ordinary submitters.

See Section 4.4.6 [The classes file], page 62, for details.

Release (text) Release or version number of the product, component or concept.

Environment (multitext) Description of the environment where the problem occurred: 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 2.2.5 [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 he or she has solved the problem.)

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

Number (enumerated) The incremental identification number for this PR. This is included in the automated reply to the submitter (if that feature of GNATS is
activated; see Section 4.3 [The dbconfig file, page 46]. It is also included in the copy of the PR that is sent to the maintainer.

The Number field is often paired with the Category field as

category/number

in subsequent email messages. This is GNATS’ way of tracking followup messages that arrive by mail so that they are filed as part of the original PR.

State (enumerated) The current state of the PR. In default GNATS installations, 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. Support organizations may also choose to temporarily ”park” PRs that are awaiting further input from the submitter under this state.
- 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 1.3 [States of Problem Reports], page 5.

Responsible (text) The person at the Support Site who is responsible for this PR. GNATS retrieves this information from the categories file (see Section 4.4.1 [The categories file], page 58).

Arrival-Date (date) The time that this PR was received by GNATS. The date is provided automatically by GNATS.

Date-Required (date) The date by which a fix is required. This is up to the maintainers at the Support Site to determine, so this field is not available until after the PR has been submitted.

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 GNATS related to this PR, in the order received. GNATS needs to find a reference to the PR in the Subject field of received email in order to be able to file it correctly, see Section 2.3.2 [Following up via direct email], page 19.

Unformatted
    Any random text found outside the fields in the original Problem Report.

    During a Problem Report’s journey from ‘open’ to ‘closed’, two more fields, Last-Modified and Closed Date (both of type date) will be added.
2 The GNATS User Tools

This chapter describes the user tools distributed with GNATS. The GNATS administrative and internal tools are described in Chapter 4 [GNATS Administration], page 43. The user tools provide facilities for initial submission, querying and editing of Problem Reports:

send-pr Used by anyone who has a problem with a body of work to submit a report of the problem to the maintainers of that work (see Section 2.2 [Submitting Problem Reports], page 13).

query-pr Used to query the GNATS database (see Section 2.4 [Querying the database], page 20).

edit-pr Used to edit Problem Reports (to record new data, to change the responsible party, etc.) (see Section 2.3 [Editing existing Problem Reports], page 17).

2.1 Environment variables and GNATS tools

All the GNATS user tools honor the GNATSDB environment variable which is used to determine which database to use. For a local database, it contains the name of the database to access.

For network access via gnatstd, it contains a colon-separated list of strings that describe the remote database in the form

server:port:dbname:username:password

Any of the fields may be omitted except for server, but at least one colon must appear; otherwise, the value is assumed to be the name of a local database.

If GNATSDB is not set and no command-line options are used to specify the database, it is assumed that the database is local and that its name is ‘default’.

2.2 Submitting Problem Reports

Use send-pr to submit Problem Reports to the database. send-pr is a shell script which composes a template for submitters to complete.

You can invoke send-pr from a shell prompt, or from within GNU Emacs using ‘M-x send-pr’ (see Section 2.5.3 [Submitting Problem Reports from Emacs], page 28.)

2.2.1 Invoking send-pr from the shell

send-pr [ -b | --batch ]
[ -d database | --database database ]
[ -f file | --file file ]
[ -p | --print ] [ --request-id ]
[ -s severity | --severity severity ]
[ -V | --version ] [ -h | --help ]

Invoking send-pr with no options assumes that you want to submit to the local GNATS database named default and calls the editor named in your environment variable EDITOR on a PR template for this database.

-b

--batch Suppresses printing of most of the messages send-pr usually prints while running.
-d database, --database database
Specifies the database to which the PR is to be submitted; if no database is specified, the local database named default is assumed. This option overrides the database specified in the GNATSDB environment variable. database can also be set to a remote database by using the format for GNATSDB described in Section 2.1 [Environment variables and gnats tools], page 13.

-f problem-report
--file problem-report
Specifies a file, problem-report, where a completed Problem Report or a PR template exists. send-pr verifies that the contents of the file constitute a valid PR and asks you if you want to edit it or send it directly. If the PR text is invalid you will be told what is wrong and be given the option to edit it. If problem-report is ‘-‘, send-pr reads from standard input.

-p
--print
Displays the PR template for the specified database, or if the -d or --database options aren’t specified, print the template for the local default database. No PR is submitted.

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

-s severity
--severity severity
Sets the initial value of the Severity field to severity.

-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.

2.2.2 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’. send-pr responds with a preconfigured Problem Report template. The Emacs interface is described in more detail in a separate section, See Section 2.5 [The Emacs interface to GNATS], page 27.

2.2.3 The Problem Report template
Invoking send-pr presents a PR template with a number of fields already filled in with default values for the database you are submitting to. 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:

---
1 If typing ‘M-x send-pr’ doesn’t work, see your system administrator for help loading gnats.el into Emacs.
• Comments
• Mail Header
• GNATS fields

The Comments section at the top of the template contains basic instructions for completing the Problem Report, as well as a list of valid entries for the Category field. One (and only one) of these values should be placed in the Category field further down in the Problem Report.

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:

The comments lines 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.)

The Mail Header section of the template contains a standard mail header constructed by send-pr. send-pr can be set up to submit PRs by e-mail or by speaking directly to the GNATS server, but since this header is part of the standard format of Problem Reports, send-pr includes it even when it is set up to speak directly to the server.

To: PR submission address
Subject: complete this field
From: your-login@your-site
Reply-To: your-login@your-site
X-send-pr-version: send-pr 4.2.0

send-pr automatically completes all the mail header fields except the Subject line with default values. (See Section 1.4 [Problem Report format], page 6.)

The GNATS fields below the mail header form the bulk of a GNATS Problem Report.

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 2.2.5 [Helpful hints], page 17, for suggestions as to what kinds of information to include.

The mechanisms send-pr uses to fill in default values is as follows: Your preconfigured Submitter-Id is taken from the local send-pr.conf configuration file. 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. If these variables aren't set in your environment, send-pr uses the values set in the local send-pr.conf configuration file, if that exists. If not, these values are left blank in the template. 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.
In this example, words in *italics* are filled in with pre-configured information:

```plaintext
>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` validates the contents and if it looks OK either submits it directly to the GNATS server or submits it by mail to the address named in the To field in the mail header.

If your PR contains one or more invalid field values, `send-pr` places the PR in a temporary file named `/tmp/pbadnnnn` 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 `*gnats-send*`; you can edit this file and then submit it with `C-c C-c`.

### 2.2.4 Submitting a Problem Report via direct e-mail

In addition to using `send-pr`, there is another way to submit a problem report. You can simply send an e-mail message to the PR submission e-mail address of the support site (This address should be published by the support site.)

When you send unformatted e-mail to this address, GNATS processes the message as a new problem report, filling in as many fields from defaults as it can:

**Synopsis** The Synopsis field is filled in by the Subject header of the e-mail message.

**Submitter ID** GNATS will try to derive the Submitter field from the address in the From header of the e-mail.

**Description** All of the text in the body of the e-mail message is put into the Description field.

Other fields, such as Category, Version, Severity, etc. are set to default values as defined by the GNATS administrator.
2.2.5 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.

2.3 Editing existing Problem Reports

Use edit-pr to make changes to existing PRs in the database. This tool can be invoked both from a shell prompt or from within GNU Emacs using ‘M-x edit-pr’.

edit-pr first examines the PR you wish to edit and locks it if it is not already locked. This is to prevent you from editing a PR at the same time as another user. If the PR you wish to edit is already in the process of being edited, edit-pr tells you the name of the person who owns the lock.
You may edit any non-readonly fields in the database. We recommend that you avoid deleting any information in the Text and MultiText fields (such as Description and How-To-Repeat (see Section 1.4 [Problem Report format], page 6). We also recommend that you record the final solution to the problem in the Fix field for future reference. Note that heavily customized installations of GNATS may have differently named fields, and sites using such installations should provide their own set of routines and instructions regarding how PRs should be treated throughout their life span.

After the PR has been edited, it is then resubmitted to the database, and the index is updated (see Section 4.6.1 [The index file], page 63). For information on pr-edit, the main driver for edit-pr, see Section 4.8 [Internal utilities], page 66.

If you change a field that requires a reason for the change, such as the Responsible or State fields in the default configuration, edit-pr prompts you to supply a reason for the change. A message is then appended to the Audit-Trail field of the PR with the changed values and the change reason.

Depending on how the database is configured, editing various fields in the PR may also cause mail to be sent concerning these changes. In the default configuration, any fields that generate ‘Audit-Trail’ entries will also cause a copy of the new ‘Audit-Trail’ message to be sent.

Mail received at the PR submission email address and recognized by GNATS as relating to an existing PR is also appended to the ‘Audit-Trail’ field, see Section 2.3.2 [follow-up via email], page 19.

2.3.1 Invoking edit-pr from the shell

The usage for edit-pr is:

```
edit-pr [ -V | --version ] [ -h | --help ]
[ -d database | --database database ] PR Number
```

Network-mode-only options:

```
[ --host host | -H host ] [ --port port ]
[ --user user | -v user ]
[ --passwd passwd | -w passwd ]
```

The options have the following meaning:

- **-h, --help**
  Prints a brief usage message for edit-pr.

- **-V, --version**
  Prints the version number for edit-pr.

- **-d database, --database database**
  Specifies the database containing the PR to be edited; if no database is specified, the database named ‘default’ is assumed. This option overrides the database specified in the GNATSDB environment variable.

- **--host host, -H host**
  Specifies the hostname of the gnatsd server to communicate with. This overrides the value in the GNATSDB environment variable.

- **--port port**
  Specifies the port number of the gnatsd server to communicate with. This overrides the value in the GNATSDB environment variable.
--user user, -v user
   Specifies the username to login with when connecting to the gnatsd server. This
   overrides the value in the GNATSDB environment variable.

--passwd passwd, -w passwd
   Specifies the password to login with when connecting to the gnatsd server. This
   overrides the value in the GNATSDB environment variable.

   _edit-pr_ calls the editor specified in your environment variable EDITOR on a temporary
   copy of that PR. (If you don’t have the variable EDITOR defined in your environment, the
   default editor vi is used.)

   Edit the PR, changing any relevant fields or adding to existing information. When you
   exit the editor, _edit-pr_ prompts you on standard input for a reason if you have changed a
   field that requires specifying a reason for the change.

2.3.2 Following up via direct email
If you have some additional information for a PR and for some reason do not want to (or
cannot) edit the PR directly, you may append the information to the Audit-Trail field by
mailing it to the PR submission address.

   In order for GNATS to be able to recognize the mail as pertaining to an existing PR
   (as opposed to a new PR, see Section 2.2.4 [Submitting via e-mail], page 16), the Subject
   mail header field must contain a reference to the PR. GNATS matches the Subject header
   against the regular expression

   \<(PR\s\#)?\-\+[\-[:alnum:]+.]+/\[0-9]+

   to determine whether such a reference is present. Any text may precede or follow the
   reference in the Subject header. If more than one reference is present, the first is used and
   the rest ignored.

   A PR reference matching the regular expression above has two parts. The second is
   the PR number (one or more digits). The first is either the capital letters ‘PR’ optionally
   followed by a separator character (blank, tab, hash mark or forward slash) or the category
   name followed by a forward slash. Following are some examples which match the regular
   expression:

   PR 123 PR4567 PR#890 gnats/4711

   The PR number and the category (if present) are checked for existence, and if the
   outcome is positive, the mail is appended to the Audit-Trail field of the PR. Note that the
   PR need not belong to the category because PRs may move between categories.

   Outgoing emails sent by GNATS itself may be configured to have a Subject header field
   that refers to the PR in question:

       Subject: Re: PR category/gnats-id: original message subject

   This makes it extremely easy to follow up on a PR by replying to such an email, see
   Section 4.3 [The dbconfig file], page 46 and the sample, default dbconfig file installed by
   mkdb.
2.4 Querying the database

Obtain information from the database by using the program query-pr. query-pr uses search parameters you provide to find matching Problem Reports in the database. You can invoke query-pr from the shell or from within Emacs. query-pr uses the same arguments whether it is invoked from the shell or from Emacs.

PRs may be selected via the use of the --expr option, directly by number, or by the use of the (now deprecated) field-specific query operators.

By default, query options are connected with a logical AND. For example,

```
query-pr --category=foo --responsible=bar
```

only prints PRs which have a Category field of ‘foo’ and a Responsible field of ‘bar’.

The --or option may be used to connect query options with a logical OR. For example,

```
query-pr --category=baz --or --responsible=blee
```

prints PRs which have either a Category field of ‘baz’ or a Responsible field of ‘blee’.

It should be emphasized, however, that the use of these field-specific options is strongly discouraged, since they exist only for compatibility with older versions of GNATS and are likely to be deleted in the next release. The expressions specified by the --expr option are much more flexible (see below).

2.4.1 Invoking query-pr

From the shell, simply type query-pr, followed by any search parameters you wish to exercise. From Emacs, type M-x query-pr. query-pr prompts you for search parameters in the minibuffer.

query-pr can also be accessed by electronic mail, if your version of GNATS is configured for this. To use this feature, simply send mail to the address ‘query-pr@your-site’ with command line arguments or options in the Subject line of the mail header. GNATS replies to your mail with the results of your query. The default settings for the query-pr mail server are

```
--restricted --state="open|analyzed|feedback|suspended"
```

To override the --state parameter, specify --state=state in the Subject line of the mail header. You can not query on confidential Problem Reports by mail.

The usage for query-pr is:

```
query-pr [--debug | -D] [--help | -h] [--version | -V]
[--output file | -o file] [--list-databases]
[--list-fields] [--list-input-fields]
[--responsible-address name] [--field-type field]
[--field-description field]
[--field-flags field]
[--adm-field field] [--adm-subfield subfield]
[--adm-key key]
[--valid-values field]
[--format format | -f format]
[--full | -F] [--summary | -q]
[--database database | -d database] [--and | -&]
[--or | -|] [--expr expr] [PR Number]
```

Non-network-mode options:

```
[--print-sh-vars] [--print-directory-for-database]
```

Network-mode-only options:
Chapter 2: The GNATS User Tools

[--host host | -H host] [--port port]
[--user user | -v user] [--passwd passwd | -w passwd]
[--print-server-addr]

Deprecated Options:
[--list-categories | -j] [--list-states | -T]
[--list-responsible | -k] [--list-submitters | -l]
[--category category | -c category]
[--synopsis synopsis | -y synopsis]
[--confidential confidential | -C confidential]
[--multitext multitext | -m multitext]
[--originator originator | -O originator]
[--release release | -A release]
[--class class | -L class] [--cases cases | -E cases]
[--quarter quarter | -Q quarter]
[--keywords keywords | -K keywords]
[--priority priority | -p priority]
[--restricted responsible | -R]
[--severity severity | -e severity]
[--skip-closed | -x] [--sql | -i] [--sql2 | -I]
[--state state | -s state]
[--submitter submitter | -S submitter]
[--text text | -t text]
[--required-before date | -u date]
[--required-after date | -U date]
[--arrived-before date | -b date]
[--arrived-after date | -a date]
[--modified-before date | -B date]
[--modified-after date | -M date]
[--closed-before date | -z date]
[--closed-after date | -Z date]

The options have the following meaning:

--help, -h
   Prints a help message.

--version, -V
   Displays the program version to stdout.

--output file, -o file
   The results of the query will be placed in this file.

--database database, -d database
   Specifies the database to be used for the query. If no database is specified,
   the database named default is assumed. (This option overrides the database
   specified in the GNATSDB environment variable; see Section 2.1 [Environment],
   page 13 for more information.)

--list-categories, -j
   Lists the available PR categories for the selected database.

--list-states, -T
   Lists the valid PR states for PRs in this database.

--list-responsible, -k
   Lists the users that appear in the database’s responsible list.
--list-submitters, -l
Lists the valid submitters for this database.

The previous -list-* options are deprecated and may be removed in future releases of GNATS; their functionality can be replaced with

```
query-pr --valid-values field
```

where `field` is one of Category, Class, Responsible,Submitter-Id, or State.

**--list-databases**
Lists the known databases.

**--list-fields**
Lists the entire set of field names for PRs in the selected database.

**--list-input-fields**
Lists the fields that should be provided when creating a new PR for the currently-specified database. The fields are listed in an order that would make sense when used in a template or form.

**--field-type field**
Returns the data type contained in PR field `field`. The current set of data types includes `'text'`, `'multitext'`, `'enum'`, `'multienum'`, `'enumerated-in-file'`, `multi-enumerated-in-file'`, `'date'` and `'integer'`.

**--field-description field**
Returns a human-readable description of the intended purpose of `field`.

**--field-flags field**
Returns the flags set for the field in the `dbconfig` file associated with the database, such as `textsearch` and `readonly`. See Section 4.3.2 [Individual field configuration], page 48.

**--adm-field field**
Used together with the `--adm-key` option, this returns a record from the administrative file (if any) associated with the field. For more material on administrative files, see [Enumerated field administrative files], page 51.

**--adm-subfield subfield**
Used together with the `--adm-field` and `--adm-key` options, this returns the contents of a particular subfield from the record specified by `--adm-field` and `--adm-key`. Subfields are treated in [Enumerated field administrative files], page 51.

**--adm-key key**
Used together with `--adm-field` to select a record from the administrative file associated with the field specified by `--adm-field`. See [Enumerated field administrative files], page 51.

**--valid-values field**
For fields of type `enum`, a list of valid values (one per line) is returned. Otherwise, a regular expression is returned that describes the legal values in `field`. 
**--responsible-address** name

The mail address of name is returned; name is assumed to be a name either appearing in the database’s responsible list, or is otherwise a user on the system.

**--print-sh-vars**

A set of /bin/sh variables is returned that describe the selected database. They include:

- **GNATSDB**  The name of the currently-selected database.
- **GNATSDB_VALID**  Set to 1 if the selected database is valid.
- **GNATSDBDIR**  The directory where the database contents are stored.
- **DEBUG_MODE**  Set to 1 if debug mode has been enabled for the database.
- **DEFAULTCATEGORY**  The default category for PRs in the database.
- **DEFAULTSTATE**  The default state for PRs in the database.

**--print-server-addr**

Prints the information about a remote server database in the format suitable for the GNATSDB environment variable. This option works only in the network mode.

**--print-directory-for-database**

Returns the directory where the selected database is located.

**--format format, -f format**

Used to specify the format of the output PRs, See Section 2.4.2 [Formatting query-pr output], page 24 for a complete description.

**--full, -F**

When printing PRs, the entire PR is displayed. This is exactly equivalent to

```
query-pr --format full
```

**--summary, -q**

When printing PRs, a summary format is used. This is exactly equivalent to

```
query-pr --format summary
```

**--debug, -D**

Enables debugging output for network queries.

**--host host, -H host**

Specifies the hostname of the gnatsd server to communicate with. This overrides the value in the GNATSDB environment variable.

**--port port**

Specifies the port number of the gnatsd server to communicate with. This overrides the value in the GNATSDB environment variable.
**--user user, -v user**

Specifies the username to login with when connecting to the gnatsd server. This overrides the value in the GNATSDB environment variable.

**--passwd passwd, -w passwd**

Specifies the password to login with when connecting to the gnatsd server. This overrides the value in the GNATSDB environment variable.

**--and, -&, --or, -|**

These options are used when connecting multiple query operators together. They specify whether the previous and subsequent options are to be logically ANDed or logically ORed.

**--expr expr**

Specifies a query expression to use when searching for PRs. See Section 2.4.3 [Query expressions], page 25.

The remaining deprecated options are not described here, since their use is fairly obvious and their functionality is completely replaced by the use of the --expr option.

### 2.4.2 Formatting query-pr output

Printing formats for PRs are in one of three forms:

- **formatname**
  
  This is a named format which is described by the database (specifically, these formats are described in the dbconfig file associated with the database). The default configuration contains five such formats: `standard`, `full`, `summary`, `sql`, and `sql2`.
  
  The first three are the ones most commonly used when performing queries.
  
  standard is the format used by default if no other format is specified.
  
  Use of the latter two are discouraged; they are merely kept for historical purposes. Other named formats may have been added by the database administrator.

- **fieldname**
  
  A single field name may appear here. Only the contents of this field will be displayed.

- **"printf string" fieldname fieldname ...**
  
  This provides a very flexible mechanism for formatting PR output. (The formatting is identical to that provided by the named formats described by the database configuration, See Section 4.3.5 [Named query definitions], page 53.
  
  The printf string can contain the following % sequences:

  %*[positionalspecifiers]s: Prints the field as a string. The positional specifiers are similar to those of printf, as +, - and digit qualifiers can be used to force a particular alignment of the field contents.

  %*[positionalspecifiers]S: Similar to %s, except that the field contents are terminated at the first space character.

  %*[positionalspecifiers]d: Similar to %s, except that the field contents are written as a numeric value. For integer fields, the value is written as a number.
  
  For enumerated fields, the field is converted into a numeric equivalent (i.e. if
the field can have two possible values, the result will be either 1 or 2). For date fields, the value is written as seconds since Jan 1, 1970.

\%F: The field is written as it would appear within a PR, complete with field header.

\%D: For date fields, the date is written in a standard GNATS format.

\%Q: For date fields, the date is written in an arbitrary "SQL" format.

An example formatted query looks as follows (note that the whole format specification should be quoted):

```
query-pr --format '"%s, %s" Synopsis State'
```

### 2.4.3 Query expressions

Query expressions are used to select specific PRs based on their field contents. The general form is

```
fieldname | "value" operator fieldname | "value" [booleanop ...]
```

*value* is a literal string or regular expression; it must be surrounded by double quotes, otherwise it is interpreted as a *fieldname*.

*fieldname* is the name of a field in the PR.

*operator* is one of:

- \(=\) The value of the left-hand side of the expression must exactly match the regular expression on the right-hand side of the expression. See Appendix D [Querying using regular expressions], page 97.

- \(\sim\) Some portion of the left-hand side of the expression must match the regular expression on the right-hand side.

- \(==\) The value of the left-hand side must be equal to the value on the right-hand side.

  The equality of two values depends on what type of data is stored in the field(s) being queried. For example, when querying a field containing integer values, literal strings are interpreted as integers. The query expression

  ```
  Number == "0123"
  ```

  is identical to

  ```
  Number == "123"
  ```

  as the leading zero is ignored. If the values were treated as strings instead of integers, then the two comparisons would return different results.

- \(!=\) The not-equal operator. Produces the opposite result of the \(==\) operator.

- \(<,>\) The left-hand side must have a value less than or greater than the right-hand side. Comparisons are done depending on the type of data being queried; in particular, integer fields and dates use a numeric comparison, and enumerated fields are ordered depending on the numeric equivalent of their enumerated values.

  *booleanop* is either ‘|’ (logical or), or ‘&’ (logical and). The query expression
Category="baz" | Responsible="blee"

selects all PRs with a Category field of ‘baz’ or a Responsible field of ‘blee’.

The not operator ‘!’ may be used to negate a test:

! Category="foo"

searches for PRs where the category is not equal to the regular expression foo.

Parentheses may be used to force a particular interpretation of the expression:

!(Category="foo" & Submitter-Id="blaz")

skips PRs where the Category field is equal to ‘foo’ and the Submitter-Id field is equal to ‘blaz’. Parentheses may be nested to any arbitrary depth.

Fieldnames can be specified in several ways. The simplest and most obvious is just a name:

Category="foo"

which checks the value of the category field for the value foo.

A fieldname qualifier may be prepended to the name of the field; a colon is used to separate the qualifier from the name. To refer directly to a builtin field name:

builtin:Number="123"

In this case, ‘Number’ is interpreted as the builtin name of the field to check. (This is useful if the fields have been renamed. For further discussion of builtin field names, see Section 4.3 [The dbconfig file], page 46.

To scan all fields of a particular type, the fieldtype qualifier may be used:

fieldtype:Text="bar"

This searches all text fields for the regular expression ‘bar’.

Note that it is not required that the right-hand side of the expression be a literal string. To query all PRs where the PR has been modified since it was closed, the expression

Last-Modified != Closed-Date

will work; for each PR, it compares the value of its Last-Modified field against its Closed-Date field, and returns those PRs where the values differ. However, this query will also return all PRs with empty Last-Modified or Closed-Date fields. To further narrow the search:

Last-Modified != Closed-Date & Last-Modified != "" & Closed-Date != ""

In general, comparing fields of two different types (an integer field against a date field, for example) will probably not do what you want.

Also, a field specifier may be followed by the name of a subfield in braces:

State[type] != "closed"

or even

builtin:State[type] != "closed"

Subfields are further discussed in Section 4.3 [The dbconfig file], page 46.
2.4.4 Example queries

The following simple query:

```bash
query-pr --expr 'Category~"rats" & State~"analyzed"
& Responsible~"fred"'
```

yields all PRs in the database which contain the field values:

- Category: rats and
- Responsible: fred and
- State: analyzed

The following query:

```bash
query-pr --expr 'State~"open|analyzed"'
```

yields all PRs in the database whose State values match either ‘open’ or ‘analyzed’ (see Appendix D [Querying using regular expressions], page 97. This search is useful as a daily report that lists all Problem Reports which require attention.

The following query:

```bash
query-pr --expr 'fieldtype:Text="The quick.*brown fox"'
```

yields all PRs whose Text fields contain the text ‘The quick’ followed by ‘brown fox’ within the same field. See Appendix D [Querying using regular expressions], page 97, which also contains further useful examples of query expressions.

2.5 The Emacs interface to GNATS

Emacs interface to GNATS provides basic access to GNATS databases, i.e. sending, editing, and querying Problem Reports. It also defines a simple major mode for editing dbconfig files.

This section provides an overview of using GNATS with Emacs. It does not describe the use of Emacs itself, for detailed instructions on using Emacs, see GNU Emacs. For installation instructions of the GNATS Emacs mode, see Section 3.2 [Installing utils], page 34.

Please note the Emacs interface was completely rewritten between GNATS 3 and GNATS 4. It now uses gnatsd, Appendix B [gnatsd], page 79, exclusively for its operations and uses modern Emacs features like faces. Its features are not complete though, you can send your suggestions and patches to the appropriate GNATS mailing list, Appendix F [Support], page 103.

2.5.1 Viewing Problem Reports

To view a particular Problem Report, use the command `M-x view-pr`. It asks for a Problem Report number and displays that Problem Report.

The displayed buffer is put in the view mode, Section “Misc File Ops” in GNU Emacs. If you decide to edit the displayed Problem Report, use the command `e (gnats-view-edit-pr).

`gnats-view-mode-hook`

Hook run when `gnats-view-mode` is entered.
2.5.2 Querying Problem Reports

Querying the database is performed by the \texttt{M-x query-pr} command. The command prompts for the query expression, Section 2.4.3 [Query expressions], page 25, and displays a buffer with the list of the matching Problem Reports.

The list of the Problem Reports is displayed in the `summary` query format, Section 2.4.2 [Formatting query-pr output], page 24. Currently, the display format cannot be changed and it must output each Problem Report's number in the first column.

The Problem Report list buffer is put in the view mode, Section “Misc File Ops” in \textit{GNU Emacs}. You can use most of the standard view mode commands in it. Additionally, the following special commands are available:

\begin{verbatim}
  v RET mouse-2  View the current Problem Report (gnats-query-view-pr), Section 2.5.1 [Emacs viewing], page 27.
  e  Edit the current Problem Report (gnats-query-edit-pr), Section 2.5.4 [Emacs editing], page 29.
  g  Update the Problem Report list (gnats-query-reread). The last performed query is executed again and the buffer is filled with the new results.
  G  Perform new query (query-pr).
  s  Send new Problem Report (send-pr), Section 2.5.3 [Emacs submitting], page 28.
  D  Change the current database (gnats-change-database), Section 2.5.6 [Emacs and databases], page 30.
  q  Bury buffer, the buffer is put at the end of the list of all buffers. This is useful for quick escape of the buffer, without killing it.
\end{verbatim}

If the value of the variable \texttt{gnats-query-reverse-listing} is non-\texttt{nil}, the listing appears in the reversed order, i.e. with the Problem Reports of the highest number first, in the buffer.

Similarly to other GNATS Emacs modes, there is a hook available for the Problem Report list.

\texttt{gnats-query-mode-hook}
  Hook run when \texttt{gnats-query-mode} is entered.

2.5.3 Submitting new Problem Reports

You can submit new Problem Reports with the command \texttt{M-x send-pr}. The command puts you to the problem editing buffer, Section 2.5.4 [Emacs editing], page 29. The buffer is prefilled with the initial report fields and their default values, if defined. You can use the usual Problem Report editing commands, Section 2.5.4 [Emacs editing], page 29. When you have filled in all the fields, you can send the Problem Report by pressing \texttt{C-c C-c}.

If you run \texttt{M-x send-pr} with a prefix argument, it runs the \texttt{gnats-change-database} command before putting you to the editing buffer, Section 2.5.6 [Emacs and databases], page 30.

You can set the following variables to get some fields pre-filled:
gnats-default-organization

Default value of the ‘Organization’ field used in new Problem Reports.

gnats-default-submitter

Default value of the ‘Submitter-Id’ field used in new Problem Reports.

2.5.4 Editing Problem Reports

To edit a particular Problem Report, use the command $M-x$ edit-pr. It asks for a Problem Report number and puts the given Problem Report in the editing buffer. See Section 2.5.5 [Emacs editing buffer], page 29, for information how to edit the Problem Report in the buffer and how to submit your changes.

Note you can also start editing of a selected Problem Report directly from within the viewing buffer, Section 2.5.1 [Emacs viewing], page 27, or the query result buffer, Section 2.5.2 [Emacs querying], page 28.

2.5.5 The Problem Report editing buffer

When you invoke a Problem Report editing command, the Problem Report is put into a special editing buffer. The Problem Report is formatted similarly to the query-pr -F output, Section 2.4.2 [Formatting query-pr output], page 24. Field identifiers are formatted as

>Field:

with the text of the field following the identifier on the same line for single-line fields or starting on the next line for multi-line fields.

The Problem Report editing mode tries to prevent you from violating the Problem Report format and the constraints put on the possible field values. Generally, you can use usual editing commands, some of them have a slightly modified behavior though. (If you encounter a very strange behavior somewhere, please report it as a bug, Appendix F [Support], page 103.)

You can move between fields easily by pressing the $TAB$ (gnats-next-field) or $M-TAB$ (gnats-previous-field) keys.

The field tags are read-only and you cannot edit them nor delete them. If you want to “remove” a field, just make its value empty.

Editing a field value depends on the type of the edited field, Section 4.3.3 [Field datatypes], page 49. For text fields, you can edit the value directly, assuming you preserve the rule about single-line and multi-line values mentioned above.

For enumerated fields, you cannot edit the value directly. You can choose it from the list of the allowed values, either from the menu popped up by pressing the middle mouse button or from within minibuffer by pressing any key on the field’s value. If the pressed key matches any of the allowed field values, that value is put as the default value after the minibuffer prompt. You can also cycle through the allowed field values directly in the editing buffer using the $SPACE$ key. Enumerated field values are marked by a special face to not confuse you; you must have enabled font lock mode to benefit from this feature, Section “Font Lock” in GNU Emacs.

Some field values can be read-only, you cannot edit them at all.
Once you have edited the Problem Report as needed, you can send it to the server with the C-c C-c command (gnats-apply-or-submit). Successful submission is reported by a message and the buffer modification flag in mode line is cleared. Then you can either kill the buffer or continue with further modifications.

\texttt{gnats-edit-mode-hook}

Hook run when \texttt{gnats-edit-mode} is entered.

### 2.5.6 Changing the database

By default, the Emacs interface connects to the default database, Section 4.2 [databases file], page 45. If you want to connect to another database, use the command \texttt{M-x gnats-change-database}. It will ask you for the database name to use, server and port it can be accessed on, and your login name.

If you want to use the \texttt{gnatsd} command, Appendix B [gnatsd], page 79, directly, without connecting to a remote server or the localhost connection port, provide your local file system full path to \texttt{gnatsd} as the server name. Port number does not matter in this case.

If the database requires a password to allow you the access to it, you are prompted for the password the first time you connect to the database. If you provide an invalid password, you cannot connect to the database anymore and you have to run the \texttt{M-x gnats-change-database} command again.

### 2.5.7 dbconfig mode

The Emacs interface defines a simple major mode \texttt{gnats-dbconfig-mode} for editing \texttt{dbconfig} files, Section 4.3 [dbconfig file], page 46. It defines basic mode attributes like character syntax and font lock keywords, it does not define any special commands now.

\texttt{gnats-dbconfig-mode-hook}

Hook run when \texttt{gnats-dbconfig-mode} is entered.

### 2.5.8 Other commands

\texttt{M-x unlock-pr}

Ask for a Problem Report number and unlock that Problem Report. This function is useful if connection to a GNATS server was interrupted during an editing operation and further modifications of the Problem Report are blocked by a stealth lock.

\texttt{M-x unlock-database}

Unlock the whole GNATS database. This function is useful in situations similar to when \texttt{unlock-pr} is used.

\texttt{M-x gnats-show-connection}

Show the connection buffer associated with the current buffer. You can view the Emacs communication with GNATSD in it. This is useful when something strange happens during the communication with the server, e.g. when sending a Problem Report with C-c C-c from a Problem Report editing buffer.

### 2.5.9 Customization

All the user variables can be customized in the customization group \texttt{gnats}, Section “Easy customization” in \texttt{GNU Emacs}.
3 Installing GNATS

See also Appendix A [Where the tools and utilities reside], page 73.

There are several steps you need to follow to fully configure and install GNATS on your system. You need `root` access in order to create a new account for `gnats` and to install the GNATS utilities. You may need `root` access on some systems in order to set up mail aliases and to allow this new account access to `cron` and `at`.

If you are updating an older version of GNATS rather than installing from scratch, see Section 3.8 [Upgrading from older versions], page 39.

GNATS installation relies on two other freely available software packages, which should be installed before you go on to configure and build GNATS. These are GNU `make` and Texinfo (version 4.2 or higher). Both are available from the GNU FTP site at `ftp://ftp.gnu.org`.

To build and install GNATS, you must:

- Run `configure`, with correct options if the defaults are unsuitable for your site. See Section 3.1 [Configuring and compiling the software], page 31. Default installation locations are in Appendix A [Where GNATS lives], page 73.
- Compile the GNATS programs on your system. See Section 3.1 [Configuring and compiling the software], page 31.
- Create an initial database by using the `mkdb` command. See Section 3.3 [Setting up the default database], page 34.
- Set up periodic jobs, using `cron`, to handle Problem Reports arriving by mail. See Section 3.4 [Setting up periodic jobs], page 35.
- Set up mail aliases for GNATS. See Section 3.5 [Setting up mail aliases], page 36.
- Install the GNATS tools and utilities locally, and install the user tools (`query-pr`, `edit-pr`, `send-pr`) on every machine in your local network. See Section 3.7 [Installing the user tools], page 37.
- Install the GNATS daemon `gnatsd`. See Section 3.6 [Installing the daemon], page 36.
- If there are people outside your organization who will be submitting PRs or who are supposed to be able to query and/or edit PRs, you may need to instruct them to obtain and build the GNATS tools `query-pr`, `edit-pr` and `send-pr` for their systems. However, for many sites, setting up a remote access interface to GNATS, such as Gnatsweb is a better solution since this requires no configuration on the remote side.

3.1 Configuring and compiling the software

1. First, unpack your distribution. We provide source code in a `tar` file which was compressed using `gzip`. The code can be extracted into a directory `unpackdir` using

   ```
   cd unpackdir
   gunzip gnats-4.2.0.tar.gz
   tar xvf gnats-4.2.0.tar
   ```

   The sources reside in a directory called `gnats-4.2.0` when unpacked. We call this the `top level` of the source directory, or `sdir`. The sources for the GNATS tools are in the subdirectory `gnats-4.2.0/gnats/*`. Lists of files included in the distribution are in each directory in the file `MANIFEST`. 

2. As of GNATS version 4, having Emacs installed on the GNATS server is no longer a requirement. If you do not have Emacs installed, disregard this step altogether.

You may wish to alter the installation directory for the Emacs lisp files. If your Emacs lisp library is not in `prefix/share/emacs/site-lisp`, edit the file `srcdir/gnats/Makefile.in`. Change the variable `lispdir` from `prefix/emacs/site-lisp` to the directory containing your Emacs lisp library. For information on `prefix`, see Section A.1 [prefix], page 73.

3. Create an account for the gnats user. You can actually name this user whatever you want to, as long as it is a valid username on your system, but we strongly recommend that you call the user gnats. If you do decide to give it some other name, remember to use the option `--enable-gnats-user` when running configure below. Below, we will anyway refer to this user by the name gnats.

This user must have an entry in the file `/etc/passwd`. As for ordinary users, create a standard home directory for the gnats user. The default PATH for this user should contain `exec-prefix/bin` and `exec-prefix/libexec/gnats`. The exec-prefix value is configurable with the `--exec-prefix` configure option described below, but for standard installations, these two directories correspond to `/usr/local/bin` and `/usr/local/libexec/gnats`.

4. Run configure. You can nearly always run configure with the simple command

   ```bash
   ./configure
   ```

and the “Right Thing” happens:

- GNATS is configured in the same directory you unpacked it in;
- when built, GNATS runs on the machine you’re building it on;
- when installed, files are installed under `/usr/local` (see Appendix A [Where GNATS lives], page 73);
- all GNATS utilities operate on the default database, assumed to be named `default` and to be located in `/usr/local/com/default`, unless you invoke the utilities with `-d databasename` or `--directory=databasename`, or set the `GNATSDB` environment variable to point to some other database.

The most common options to configure are listed below:

```bash
configure [ --prefix=prefix ]
    [ --exec-prefix=exec-prefix ]
    [ --enable-gnats-service=service-name ]
    [ --enable-gnats-user=username ]
    [ --enable-gnatsd-user-access-file=path ]
    [ --enable-gnatsd-host-access-file=path ]
    [ --enable-gnatsd-dblist-file=path ]
    [ --enable-gnatsd-default-db=path ]
    [ --with-kerberos ] [ --with-krb4 ]
    [ --verbose ]
```

`--prefix=prefix`

All host-independent programs and files are to be installed under `prefix`. (Host-dependent programs and files are also installed in `prefix` by default.) The default for `prefix` is `/usr/local`. See Appendix A [Where GNATS lives], page 73.
--exec-prefix=exec-prefix
   All host-dependent programs and files are to be installed under exec-prefix.
The default for exec-prefix is prefix. See Appendix A [Where GNATS lives],
   page 73.

--enable-gnats-service=service-name
   Set service-name to be the GNATS network service. Default name is support.

--enable-gnats-user=username
   Set username to be the user name for GNATS. Default username is gnats.

--enable-gnatsd-user-access-file=path
   Set global (across all databases) gnatsd user access file to path. Default is
       /usr/local/etc/gnats/gnatsd.user_access. Per-database user access permissions are set
       in a gnatsd.user_access file in the gnats-adm subdirectory of each database.

--enable-gnatsd-host-access-file=path
   Set global (across all databases) gnatsd host access file to path. Default is
       /usr/local/etc/gnats/gnatsd_host.access. There is currently no way to specify host access permissions on a per-database basis.

--enable-gnats-dblist-file=path
   Specify the file containing the list of databases.
       Default is prefix/etc/gnats/databases.

--enable-gnats-default-db=path
   Specify the default database to use when GNATS tools are invoked without the -d or --databasename option, and when the GNATSDB environment variable hasn’t been set. Default is /prefix/com/gnatsdb.

--with-kerberos
   Include code for Kerberos authentication.

--with-krb4
   Support Kerberos 4.

--verbose
   Give verbose output while configure runs.

configure supports several more options which allow you to specify in great detail
where files are installed. For a complete list of options, run ./configure --help in the source directory.

You can build GNATS in a different directory (objdir) from the source code by calling
the configure program from the new directory, as in

    mkdir objdir
    cd objdir
    srcdir/configure ...

By default, make compiles the programs in the same directory as the sources (srcdir).

5. Make sure you have GNU make, then run
3.2 Installing the utilities

The following steps are necessary for a complete installation. You may need root access for these.

1. Install the utilities by invoking

   make install install-info

   These targets indicate:

   install  Installs all programs into their configured locations (see Appendix A [Where GNATS lives], page 73).

   install-info  Installs ‘info’ files into their configured locations (see Appendix A [Where GNATS lives], page 73).

   After you have installed GNATS, you can remove the object files with

   make clean

2. If you do not have Emacs installed on your GNATS server, this step should be skipped.

   Place the following lines in the file default.el in your Emacs lisp library, or instruct your local responsible parties to place the lines in their .emacs:

   (autoload 'send-pr "gnats" "Command to create and send a problem report." t)
   (autoload 'edit-pr "gnats" "Command to edit a problem report." t)
   (autoload 'view-pr "gnats" "Command to view a problem report." t)
   (autoload 'query-pr "gnats" "Command to query information about problem reports." t)
   (autoload 'unlock-pr "gnats" "Unlock a problem report." t)
   (autoload 'gnats-dbconfig-mode "gnats" "Major mode for editing the ‘dbconfig’ GNATS configuration file." t)
   (add-to-list 'auto-mode-alist '("\<dbconfig\>" . gnats-dbconfig-mode))

3. If you want people who are logged into the GNATS server itself to be able to use the send-pr tool to submit problem reports, you need to create a configuration file for send-pr on the server. See Section 4.5 [The send-pr.conf configuration file], page 62.

3.3 Installing the default database

For the following steps, log in as the user gnats.

We are now going to initialize the default GNATS database. Run the following command:

   mkdb default

make all info

from the directory where configure created a Makefile (this is objdir if you used it, otherwise srcdir.) These targets indicate:

all  Compile all programs
info  Create ‘info’ files using makeinfo.
This creates a database named `default`, with all its data stored below the
directory `prefix/com/gnatsdb`, in a default installation this corresponds to
`/usr/local/com/gnatsdb`. If you specified the `--enable-gnats-default-db` option
when running configure, the default database will be created under the directory you
specified instead. `mkdb` creates the database directory itself, together with three different
subdirectories:

- A directory for the mandatory GNATS category `pending`.
- A `gnats-queue` directory for queueing new messages to GNATS before they are processed
  by `file-pr`.
- The administrative directory `gnats-adm`. This directory is populated with default
  configuration files from the `prefix/etc/gnats/defaults` directory.

The next configuration step is to edit the default files copied to the database’s `gnats-adm` directory by `mkdb`.

The default `dbconfig` file installed by `mkdb` provides a good basis for many GNATS
databases. The default file causes similar behaviour to the 3.x versions of GNATS. However,
even if this might be precisely what you want, you should still go through the file and check
that the default settings suit your needs. See Section 4.3 [The `dbconfig` file], page 46.

Then edit the files `categories`, `responsible`, and `submitters` in the `gnats-adm` directory (see Section 4.4 [Other database-specific config files], page 58) to reflect your local
needs. For special configurations, you may also have to edit the `states` and `classes` files.

If you used the `--enable-gnats-default-db` option in the pre-build configure
to change the location of the default database, you need to edit the `databases`
config file, see Section 4.2 [The `databases` file], page 45. This file is by default
located in the `prefix/etc/gnats` directory, but may have been changed by the option
`--enable-gnats-dblist-file` option during configure.

### 3.4 Setting up periodic jobs

Allow the new user `gnats` access to `cron` and `at`. To do this, add the name `gnats` to the files `cron.allow` and `at.allow`, which normally reside in the directory `/var/spool/cron.
If these files do not exist, make sure `gnats` does not appear in either of the files `cron.deny`
and `at.deny` (in the same directory). If you changed the name of the GNATS user during
configure, remember to substitute as appropriate in the previous steps.

Create a `crontab` entry that periodically runs the program `queue-pr` with the `--run`
option (see Section 4.8.1 [queue-pr], page 66). For example, to run `queue-pr --run`
every ten minutes, create a file called `.mycron` in the home directory of the user `gnats`
which contains the line:

```
0,10,20,30,40,50 * * * * exec-prefix/libexec/gnats/queue-pr --run
```

(Specify the full path name for `queue-pr`.) Then run

```
crontab .mycron
```

See the `man` pages for `cron` and `crontab` for details on using `cron`.

---

1 Upgraders from older versions of GNATS should note that category directories are now created “on-the-fly”
as needed by default.
3.5 Setting up mail aliases

The following mail aliases must be added on the machine where the GNATS server is installed. The instructions below are for Sendmail or Sendmail-like mail systems. If these instructions don’t fit your system, particularly if you do not have an aliases file, ask your mail administrator for advice.

The following aliases should be placed in the file /etc/aliases. You may need root access to add these aliases:

- Create an alias for the GNATS administrator. This address should point to the address of the person in charge of administrating GNATS:

  gnats-admin: address

- Create an alias to redirect incoming Problem Reports. This alias should redirect incoming mail via a pipe to the program 'queue-pr -q'. For example, if Problem Reports coming to your site are to arrive at the address ‘bugs@your.company.com’, create an alias to the effect of:

  bugs: "| exec-prefix/libexec/gnats/queue-pr -q"

This places incoming Problem Reports in the gnats-queue directory of your database. Remember to fill in the full path of the queue-pr command as appropriate for your installation.

- You may also wish to forward a copy of each incoming Problem Report to a log. This can be accomplished with something like:

  bug-q: "| exec-prefix/libexec/gnats/queue-pr -q"
  bug-log: /some/path/bugs.log
  bugs: bug-q, bug-log

This configuration archives incoming Problem Reports in the file bug.log, and also feeds them to the program queue-pr. (Remember, bug.log needs to be world-writable, and should be pruned regularly; see Chapter 4 [GNATS Administration], page 43.) In order for the log file to protect fully against data loss in case a disk runs full, try to place it on a different disk volume than the GNATS database.

- If you want your users to be able to query the database by mail, use the following alias:

  query-pr: "| exec-prefix/libexec/gnats/mail-query"

The mail-query program uses ‘--restricted’ to search on the database, and by default only searches for PRs that aren’t closed (see Section 2.4 [Querying the database], page 20).

3.6 Installing the daemon

By default, the daemon and clients are set to use port 1529. Add the line

```bash
support 1529/tcp # GNATS
```

to your /etc/services file. If you want a different service name, configure GNATS with

```
--enable-gnats-service=servicename
```

In your inetd.conf file, add the line

```
support stream tcp nowait gnats /usr/local/libexec/gnats/gnatsd gnatsd
```

adjusting the path accordingly if you used configure options to make changes to the defaults. To make inetd start spawning the GNATS daemon when connected on that port, send it a hangup signal (HUP).
Some operating systems have replaced inetd with the more modern xinetd. Instead of editing inetd.conf, you should create the file /etc/xinetd.d/support, containing something like the following:

```
service support {
    disable = no
    socket_type = stream
    protocol = tcp
    wait = no
    user = gnats
    server = /usr/local/libexec/gnats/gnatsd
}
```

If you specified a different service name when running configure, you need to give the file the same name as the service name, and you need to adjust the service line above. If the --prefix or --exec-prefix options were passed to configure, adjust the server line above, and if you used the --enable-gnats-user option, adjust the user line.

Then restart xinetd to make the new configuration current.

If you use an Internet superserver different from inetd or xinetd, please refer to its documentation for information how to configure it.

At this point, you will probably want to set the access permissions of the different hosts that are going to be accessing your databases. The access permissions can currently only be set on a global scale (that is, across all the databases on a GNATS server). The location and name of the global host access configuration file can be set during the pre-build configure as shown above, but by default the file is /usr/local/etc/gnats/gnatsd_host.access. It lists the hosts allowed to access your server, and what their default access levels are. Each line in the file denotes one server, or one part of a network domain. There are three fields on each line, but only two are currently used. To grant all hosts from the domain site.com edit access, use this line:

```
site.com:edit
```

If you run a GNATS web interface or similar tool on the same machine as the server is running on, you probably want to grant localhost edit access:

```
localhost:edit
```

If you are using Kerberos, the gnatsd_host.access file shows the sites that don’t require Kerberos authentication.

The third field might in the future be used for things like controlling what categories, submitter-id’s PRs, etc., can be accessed from that site. Access attempts that are denied are logged to the syslog messages file (/var/adm/messages on many systems).

### 3.7 Installing the user tools

When you install the GNATS utilities, the user tools send-pr, query-pr and edit-pr are installed on the server machine. If your machine is part of a network, however, you may wish to install the user tools on each machine in the network so that responsible parties on those machines can submit new Problem Reports, query the database, and edit existing PRs. In the following discussion, machines with the GNATS user tools installed are referred to as client machines. In general, there are three distinct types of client that a GNATS server may have to cater for:
Machines that are on the same local network as the GNATS server, i.e. machines that are under the same administrative control.

- Machines on the general, open Internet.
- Machines behind firewalls etc. which deny direct access over the network to the GNATS server.

Each type of client requires a different approach when it comes to providing access.

### 3.7.1 User tools on a local network

If all the machines involved reside on the same local network as the GNATS server, you can simply share out the directories on the server that contain the user tools, by default `/usr/local/bin` and the directory which contains the `send-pr.conf` configuration file (see Section 4.5 [The send-pr.conf configuration file], page 62), by default `/usr/local/etc/gnats`. If you have a heterogeneous environment, i.e. hosts running different operating systems, you need to create several shared GNATS installations, one for each platform. The `send-pr.conf` file is platform-independent, though.

In order to submit a new PR, `send-pr` would then be invoked as follows on the client machines:

```bash
send-pr -d hostname:port:database:username:password
```

Or by first setting the environment variable `GNATSDB` as follows (the exact syntax will vary depending on what shell you use):

```bash
export GNATSDB=hostname:port:database:username:password
```

Then, `send-pr` can simply be invoked without any options.

The other tools, `query-pr` and `edit-pr`, work in similar ways, honoring the `-d` option as well as the `GNATSDB` environment variable. See Chapter 2 [GNATS user tools], page 13.

### 3.7.2 User tools for remote users

When client machines reside on the general Internet, both security and practical considerations may make it impossible to provide a shared installation of the GNATS tools. In this case, you may choose to only provide access through a web interface such as Gnatsweb.

For clients that need the GNATS tools, the following needs to be carried out on the remote machines:

1. Configure and build GNATS on the client machine
2. Configure `send-pr` on the client machine

You should unpack the distribution and run `configure` on the client machine in the same way as described in Section 3.1 [Configuring and compiling the software], page 31. Note, however, that you do not need to create a `gnats` user on the client and you should not use the `make all info` command to build. Instead, issue the following commands from the top level directory of the source distribution:

```bash
cd gnats
make install-tools
cd ../send-pr
make all install
```

This builds and installs the `send-pr`, `query-pr` and `edit-pr` tools on the client machine. You should now configure `send-pr` by editing the `send-pr.conf` file (see Section 4.5 [The send-pr.conf configuration file], page 62.)
Users on the client machine can now either use the send-pr syntax or the GNATSDB environment variable described in the previous section.

For sites that need to submit Problem Reports by having send-pr send e-mail instead of speaking directly over the network to the GNATS server, you need to create a problem report template on the GNATS server and have that template copied to a suitable location on the client machine (any filename and any location will do, as long as send-pr on the client machine can read the file). On the GNATS server, use the command

```
send-pr -p > filename
```

The file `filename` now contains a PR template for your database. Copy this file to the client. Then edit the `send-pr.conf` file that you created on the client, set the TEMPLATE variable to point to the template file (see Section 4.5 [The send-pr.conf configuration file], page 62) and make sure that the MAILPROG and MAILADDR variables in `send-pr.conf` are correctly set. You should now have a working remote tool installation.

For clients that have no direct network access to your GNATS server, such as those that are located behind strict firewalls, you either need to set up a web interface such as Gntsweb (provided that the firewall lets web traffic through) or use the procedure above which sets up send-pr to submit Problem Reports by e-mail. In order to query PRs, users on the remote machines will then have to use the the e-mail functionality of query-pr (see Section 2.4.1 [Invoking query-pr], page 20. Editing PRs by e-mail is not possible, so clients in this group who need edit access have to get access through a web interface if possible.

Note that when send-pr is set up to work over e-mail, the GNATSDB environment variable and the -d command line option have no effect since send-pr is tied to a specific database by way of the value of MAILADDR in the send-pr.conf file.

### 3.8 Upgrading from older versions

The following procedure covers an upgrade from all GNATS 3 versions newer than 3.108. If your installation is an older 3.10x version, or even the ancient 3.2 version, you need to review the UPGRADING.old file in the GNATS distribution before carrying out the steps detailed here.

#### 3.8.1 Overview

Although almost all of the GNATS internals have been redesigned and rewritten for GNATS 4, little has changed in the format and structure of the database data. The only change that needs to be taken into account when upgrading is the fact that the database index format is binary in a default installation of GNATS 4. Thus, you will need to regenerate your database index by using the gen-index tool. In addition, if your old GNATS installation was so-called “release-based”, you need to make some simple modifications to the database setup file `dbconfig`. See below for details.

Apart from building and installing new binaries, the major changes which impinge on the upgrade procedure are all on the configuration side. The main database configuration file, `dbconfig`, is far more complex and powerful than the old `config` file, and while the installation process creates a sensible set of default values which are similar to GNATS 3.11x’s defaults, you still need to migrate any changes you may have made to your own local configuration.
Another aspect which needs consideration are remote submitter sites. Such sites either
need to be instructed to upgrade their locally installed copies of the GNATS user tools
(send-pr, edit-pr and query-pr), or they should be given access through interfaces such as Gnatsweb.

Since the GNATS network daemon has been completely reworked, with an entirely new
command set, all network-based interfaces, such as Gnatsweb and TkGnats need to be
upgraded to versions that support GNATS 4. The contrib directory of this distribution
contains some third-party interfaces, and the README file contains pointers to where you
can obtain the newest versions of these tools.

This document only deals with upgrading GNATS itself. Third-party tools should have
separate upgrading instructions in their distributions.

3.8.2 Upgrading

1. Before you begin, make a backup of your entire GNATS database directory hierar-
chy, the GNATS executables directory and the GNATS user tools (send-pr, query-pr
etc.) The locations of these may vary, but in a default GNATS 3 installation, the
database(s) reside under /usr/local/share/gnats, the executables are located in
/usr/local/libexec/gnats and the user tools reside in /usr/local/bin.

2. (optional) In order to avoid confusing your users, you may want to remove the old
GNATS 3 executables and tools, especially if you plan to install GNATS 4 in a different
location than version 3.

3. Build and install GNATS 4. See Chapter 3 [Installing GNATS], page 31. It is
recommended that you use the --enable-gnats-default-db option when running
configure, in order to set the default database to be one of your already existing
GNATS 3 databases.

4. Edit the GNATS databases file and add entries for all your old
GNATS 3 databases and tools, especially if you plan to install GNATS 4 in a different
location than version 3.

5. In GNATS 3, the file gnatsd.conf specifies minimum access levels for the different hosts
accessing the GNATS daemon, gnatsd. There is one gnatsd.conf for each database.
In GNATS 4, these files have been replaced by a single file named gnatsd.host_access
which contains settings that apply across all the databases on the server. This file is
located in the same directory as the databases file. You need to combine the host access
settings from all your GNATS 3 databases and add them to the gnatsd.host_access
file. Note that you are no longer able to control host access on a per-database basis.
Optionally, you may delete the old gnatsd.conf files. See Appendix C [Controlling
access to GNATS databases], page 93.

6. Next, you need to migrate the settings in the old config files of your data-
bases to corresponding dbconfig files. The database you specified with the
--enable-gnats-default-db configure option got a default dbconfig installed. This
default file contains field definitions etc. which makes this version of GNATS behave
almost exactly like older versions. Copy this default file to the gnats-adm directories
of any other GNATS databases that you may have on your host before you proceed to
migrate your old configuration settings.
The following is a list of the configuration directives that may be present in a config file and their counterparts (if any) in GNATS 4.

**GNATS_ADDR**
This setting has no counterpart in GNATS 4, since GNATS no longer needs to know its own mail address.

**GNATS_ADMIN**
This setting is now set in the responsible file in the gnats-adm directory of your database(s).

**GNATS_SITE**
GNATS 4 has no concept of a named ‘site’, so this directive is obsolete.

**SUBMITTER**
Obsolete, since it relates to GNATS_SITE.

**DEFAULT_RELEASE**
**DEFAULT_ORGANIZATION**
The GNATS 4 dbconfig file has separate configuration sections for each defined field. Field defaults are set with the default keyword in these sections. See Section 4.3 [The dbconfig file], page 46.

**NOTIFY**
Controlled by the notify-about-expired-prs setting in the dbconfig file.

**ACKNOWLEDGE**
Controlled by the send-submitter-ack setting in the dbconfig file.

**DEFAULT_SUBMITTER**
The default submitter is now always the first entry in the submitters file of your database.

**KEEP_RECEIVED_HEADERS**
Controlled by the keep-all-received-headers setting in the dbconfig file.

**DEBUG_MODE**
Controlled by the debug-mode setting in the dbconfig file.

**BDAY_START**
**BDAY_END**
**BWEKETE Starter**
**BWEKET_Write**
Controlled by the settings business-day-hours and business-week-days in the dbconfig file.

**DEFINE_CATEGORY**
The default category for PRs that arrive without one is now the first category listed in the categories file of your database.

After your are done migrating the settings, you may optionally delete the old config files. Since there are many more configuration settings available in the GNATS 4 dbconfig file, you should take some time to review them all before proceeding. See Section 4.3 [The dbconfig file], page 46.
If your old GNATS installations was release-based, i.e. it included the fields Quarter, Keywords and Date-Required, you need to define those fields in the dbconfig file by following the instructions in [Supporting old GNATS “release-based” fields], page 100.

7. The file gnatd.access has been renamed to gnatd.user_access. Furthermore, GNATS 4 uses a different password format in the gnatd.user_access file than older versions, since it supports crypt() and MD5 passwords (see Appendix C [Controlling access to GNATS databases], page 93). You need to translate your old gnatd.user_access files to the new format by using the gnatspwconv tool which was installed in the EXEC-PREFIX/libexec/gnats directory, typically /usr/local/libexec/gnats. See Section 4.7.6 [Managing user passwords], page 66.

8. The final step involves regenerating the indexes of your databases. For this, log in as the user gnats. Then run the gen-index command for each of your databases. See Section 4.7 [Administrative Utilities], page 64 for details on how to use gen-index.

9. Sit back and enjoy your new GNATS 4 setup...
Chapter 4: GNATS Administration

4 GNATS Administration

In daily usage, GNATS is self-maintaining. However, there are various administrative duties which need to be performed periodically. Also, requirements may change with time, so it may be necessary to make changes to the GNATS configuration at some point:

emptying the pending directory
If a Problem Report arrives with a Category value that is unrecognized by the categories file, or if that field is missing, GNATS places the PR in the pending directory (see Appendix A [Where GNATS lives], page 73). PRs submitted in free-form by email will always be filed in the pending directory. If so configured, GNATS sends a notice to the gnats-admin and to the party responsible for that submitter (as listed in the submitters file) when this occurs. To have these "categoryless" PRs filed correctly, you can then use a GNATS tool such as edit-pr to set the correct category of each PR in the pending directory.

In order to protect yourself from problems caused by full disks, you should arrange to have all mail that is sent to the GNATS database copied to a log file (Section 3.5 [Setting up mail aliases], page 36). Then, should you run out of disk space, and an empty file ends up in the database's pending directory, you need only look in the log file, which should still contain the full message that was submitted.

adding another database
GNATS supports multiple databases. If you find at some point that you need to add another database to your server, the mkdb tool does most of the work for you. See Section 4.7.1 [Adding another database], page 64.

adding new categories
Most installations of GNATS will only require you to add a new line to the categories file. The category directory will then be created automatically as needed. However, if automatic directory creation has been switched off in the dbconfig file (see Section 4.3 [The dbconfig file], page 46), you need to use the mkcat program.

removing categories
To remove a category, you need to make sure the relevant subdirectory is empty (in other words, make sure no PRs exist for the category you wish to remove). You can then remove the category listing from the categories file, and invoke rmcat category...

to remove category (any number of categories may be specified on the command line to rmcat, so long as they abide by the above constraints).

adding and removing maintainers
Edit the responsible file to add a new maintainer or to remove an existing maintainer. See Section 4.4.2 [The responsible file], page 59.

building a new index
If your index becomes corrupted, or if you wish to generate a new one for some reason, use the program gen-index (see Section 4.7.4 [Regenerating the index], page 64).
pruning log files

Log files often grow to unfathomable proportions. As with gardening, it is best to prune these as they grow, lest they take over your disk and leave you with no room to gather more Problem Reports. If you keep log files, be sure to keep an eye on them. (See Section 3.5 [Setting up mail aliases], page 36.)

BACKING UP YOUR DATA

Any database is only useful if its data remains uncorrupted and safe. Performing periodic backups ensures that problems like disk crashes and data corruption are reversible.

See Appendix A [Where GNATS lives], page 73.

4.1 Overview of GNATS configuration

See Appendix A [Where GNATS lives], page 73.

GNATS has two, well, actually three, different kinds of configuration file. The site-wide configuration files determine overall behaviour across all the databases on your machine, while the database-specific configuration files determine how GNATS behaves when dealing with a specific database. In addition, there is a single file that needs to be set up for the send-pr tool to work properly. These files can be edited at any time — the next time a GNATS tool is invoked, the new parameters will take effect.

These are the site-wide configuration files used by GNATS:

- **databases**
  Specifies database names and their associated parameters, such as in which directory they are located. This file is used by the GNATS clients to determine the location of a database referred to by name. See Section 4.2 [The databases file], page 45.

- **defaults**
  This directory contains the set of default per-database configuration files used when a new database is created with mkdb.

- **gnatsd.host_access**
  Controls access levels for the different machines that will do lookups in the databases on this machine. See Appendix C [GNATS access control], page 93.

- **gnatsd.user_access**
  Controls user access levels for the databases on this server. The settings apply to all databases (there is also a database-specific user access level file). See Appendix C [GNATS access control], page 93.

The database-specific configuration is determined by the following files in the gnats-adm subdirectory of the database directory:

- **dbconfig**
  Controls most aspects of how GNATS behaves when dealing with your database. See Section 4.3 [The dbconfig file], page 46.

- **categories**
  The list of categories that GNATS accepts as valid for the Category field, and the maintainers responsible for each category. Update this file whenever you have a new category, or whenever a category is no longer valid. You must also update
this file whenever responsibility for a category changes, or if a maintainer is no longer valid. See Section 4.4.1 [The categories file], page 58.

**responsible**

An alias list mapping names to their associated mailing addresses. The names in this list can have multiple email addresses associated with them. If a responsible user does not show up in this list, they are assumed to be a user local to the system. This list is not associated with just the responsible user field; all email addresses are mapped through this file whenever mail is sent from GNATS. See Section 4.4.2 [The responsible file], page 59.

**submitters**

Lists sites from whom GNATS accepts Problem Reports. The existence of this file is mandatory, although the feature it provides is not; see Section 4.4.3 [The submitters file], page 59.

**addresses**

Mappings between submitter IDs and submitters’ e-mail addresses. Use of this file is optional. If you get Problem reports where the Submitter field is not filled in, GNATS will use entries in this file to try to derive the submitter ID from the e-mail headers. See Section 4.4.5 [The addresses file], page 61.

**states**

Lists the possible states for Problem Reports, typically ranging from open to closed. See Section 4.4.5 [The states file], page 61.

**classes**

Lists the possible classes of Problem Report. This provides an easy way to have “subcategories”, for example by setting up classes such as sw-bug, doc-bug, change-request etc. See Section 4.4.6 [The classes file], page 62.

**gnatsd.user_access**

Specify the access levels for different users to your database. See Appendix C [GNATS access control], page 93.

The last file in this menagerie is the send-pr configuration file send-pr.conf. This file contains some defaults that need to be known in order for send-pr to work. The file needs to be present on all hosts where send-pr is to be used. See Section 4.5 [the send-pr.conf file], page 62.

### 4.2 The databases file

The databases configuration file is a site-wide configuration file containing the list of GNATS databases that are available either on the host itself or remotely over the network, together with some parameters associated with each database.

The file contains one line for each database. For databases located on the host itself, each line consists of three fields separated by colons:

```
database name:short description of database:path/to/database
```

The first field is the database name. This is the name used to identify the database when invoking programs such as query-pr or send-pr, either by using the --database option of the program or by setting the GNATSDB environment variable to the name of the database. The second field is a short human-readable description of the database contents, and the final field is the directory where the database contents are kept.
For a database that is located across a network, but which should be accessible from this host, the entry for the database should look like this:

```
database name:short description of database::hostname:port
```

The first two fields are the same as for local databases, the third field is empty (notice the two adjacent ': ' symbols, indicating an empty field), the fourth field is the hostname of the remote GNATS server, and the fifth field is the port number that the remote GNATS is running on.

If GNATS was built with default options, the `databases` file will be located in the `/usr/local/etc/gnats` directory. However, if the option `--enable-gnats-dblist-file` was used during building of GNATS, the `databases` file has the name and location given to this option. A sample `databases` file is installed together with GNATS.

Note that if you add a new local database, you must create its data directory, including appropriate subdirectories and administrative files. This is best done using the `mkdb` tool, See Section 4.7.1 [mkdb], page 64.

4.3 The `dbconfig` file

The `dbconfig` configuration file controls the configuration of a GNATS database. Each database has its own individual copy of this file, which is located in the `gnats-adm` subdirectory of the database.

The file consists of standard plain text. Whitespace is completely optional and is ignored. Sets of braces `{` are used to delimit the different sections, and all non-keyword values must be surrounded with double quotes. The values in `dbconfig` can be changed at any time; the new values take effect for all subsequent iterations of GNATS tools.

The `dbconfig` file contains 6 major sections, which must appear in the following order:

- Overall database configuration
- Individual field configuration
- Named query definitions
- Audit-trail and outgoing email formats
- Index file description
- Initial Problem Report input fields

The different sections are described below. While reading the following sections, it will be useful to refer to the sample `dbconfig` file which is installed when a new database is initialized with the `mkdb` tool. In fact, the sample file provides a configuration that should be usable for a great range of sites, since it reproduces the behaviour of the older, less customizable 3.x versions of GNATS.

4.3.1 Overall database configuration

The overall database options are controlled by settings in the `database-info` section of the `dbconfig` file. The following is the general format of this section:

```
database-info {
    [options]
}
```

The following options and values may be used in the `database-info` section:
debug-mode  true | false
If set to true, the database is placed into debug mode. This causes all outgoing email to be sent to the gtns-admin user listed in the responsible file of the database. The default value is false.

keep-all-received-headers  true | false
If set to true, all of the Received: headers for PRs submitted via email are kept in the PR. Otherwise, only the first one is kept. The default value is false.

notify-about-expired-prs  true | false
If set to true, notification email about expired PRs is sent via the at-pr command. Otherwise, required times for PR fixes are not used. The default value is false.

send-submitter-ack  true | false
When new PRs are submitted to the database, an acknowledgment email will be sent to the submitter of send-submitter-ack is set to true. This is in addition to the normal notification mail to the person(s) responsible for the new PR. The default value is false.

libexecdir  "directory"
Specifies the directory where the GNATS administrative executables are located. In particular, at-pr and mail-pr are invoked from this directory. The default value is the empty string, which is unlikely to be useful.

business-day-hours  day-start - day-end
Used to specify the hours that define a business day. The values are inclusive and should be given in 24-hour format, with a dash separating them. GNATS uses these values to determine whether the required completion time for a PR has passed. The default values are 8 for day-start and 17 for day-end.

business-week-days  week-start - week-end
Specifies the start and ending days of the business week, where 0 is Sunday, 1 is Monday, etc. The days are inclusive, and the values should be given with a dash separating them. GNATS uses these values to determine whether the required completion time for a PR has passed. The default values are 1 for week-start and 5 for week-end.

create-category-dirs  true | false
If set to true, database directories for categories are automatically created as needed. Otherwise, they must be created manually (usually with the mkcat script). It is recommended that the default value of true be kept.

category-dir-perms  mode
Standard octal mode specification specifying the permissions to be set on auto-created category directories. Default is 0750, yielding user read, write and execute, and group read and execute. Note that if you have local users on the GNATS server itself, running for instance query-pr, you may need to change the permissions to 0755.
### 4.3.2 Individual field configuration

Each type of field in a PR must be described with a `field` section in the `dbconfig` file. These sections have the following general structure:

```
field "fieldname" {
    description "string"
    [ field-options ... ]
    datatype [ datatype-options ... ]
    [ on-change { edit-options ... } ]
}
```

`fieldname` is used as the field header in the PR. The characters > and : are used internally as field markers by GNATS, so they must not be used in fieldnames.

The order in which the `field` sections appear in the `dbconfig` file determines the order in which they appear in the PR text. There is no required order, unlike previous versions of GNATS — the Unformatted field and multitext fields may appear anywhere in the PR.

The following `field-options` may be present within a `field` section:

- **builtin-name "name"**
  Indicates that this field corresponds to one of the GNATS built-in fields.
  GNATS has several fields which are required to be present in a PR, and this option is used to map their external descriptions to their internal usage. The external field names are:

  - **arrival-date**
    The arrival date of the PR
  - **audit-trail**
    The audit-trail recording changes to the PR
  - **category**
    The category that the PR falls into
  - **closed-date**
    The date that the PR was closed
  - **confidential**
    If set to yes, the PR is confidential
  - **description**
    A description of the problem
  - **last-modified**
    The date the PR was last modified
  - **number**
    The PR’s unique numeric identifier
  - **originator**
    The originator of the PR
  - **priority**
    Priority of the PR
  - **responsible**
    The person responsible for handling the PR
  - **severity**
    Severity of the problem described by the PR
state The current state of the PR
submitter-id The user that submitted the PR
synopsis The one-line description of the PR
unformatted PR text which cannot be parsed and associated with other fields.

For these built-in fields, a matching field description must appear in the dbconfig file. Otherwise, the configuration will be considered invalid, and errors will be generated from the GNATS clients and gnatsd. We also recommend that you leave the actual fieldnames of these fields at their default values (i.e. capitalized versions of their built-in names), since some clients may depend on these names.

description "description text"
A one-line human-readable description of the field. Clients can use this string to describe the field in a help dialog. The string is returned from the FDSC command in gnatsd and is also available via the --field-description option in query-pr.
This entry must be present in the field description, and there is no default value.

query-default exact-regexp | inexact-regexp
Used to specify the default type of searches performed on this field. This is used when the ^ search operator appears in a query, and is also used for queries in query-pr that use the old --field query options.
If the option is not given, the default search is exact-regexp.

textsearch
If this option is present, the field will be searched when the user performs a --text search from query-pr. The field is also flagged as a textsearch field in the set of field flags returned by the FIELDFLAGS command in gnatsd.
By default, fields are not marked as textsearch fields.

read-only
When this option is present, the field contents may not be edited — they must be set when the PR is initially created. In general, this should only be used for fields that are given as internal values rather than fields supplied by the user.
By default, editing is allowed.

4.3.3 Field datatypes
Each field description has to contain a datatype declaration which describes what data are to be store in the field. The general format for such a declaration is
datatype [ options ... ]
The available datatypes are:
text [ matching { "regexp" [ "regexp" ... ] } ]
The text datatype is the most commonly used type; it is a one-line text string.
If the **matching** qualifier is present, the data in the field must match at least one of the specified regexps. This provides an easy and flexible way to limit what text is allowed in a field. If no **matching** qualifier is present, no restriction is placed on what values may appear in the field.

**multitext** \[ \{ default "string" \} \]
The field can contain multiple lines of text.

If the **default** option is present, the field will default to the specified **string** if the field is not given a value when the PR is initially created. Otherwise, the field will be left empty.

**enum** \{
  values \{
    "string" [ "string" ... ]
  }
  [ default "string" ]
}\}
Defines an enumerated field, where the values in the PR field must match an entry from a list of specified values. The list of allowed values is given with the **values** option. The **values** option is required for an enumerated field.

If a **default** option is present, it is used to determine the initial value of the field if no entry for the field appears in an initial OR (or if the value in the initial PR is not one of the acceptable values). However, the value in the **default** statement is not required to be one of the accepted values; this can be used to allow the field to be initially empty, for example.

If no **default** option is specified, the default value for the field is the first value in the **values** section.

**multienum** \{
  values \{
    "string" [ "string" ... ]
  }
  [ separators "string" ]
  [ default "string" ]
}\}
The **multienum** datatype is similar to the **enum** datatype, except that the field can contain multiple values, separated by one or more characters from the **separators** list. If no **separators** option is present, the default separators are space (’ ’) and colon (’:’).

The values in the **default** string for this field type should be separated by one of the defined separators. An example clarifies this. If we have a field named **ingredients** where the default values should be ‘sugar’, ‘flour’ and ‘baking powder’ and the separator is a colon (’:’), the following sets these defaults:

```
default "sugar:flour:baking powder"
```
The `enumerated-in-file` type is used to describe an enumerated field with an associated administrative file which lists the legal values for the field, and may optionally contain additional fields that can be examined by query clients or used for other internal purposes. It is similar to the `enum` datatype, except that the list of legal values is stored in a separate file. An example of this kind of field is the built-in Category field with its associated categories administrative file.

`filename` is the name of a file in the `gnats-adm` administrative directory for the database.

The format of the administrative file should be simple ASCII. Subfields within the file are separated with colons (`:`). Lines beginning with a hash sign (`#`) are ignored as comments. Records within the file are separated with newlines.

The `field` option is used to name the subfields in the administrative file. There must be at least one subfield, which is used to list the legal values for the field. If the administrative file is empty (or does not contain any non-empty non-comment lines), the PR field must be empty.

The `key` option is used to designate which field in the administrative file should be used to list the legal values for the PR field. The value must match one of the field names in the `field` option.

If the `allow-any-value` option is present, the value of the PR field is not required to appear in the administrative file — any value will be accepted.

Note that there is no `default` keyword for `enumerated-in-file`. These fields get their default value from the first entry in the associated administrative file.

`multi-enumerated-in-file` is to `multienum` what `enumerated-in-file` is to `enum`. Its options have the same meaning as their counterparts in the `multienum` and `enumerated-in-file` fields.

**NOTE:** Keywords may appear in any sequence, with one exception – the `separators` keyword, if present, has to come last. This rule only applies to fields of type `multi-enumerated-in-file`. 
date

The **date** datatype is used to hold dates. Date fields must either be empty or contain a correctly formatted date.

No defaults or other options are available. The field is left empty if no value for the field is given in the initial PR.

**integer** [ { default "integer" } ]

Integer fields are used to hold numbers. They must either be empty or contain a value composed entirely of digits, with an optional leading sign.

If the `default` option is present, the field will have the value of `integer` if the field is not given a value when the PR is initially created. Otherwise, the field will be left empty.

### 4.3.4 Edit controls

The **on-change** subsection of a **fields** section specifies one or more actions to be performed when the field value is edited by the user. It has the general form

```plaintext
on-change [ "query-expression" ] {
    [ add-audit-trail ]
    [ audit-trail-format {
        format "formatstring"
        [ fields { "fieldname" ... } ]
    }]
    [ require-change-reason ]
    [ set-field | append-to-field "fieldname" {
        "format-string" [ fieldlist ]
    }]
    [ require { "fieldname" ... } ]
}
```

The optional `query-expression` controls whether or not the actions in the **on-change** section are taken. If the expression fails to match, the actions are skipped.

The **add-audit-trail** option indicates that an entry should be appended to the PR’s audit-trail when this field is changed. The format of the entry is controlled by the optional **audit-trail-format** section within the field, or by the global **audit-trail-format** section. See Section 4.3.6 [Audit-trail formats], page 53 and Section 4.3.7 [Outgoing email formats], page 54.

The **require-change-reason** option specifies that a change reason must be present in the PR when this field is edited. This option only makes sense if an audit-trail entry is required, as the change reason is otherwise unused.

The **set-field** and **append-to-field** options are used to change the value of the field **fieldname** in the PR. The supplied format is used to format the value that will be placed in the field.

**append-to-field** appends the resulting formatted string to the existing, while **set-field** completely replaces the contents.

Any field may be edited by the **set-field** or **append-to-field** option (the **read-only** option on a field is ignored). However, the changes are subject to the usual field content checks.
The **require** option specifies that one or more fields must have a (non-blank) value when this field is changed.

A field may be enforced to have a (non-blank) value at all times by including it in the set of fields required for the initial PR, see Section 4.3.9 [Initial PR input fields], page 57, as well as in the set of fields required on change of the field itself.

There is also a global **on-change** section that is executed once for each PR edit. A typical use for such a section is to set the last-modified date of the PR.

### 4.3.5 Named query definitions

When queries are performed via **query-pr**, they can refer to a query format described by a query section in the **dbconfig** file:

```plaintext
query "queryname" {
  format "formatstring"
  [fields { "fieldname" [ "fieldname" ... ] } ]
}
```

*formatstring* is as described in Section 2.4.2 [Formatting query-pr output], page 24. It basically contains a string with printf-like % escapes. The output of the query is formatted as specified by this format string.

The **fields** option lists the fields to be used with the format string. If the **fields** option is present without a **format** option, the contents of the listed fields are printed out, separated by newlines.

The named query formats **full**, **standard** and **summary** must be present in the **dbconfig** file. **full** and **summary** correspond to the **query-pr** options --full and --summary, while **standard** is used when no format option is given to **query-pr**.

### 4.3.6 Audit-trail formats

These formats are similar to the named query formats, but they include more options. They are used for formatting audit-trail entries and for outgoing email messages.

There is currently only one audit-trail format, defined by the **audit-trail-format** option:

```plaintext
audit-trail-format {
  format "formatstring"
  [ fields { "fieldname" [ "fieldname" ... ] } ]
}
```

For those fields that require an audit-trail entry, the audit-trail text to be appended is formatted as described by this format. The per-field **audit-trail-format** is used in preference to this one, if it exists.

*formatstring* and *fieldname* are similar to those used by the named query format. *fieldname* may also be a **format parameter**, which is a context-specific value. (Format parameters are distinguished from fieldnames by a leading dollar sign ($$)).

The following format parameters are defined for **audit-trail-format** entries:

**$Fieldname**

The name of the field for which an audit-trail entry is being created.
$OldValue
The old value of the field.

$NewValue
The new field value.

$EditUserEmailAddr
The email address of the user editing the field. Set by the EDITADDR gnatd command or from the responsible file; if not available, user’s local address is used.

$CurrentDate
The current date.

$ChangeReason
The reason for the change; may be blank if no reason was supplied.

These parameters may be used anywhere a fieldname can appear.

4.3.7 Outgoing email formats
During the life of a PR, GNATS can be configured to send out a range of email messages. When a PR first arrives, an acknowledgment message is sent out if the send-submitter-ack parameter above is set to true. Certain edits to the PR may also cause email to be sent out to the various parties, and if a PR is deleted, GNATS may send notification email.

The formats of the email messages are controlled by mail-format sections in the dbconfig file. The general structure of a mail-format section is as follows:

```
mail-format "format-name" {
  from-address {
    [ fixed-address "address" ]
    [ email-header-name | [ mail-header-name | ... ] ]
  }
  to-address {
    [ fixed-address "address" ]
    [ "email-header-name" | [ "mail-header-name" | ... ] ]
  }
  reply-to {
    [ fixed-address "address" ]
    [ "email-header-name" | ... ] | [ "gnats-field-name" | ... ]
  }
  header {
    format "formatstring"
    [ fields { "fieldname" [ "fieldname" ... ] } ]
  }
  body {
    format "formatstring"
    [ fields { "fieldname" [ "fieldname" ... ] } ]
  }
}
```

gnats recognizes and uses 6 different format-name values:
initial-response-to-submitter
Format of the message used when mailing an initial response back to the PR submitter. This message will only be sent if send-submitter-ack in the overall database configuration is set to true.

initial-pr-notification
Format of the message sent to the responsible parties when a new PR with category different from “pending” arrives.

initial-pr-notification-pending
Format of the message sent to the responsible parties when a new PR that ends up with category “pending” arrives.

appended-email-response
Format of the notification message sent out when a response to a PR is received via email.

audit-mail
Format of the message sent out when a PR edit generates an Audit-Trail entry.

deleted-pr-mail
Format of the message sent out when a PR is deleted.

The from-address, to-address and reply-to subsections of a mail-format section specify the contents of the To:, From: and Reply-To: headers in outgoing email. There are two ways to specify the contents: by using a fixed-address specification, or by specifying email-header-names or gnis-field-names.

When an email-header-name or gnis-field-name value is given, GNATS will attempt to extract an email address from the specified location. If several values are given on the same line, separated by ‘|’ characters, GNATS will try to extract an address from each location in turn until it finds a header or field which is nonempty. The following example should clarify this:

```
mail-format "initial-response-to-submitter" {
    from-address {
        fixed-address "gnats-admin"
    }
    to-addresses {
        "Reply-To:" | "From:" | "Submitter-Id"
    }
} ...
```

This partial mail-format section specifies the format of the address headers in the email message that is sent out as an acknowledgment of a received PR. The From: field of the message will contain the email address of the GNATS administrator, as specified by the gnis-admin line in the responsible file. To fill in the To: header, GNATS will first look for the mail header Reply-To: in the PR and use the contents of that, if any. If that header doesn’t exist or is empty, it will look for the contents of the From: email header, and if that yields nothing, it will look for the GNATS Submitter-Id field and use the contents of that.

Other email headers to be included in messages sent out by GNATS can be specified by header subsections of the mail-header section. formatstring andfieldname are similar to those used by the named query format. Each header line must have a newline character ('\n') at the end.
The email message body is specified in the `body` subsection of the `mail-format` section. Just as for a `header` section, the `body` section must contain a `formatstring` and `fieldname` values.

For some of the formats that GNATS recognizes, special variables are available for use. The following table lists the formats that provide special variables. See the example below for an illustration of how they are used.

### appended-email-response

- **$MailFrom**  
The From: line of the original message.
- **$MailTo**  
The To: line of the original message.
- **$MailSubject**  
The Subject: line of the original message.
- **$MailCC**  
The CC: line of the original message.
- **$NewAuditTrail**  
The text of the new audit trail entry (corresponds to the body of the message).

### audit-mail

- **$EditUserEmailAddr**  
The email address of the user editing the PR. Set by the `EDITADDR` `gnatsd` command or from the `responsible` file; if not available, user's local address is used.
- **$OldResponsible**  
The previous Responsible field entry, if it was changed.
- **$NewAuditTrail**  
The Audit-Trail: entries that have been appended by the edits.

### deleted-pr-mail

- **$EditUserEmailAddr**  
The email address of the user deleting the PR. Set by the `EDITADDR` `gnatsd` command or from the `responsible` file; if not available, user's local address is used.
- **$PRNum**  
The number of the PR that was deleted.

The following example illustrates the use of these special variables:

```plaintext
mail-format "deleted-pr-mail" {
  from-address {
    "$EditUserEmailAddr"
  }
  to-addresses {
    fixed-address "gnats-admin"
  }
  header {
    format "Subject: Deleted PR %s\n"
  }
}
This mail-format section specifies the format of the email message that is sent out when a PR is deleted. The From: field is set to the email address field of the user who deleted the PR, the subject of the message contains the number of the deleted PR, and the message body contains both the PR number and the user's email address.

### 4.3.8 Index file description

The index section of the dbconfig file lists the fields that appear in the database index. The index is always keyed by PR number. The general format for the index section is:

```plaintext
index {
  path "file"
  fields { "fieldname" [ "fieldname" ... ] }
  binary-index true | false
  [ separator "symbol" ]
}
```

The path parameter gives the name of the index file in the gnats-adm directory of the database. Only one index is allowed per database, so only one path entry is allowed.

The fields parameter controls what fields will appear, and in what order, in the index. Fields are listed by their names, separated by spaces ('). Fields will appear in the order they are listed.

The binary-index parameter controls whether the index is supposed to be in plaintext or binary format. Binary format is recommended, as it avoids potential problems when field separators appear as bona-fide field contents.

When plaintext format is used, by setting binary-index false, the symbol (') is used as a field separator in the index, unless the optional separator parameter is used to redefine the separator character.

### 4.3.9 Initial PR input fields

An initial-entry section in the dbconfig file is used to describe which fields should be present on initial PR entry; this is used by tools such as send-pr to determine which fields to include in a “blank” PR template. An optional require parameter can be defined to specify a subset of the initial-entry fields which must be assigned a value upon initial creation of the PR.

The general format for the initial-entry section is:

```plaintext
initial-entry {
  fields { "fieldname" [ "fieldname" ... ] }
  [ require { "fieldname" [ "fieldname" ... ] } ]
}
```
4.4 Other database-specific config files

4.4.1 The categories file

The categories file contains a list of problem categories, specific to the database, which GNATS tracks. This file also matches responsible people with these categories. You must edit this file initially, creating valid categories. In most installations, GNATS is configured to create directories on disk for valid categories automatically as needed (see Section 4.3.1 [Overall database configuration], page 46). If GNATS isn’t set up to do this, you need to run mkcat to create the corresponding subdirectories of the database directory. For instructions on running mkcat, see Section 4.7.2 [Adding a problem category], page 64.

To create a new category, log in as gnats, add a line to this file, and run mkcat if applicable. Lines beginning with ‘#’ are ignored.

A line in the categories file consists of four fields delimited by colons, as follows:

```plaintext
category: description: responsible: notify
```

category  A unique category name, made up of text characters. This name cannot contain spaces or any of the following characters: 

! $ * ( ) { } [ ] ' " ; : < > ~

Ideally, category names should not contain commas or begin with periods. Each line has a corresponding subdirectory in the database directory.

description  A terse textual description of the category.

responsible  The name tag of the party responsible for this category of problems, as listed in the responsible file (see Section 4.4.2 [The responsible file], page 59).

notify  One or more other parties which should be notified when a Problem Report with this category arrives, such as a project manager, other members of the same project, other interested parties, or even log files. These should be separated with commas.

A good strategy for configuring this file is to have a different category for each product your organization supports or wishes to track information for.

```plaintext
rock:ROCK program:me:myboss,fred
stone:STONE utils:barney:fred
iron:IRON firewall:me:firewall-log
```

In the above example, the nametags ‘myboss’, ‘me’, ‘fred’, and ‘barney’ must be defined in the responsible file (see Section 4.4.2 [The responsible file], page 59).

Problem Reports with a category of ‘rock’ are sent to the local mail address (or alias) ‘me’, and also sent to the addresses ‘myboss’ and ‘fred’. PRs with a category of ‘stone’ are sent to the local addresses ‘barney’ and ‘fred’ only, while PRs with the category ‘iron’ are sent only to ‘me’, and are also filed in firewall-log (in this case, a mail alias should be set up, see Section 3.5 [Setting up mail aliases], page 36.

If you want to separate PRs in each problem category into specific subsets, say documentation and software bugs, using the classes file is recommended. See Section 4.4.6 [The classes file], page 62.

Only one category must be present for GNATS to function:
pending: Non-categorized PRs: gnats-admin:

The pending directory is created automatically when you run `mkdb` to initialize a new database. (see Section 3.1 [Configuring and compiling the software], page 31).

4.4.2 The responsible file

This file contains a list of the responsible parties. Lines beginning with ‘#’ are ignored. Each entry contains three fields, separated by colons:

```
responsible: full-name: mail-address
```

- **responsible**
  A name-tag description of the party in question, such as her or his user name, or the name of the group. This name is listed in the PR in the Responsible field.

- **full-name**
  The full name of the party (“Charlotte Bronte”; “Compiler Group”).

- **mail-address**
  The full, valid mail address of the party. This field is only necessary if the responsible party has no local mail address or alias.

A sample responsible listing might be:

```
ren: Ren Hoek:  
stimpy: Stimpson J. Cat: stimpy@lederhosen.org
```

Here, ren is a local user. stimpy is remote, so his full address must be specified.

The following entry must be present for GNATS to function:

```
gnats-admin: GNATS administrator:
```

*gnats-admin* is usually defined as a mail alias when GNATS is installed, so for this purpose *gnats-admin* is a local address. However, this line can also be used to redefine the email address of the GNATS administrator, by adding the desired address at the end of the line.

4.4.3 The submitters file

This is a database of sites which submit bugs to your support site. It contains six fields delineated by colons. Lines beginning with ‘#’ will be ignored.

Entries are of the format:

```
submitter-id: name: type: resp-time: contact: notify
```

- **submitter-id**
  A unique identifier for a specific site or other entity who submits Problem Reports. The first submitter-id listed in the file will be the default for PRs that arrive with invalid or empty submitter fields.

- **name**
  The full name or a description of this entity.

- **type**
  Optional description for the type of relationship of this submitter to your support site. This could indicate a contract type, a level of expertise, etc., or it can remain blank.

- **resp-time**
  Optional quoted response time in business hours. If the database `dbconfig` file has the `notify-about-expired-prs` entry set to true (see Section 4.3.1 [Overall database configuration], page 46, GNATS will use this field to schedule when it
should notify the gnats-admin, responsible person and submitter contact that the PR wasn’t analyzed within the agreed response time. Business hours and business-week days are set in the dbconfig file. For information on at-pr, the program which sends out this reminder, see Section 4.8.3 [Timely Reminders], page 68.

**contact** The name tag of the main contact at the Support Site for this submitter. This contact should be in the responsible file (see Section 4.4.2 [The responsible file], page 59). Incoming bugs from submitter are sent to this contact. Optionally, this field can be left blank.

**notify** Any other parties who should receive copies of Problem Reports sent in by submitter. They need not be listed in the responsible file.

A few example entries in the submitters file:

```
univ-hell:University of Hades:eternal:3:beelzebub:lucifer
tta:Telephones and Telegraphs of America:720:dave:
```

In this example, when a PR comes in from the University of Hades (who has an eternal contract), it should have ‘univ-hell’ in its Submitter-Id field. This Problem Report goes to beelzebub (who should be in the responsible file), and if it is not analyzed within three business hours a reminder message is sent. lucifer also receives a copy of the bug, and a copy of the reminder message as well (if it is sent). When Telephones and Telegraphs of America utilizes their support contract and submits a bug, a copy is sent only to dave, who has 720 business hours to return an analysis before a reminder is sent.

To disable the feature of GNATS which tracks the Submitter-Id, simply alter the submitters file to only contain one submitter-id value, and instruct your submitters to ignore the field.

### 4.4.4 The states file

This file lists the possible states for Problem Reports. Each entry has up to three fields, separated by colons. Lines beginning with ‘#’ will be ignored.

```
state:type:description
```

**state** The name of the state. It may contain alphanumerics as well as ‘-’ (hyphen), ‘_’ (underscore), or ‘.’ (period), but no other characters.

**type** This is the type of the state. This field is optional and it may contain alphanumerics as well as ‘-’ (hyphen), ‘_’ (underscore), or ‘.’ (period), but no other characters.

The concept of the type of a state recognizes that there may for instance be several possible states for a Problem Report which effectively means that the PR is closed and that there may be certain actions that need to be taken when a PR reaches a “closed state”. The problem may have been resolved, it might have been decided that the problem is unsolvable or simply that it won’t be solved. Some organizations may for instance wish to consider the “suspended” state as a state of type “closed”.

Currently, the only defined state types are “open” and “closed”, the “open” type isn’t currently used for anything while the “closed” type is only used to control the Closed-Date field of PRs. Changing the state of a PR to any state
of type “closed” will set the Closed-Date field with a time stamp and changing
the state of a PR from one “closed” state to another will leave the Closed-Date
field as it was. Changing the state of a PR from any state of type “closed” to
a non-closed state will clear the Closed-Date field.
The --skip-closed option of query-pr refers to all states of type “closed”,
ot to a specific state name of “closed”.

description
This is an optional one-line description of what the state means. Any charac-
ter is okay in the description; a newline ends it. GNATS itself does not currently
use the description for anything, but certain external tools (such as TkGnats
and Gnatsweb) look for it, so it’s a good idea to include one for every state.

The first state listed will be the state automatically assigned to Problem Reports when
they arrive; by default this is named “open”. The last state listed is the end state for
Problem Reports — one should usually assume that a PR in this state is not being actively
worked on; by default this state is named “closed”. Even if a different name has been chosen
for this state, GNATS will force this state to be of type “closed”.

It is recommended that you keep the default names of “open” and “closed” for the first
and last states respectively, since there may be external tools that depend on these names.

4.4.5 The addresses file
This file contains mappings between submitter IDs and corresponding e-mail addresses.

When a PR comes in without a submitter ID (if someone sends unformatted e-mail to
the PR submission email address), GNATS will try to derive the submitter ID from the
address in the "From:" header. The entries in this file consist of two fields, separated by a
colon:

submitter-id:address-fragment

submitter-id
A valid submitter ID

address-fragment
Part of, or all of the e-mail address to be matched

Here is an example of an addresses file:

# Addresses for Yoyodine Inc
yoyodine:yoyodine.com
yoyodine:yoyodine.co.uk
# Addresses for Foobar Inc.
foobar1:sales.foobar.com
foobar2:admin.foobar.com
foobar3:clark@research.foobar.com

GNATS checks each line in the addresses file, comparing address-fragment to the end of
the "From:" header, until it finds a match. If no match is found, GNATS uses the default
submitter ID.

You can only have one address fragment per line, but you can have more than one line
for a given submitter ID. An address fragment can be a domain (i.e. yoyodine.com), a
machine location (admin.foobar.com), or a full e-mail address (clark@research.foobar.com).

GNATS can match addresses in three e-mail formats:
'From: name@address.com'
   The address by itself without a full name, not enclosed in brackets

'From: Real Person <name@address.com>'
   A full name (optional, with or without quotation marks), followed by the ad-
   dress enclosed in angle brackets

'From: name@address.com (Real Person)'
   An address, followed by a name or comment in parentheses

If gnats sees other e-mail address formats, it uses the default submitter ID.

4.4.6 The classes file
This file lists the possible classes of Problem Reports. Each line consists of a class name,
followed by a colon and an optional class type name (the class type name is not used for
anything in the current implementation of gnats, so it may be left blank. The class and
class-type-name fields may only contain alphanumerics, ‘-‘, ‘_’, and ‘.’, but no other
characters.

Then comes another colon, followed by an optional one-line description of the class.
Gnats itself does not use the class description, but external tools such as Gnatsweb and
TkGnats may use it. Therefore, a line in this file should at least contain the following:

    class::class description

Lines beginning with ‘#’ will be ignored, and the first listed class is the default class for
an incoming Problem Report.

4.5 The send-pr.conf file
This file contains some default values that need to be known in order for send-pr to work
properly. This file needs to be copied to all hosts where send-pr will be used.

If gnats was built with default options, the send-pr.conf file should be placed in the
/usr/local/etc/gnats directory. However, if the option --sysconfdir was used during
building of gnats, the send-pr.conf file resides at the location given to this option.

Entries in this file are on the format

    variable=value

The valid variables are:

SUBMITTER
   The default value to be used for the Submitter-Id field when send-pr is invoked.

DEFAULT_RELEASE
   The default value to be used for the Release field (only applicable if the Release
   field is defined in the dbconfig file.

DEFAULT_ORGANIZATION
   The default value to be used for the Organization field. (only applicable if the
   Organization field is defined in the dbconfig file.

MAILPROG
   If the gnats server can’t be reached directly over the network, i.e. it is behind
   a firewall or suchlike, send-pr can be set up to submit Problem Reports by
   e-mail. This is done by setting the MAILPROG variable to point to a mailer such
as Sendmail. If MAILPROG needs to have the address that the mail is being sent to specified on the command line, it should be specified here as well (for example, ‘MAILPROG=’mail bugs@foo.bar.com’’ should work). If Sendmail is used, use ‘MAILPROG=’/usr/lib/sendmail -oi -t’’. See also MAILADDR and TEMPLATE below.

MAILADDR If using e-mail to submit PRs, this is the address that PRs should be sent to.

TEMPLATE When invoked, send-pr communicates directly over the network with the GNATS server to determine what fields to include in a correctly formatted Problem Report so that it can present the user with a template. If the GNATS server can’t be reached directly over the network, a template must be provided. Set the TEMPLATE variable to point to a template file created on the GNATS server by using the command send-pr -p. See Section 3.7 [Installing the user tools], page 37.

4.6 Administrative data files

The following files are database-specific and are located in the gnats-adm subdirectory of the database directory. These files are maintained by GNATS; you should never need to touch them.

4.6.1 The index file

The index is used to accelerate searches on the database by query-pr and edit-pr. This file is not created until the first PR comes in. It is then kept up to date by GNATS; you should never touch this file.

Searches on subjects contained in the index are much faster than searches which depend on data not in the index. Inexes come in two different formats: binary and plain-text. Binary indexes are safer, in that they avoid certain problems that may crop up if the field separators used by plain-text indexes appear in field data.

A plain-text index contains single-line entries for all PR fields except for the multitext fields such as Description, How-To-Repeat, etc. Fields are separated by bars (‘|’) except for Category and Number, which are separated by a slash (‘/’).

Binary indexes are not meant to be human-readable, but they are safer than the plain-text variety, in that they avoid certain problems that may crop up if the field separators used by plain-text indexes appear in field data.

The format of the index for a database is set in the dbconfig file. See Section 4.3.8 [Index file description], page 57.

Should the index file become corrupted, the gen-index utility can be used to regenerate it. See Section 4.7.4 [Regenerating the index], page 64.

4.6.2 The current file

This file contains the last serial number assigned to an incoming PR. It is used internally by GNATS; you need never touch this file.
4.7 Administrative utilities

These tools are used by the GNATS administrator as part of the periodic maintenance and configuration of GNATS. See Chapter 4 [GNATS Administration], page 43.

4.7.1 Adding another database

To initialize a new GNATS database:

1. Add a line for the new database in the `databases` file (see Appendix A [Where GNATS lives], page 73).
2. Run `mkdb`, using

```
    mkdb database
```

where `database` is the database you specified in the `databases` file. `mkdb` creates the database directory and populates it with the directories `pending`, `gnats-queue` and `gnats-adm`. A full set of sample configuration files is copied to the `gnats-adm` directory.

4.7.2 Adding a problem category

To add new categories to the database:

1. Add a line to for each new category to the `categories` file in the `gnats-adm` directory of the database. See Section 4.4.1 [The categories file], page 58.
2. Run `mkcat` If applicable. If `create-category-dirs` is set to `false` in the database `dbconfig` file, you need to create category directories with `mkcat`. `mkcat` creates a subdirectory under the database directory for any new categories which appear in the `categories` file.

4.7.3 Removing a problem category

To remove a category from the database:

1. Remove the Problem Reports from the subdirectories corresponding to the categories you wish to remove, or assign the PRs to new categories. All PRs for a given category reside in a subdirectory of the database directory, with the same name as the category.
2. Run `rmcat` using

```
    rmcat category [ category... ]
```

where `category` is the category you wish to remove. You can specify as many categories as you wish as long as each category has no PRs associated with it. `rmcat` removes the directory where the Problem Reports for that category had been stored.

4.7.4 Regenerating the index

If your `index` file becomes corrupted, or if you need a copy of the current index for some reason, use

```
```

With no options, `gen-index` generates an index that is sorted by the order that the categories appear in the `categories` file. The index is generated in plaintext or binary format.
Chapter 4: GNATS Administration

according to the value of `binary-index` in the `dbconfig` file (see Section 4.3.8 [Index file description], page 57). The results are printed to standard output. The options are:

- `n`  
  --numeric
  Sorts index entries numerically.

- `d` `databasename`
  --database=`databasename`
  Specifies the database to index. If this option is left out, `gen-index` attempts to index the database with name taken from the the `GNATSDB` environment variable, and if that is undefined, the default database, as set when GNATS was built (usually `default`).

- `o` `filename`
  --outfile=`filename`
  Places output in `filename` rather than sending it to standard output.

- `i`
  --import
  Import the existing index file instead of re-indexing the database.

- `e`
  --export
  Force plaintext output.

- `h`
  --help
  Prints the usage for `gen-index`.

- `V`
  --version
  Prints the version number for `gen-index`.

### 4.7.5 Checking database health

The `check-db` script is useful for performing periodic checks on database health. It accepts the following options:

- `d` `databasename`
  --database=`databasename`
  Determines the database which to operate on.

- `--all-databases`
  Check all GNATS databases on the system. This option takes precedence over the --database option.

If no option is given, the default database is checked.

During its operation, `check-db` first attempts to lock `database`. If this is not possible, it repeats the locking attempts for five minutes; if it fails, it sends a mail message notifying the administrator of the failure and exits.

Once the database is locked, the script searches the database for lock files that are more than 24 hours old. Any old lock files are reported to the administrator in a mail message.

After checking for old lock files, it calls `gen-index` (see Section 4.7.4 [Regenerating the index], page 64) and compares the results with the current index file of the database; any inconsistencies are reported to the administrators in a mail message.
After checking the index file for inconsistencies, the script unlocks the database and
exits.

4.7.6 Managing user passwords
Older versions of Gnats, up to and including version 3.x, stored user passwords in plaintext in the `gnatsd.user_access` files. Version 4 has the options of storing the password as MD5 or standard DES crypt() hashes (as most UNIX versions do in the system password files) as well as in plaintext. Since the password strings require a prefix to indicate how they are encrypted, one is forced to convert the old password files to a new format when upgrading to Gnats version 4. See Section 3.8 [Upgrading from older versions], page 39.

The `gnats-pwconv` tool takes care of converting the old password files to the new format:

```
$ gnats-pwconv [ -c | --crypt ] [ -m | --md5 ] [ -p | --plaintext ]
          [ -h | --help ] [ -V | --version ] INFILE [OUTFILE]
```

Unless the --version or --help options are given, exactly one encryption method must be specified, as well as an input file. The output file parameter is optional. If one is not specified, results will be printed on standard output.

4.8 Internal utilities
These tools are used internally by Gnats. You should never need to run these by hand; however, a complete understanding may help you locate problems with the Gnats tools or with your local implementation.

4.8.1 Handling incoming traffic
The program `queue-pr` handles traffic coming into Gnats. `queue-pr` queues incoming Problem Reports in the `gnats-queue` directory of the database, and then periodically (via `cron`) passes them on to `file-pr` to be filed in the Gnats database. See Chapter 3 [Installing Gnats], page 31.

The usage for `queue-pr` is as follows:

```
$ queue-pr [ -q | --queue ] [ -r | --run ]
                  [ -f filename | --file=filename ]
                  [ -m kbytes | --max-size=kbytes ]
                  [ -d databasename | --database=databasename ]
                  [ -h | --help ] [ -V | --version ]
```

One of `-q` or `-r` (or their longer-named counterparts) must be present upon each call to `queue-pr`. These options provide different functions, as described below.

- `q`  
  --queue  Accepts standard input as an incoming mail message, placing this message in an incrementally numbered file in the `gnats-queue` directory under the database directory (see Appendix A [Where Gnats lives], page 73).

- `r`  
  --run  Redirects files in the `gnats-queue` directory into the program `file-pr` one by one.

- `f filename`  
  --file=filename  
  Used with `-q` (or `--queue`), accepts the file denoted by `filename` as input rather than reading from standard input.
-m kbytes
   --max-size=kbytes
   Do not process messages larger then kbytes kilobytes. Files larger than the
   limit are left for human intervention.

-d databasename
   --directory=databasename
   Specifies database to operate on. If this option is left out, the value of the
   GNATSDB environment variable is used, and if that is undefined, the default
   database name set when GNATS was built is used (usually default).

-h
   --help
   Prints the usage for gen-index.

-V
   --version
   Prints the version number for gen-index.

4.8.2 Processing incoming traffic

queue-pr hands off queued Problem Reports to file-pr one at a time. file-pr checks
each Problem Report for correct information in its fields (particularly a correct Category),
assigns it an identification number, and files it in the database under the appropriate cate-
gory.

If the Category field does not contain a valid category value (i.e., matching a line in
the categories file; see Section 4.4.1 [The categories file], page 58), the PR is assigned
to the default category, as set in the dbconfig file. If there is no default category defined,
the PR is given a Category value of ‘pending’ and is placed in the pending directory. The
GNATS administrator is notified of the unplaceable PR.

file-pr assigns the Problem Report an identification number, files it in the GNATS
database (under the default, if the Category field contains an invalid category), and sends
acknowledgments to appropriate parties. For the default GNATS configuration, the person
responsible for that category of problem (see Section 4.4.1 [The categories file], page 58)
and the person responsible for the submitter site where the PR originated (see Section 4.4.3
[The submitters file], page 59) receive a copy of the PR in its entirety. Optionally, the
originator of the PR receives an acknowledgment that the PR arrived and was filed (see
Section 4.1 [Changing your GNATS configuration], page 44).

The usage for file-pr is as follows:

    file-pr [ -f filename | --file=filename ]
              [ -d databasename | --database=databasename ]
              [ -h | --help ] [ -V | --version ]

network options:
    [ -H host | --host=host ]
    [ -P port | --port=port ]
    [ -v username | --user=username ]
    [ -w password | --passwd=password ]

file-pr requires no options in order to operate, and takes input from standard input
(normally, the output of `queue-pr -r`) unless otherwise specified. The options include:
-f filename
  --filename=filename
  Uses filename as input rather than standard input.

-d databasename
  --database=databasename
  Performs refiling operations on database. If this option is left out, the value
  of the GNATSDB environment variable is used, and if that is undefined, the
default database name set when GNATS was built is used (usually default).

-h
  --help
  Prints the usage for file-pr.

-V
  --version
  Prints the version number for file-pr.

file-pr can file PRs across a network, talking to a remote gnatsd. The following options
relate to network access:

-H host
  --host=host
  Hostname of the GNATS server.

-P port
  --port=port
  The port that the GNATS server runs on.

-v username
  --username=username
  Username used to log into the GNATS server.

-w password
  --passwd=password
  Password used to log into the GNATS server.

4.8.3 Timely reminders

at-pr creates a queued job using at which, after an allotted response time is past, checks
the PR to see if its state has changed from ‘open’. When the PR is originally filed, file-pr
checks the notify-about-expired-prs parameter in the dbconfig file. If this parameter
is set to true, file-pr calls at-pr, which sets up the expiry check.

The submitters file contains the response time for each >Submitter-Id: (see Section 4.4.3 [The submitters file], page 59). The time is determined in business hours,
which are defined in the database’s dbconfig file (see Section 4.3.1 [Overall database configuration], page 46).

If the PR is urgent and is still open after the requisite time period has passed, at-pr sends
a reminder to the GNATS administrator, to the maintainer responsible for that submitter,
and to the maintainer responsible for the PR with the following message:

To: submitter-contact responsible gnats-admin
Subject: PR gnats-id not analyzed in #hours hours
PR gnats-id was not analyzed within the acknowledgment period of #hours business hours. The pertinent information is:

Submitter-Id: submitter
Originator: full name of the submitter
Synopsis: synopsis
Person responsible for the PR: responsible

The GNU Problem Report Management System (GNATS)

The PR is urgent if its priority is ‘critical’ or if its priority is ‘serious’ and the severity is ‘high’.

4.8.4 The edit-pr driver

pr-edit does the background work for edit-pr, including error-checking and refiling newly edited Problem Reports, handling file and database locks and deletion of PRs. It can be called interactively, though it has no usable editing interface.

The usage for pr-edit is:

```
pr-edit [ -l username | --lock=username ] [ -u | --unlockdb ]
[ -L | --lockdb ] [ -U | --unlockdb ] [ -c | --check ]
[ -C | --check-initial ] [ -s | --submit [ --show-prnum ] ]
[ -a field | --append field=field ]
[ -r field | --replace=field ] [ --delete-pr ]
[ -R reason | --reason=reason ]
[ -p process-id | --process=process-id ]
[ -d database | --database=database ]
[ -f filename | --filename=filename ]
[ -V | --version ]
[ -h | --help ] [ -v username | --user=username ]
[ -w passwd | --passwd=passwd ]
[ -H host | --host=host ]
[ -P port | --port=port ]
[ -D | --debug ] [ PR number ]
```

A lock is placed on a Problem Report while the PR is being edited. The lock is simply a file in the locks subdirectory of the gnats-adm directory of the database, with the name gnats-id.lock, which contains the name of the user who created the lock. user then “owns” the lock, and must remove it before the PR can be locked again, even by the same user. If a PR is already locked when you attempt to edit it, pr-edit prints an error message giving the name of the user who is currently editing the PR.

If you do not specify PR number, pr-edit reads from standard input. You must specify PR number for the functions which affect PR locks, ‘--lock= username’ and ‘--unlock’.

-L
--lockdb Locks the database specified with the --database or -d option. No PRs may be edited, created or deleted while the database is locked. This option is generally used when editing the index file.

---

1 This approach may seem heavy-handed, but it ensures that changes are not overwritten.
-U
--unlockdb
Unlocks the specified database. No check is made that the invoking user actually had locked the database in the first place; hence, it is possible for anyone to steal a database lock.

-c
--check
-C
--check-initial
The --check options are used to verify that a proposed PR’s field contents are valid. The PR is read in (either from stdin or a file specified with --filename), and its fields are compared against the rules specified by the database configuration of the selected database. Warnings are given for enumerated fields whose contents do not contain one of the required values or fields that do not match required regexps. --check-initial is used to verify initial PRs, rather than proposed edits of existing PRs.

-s
--submit
Used to submit a new PR to the database. The PR is read in and verified for content; if the PR is valid as an initial PR, it is then added to the database. If the submission is successful a zero exit code is returned. Otherwise, the reason(s) for the PR being rejected are printed, and a non-zero exit code is returned.

--show-prnum
This option is used with the --submit option to display the PR number associated with the submitted PR.

The following options require a PR number to be given.

--delete-pr
Deletes the specified PR from the database. The PR must be in a closed state, and not locked. Only the user gnats (or the user name specified instead of gnats during the building of GNATS) is permitted to delete PRs.

-l username
--lock=username
Locks the PR. username is associated with the lock, so the system administrator can determine who actually placed the lock on the PR. However, anyone is permitted to remove locks on a PR. If the optional --process or -p option is also given, that process-id is associated with the lock.

-u
--unlock
Unlocks the specified PR.

-a field
--append=field
-r field
--replace=field
--append and --replace are used to append or replace content of a specific field within a PR. The new field content is read in from stdin (or from the file
specified with the `--filename` option), and either appended or replaced to the specified field. The field contents are verified for correctness before the PR is rewritten. If the edit is successful, a zero exit status is returned. If the edit failed, a non-zero exit status is returned, and the reasons for the failure are printed to stdout.

`-R reason`
`--reason=reason`

Certain PR fields are configured in the database configuration to require a short text describing the reason of every change that happens to them, See Section 4.3 [dbconfig file], page 46. If you edit a problem and change any of such fields, you must issue a short text, the `reason` of the change, through this option. If the option is used and no change-reason requiring field is actually changed, the option has no effect.

**PR number**

If only a PR number is specified with no other options, a replacement PR is read in (either from stdin or the file specified with `--filename`). If the PR contents are valid and correct, the existing PR is replaced with the new PR contents. If the edit is successful, a zero exit status is returned. If the edit failed, a non-zero exit status is returned, and the reasons for the failure are printed to stdout.

`-d database`
`--database=database`

Specifies the database which is to be manipulated. If no database is specified, the default database name set when GNATS was built is used (usually `default`). This option overrides the database specified in the GNATSDB environment variable.

`-f filename`
`--filename=filename`

For actions that require reading in a PR or field content, this specifies the name of a file to read. If `--filename` is not specified, the PR or field content is read in from stdin.

`-h`
`--help`

Prints the usage for `pr-edit`.

`-V`
`--version`

Prints the version number for `pr-edit`.

`pr-edit` can edit PRs across a network, talking to a remote gnatsd. The following options relate to network access:

`-H host`
`--host=host`

Hostname of the GNATS server.

`-P port`
`--port=port`

The port that the GNATS server runs on.
-v username
--username=username
   Username used to log into the GNATS server.

-w password
--passwd=password
   Password used to log into the GNATS server.

-D
--debug   Used to debug network connections.

4.8.5 The diff-prs tool

The diff-prs tool is invoked as follows:

    diff-prs prfile1 prfile2

    diff-prs simply reads the PRs contained in prfile1 and prfile2 and returns a list of the fields that are different between the two. No output is produced if the PRs are identical.

4.8.6 The pr-age tool

The pr-age tool reports the time, in days and hours, since the PR arrived. Usage is

    pr-age [ -d databasename | --database=databasename ]
    [ -H host | --host=host ]
    [ -P port | --port=port ]
    [ -v username | --user=username ]
    [ -w password | --passwd=password ]
    [ -h | --help ] [ -V | --version ]

    For an explanation of the arguments listed above, please refer to the usage description for file-pr (Section 4.8.2 [file-pr], page 67).
Appendix A Where GNATS lives

We use a few conventions when referring to the installation structure GNATS uses. These values are adjustable when you build and install GNATS (see Chapter 3 [Installing GNATS], page 31).

A.1 prefix

prefix corresponds to the variable ‘prefix’ for configure, which passes it on to the Makefile it creates. prefix sets the root installation directory for host-independent files as follows:

the directory path of the default database

prefix/com

site-wide configuration files

prefix/etc/gnats

man pages prefix/man

info documents

prefix/info

include files

prefix/include

etc...

The default value for prefix is /usr/local, which can be changed on the command line to configure using

configure --prefix=prefix ...

See Section 3.1 [Configuring and compiling the software], page 31.

A.2 exec-prefix

exec-prefix corresponds to the variable ‘exec-prefix’ for configure, which passes it on to the Makefile it creates. exec-prefix sets the root installation for host-dependent files as follows:

GNATS user tools

exec-prefix/bin

administrative and support utilities

exec-prefix/libexec/gnats

compiled libraries

exec-prefix/lib

etc...
configure supports several more options which allow you to specify in great detail where different files are installed. The locations given in this appendix do not take into account highly customized installations, but fairly ordinary GNATS installations should be covered by the material here. For a complete list of options accepted by configure, run ./configure --help in the gnats subdirectory of the distribution.

Since most installations are not intended to be distributed around a network, the default value for exec-prefix is the value of 'prefix', i.e., /usr/local. However, using exec-prefix saves space when you are installing a package on several different platforms for which many files are identical; rather than duplicate them for each host, these files can be shared in a common repository, and you can use symbolic links on each host to find the host-dependent files.

Use exec-prefix in conjunction with prefix to share host-independent files, like libraries and info documents. For example:

```
for each host:
    configure --prefix=/usr/gnu --exec-prefix=/usr/gnu/H-host
    make all install ...
```

Using this paradigm, all host-dependent binary files are installed into /usr/gnu/H-host/bin, while files which do not depend on the host type for which they were configured are installed into /usr/gnu.

You can then use a different symbolic link for /usr/gnu on each host (/usr is usually specific to a particular machine; it is always specific to a particular architecture).

```
on host-1:
    ln -s /usr/gnu/H-host-1 /usr/gnu
on host-2:
    ln -s /usr/gnu/H-host-2 /usr/gnu
```

To the end user, then, placing /usr/gnu/bin in her or his PATH simply works transparently for each host type.

You can change exec-prefix on the command line to configure using

```
configure --exec-prefix=exec-prefix ...
```

We recommend that you consult Section “Using configure” in Cygnus configure, before attempting this.

A.3 The gnats-adm directory

Each GNATS database located on a server has its own directory, as listed in the databases (see Section 4.2 [The databases file], page 45 and given when the mkdb utility is invoked to initialize the database (see Section 4.7.1 [Initializing a new database], page 64). This directory has several subdirectories, one of which is named gnats-adm. This directory contains all configuration files related to this specific database, including the categories, submitters, responsible, states, classes, dbconfig, addresses, states and gnatsd.user_access, as well as two files generated and maintained by GNATS, index and current.

A.4 Default installation locations

prefix defaults to /usr/local; change using configure (see Section 3.1 [Configuring and compiling the software], page 31).
Appendix A: Where GNATS lives

exec-prefix

defaults to prefix; change using configure (see Section 3.1 [Configuring and compiling the software], page 31).

GNATS installs tools, utilities, and files into the following locations.

exec-prefix/bin
	send-pr See Section 2.2 [Submitting Problem Reports], page 13.
	edit-pr See Section 2.3 [Editing existing Problem Reports], page 17.
	query-pr See Section 2.4 [Querying the database], page 20.

exec-prefix/libexec/gnats
	at-pr See Section 4.8.3 [Timely reminders], page 68.
	check-db See Section 4.7.5 [Checking database health], page 65.

del-te-pr
	Tool for deleting PRs. Deprecated. Use the –delete-pr option of pr-edit instead (see Section 4.8.4 [The edit-pr driver], page 69).

diff-prs See Section 4.8.5 [The diff-prs tool], page 72.
	file-pr See Section 4.8.2 [Interface to pr-edit for filing new PRs], page 67.

gnatsd The GNATS daemon.

gnats-pwconv
	See Section 4.7.6 [Converting old password files], page 66.

mail-query
	See Section 3.5 [Setting up mail aliases], page 36.

mkcat See Section 4.7.2 [Adding a problem category], page 64.

mkdb See Section 4.7.1 [Script for creating new databases], page 64.

pr-age See Section 4.8.6 [The pr-age tool], page 72.

pr-edit See Section 4.8.4 [The main PR processor], page 69.

queue-pr See Section 4.8.1 [Handling incoming traffic], page 66.

rmcat See Section 4.7.3 [Removing categories], page 64.

exec-prefix/lib/libiberty.a
	The GNU libiberty library.

prefix/etc/gnats

databases
	See Section 4.2 [The databases file], page 45.

defaults See Section 4.1 [Overview of GNATS configuration], page 44.

gnatsd.host_access
	See Section C.3 [The gnatsd.host_access file], page 93.
gnatsd.user_access
   See Section C.4 [The gnatsd.user_access file], page 94.

prefix/share/emacs/site-lisp
gnats.el
gnats.elc
   The Emacs versions of the programs send-pr, query-pr, edit-pr, and view-pr. See Chapter 2 [The GNATS user tools], page 13. To change this directory you must change the lispdir variable in Makefile.in; see Section 3.1 [Configuring and compiling the software], page 31.

prefix/info
gnats.info
send-pr.info
   The GNATS manuals, in a form readable by info (the GNU hypertext browser). See Section “Reading GNU Online Documentation” in GNU Online Documentation.

prefix/man/man1
prefix/man/man8
   man pages for all the GNATS tools and utilities.
   See Chapter 2 [The GNATS user tools], page 13.

Per-database directory
gnats-adm
   Administration and configuration data files that define behaviour of the particular database. The files
categories
submitters
responsible
states
classes
dbconfig
addresses
states
gnatsd.user_access
index       (This file is created by GNATS.)
current     (This file is created by GNATS.)
   exist here. See Section 4.4 [Other database-specific config files], page 58, Section 4.6 [Administrative data files], page 63 and Appendix C [Controlling access to databases], page 93.

gnats-queue
   Incoming Problem Reports are queued here until the next iteration of `queue-pr -r' (see Section 4.8.1 [Handling incoming traffic], page 66).
pending  If no default category is set, problem reports without a category are reassigned to the category 'pending' and placed here pending intervention by GNATS administrators. See Chapter 4 [GNATS administration], page 43.

category  Each valid category has a corresponding subdirectory in the database. All Problem Reports associated with that category are kept in that subdirectory.
Appendix B The GNATS network server – gnatsd

This section describes in details how the GNATS network daemon works. This information is mainly assumed to be useful for developers of GNATS client software.

B.1 Description of gnatsd

The gnatsd network daemon is used to service remote GNATS requests such as querying PRs, PR creation, deletion, and editing, and miscellaneous database queries. It uses a simple ASCII-based command protocol (similar to SMTP or POP3) for communicating with remote clients.

It also provides a security model based either on IP-based authentication (generally considered very weak) or username/passwords, where passwords may be in cleartext, UNIX crypt or MD5 hash format. Access through gnatsd is granted according to certain predefined access levels. Access levels are further discussed in Appendix C [Controlling access to databases], page 93. It should be emphasized that security has not been a focus of development until now, but future versions are expected to address this more thoroughly.

All of the GNATS clients are capable of communicating via the GNATS remote protocol to perform their functions.

gnatsd is usually started from the inetd facility and should run as the gnats user (the actual username of this user is configurable during installation, see Section 3.1 [Configuring and compiling the software], page 31 for details.)

B.2 gnatsd options

The daemon supports the following command-line options:

`gnatsd [--database database | -d database]
[--not-inetd | -n] [--max-access-level level | -m level]
[--version | -V] [--help | -h]`

-V, --version
Prints the program version to stdout and exits.

-h, --help
Prints a short help text to stdout and exits.

-d, --database
Specifies the default database which is to be serviced by this invocation of gnatsd. (The selected database may be changed via the CHDB command; the name set with this option is simply the default if no CHDB command is issued.)
If no database is specified, the database named default is assumed. This option overrides the database specified in the GNATSDB environment variable.

-n, --not-inetd
As its name suggests, indicates that gnatsd is not being invoked from inetd. This can be used when testing gnatsd, or if it is being run via ssh or some other mechanism.
This has the effect of using the local hostname where gnatsd is being invoked for authentication purposes, rather than the remote address of the connecting client.
--max-access-level, -m

Specifies the maximum access level that the connecting client can authenticate to. Authentication is as normal but if the user or host authenticates at a higher level, access level is still forced to this level. See Appendix C [Controlling access to databases], page 93 for details on access levels.

B.3 gnatsd command protocol

Commands are issued to gnatsd as one or more words followed by a carriage-return/linefeed pair. For example, the CHDB (change database) command is sent as `CHDB database<CR><LF>` (the CRLF will not be explicitly written for future examples.)

Replies from gnatsd are returned as one or more response lines containing a 3-digit numeric code followed by a human-readable string; the line is terminated with a `<CR><LF>` pair. For example, one possible response to the CHDB command above would be:

```
210 Now accessing GNATS database 'database'.
```

The three-digit code is normally followed by a single ASCII space (character 0x20). However, if additional response lines are to be returned from the server, there will be a single dash ‘-‘ instead of the space character after the three-digit code.

Response code values are divided into ranges. The first digit reflects the general type of response (such as "successful" or "error"), and the subsequent digits identify the specific type of response.

**Codes 200-299**

Positive response indicating that the command was successful. No subsequent data will be transmitted with the response. In particular, code 210 (`CODE_OK`) is used as the positive result code for most simple commands.

Commands that expect additional data from the client (such as SUBM or VFLD) use a two-step mechanism for sending the data. The server will respond to the initial command with either a 211 (`CODE_SEND_PR`) or 212 (`CODE_SEND_TEXT`) response line, or an error code if an error occurred with the initial command. The client is then expected to send the remaining data using the same quoting mechanism as described for server responses in the 300-349 range. The server will then send a final response line to the command.

**Codes 300-399**

Positive response indicating that the query request was successful, and that a PR or other data will follow. Codes 300-349 are used when transmitting PRs, and 350-399 are used for other responses.

Codes in the 300-349 range are followed by a series of CRLF-terminated lines containing the command response, usually a PR. The final line of the result is a single period ‘.’. Result lines that begin with a period have an extra period prepended to them.

Codes in the 350-399 range use a different scheme for sending their responses. The three-digit numeric code will be followed by either a dash ‘-‘ or a single space. If the code is followed by a dash, that indicates that another response line will follow. The final line of the response has a single space after the three-digit code.
In previous versions of the protocol the first line of a CODE_INFORMATION (310) response was to be ignored. This is no longer the case. Instead, any lines marked with code CODE_INFORMATION_FILLER (351) are to be ignored. This allows the server to transmit additional headers or other human-readable text that can be safely ignored by the clients.

**Codes 400-599**

An error occurred, usually because of invalid command parameters or invalid input from the client, missing arguments to the command, or a command was issued out of sequence. The human-readable message associated with the response line describes the general problem encountered with the command. Multiple error messages may be returned from a command; in this case the '--' continuation character is used on all but the last response line.

**Codes 600-799**

An internal error occurred on the server, a timeout occurred reading data from the client, or a network failure occurred. These errors are of the "this should not occur" nature, and retrying the operation may resolve the problem. Fortunately, most GNATS transactions are idempotent; unfortunately, locking the database or a PR are not repeatable actions (we cannot determine if an existing lock is the one we originally requested, or someone else’s).

**B.4 gnatsd commands**

Note that the set of GNATS commands and their responses is somewhat inconsistent and is very much in flux. At present the GNATS clients are rather simple-minded and not very strict about processing responses. For example, if the server were to issue a code 300 (CODE_PR_READY) response to a CHDB command, the client would happily expect to see a PR appear (and would print it out if one was sent).

It is thus suggested that any clients that use the GNATS protocol be equally flexible about the way received responses are handled; in particular, only the first digit of the response code should be assumed to be meaningful, although subsequent digits are needed in some cases (codes 300-399). No attempt should be made to parse the message strings on error response lines; they are only intended to be read by humans, and will be changed on a regular basis.

Almost every command may result in the response 440 (CODE_CMD_ERROR). This indicates that there was a problem with the command arguments, usually because of insufficient or too many arguments being specified.

Access to most gnatsd commands requires a certain access level. For details of this, see Section C.5 [Privileged gnatsd commands], page 95.

**USER [userid password]**

Specifies the userid and password for database access. Either both a username and password must be specified, or they both may be omitted; in the latter case, the current access level is returned.

The possible server responses are:

350 (CODE_INFORMATION)
The current access level is specified.
422 (CODE_NO_ACCESS)
A matching username and password could not be found.
200 (CODE_OK)
A matching username and password was found, and the login was successful.

QUIT
Requests that the connection be closed. Possible responses:
201 (CODE_CLOSING)
Normal exit.
The QUIT command has the dubious distinction of being the only command that cannot fail.

LIST list type
Describes various aspects of the database. The lists are returned as a list of records, one per line. Each line may contain a number of colon-separated fields. Possible values for list type include

Categories
Describes the legal categories for the database.
Submitters
Describes the set of submitters for the database.
Responsible
Lists the names in the responsible administrative file, including their full names and email addresses.
States
Lists the states listed in the state administrative file, including the state type (usually blank for most states; the closed state has a special type).
FieldNames
Lists the entire set of PR fields.
InitialInputFields
Lists the fields that should be present when a PR is initially entered.
InitialRequiredFields
Lists fields that have to be present and nonempty when a PR is initially entered (fields containing only blank characters such as spaces or newlines are considered empty.)
Databases
Lists the set of databases.
The possible responses are:
301 (CODE_TEXT_READY)
Normal response, followed by the records making up the list as described above.
416 (CODE_INVALID_LIST)
The requested list does not exist.

FTYP field [field ...]
Describes the type of data held in the field(s) specified with the command. The currently defined data types are:

Text
A plain text field, containing exactly one line.
MultiText
A text field possibly containing multiple lines of text.

Enum
An enumerated data field; the value is restricted to one entry out of a list of
values associated with the field.

MultiEnum
The field contains one or more enumerated values. Values are separated with
spaces or colons ‘:’.

Integer
The field contains an integer value, possibly signed.

Date
The field contains a date.

TextWithRegex
The value in the field must match one or more regular expressions associated
with the field.

The possible responses are:
350 (CODE_INFORMATION)
The normal response; the supplied text is the data type.

410 (CODE_INVALID_FIELD_NAME)
The specified field does not exist.

If multiple field names were given, multiple response lines will be sent, one for
each field, using the standard continuation protocol; each response except the
last will have a dash ‘-’ immediately after the response code.

FTYPINFO field property
Provides field-type-related information. Currently, only the property
‘separators’ for MultiEnum fields is supported. When ‘separators’ is
specified, the possible return codes are:

350 (CODE_INFORMATION) A proper MultiEnum field was specified and the re-
turned text is the string of separators specified for the field in the dbconfig file
(see Section 4.3.3 [Field datatypes], page 49) quoted in ‘’s.

435 (CODE_INVALID_FTYPE_PROPERTY) The ‘separators’ property is not de-
defined for this field, i.e. the specified field is not of type MultiEnum.

Currently, specifying a different property than ‘separators’ results in return
code 435 as above.

FDSC field [field ...]
Returns a human-readable description of the listed field(s). The possible re-
sponses are:

350 (CODE_INFORMATION) The normal response; the supplied text is the field
description.

410 (CODE_INVALID_FIELD_NAME) The specified field does not exist.

Like the FVLD command, the standard continuation protocol will be used if
multiple fields were specified with the command.
FIELDFLAGS field [field ... ]
Returns a set of flags describing the specified field(s). The possible responses
are either
410 (CODE_INVALID_FIELD_NAME)
meaning that the specified field is invalid or nonexistent, or
350 (CODE_INFORMATION)
which contains the set of flags for the field. The flags may be blank, which
indicate that no special flags have been set for this field.
Like the FDSC and FTYP commands, multiple field names may be listed with the
command, and a response line will be returned for each one in the order that
the fields appear on the command line.
The flags include:
textsearch
The field will be searched when a text field search is requested.
allowAnyValue
For fields that contain enumerated values, any legal value may be used in the
field, not just ones that appear in the enumerated list.
requireChangeReason
If the field is edited, a reason for the change must be supplied in the new PR
text describing the reason for the change. The reason must be supplied as a
multitext PR field in the new PR whose name is field-Changed-Why (where
field is the name of the field being edited).
readonly
The field is read-only, and cannot be edited.

FVLD field

Returns one or more regular expressions or strings that describe the valid types
of data that can be placed in field. Exactly what is returned is dependent
on the type of data that can be stored in the field. For most fields a regular
expression is returned; for enumerated fields, the returned values are the list of
legal strings that can be held in the field.
The possible responses are:
301 (CODE_TEXT_READY)
The normal response, which is followed by the list of regexps or strings.
410 (CODE_INVALID_FIELD_NAME) The specified field does not exist.

VFLD field

VFLD can be used to validate a given value for a field in the database. The
client issues the VFLD command with the name of the field to validate as an
argument. The server will either respond with 212 (CODE_SEND_TEXT), or 410
(CODE_INVALID_FIELD_NAME) if the specified field does not exist.
Once the 212 response is received from the server, the client should then send
the line(s) of text to be validated, using the normal quoting mechanism de-
scribed for PRs. The final line of text is followed by a line containing a single
period, again as when sending PR text.
The server will then either respond with 210 (CODE_OK), indicating that the text is acceptable, or one or more error codes describing the problems with the field contents.

**INPUTDEFAULT field [field ...]**
Returns the suggested default value for a field when a PR is initially created. The possible responses are either 410 (CODE_INVALID_FIELD_NAME), meaning that the specified field is invalid or nonexistent, or 350 (CODE_INFORMATION) which contains the default value for the field.

Like the FDSC and FTYP commands, multiple field names may be listed with the command, and a response line will be returned for each one in the order that the fields appear on the command line.

**RSET**
Used to reset the internal server state. The current query expression is cleared, and the index of PRs may be reread if it has been updated since the start of the session. The possible responses are:
- **200 (CODE_OK)**: The state has been reset.
- **440 (CODE_CMD_ERROR)**: One or more arguments were supplied to the command.
- **6xx (internal error)**: There were problems resetting the state (usually because the index could not be reread). The session will be immediately terminated.

**LKDB**
Locks the main GNATS database. No subsequent database locks will succeed until the lock is removed. Sessions that attempt to write to the database will fail. The possible responses are:
- **200 (CODE_OK)**: The lock has been established.
- **440 (CODE_CMD_ERROR)**: One or more arguments were supplied to the command.
- **431 (CODE_GNATS_LOCKED)**: The database is already locked, and the lock could not be obtained after 10 seconds.
- **6xx (internal error)**: An internal error occurred, usually because of permission or other filesystem-related problems. The lock may or may not have been established.

**UNDB**
Unlocks the database. Any session may steal a database lock; no checking of any sort is done. The possible responses are:
- **200 (CODE_OK)**: The lock has been removed.
- **432 (CODE_GNATS_NOT_LOCKED)**: The database was not locked.
- **440 (CODE_CMD_ERROR)**: One or more arguments were supplied to the command.
- **6xx (internal error)**: The database lock could not be removed, usually because of permissions or other filesystem-related issues.
LOCK PR user [pid]
Locks the specified PR, marking the lock with the user name and the optional pid. (No checking is done that the user or pid arguments are valid or meaningful; they are simply treated as strings.)
The EDIT command requires that the PR be locked before it may be successfully executed. However, it does not require that the lock is owned by the editing session, so the usefulness of the lock is simply as an advisory measure.
The APPN and REPL commands lock the PR as part of the editing process, and they do not require that the PR be locked before they are invoked.
The possible responses are:
440 (CODE_CMD_ERROR)
Insufficient or too many arguments were specified to the command.
300 (CODE_PR_READY)
The lock was successfully obtained; the text of the PR (using the standard quoting mechanism for PRs) follows.
400 (CODE_NONEXISTENT_PR)
The PR specified does not exist.
430 (CODE_LOCKED_PR)
The PR is already locked by another session.
6xx (internal error)
The PR lock could not be created, usually because of permissions or other filesystem-related issues.

UNLK PR
Unlocks PR. Any user may unlock a PR, as no checking is done to determine if the requesting session owns the lock.
The possible responses are:
440 (CODE_CMD_ERROR)
Insufficient or too many arguments were specified to the command.
200 (CODE_OK)
The PR was successfully unlocked.
433 (CODE_PR_NOT_LOCKED)
The PR was not locked.
6xx (internal error)
The PR could not be unlocked, usually because of permission or other filesystem-related problems.

DELETE PR
Deletes the specified PR. The user making the request must have admin privileges (see Appendix C [Controlling access to databases], page 93). If successful, the PR is removed from the filesystem and the index file; a gap will be left in the numbering sequence for PRs. No checks are made that the PR is closed.
The possible responses are:
200 (CODE_OK)
The PR was successfully deleted.
422 (CODE_NO_ACCESS)
The user requesting the delete does not have admin privileges.
430 (CODE_LOCKED_PR)
The PR is locked by another session.

431 (CODE_GNATS_LOCKED)
The database has been locked, and no PRs may be updated until the lock is cleared.

6xx (internal error)
The PR could not be successfully deleted, usually because of permission or other filesystem-related problems.

CHEK [initial]
Used to check the text of an entire PR for errors. Unlike the VFLD command, it accepts an entire PR at once instead of the contents of an individual field.
The initial argument indicates that the PR text to be checked is for a PR that will be newly created, rather than an edit or replacement of an existing PR.
After the CHEK command is issued, the server will respond with either a 440 (CODE_CMD_ERROR) response indicating that the command arguments were incorrect, or a 211 (CODE_SEND_PR) response code will be sent.
Once the 211 response is received from the server, the client should send the PR using the normal PR quoting mechanism; the final line of the PR is then followed by a line containing a single period, as usual.
The server will then respond with either a 200 (CODE_OK) response, indicating there were no problems with the supplied text, or one or more error codes listing the problems with the PR.

EDIT PR
Verifies the replacement text for PR. If the command is successful, the contents of PR are completely replaced with the supplied text. The PR must previously have been locked with the LOCK command.
The possible responses are:

431 (CODE_GNATS_LOCKED)
The database has been locked, and no PRs may be updated until the lock is cleared.

433 (CODE_PR_NOT_LOCKED)
The PR was not previously locked with the LOCK command.

400 (CODE_NONEXISTENT_PR)
The specified PR does not currently exist. The SUBM command should be used to create new PRs.

211 (CODE_SEND_PR)
The client should now transmit the replacement PR text using the normal PR quoting mechanism. After the PR has been sent, the server will respond with either 200 (CODE_OK) indicating that the edit was successful, or one or more error codes listing problems either with the replacement PR text or errors encountered while updating the PR file or index.

EDITADDR address
Sets the e-mail address of the person communicating with gnatsd. The command requires at least the edit access level.
The possible responses are:

200 (CODE_OK)
The address was successfully set.

440 (CODE_CMD_ERROR)
Invalid number of arguments were supplied.

**APPN PR field**
**REPL PR field**
Appends to or replaces the contents of `field` in `PR` with the supplied text. The command returns a 201 (CODE_SEND_TEXT) response; the client should then transmit the new field contents using the standard PR quoting mechanism. After the server has read the new contents, it then attempts to make the requested change to the PR.

The possible responses are:

200 (CODE_OK)
The PR field was successfully changed.

400 (CODE_NONEXISTENT_PR)
The PR specified does not exist.

410 (CODE_INVALID_FIELD_NAME)
The specified field does not exist.

402 (CODE_UNREADABLE_PR)
The PR could not be read.

431 (CODE_GNATS_LOCKED)
The database has been locked, and no PRs may be updated until the lock is cleared.

430 (CODE_LOCKED_PR)
The PR is locked, and may not be altered until the lock is cleared.

413 (CODE_INVALID_FIELD_CONTENTS)
The supplied (or resulting) field contents are not valid for the field.

6xx (internal error)
An internal error occurred, usually because of permission or other filesystem-related problems. The PR may or may not have been altered.

**SUBM**
Submits a new PR into the database. The supplied text is verified for correctness, and if no problems are found a new PR is created.

The possible responses are:

431 (CODE_GNATS_LOCKED)
The database has been locked, and no PRs may be submitted until the lock is cleared.

211 (CODE_SEND_PR)
The client should now transmit the new PR text using the normal quoting mechanism. After the PR has been sent, the server will respond with either 351 (CODE_INFORMATION_FILLER) and 350 (CODE_INFORMATION) responses indicating that the new PR has been created and supplying the number assigned to it, or one or more error codes listing problems with the new PR text.
Appendix B: The GNATS network server – gnatsd

CHDB database
Switches the current database to the name specified in the command.
The possible responses are:
422 (CODE_NO_ACCESS)
The user does not have permission to access the requested database.
417 (CODE_INVALID_DATABASE)
The database specified does not exist, or one or more configuration errors in
the database were encountered.
220 (CODE_OK)
The current database is now database. Any operations performed will now be
applied to database.

DBLS
Lists the known set of databases.
The possible responses are:
6xx (internal error)
An internal error was encountered while trying to obtain the list of available
databases, usually due to lack of permissions or other filesystem-related prob-
lems, or the list of databases is empty.
301 (CODE_TEXT_READY)
The list of databases follows, one per line, using the standard quoting mecha-
nism. Only the database names are sent.
The gnatsd access level ‘listdb’ denies access until the user has authenticated
with the USER command. The only other command available at this access
level is DBLS. This access level provides a way for a site to secure its GNATS
databases while still providing a way for client tools to obtain a list of the
databases for use on login screens etc. See Appendix C [Controlling access to
databases], page 93.

DBDESC database
Returns a human-readable description of the specified database.
Responses include:
6xx (internal error)
An internal error was encountered while trying to read the list of available data-
bases, usually due to lack of permissions or other filesystem-related problems,
or the list of databases is empty.
350 (CODE_INFORMATION)
The normal response; the supplied text is the database description.
417 (CODE_INVALID_DATABASE)
The specified database name does not have an entry.

EXPR query expression
Specifies a query expression used to limit which PRs are returned from the
QUER command. The expression uses the normal query expression syntax, (see
Section 2.4.3 [Query expressions], page 25).
Multiple EXPR commands may be issued; the expressions are boolean ANDed
together.
Expressions are cleared by the `RSET` command.
Possible responses include:

415 (CODE_INVALID_EXPR)
The specified expression is invalid, and could not be parsed.
200 (CODE_OK)
The expression has been accepted and will be used to limit the results returned from `QUER`.

**QFMT query format**

Use the specified *query format* to format the output of the `QUER` command. The query format may be either the name of a query format known to the server (see Section 4.3.5 [Named query definitions], page 53), or an actual query format (see Section 2.4.2 [Formatting query-pr output], page 24). The possible responses are:

200 (CODE_OK)
The normal response, which indicates that the query format is acceptable.
440 (CODE_CMD_ERROR)
No query format was supplied.
418 (CODE_INVALID_QUERY_FORMAT)
The specified query format does not exist, or could not be parsed.

**QUER [PR] [PR] […]**

Searches the contents of the database for PRs that match the (optional) specified expressions with the `EXPR` command. If no expressions were specified with `EXPR`, the entire set of PRs is returned.
If one or more PRs are specified on the command line, only those PRs will be searched and/or output.
The format of the output from the command is determined by the query format selected with the `QFMT` command.
The possible responses are:

418 (CODE_INVALID_QUERY_FORMAT)
A valid format was not specified with the `QFMT` command prior to invoking `QUER`.
300 (CODE_PR_READY)
One or more PRs will be output using the requested query format. The PR text is quoted using the normal quoting mechanisms for PRs.
220 (CODE_NO_PRS_MATCHED)
No PRs met the specified criteria.

**ADMV field key [subfield]**

Returns an entry from an administrative data file associated with *field*. *key* is used to look up the entry in the data file. If *subfield* is specified, only the value of that subfield is returned; otherwise, all of the fields in the adm data file are returned, separated by colons `:`.
The responses are:

410 (CODE_INVALID_FIELD_NAME) The specified field does not exist.
221 (CODE_NO_ADM_ENTRY) An adm entry matching the key was not found, or the field does not have an adm file associated with it.

350 (CODE_INFORMATION) The normal response; the supplied text is the requested field(s).

B.5 gnatsd environment variables

gnatsd supports the GNATSDB environment variable, See Section 2.1 [Environment], page 13, in almost the same way as the GNATS tools do. This variable is used to determine which database to use. For a local database, it contains the name of the database to access. gnatsd cannot service remote databases (though it might be interesting if it could) so the database is always assumed to be local.

If GNATSDB is not set and the --database option is not supplied, it is assumed that the database is local and that its name is 'default'.
Appendix C Controlling access to databases

C.1 Overview

GNATS supports granting various levels of access to the GNATS databases served by the network daemon, gnatsd.

GNATS access can be controlled at these levels:

- **deny**: gnatsd closes the connection
- **none**: no further access until userid and password given
- **listdb**: only listing of available databases is allowed
- **view**: query and view PRs with Confidential=no only
- **viewconf**: query and view PRs with Confidential=yes
- **edit**: full edit access
- **admin**: full admin access

These access levels are used in the following settings:

- overall gnatsd access level
- overall access level set by host name or IP address
- overall access level set by userid and password
- per-database access level set by userid and password

C.2 Overall gnatsd access level

The overall gnatsd access level is set by starting gnatsd with the option

```
-m level or --maximum-access-level=level,
```

where *level* is one of the six access levels listed above. This restricts any access to the GNATS daemon to levels up to and including *level*, regardless of the settings in the access control files discussed below. If this option is left out, any access levels set in the access control files will be allowed.

The discussion below assumes that the pre-build configure of GNATS was done without altering the default values for the `--enable-gnatsd-user-access-file` and `--enable-gnatsd-host-access-file` options. If non-default values were given, substitute as appropriate below.

C.3 Overall access levels per host

The host access file (by default `/usr/local/etc/gnats/gnatsd.host_access`) controls overall access levels on a per-host basis, meaning that settings in this file apply across all databases on the server. Entries in this file are in the following format:

```
host:access-level:whatever
```

*host* is the hostname or IP address of the host contacting gnatsd. Wildcard characters are supported: ‘*’ matches anything; ‘?’ matches any single character. By using wildcards, you
can specify access levels for entire network subnets and domains. Note that when GNATS authenticates hosts, it reads the entries in this file in sequence until a match is found. This means that wildcard entries must be placed near the end of the file, otherwise, they will override non-wildcard entries appearing after the wildcard ones.

The second field is the access level of host. The default is `deny`. If the user’s hostname isn’t in the file or its access level is set to `deny`, the connection is closed immediately.

GNATS currently doesn’t make use of the third field. Remember to still include the second ‘:’ on the line if you choose to leave the third field empty.

Whenever a CHDB command is processed (or defaulted), the user’s access level is set to the level for their host, as determined by the values in the `gnatsd.host_access` file. However, even if a host is given the none access level, an individual can still give the USER command to possibly gain a higher (but never lower) access than is set for their host. The gnatsd USER command takes two arguments: `USER <userid> <passwd>`.

### C.4 Access levels per user

Access levels per user can be set both across all databases on the server or on a per-database basis. The `gnatsd.user_access` file in a database’s `gnats-adm` directory specifies the user access rules for that database. If it doesn’t exist, or doesn’t contain the user name given to gnatsd, then the overall user access file (by default `/usr/local/etc/gnats/gnatsd.user_access`) specifying the per-user access levels across all the databases on the server is checked.

The user access files can only increase the access level defined in the host access files for the given host, they can never lower it.

If the access level is `none` after processing the userid and password, the connection is closed.

The `gnatsd.user_access` files can contain plain text passwords, in such a case they should be owned by the GNATS user with file permission 600.

Wildcard characters are supported for the userid and password with plain text passwords. A null string or ‘*’ matches anything; ‘?’ matches any one character. Note that when GNATS authenticates users, it reads the entries in this file in sequence until a match is found. This means that wildcard entries must be placed near the end of the file, otherwise, they will override non-wildcard entries appearing after the wildcard ones.

Entries in the database-specific `gnatsd.user_access` user access file in the `gnats-adm` directory of the database have the following general format:

```
userid:password:access-level
```

The overall `gnatsd.user_access` user access file adds a fourth `databases` field:

```
userid:password:access-level:databases
```

Password should either be in plain text, DES `crypt()` \(^1\) or MD5 hash format \(^2\).

---

\(^1\) DES crypt is the standard password encryption format used by most UNIX systems

\(^2\) MD5 is only supported on platforms that have a `crypt()` function that supports MD5. Among others, this currently includes GNU Linux and OpenBSD.
If the password is in plain text format, it must be prefixed by ‘$0$’ and if it is in MD5 format, it needs to be prefixed by the string ‘$1$’.\footnote{Some systems support even more encryption methods. In FreeBSD, for instance, a prefix of ‘$2$’ implies Blowfish encoding. GNATS will happily accept any encryption that the OS supports.} Passwords encrypted by \texttt{crypt()} should have no prefix. If no password is given then users can login with an empty password string.

A \texttt{gnats-passwd} tool to manage \texttt{gnatsd.user_access} files is planned. In the meantime, \texttt{crypt()} passwords can be generated by using standard UNIX passwords tools, while MD5 passwords can be generated with the following little Perl snippet:

\begin{verbatim}
perl -e 'use Crypt::PasswdMD5 ; print Crypt::PasswdMD5::unix_md5_crypt "password" , time() % 100000000'
\end{verbatim}

If your Perl installation doesn’t have the Crypt module installed, you need to install it. On most systems, the following command achieves this:

\begin{verbatim}
perl -MCPAN -e 'install Crypt::PasswdMD5'
\end{verbatim}

A tool for conversion of pre-version 4 \texttt{gnatsd.user_access} files is distributed with GNATS 4. See Section 4.7.6 [Converting old password files], page 66.

The access-level field should contain one of the values listed at the beginning of this appendix. This overrides (increases but never lowers) the access level given as the default for the user’s host in the global \texttt{gnatsd.host_access} file.

The following shows an example \texttt{gnatsd.user_access} file with plain text passwords:

\begin{verbatim}
rickm:$0$ruckm:edit
pablo:$0$pueblo:view
*:none
\end{verbatim}

And this is the same file with MD5-encrypted passwords:

\begin{verbatim}
rickm:$1$92388613$D7ZIYikzTUqd./dODTFrI.:edit
pablo:$1$92388652$QRfAhIBG5elT.FQjQKhj80:view
*:none
\end{verbatim}

In these examples, anybody other than rickm and pablo get denied access, assuming that the host access level is also \texttt{none}. You could set the catch-all rule at the end to be \texttt{*:view} to allow view access to anyone who does not supply a password. Note the important detail that such a rule would allow view access only to persons who do not supply a password at all, i.e. if rickm or pablo tries to log in but mistypes his password, this rule would not apply and they would be denied access entirely. This is by design, since people might be surprised if they suddenly found themselves logged in, but with a lower access level than they usually have.

The \texttt{databases} field contains a comma-separated list of database names, as defined in the \texttt{databases} file (see Section 4.2 [The \texttt{databases} file], page 45. Wildcard characters are supported. The databases listed in this field are the ones to which the other settings on the same line will be applied.

\section*{C.5 Privileged \texttt{gnatsd} commands}

Every \texttt{gnatsd} command has a minimum access level attached to it. If your access level is too low for a command, you get this response:
LOCK 12

422 You are not authorized to perform this operation (LOCK).

The commands CHDB, USER and QUIT are unrestricted.
The DBLS command requires at least listdb access.
A user must have at least edit access for these commands:

LKDB lock the main GNATS database.
UNDB unlock the main GNATS database.

LOCK PR user pid
lock PR for user and optional pid and return PR text.

UNLK PR unlock PR.
EDIT PR check in edited PR.
APPN PR field, REPL PR field
Appends to or replaces the contents of field in PR.

The DELETE PR command is special in that it requires admin access.
All other commands require view access.

edit-pr and query-pr accept the command line arguments -v|--user and
Appendix D  Querying using regular expressions

See also Section 2.4.3 [Query expressions], page 25.

Unfortunately, we do not have room in this manual for a complete exposition on regular expressions. The following is a basic summary of some regular expressions you might wish to use.

*NOTE:* When you use query expressions containing regular expressions as part of an ordinary query-pr shell command line, you need to quote them with '', otherwise the shell will try to interpret the special characters used, yielding highly unpredictable results.

See Section “Regular Expression Syntax” in *Regex*, for details on regular expression syntax. Also see Section “Syntax of Regular Expressions” in *GNU Emacs Manual*, but beware that the syntax for regular expressions in Emacs is slightly different.

All search criteria options to *query-pr* rely on regular expression syntax to construct their search patterns. For example,

```
query-pr --expr 'State="open"' --format full
```

matches all PRs whose *State* values match with the regular expression ‘open’.

We can substitute the expression ‘o’ for ‘open’, according to GNU regular expression syntax. This matches all values of *State* which begin with the letter ‘o’.

We see that

```
query-pr --expr 'State="o"' --format full
```

is equivalent to

```
query-pr --expr 'State="open"' --format full
```

in this case, since the only value for *State* which matches the expression ‘o’ in a standard installation is ‘open’. ‘State="o"’ also matches ‘o’, ‘oswald’, and even ‘oooooo’, but none of those values are valid states for a Problem Report in default G-Nats installations.

We can also use the expression operator ‘|’ to signify a logical OR, such that

```
query-pr --expr 'State="o|a"' --format full
```

matches all ‘open’ or ‘analyzed’ Problem Reports.

Regular expression syntax considers a regexp token surrounded with parentheses, as in ‘(regexp)’, to be a group. This means that ‘(ab)*’ matches any number (including zero) of contiguous instances of ‘ab’. Matches include ‘’, ‘ab’, and ‘ababab’.

Regular expression syntax considers a regexp token surrounded with square brackets, as in ‘[regexp]’, to be a list. This means that ‘Char[(ley)(lene)(broiled)]’ matches any of the words ‘Charley’, ‘Charlene’, or ‘Charbroiled’ (case is significant; ‘charbroiled’ is not matched).

Using groups and lists, we see that

```
query-pr --expr 'Category="gcc|gdb|gas"' --format full
```

is equivalent to

```
query-pr --expr 'Category="g(cc|db|as)"' --format full
```

and is also very similar to

```
query-pr --expr 'Category="g[cda]"' --format full
```

with the exception that this last search matches any values which begin with ‘gc’, ‘gd’, or ‘ga’.
The ‘.’ character is known as a wildcard. ‘.’ matches on any single character. ‘*’ matches the previous character (except newlines), list, or group any number of times, including zero. Therefore, we can understand ‘.*’ to mean “match zero or more instances of any character.”

query-pr --expr 'State=”.*a”' --format full

matches all values for State which contain an ‘a’. (These include ‘analyzed’ and ‘feedback’.)

Another way to understand what wildcards do is to follow them on their search for matching text. By our syntax, ‘.*’ matches any character any number of times, including zero. Therefore, ‘.*a’ searches for any group of characters which end with ‘a’, ignoring the rest of the field. ‘.*a’ matches ‘analyzed’ (stopping at the first ‘a’) as well as ‘feedback’.

*Note:* When using ‘fieldtype:Text’ or ‘fieldtype:Multitext’ (see Section 2.4.3 [Query expressions], page 25), you do not have to specify the token ‘.*’ at the beginning of your expression to match the entire field. For the technically minded, this is because these queries use ‘re_search’ rather than ‘re_match’. ‘re_match’ anchors the search at the beginning of the field, while ‘re_search’ does not anchor the search.

For example, to search in the >Description: field for the text

The defrobulator component returns a nil value.

we can use

query-pr --expr 'fieldtype:Multitext="defrobulator.*nil”' --format full

To also match newlines, we have to include the expression ‘(.|^M)’ instead of just a dot (‘.’). ‘(.|^M)’ matches “any single character except a newline (‘.’) or (‘|’) any newline (‘^M”).” This means that to search for the text

The defrobulator component enters the bifrabulator routine

and returns a nil value.

we must use

query-pr --expr 'fieldtype:Multitext="defrobulator(.|^M)*nil”'

--format full

To generate the newline character ‘^M’, type the following depending on your shell:

csh

‘control-V control-M’

tcsh

‘control-V control-J’

sh (or bash)

Use the RETURN key, as in

( . | )

Again, see Section “Regular Expression Syntax” in Regex, for a much more complete discussion on regular expression syntax.
Appendix E: dbconfig recipes

The dbconfig file (Section 4.3 [The dbconfig file], page 46) is the heart of any GNATS installation. It contains some very powerful machinery, something which this appendix tries to illustrate.

We provide a range of examples that are both intended to be useful in their own right and to serve as starting points or building blocks for your own modifications.

Provide Gnatsweb URL in initial response

Sites that have Gnatsweb installed may wish to modify the response e-mail which is sent to the submitter of a PR so that it includes a URL where the status of the PR can be monitored. In order to allow this, you should first create an entry in gnatsd.user_access which allows viewing of PRs in your database (See Section C.4 [The gnatsd.user_access file], page 94.)

Next, locate the entry mail-format "initial-response-to-submitter" in the dbconfig file of your database and add the following before the line reading “The individual assigned...” in the body section:

```
\nYou can follow the status of this report on\nhttp://hostname/cgi-bin/scriptname?\n  cmd=view&database=dbname&user=username&\n  password=passwd&pr=%s\n\nSubstitute hostname, cgi-bin and scriptname as appropriate for the setup of your web server. The part before the ‘?’ would typically look something like http://www.example.com/cgi-bin/gnatsweb.pl. Substitute the name of your database for dbname, and the username and password of the user with view rights for username and passwd.
```

Next, add a Number to the fields list statement inside the body so it reads as follows:

```
fields { "Category" "Number" "Number" "Responsible"
  "Category" "Responsible" "Synopsis"
  "Arrival-Date" }
```

State full name of responsible in initial response

The initial e-mail response to the submitter of a PR identifies the responsible person assigned to the PR as follows: “The individual assigned to look at your report is: GNATS username”. Some sites may wish to modify this so that the full name of the responsible person is used instead of the GNATS user name.

The full name is contained in the fullname subfield of the user’s entry in the responsible file and can be accessed as Responsible[fullname] (see [Enumerated field administrative files], page 51.)

The change is achieved by editing the dbconfig item mail-format "initial-response-to-submitter" and changing the fields part of the Body from

```
fields { "Category" "Number" "Responsible"
  "Category" "Responsible" "Synopsis"
  "Arrival-Date" }
```

to
Append-only Audit-Trail

The Audit-Trail of a PR is by default editable. For some applications, one might want to make the Audit-Trail append-only, so it provides a full and unchangeable case history. Also by default, only certain changes, such as change of state and change of responsible gets recorded in the Audit-Trail. In some cases, it might also be convenient to have a way of inserting comments directly into the Audit-Trail.

The following procedure creates such an append-only Audit-Trail and adds a PR field which makes it possible to register comments in the Audit-Trail.

First, add the keyword `read-only` to the Audit-Trail field definition in `dbconfig`.

Then, add the following field definition to `dbconfig`:

```plaintext
field "Add-To-Audit-Trail" {
    description "Add a log entry to the Audit Trail"
    multitext { default "\n" }
    on-change {
        add-audit-trail
        audit-trail-format {
            format "**** Comment added by %s on %s ****\n %s\n\n"
            fields { "$EditUserEmailAddr" "$CurrentDate" "$NewValue" }
        }
    }
    on-change {
        set-field "Add-To-Audit-Trail" { "\n" }
    }
}
```

Supporting GNATS “release-based” fields

When installing GNATS version 3.x, it was possible to choose whether to enable three optional fields: `Quarter`, `Keywords` and `Date-Required`. Default installations had these fields switched off, and installations which had them were called “release-based”.

The default `dbconfig` shipped with GNATS version 4 or newer does not have these fields, so if you are upgrading from an old release-based system, you need to add the following field definitions to your `dbconfig` file:

```plaintext
field "Quarter" {
    description "What quarter does the PR fall into?"
    text
    query-default inexact-regexp
textsearch
}
```

```plaintext
field "Keywords" {
```
description "Keywords used to index this PR"
    text
    query-default inexact-regexp
    textsearch
}

field "Date-Required" {
    description "Date that the PR must be fixed by"
    date
}

A side note: Pre-release versions of GNATS 4 also had a field named Cases. For those who may need it, here is the field definition of Cases:

field "Cases" {
    text
    query-default inexact-regexp
    textsearch
}
Appendix F  GNATS support

The GNATS home page is located at http://www.gnu.org/software/gnats. It contains all the important references to the available information about GNATS and the related software.

There is also a special page dedicated to the GNATS development at http://savannah.gnu.org/projects/gnats.

There are several GNATS mailing lists. The most important ones are:

info-gnats@gnu.org
Announcements and other important information about GNATS and the related software. This is a very low volume moderated list.

bug-gnats@gnu.org
The bug reporting mailing list on the GNATS itself. Please note that the preferred way to report GNATS bugs is to submit them via the web interface at http://bugs.gnu.org/cgi-bin/gnatsweb.pl?database=gnats. New bug reports submitted via the web interface are copied to the mailing list automatically.

help-gnats@gnu.org
General discussion about GNATS. Anything related to GNATS (user questions, development, suggestions, etc.) can be discussed there.

The complete list of GNATS related mailing lists is available from the web page at http://savannah.gnu.org/project/gnats.

When you report problems concerning GNATS itself, please do not forget to provide especially the following information:

• The GNATS version you are using.
• The exact way how to reproduce the bug.
• Your configuration.
• If you encounter a compilation or build problem, it is especially important to mention the operating system, compiler and possibly other build utilities you use.

Providing this information in the initial report avoids further unnecessary communication, saves our limited development resources and helps to track down and fix the problem soon.
Appendix G  Copying This Manual

G.1 GNU Free Documentation License

Version 1.2, November 2002
51 Franklin St, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE
The purpose of this License is to make a manual, textbook, or other functional and useful document free in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS
This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.
The “Invariant Sections” are certain Secondary Sections whose titles are designated, as
being those of Invariant Sections, in the notice that says that the Document is released
under this License. If a section does not fit the above definition of Secondary then it is
not allowed to be designated as Invariant. The Document may contain zero Invariant
Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover
Texts or Back-Cover Texts, in the notice that says that the Document is released under
this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may
be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented
in a format whose specification is available to the general public, that is suitable for
revising the document straightforwardly with generic text editors or (for images com-
posed of pixels) generic paint programs or (for drawings) some widely available drawing
editor, and that is suitable for input to text formatters or for automatic translation to
a variety of formats suitable for input to text formatters. A copy made in an otherwise
Transparent file format whose markup, or absence of markup, has been arranged to
thwart or discourage subsequent modification by readers is not Transparent. An image
format is not Transparent if used for any substantial amount of text. A copy that is
not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without
markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly
available DTD, and standard-conforming simple HTML, PostScript or PDF designed
for human modification. Examples of transparent image formats include PNG, XCF and
JPG. Opaque formats include proprietary formats that can be read and edited
only by proprietary word processors, SGML or XML for which the DTD and/or
processing tools are not generally available, and the machine-generated HTML,
PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following
pages as are needed to hold, legibly, the material this License requires to appear in the
title page. For works in formats which do not have any title page as such, “Title Page”
means the text near the most prominent appearance of the work’s title, preceding the
beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either
is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in
another language. (Here XYZ stands for a specific section name mentioned below, such
as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve
the Title” of such a section when you modify the Document means that it remains a
section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that
this License applies to the Document. These Warranty Disclaimers are considered to
be included by reference in this License, but only as regards disclaiming warranties:
any other implication that these Warranty Disclaimers may have is void and has no
effect on the meaning of this License.

2. VERBATIM COPYING
You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,
be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their
titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements.”

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.
7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.
G.1.1 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the
document and put the following copyright and license notices just after the title page:

Copyright (C) year your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts. A copy of the license is included in the section entitled "GNU
Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the
"with...Texts." line with this:

with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.

If you have Invariant Sections without Cover Texts, or some other combination of the
three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing
these examples in parallel under your choice of free software license, such as the GNU
General Public License, to permit their use in free software.
## Index

- `--enable-gnats-dblist-file` ........................................ 33  
- `--enable-gnats-default-db` ......................................... 33  
- `--enable-gnats-service` ............................................ 33  
- `--enable-gnats-user` ................................................ 33  
- `--enable-gnatsd-host-access-file` ................................. 33  
- `--enable-gnatsd-user-access-file` ................................ 33  
- `--with-kerberos` .................................................... 33  
- `--with-krb4` .......................................................... 33  

>  

> `Submitter-Id` ......................................................... 16  

### A  

adding a problem category ................................. 43, 64  
adding and removing maintainers .......................... 43  
adding another database ......................................... 43  
`addresses` file .................................................. 45, 61  
admin files .......................................................... 63  
administering GNATS ............................................. 43  
administrative utilities .......................................... 64  
age of PR .......................................................... 72  
alias for incoming Problem Reports ........................ 36  
`aliases` .............................................................. 36  
`analyzed-state` .................................................... 5  
appende-mail-response .......................................... 55  
`arrival-date` ....................................................... 48  
`Arrival-Date field` ................................................. 11  
at ................................................................. 35  
at--pr ............................................................... 68  
audit-mail .......................................................... 55  
audit-trail .......................................................... 48  
Audit-Trail field .................................................. 11  
Audit-trail format ................................................ 53  
autoloader commands ............................................. 34  
automatic notification ........................................... 5, 68  

### B  

BACK UP YOUR DATA .................................................. 44  
bad Problem Reports ................................................. 16  
bug alias .............................................................. 36  
bug reporting ......................................................... 103  
building a new index .............................................. 43  
building GNATS ..................................................... 31  
building in a different directory ......................... 33  
`builtin-name` ....................................................... 48  
`bury-buffer` ........................................................ 28  
`business-day-hours` ............................................... 47  
`business-week-days` .............................................. 47  

### C  

categories file .................................................... 44, 58, 64  
category ............................................................. 48  
Category field ...................................................... 10  
category-dir-perms ................................................ 47  
`change-request` class ........................................... 10  
check-db ............................................................ 65  
Class field .......................................................... 10  
classes file ........................................................ 45, 62  
closed state ........................................................ 6  
closed-date ........................................................ 48  
command line options ........................................... 13  
comment section in the PR template ...................... 15  
compiling the software ........................................... 31  
confidential .......................................................... 48  
Confidential field ................................................ 9  
confidentiality in PRs ............................................. 9  
configure ............................................................ 31, 32  
configuring and compiling the software .................. 31  
configuring GNATS ................................................ 31  
configuring GNATS for remote users ....................... 38  
configuring GNATS on a local network ..................... 38  
configuring GNATS on a network .............................. 37  
create-category-dirs .............................................. 47  
creating an account for the GNATS user .................. 32  
critical severity problems ....................................... 9  
cron ................................................................. 35  
crontab ............................................................. 35  
current file ........................................................ 63  

### D  
daemon .............................................................. 36  
database paradigm ................................................. 3  
database rationale ................................................ 3  
database similarities ............................................. 6  
databases ........................................................... 44  
databases file ...................................................... 45  
datatype ............................................................ 49  
date ................................................................. 49  
Date-Required ..................................................... 11  
Date-Required field ............................................... 11  
dbconfig ............................................................ 99  
dbconfig file ....................................................... 44, 46  
dbconfig mode ...................................................... 30  
dbconfig recipes .................................................. 99  
dbconfig ............................................................ 99  
dbdebug-mode ....................................................... 47  
default installation locations ............................... 73, 74  
defaults ............................................................. 44  
deleted-pr-mail .................................................... 55  
description ........................................................ 48  
Description field ................................................ 10  
diff-prs ............................................................ 72  
Direct e-mail ...................................................... 16
Responsible-Changed-Why in Audit-Trail: 12
rmcat: 43, 64

S
sample Problem Report: 6
send-pr: 13, 28
send-pr fields: 14, 16
send-pr within Emacs: 14
send-pr.conf file: 62
send-submitter-ack: 47
serious severity problems: 9
setting up GNATS: 31
severity: 48
Severity field: 9
shell invocation: 13
Site wide configuration files: 44
so what does it do: 3
state: 49
State field: 11
state—analyzed: 5
state—closed: 6
state—feedback: 6
state—open: 5
state—suspended: 6
State-Changed-<From>-<To> in Audit-Trail: 11
State-Changed-By in Audit-Trail: 11
State-Changed-When in Audit-Trail: 11
State-Changed-Why in Audit-Trail: 12
states file: 45, 60
states of Problem Reports: 5
Subject header: 8
submitter-id: 8, 16
Submitter-Id field: 49
submitters file: 45, 59
Submitting a PR via e-mail: 16
subsequent mail: 12, 19
support class: 10
support site: 3
suspended state: 6
su-bug class: 10
synopsis: 49
Synopsis field: 9

T
syntax of regexps: 97
template: 14
template comment section: 15
text: 49
the section on query-by-mail needs to be relocated: 20
timely reminders: 68
To header: 8

U
unformatted: 49
Unformatted field: 12
unlock-database: 30
unlock-pr: 30
unpacking the distribution: 31
unparseable incoming PRs: 4
upgrade, procedure: 40
upgrading from older versions: 39
upgrading, overview: 39
usage for the GNATS user tools: 13
Using and Porting GNU CC: 17
using edit-pr: 17
using gnats over a local network: 38
using gnats over a network: 36, 37
using gnats remotely: 38
using query-pr: 20
using send-pr: 13
using send-pr from within Emacs: 14

V
view-pr: 27
visual map of data flow: 5

W
what is a PR: 3
where GNATS lives: 73
why GNATS: 3