The Bugzilla Guide - 2.16.4 Release

Matthew P. Barnson

The Bugzilla Team

The Bugzilla Guide - 2.16.4 Release

by Matthew P. Barnson and The Bugzilla Team

Published 2003-11-01

This is the documentation for Bugzilla, the mozilla.org bug-tracking system. Bugzilla is an enterprise-class piece of software that powers issue-tracking for hundreds of organizations around the world, tracking millions of bugs.

This documentation is maintained in DocBook 4.1.2 XML format. Changes are best submitted as plain text or XML diffs, attached to a bug filed in mozilla.org's Bugzilla (http://bugzilla.mozilla.org/enter_bug.cgi?product=Bugzilla&component=Documentation).

The most current version of this document can always be found on the Bugzilla Documentation Page (http://www.bugzilla.org/documentation.html).

Table of Contents

1. About This Guide **7**
1.1. Copyright Information 7
1.2. Disclaimer 7
1.3. New Versions 7
1.4. Credits 8
1.5. Document Conventions 9
2. Introduction **11**
2.1. What is Bugzilla? 11
2.2. Why Should We Use Bugzilla? 11
3. Using Bugzilla **13**
3.1. How do I use Bugzilla? 13
3.2. Hints and Tips 16
3.3. User Preferences 17
4. Installation **19**
4.1. Step-by-step Install 19
4.2. Optional Additional Configuration 27
4.3. Win32 Installation Notes 30
4.4. Mac OS X Installation Notes 38
4.5. Troubleshooting 39
5. Administering Bugzilla **43**
5.1. Bugzilla Configuration 43
5.2. User Administration 45
5.3. Product, Component, Milestone, and Version Administration 47
5.4. Voting 49
5.5. Groups and Group Security 50
5.6. Bugzilla Security 51
5.7. Template Customisation 54
5.8. Upgrading to New Releases 58
5.9. Integrating Bugzilla with Third-Party Tools 61
A. The Bugzilla FAQ **63**
B. The Bugzilla Database **75**
B.1. Database Schema Chart 75
B.2. MySQL Bugzilla Database Introduction 75
C. Useful Patches and Utilities for Bugzilla **83**
C.1. Apache `mod_rewrite` magic 83
C.2. Command-line Bugzilla Queries 83

D. Bugzilla Variants and Competitors 85
D.1. Red Hat Bugzilla 85
D.2. Loki Bugzilla (Fenris) 85
D.3. Issuezilla 85
D.4. Scarab 85
D.5. Perforce SCM 86
D.6. SourceForge 86
E. GNU Free Documentation License 87
0. PREAMBLE 87
1. APPLICABILITY AND DEFINITIONS 87
2. VERBATIM COPYING 88
3. COPYING IN QUANTITY 88
4. MODIFICATIONS 89
5. COMBINING DOCUMENTS 91
6. COLLECTIONS OF DOCUMENTS 91
7. AGGREGATION WITH INDEPENDENT WORKS 91
8. TRANSLATION 92
9. TERMINATION 92
10. FUTURE REVISIONS OF THIS LICENSE 92
How to use this License for your documents 92
Glossary 95

List of Figures

4-1. Other File::Temp error messages....41
4-2. Patch for File::Temp in Perl 5.6.0....41

List of Examples

4-1. Installing ActivePerl ppd Modules on Microsoft Windows 32
4-2. Installing OpenInteract ppd Modules manually on Microsoft Windows 33
5-1. Upgrading using CVS....59
5-2. Upgrading using the tarball 59
5-3. Upgrading using patches 60

Chapter 1. About This Guide

1.1. Copyright Information

If you have any questions regarding this document, its copyright, or publishing this document in non-electronic form, please contact The Bugzilla Team.

1.2. Disclaimer

No liability for the contents of this document can be accepted. Use the concepts, examples, and other content at your own risk. This document may contain errors and inaccuracies that may damage your system, cause your partner to leave you, your boss to fire you, your cats to pee on your furniture and clothing, and global thermonuclear war. Proceed with caution.

All copyrights are held by their respective owners, unless specifically noted otherwise. Use of a term in this document should not be regarded as affecting the validity of any trademark or service mark.

Naming of particular products or brands should not be seen as endorsements, with the exception of the term "GNU/Linux". We wholeheartedly endorse the use of GNU/Linux in every situation where it is appropriate. It is an extremely versatile, stable, and robust operating system that offers an ideal operating environment for Bugzilla.

You are strongly recommended to make a backup of your system before installing Bugzilla and at regular intervals thereafter. If you implement any suggestion in this Guide, implement this one!

Although the Bugzilla development team has taken great care to ensure that all easily-exploitable bugs or options are documented or fixed in the code, security holes surely exist. Great care should be taken both in the installation and usage of this software. Carefully consider the implications of installing other network services with Bugzilla. The Bugzilla development team members, Netscape Communications, America Online Inc., and any affiliated developers or sponsors assume no liability for your use of this product. You have the source code to this product, and are responsible for auditing it yourself to ensure your security needs are met.

1.3. New Versions

This is the 2.16.4 version of The Bugzilla Guide. It is so named to match the version of Bugzilla it is disributed with. If you are reading this from any source other than those below, please check one of these mirrors to make sure you are reading an up-to-date version of the Guide.

The newest version of this guide can always be found at bugzilla.org (http://www.bugzilla.org); including documentation for past releases and the current development version.

The documentation for the most recent stable release of Bugzilla can also be found at The Linux Documentation Project (http://www.tldp.org).

The latest version of this document can always be checked out via CVS. Please follow the instructions available at the Mozilla CVS page (http://www.mozilla.org/cvs.html), and check out the `mozilla/webtools/bugzilla/docs/` subtree.

The Bugzilla Guide is currently only available in English. If you would like to volunteer to translate it, please contact Dave Miller (mailto:justdave@syndicomm.com).

1.4. Credits

The people listed below have made enormous contributions to the creation of this Guide, through their writing, dedicated hacking efforts, numerous e-mail and IRC support sessions, and overall excellent contribution to the Bugzilla community:

Matthew P. Barnson <`mbarnson@sisna.com`>

> for the Herculaean task of pulling together the Bugzilla Guide and shepherding it to 2.14.

Terry Weissman <`terry@mozilla.org`>

> for initially writing Bugzilla and creating the README upon which the UNIX installation documentation is largely based.

Tara Hernandez <`tara@tequilarists.org`>

> for keeping Bugzilla development going strong after Terry left mozilla.org and for running landfill.

Dave Lawrence <`dkl@redhat.com`>

> for providing insight into the key differences between Red Hat's customized Bugzilla, and being largely responsible for Section D.1.

Dawn Endico <endico@mozilla.org>

for being a hacker extraordinaire and putting up with Matthew's incessant questions and arguments on irc.mozilla.org in #mozwebtools

Jacob Steenhagen <jake@bugzilla.org>

for taking over documentation during the 2.17 development period and backporting relevent docs changes to this 2.16 branch.

Last but not least, all the members of the news://news.mozilla.org/netscape/public/mozilla/webtools newsgroup. Without your discussions, insight, suggestions, and patches, this could never have happened.

Thanks also go to the following people for significant contributions to this documentation (in alphabetical order): Andrew Pearson, Ben FrantzDale, Eric Hanson, Gervase Markham, Joe Robins, Kevin Brannen, Ron Teitelbaum, Spencer Smith, Zach Liption .

1.5. Document Conventions

This document uses the following conventions:

Descriptions	Appearance
Use caution	**Caution** Don't run with scissors!
Hint	**Tip:** Would you like a breath mint?
Notes	**Note:** Dear John...
Warnings	**Warning** Read this or the cat gets it.
File Names	`filename`
Directory Names	`directory`
Commands to be typed	**command**
Applications Names	application
Prompt of users command under bash shell	bash$
Prompt of root users command under bash shell	bash#

Descriptions	Appearance
Prompt of user command under tcsh shell	tcsh$
Environment Variables	VARIABLE
Emphasized word	*word*
Term found in the glossary	*Bugzilla*
Code Example	`<para> Beginning and end of paragraph </para>`

Chapter 2. Introduction

2.1. What is Bugzilla?

Bugzilla is a bug- or issue-tracking system. Bug-tracking systems allow individual or groups of developers effectively to keep track of outstanding problems with their product. Bugzilla was originally written by Terry Weissman in a programming language called *TCL*, to replace a rudimentary bug-tracking database used internally by Netscape Communications. Terry later ported Bugzilla to Perl from TCL, and in Perl it remains to this day. Most commercial defect-tracking software vendors at the time charged enormous licensing fees, and Bugzilla quickly became a favorite of the open-source crowd (with its genesis in the open-source browser project, Mozilla). It is now the de-facto standard defect-tracking system against which all others are measured.

Bugzilla boasts many advanced features. These include:

- Powerful searching
- User-configurable email notifications of bug changes
- Full change history
- Inter-bug dependency tracking and graphing
- Excellent attachment management
- Integrated, product-based, granular security schema
- Fully security-audited, and runs under Perl's taint mode
- A robust, stable RDBMS back-end
- Web, XML, email and console interfaces
- Completely customisable and/or localisable web user interface
- Extensive configurability
- Smooth upgrade pathway between versions

2.2. Why Should We Use Bugzilla?

For many years, defect-tracking software has remained principally the domain of large software development houses. Even then, most shops never bothered with bug-tracking software, and instead simply relied on shared lists and email

to monitor the status of defects. This procedure is error-prone and tends to cause those bugs judged least significant by developers to be dropped or ignored.

These days, many companies are finding that integrated defect-tracking systems reduce downtime, increase productivity, and raise customer satisfaction with their systems. Along with full disclosure, an open bug-tracker allows manufacturers to keep in touch with their clients and resellers, to communicate about problems effectively throughout the data management chain. Many corporations have also discovered that defect-tracking helps reduce costs by providing IT support accountability, telephone support knowledge bases, and a common, well-understood system for accounting for unusual system or software issues.

But why should *you* use Bugzilla?

Bugzilla is very adaptable to various situations. Known uses currently include IT support queues, Systems Administration deployment management, chip design and development problem tracking (both pre-and-post fabrication), and software and hardware bug tracking for luminaries such as Redhat, NASA, Linux-Mandrake, and VA Systems. Combined with systems such as CVS (http://www.cvshome.org), Bonsai (http://www.mozilla.org/bonsai.html), or Perforce SCM (http://www.perforce.com), Bugzilla provides a powerful, easy-to-use solution to configuration management and replication problems.

Bugzilla can dramatically increase the productivity and accountability of individual employees by providing a documented workflow and positive feedback for good performance. How many times do you wake up in the morning, remembering that you were supposed to do *something* today, but you just can't quite remember? Put it in Bugzilla, and you have a record of it from which you can extrapolate milestones, predict product versions for integration, and follow the discussion trail that led to critical decisions.

Ultimately, Bugzilla puts the power in your hands to improve your value to your employer or business while providing a usable framework for your natural attention to detail and knowledge store to flourish.

Chapter 3. Using Bugzilla

3.1. How do I use Bugzilla?

This section contains information for end-users of Bugzilla. There is a Bugzilla test installation, called Landfill (http://landfill.bugzilla.org/), which you are welcome to play with (if it's up.) However, it does not necessarily have all Bugzilla features enabled, and often runs cutting-edge versions of Bugzilla for testing, so some things may work slightly differently than mentioned here.

3.1.1. Create a Bugzilla Account

If you want to use Bugzilla, first you need to create an account. Consult with the administrator responsible for your installation of Bugzilla for the URL you should use to access it. If you're test-driving Bugzilla, use this URL: http://landfill.bugzilla.org/bugzilla-tip/ (http://landfill.bugzilla.org/bugzilla-tip/)

1. Click the "Open a new Bugzilla account" link, enter your email address and, optionally, your name in the spaces provided, then click "Create Account" .
2. Within moments, you should receive an email to the address you provided above, which contains your login name (generally the same as the email address), and a password you can use to access your account. This password is randomly generated, and can be changed to something more memorable.
3. Click the "Log In" link in the yellow area at the bottom of the page in your browser, enter your email address and password into the spaces provided, and click "Login".

You are now logged in. Bugzilla uses cookies for authentication so, unless your IP address changes, you should not have to log in again.

3.1.2. Anatomy of a Bug

The core of Bugzilla is the screen which displays a particular bug. It's a good place to explain some Bugzilla concepts. Bug 1 on Landfill (http://landfill.bugzilla.org/bugzilla-tip/show_bug.cgi?id=1) is a good example. Note that the labels for most fields are hyperlinks; clicking them will take you to context-sensitive help on that particular field. Fields marked * may not be present on every installation of Bugzilla.

1. *Product and Component*: Bugs are divided up by Product and Component, with a Product having one or more Components in it. For example, bugzilla.mozilla.org's "Bugzilla" Product is composed of several Components:

Administration: Administration of a Bugzilla installation.
Bugzilla-General: Anything that doesn't fit in the other components, or spans multiple components.
Creating/Changing Bugs: Creating, changing, and viewing bugs.
Documentation: The Bugzilla documentation, including The Bugzilla Guide.
Email: Anything to do with email sent by Bugzilla.
Installation: The installation process of Bugzilla.
Query/Buglist: Anything to do with searching for bugs and viewing the buglists.
Reporting/Charting: Getting reports from Bugzilla.
User Accounts: Anything about managing a user account from the user's perspective. Saved queries, creating accounts, changing p
User Interface: General issues having to do with the user interface cosmetics (not functionality) including cosmetic issues, HTML

2. *Status and Resolution:* These define exactly what state the bug is in - from not even being confirmed as a bug, through to being fixed and the fix confirmed by Quality Assurance. The different possible values for Status and Resolution on your installation should be documented in the context-sensitive help for those items.
3. *Assigned To:* The person responsible for fixing the bug.
4. **URL:* A URL associated with the bug, if any.
5. *Summary:* A one-sentence summary of the problem.
6. **Status Whiteboard:* (a.k.a. Whiteboard) A free-form text area for adding short notes and tags to a bug.
7. **Keywords:* The administrator can define keywords which you can use to tag and categorise bugs - e.g. The Mozilla Project has keywords like crash and regression.
8. *Platform and OS:* These indicate the computing environment where the bug was found.
9. *Version:* The "Version" field is usually used for versions of a product which have been released, and is set to indicate which versions of a Component have the particular problem the bug report is about.
10. *Priority:* The bug assignee uses this field to prioritise his or her bugs. It's a good idea not to change this on other people's bugs.
11. *Severity:* This indicates how severe the problem is - from blocker ("application unusable") to trivial ("minor cosmetic issue"). You can also use this field to indicate whether a bug is an enhancement request.
12. **Target:* (a.k.a. Target Milestone) A future version by which the bug is to be fixed. e.g. The Bugzilla Project's milestones for future Bugzilla versions are 2.18, 2.20, 3.0, etc. Milestones are not restricted to numbers, thought - you can use any text strings, such as dates.
13. *Reporter:* The person who filed the bug.
14. *CC list:* A list of people who get mail when the bug changes.
15. *Attachments:* You can attach files (e.g. testcases or patches) to bugs. If there are any attachments, they are listed in this section.
16. **Dependencies:* If this bug cannot be fixed unless other bugs are fixed (depends on), or this bug stops other bugs being fixed (blocks), their numbers are recorded here.

17. **Votes:* Whether this bug has any votes.
18. *Additional Comments:* You can add your two cents to the bug discussion here, if you have something worthwhile to say.

3.1.3. Searching for Bugs

The Bugzilla Search page is is the interface where you can find any bug report, comment, or patch currently in the Bugzilla system. You can play with it here: landfill.bugzilla.org/bugzilla-tip/query.cgi (http://landfill.bugzilla.org/bugzilla-tip/query.cgi) .

The Search page has controls for selecting different possible values for all of the fields in a bug, as described above. Once you've defined a search, you can either run it, or save it as a Remembered Query, which can optionally appear in the footer of your pages.

Highly advanced querying is done using Boolean Charts, which have their own context-sensitive help (http://landfill.bugzilla.org/bugzilla-tip/booleanchart.html) .

3.1.4. Bug Lists

If you run a search, a list of matching bugs will be returned. The default search is to return all open bugs on the system - don't try running this search on a Bugzilla installation with a lot of bugs!

The format of the list is configurable. For example, it can be sorted by clicking the column headings. Other useful features can be accessed using the links at the bottom of the list:

Long Format: this gives you a large page with a non-editable summary of the fields of each bug.
Change Columns: change the bug attributes which appear in the list.
Change several bugs at once: If your account is sufficiently empowered, you can make the same change to all the bugs in the list - for
Send mail to bug owners: Sends mail to the owners of all bugs on the list.
Edit this query: If you didn't get exactly the results you were looking for, you can return to the Query page through this link and make

3.1.5. Filing Bugs

Years of bug writing experience has been distilled for your reading pleasure into the Bug Writing Guidelines (http://landfill.bugzilla.org/bugzilla-tip/bugwritinghelp.html). While some of the advice is Mozilla-specific, the basic principles of reporting Reproducible, Specific bugs, isolating the Product you are using, the Version of the Product, the Component which failed, the Hardware Platform, and Operating System you were using at the time of the failure go a long way toward ensuring accurate, responsible fixes for the bug that bit you.

The procedure for filing a test bug is as follows:

1. Go to Landfill (http://landfill.bugzilla.org/bugzilla-tip/) in your browser and click Enter a new bug report (http://landfill.bugzilla.org/bugzilla-tip/enter_bug.cgi).
2. Select a product - any one will do.
3. Fill in the fields. Bugzilla should have made reasonable guesses, based upon your browser, for the "Platform" and "OS" drop-down boxes. If they are wrong, change them.
4. Select "Commit" and send in your bug report.

3.2. Hints and Tips

This section distills some Bugzilla tips and best practices that have been developed.

3.2.1. Autolinkification

Bugzilla comments are plain text - so posting HTML will result in literal HTML tags rather than being interpreted by a browser. However, Bugzilla will automatically make hyperlinks out of certain sorts of text in comments. For example, the text http://www.bugzilla.org will be turned into http://www.bugzilla.org. Other strings which get linkified in the obvious manner are:

bug 12345
bug 23456, comment 53
attachment 4321
mailto:george@example.com
george@example.com
ftp://ftp.mozilla.org
Most other sorts of URL

A corollary here is that if you type a bug number in a comment, you should put the word "bug" before it, so it gets autolinkified for the convenience of others.

3.2.2. Quicksearch

Quicksearch is a single-text-box query tool which uses metacharacters to indicate what is to be searched. For example, typing "`foo|bar`" into Quicksearch would search for "foo" or "bar" in the summary and status whiteboard of a bug; adding "`:BazProduct`" would search only in that product.

You'll find the Quicksearch box on Bugzilla's front page, along with a Help (../../quicksearch.html) link which details how to use it.

3.2.3. Comments

If you are changing the fields on a bug, only comment if either you have something pertinent to say, or Bugzilla requires it. Otherwise, you may spam people unnecessarily with bug mail. To take an example: a user can set up their account to filter out messages where someone just adds themselves to the CC field of a bug (which happens a lot.) If you come along, add yourself to the CC field, and add a comment saying "Adding self to CC", then that person gets a pointless piece of mail they would otherwise have avoided.

Don’t use sigs in comments. Signing your name ("Bill") is acceptable, particularly if you do it out of habit, but full mail/news-style four line ASCII art creations are not.

3.2.4. Attachments

Use attachments, rather than comments, for large chunks of ASCII data, such as trace, debugging output files, or log files. That way, it doesn’t bloat the bug for everyone who wants to read it, and cause people to receive fat, useless mails.

Trim screenshots. There’s no need to show the whole screen if you are pointing out a single-pixel problem.

Don’t attach simple test cases (e.g. one HTML file, one CSS file and an image) as a ZIP file. Instead, upload them in reverse order and edit the referring file so that they point to the attached files. This way, the test case works immediately out of the bug.

3.2.5. Filing Bugs

Try to make sure that everything said in the summary is also said in the first comment. Summaries are often updated and this will ensure your original information is easily accessible.

You do not need to put "any" or similar strings in the URL field. If there is no specific URL associated with the bug, leave this field blank.

If you feel a bug you filed was incorrectly marked as a DUPLICATE of another, please question it in your bug, not the bug it was duped to. Feel free to CC the person who duped it if they are not already CCed.

3.3. User Preferences

Once you have logged in, you can customise various aspects of Bugzilla via the "Edit prefs" link in the page footer. The preferences are split into four tabs:

3.3.1. Account Settings

On this tab, you can change your basic account information, including your password, email address and real name. For security reasons, in order to change anything on this page you must type your *current* password into the “Password” field at the top of the page. If you attempt to change your email address, a confirmation email is sent to both the old and new addresses, with a link to use to confirm the change. This helps to prevent account hijacking.

3.3.2. Email Settings

On this tab you can reduce or increase the amount of email sent you from Bugzilla, opting in our out depending on your relationship to the bug and the change that was made to it. (Note that you can also do client-side filtering using the X-Bugzilla-Reason header which Bugzilla adds to all bugmail.)

By entering user email names, delineated by commas, into the "Users to watch" text entry box you can receive a copy of all the bugmail of other users (security settings permitting.) This powerful functionality enables seamless transitions as developers change projects or users go on holiday.

> **Note:** The ability to watch other users may not be available in all Bugzilla installations. If you can’t see it, ask your administrator.

3.3.3. Page Footer

On the Search page, you can store queries in Bugzilla, so if you regularly run a particular query it is just a drop-down menu away. Once you have a stored query, you can come here to request that it also be displayed in your page footer.

3.3.4. Permissions

This is a purely informative page which outlines your current permissions on this installation of Bugzilla - what product groups you are in, and whether you can edit bugs or perform various administration functions.

Chapter 4. Installation

4.1. Step-by-step Install

4.1.1. Introduction

Bugzilla has been successfully installed under Solaris, Linux, and Win32. Win32 is not yet officially supported, but many people have got it working fine. Please see the Win32 Installation Notes for further advice on getting Bugzilla to work on Microsoft Windows.

4.1.2. Package List

> **Note:** If you are running the very most recent version of Perl and MySQL (both the executables and development libraries) on your system, you can skip these manual installation steps for the Perl modules by using Bundle::Bugzilla; see Using Bundle::Bugzilla instead of manually installing Perl modules.

The software packages necessary for the proper running of Bugzilla (with download links) are:

1. MySQL database server (http://www.mysql.com/) (3.22.5 or greater)
2. Perl (http://www.perl.org) (5.005 or greater, 5.6.1 is recommended if you wish to use Bundle::Bugzilla)
3. Perl Modules (minimum version):
    a. Template (http://www.template-toolkit.org) (v2.07)
    b. AppConfig (http://www.cpan.org/modules/by-module/AppConfig/) (v1.52)
    c. Text::Wrap (http://www.cpan.org/authors/id/MUIR/modules/Text-Tabs%2BWrap-2001.0131.tar.gz) (v2001.0131)
    d. File::Spec (http://search.cpan.org/search?dist=File-Spec) (v0.8.2)
    e. Data::Dumper (http://www.cpan.org/modules/by-module/Data/) (any)
    f. DBD::mysql (http://www.cpan.org/modules/by-module/Mysql/) (v1.2209)
    g. DBI (http://www.cpan.org/modules/by-module/DBI/) (v1.13)
    h. Date::Parse (http://www.cpan.org/modules/by-module/Date/) (any)

i. CGI::Carp (any)

and, optionally:

a. GD (http://www.cpan.org/modules/by-module/GD/) (v1.19) for bug charting

b. Chart::Base (http://www.cpan.org/modules/by-module/Chart/) (v0.99c) for bug charting

c. XML::Parser (any) for the XML interface

d. MIME::Parser (any) for the email interface

4. The web server of your choice. Apache (http://www.apache.org/) is highly recommended.

> **Warning**
>
> It is a good idea, while installing Bugzilla, to ensure that there is some kind of firewall between you and the rest of the Internet, because your machine may be insecure for periods during the install. Many installation steps require an active Internet connection to complete, but you must take care to ensure that at no point is your machine vulnerable to an attack.

Note: Linux-Mandrake 8.0 includes every required and optional library for Bugzilla. The easiest way to install them is by using the `urpmi` utility. If you follow these commands, you should have everything you need for Bugzilla, and `checksetup.pl` should not complain about any missing libraries. You may already have some of these installed.

```
bash# urpmi perl-mysql
bash# urpmi perl-chart
bash# urpmi perl-gd
bash# urpmi perl-MailTools (for Bugzilla email integration)
bash# urpmi apache-modules
```

4.1.3. MySQL

Visit the MySQL homepage at www.mysql.com (http://www.mysql.com) to grab and install the latest stable release of the server.

Note: Many of the binary versions of MySQL store their data files in `/var`. On some Unix systems, this is part of a smaller root partition, and may not have room for your bug database. You can set the data directory as an option to `configure` if you build MySQL from source yourself.

If you install from something other than an RPM or Debian package, you will need to add `mysqld` to your init scripts so the server daemon will come back up whenever your machine reboots. Further discussion of UNIX init sequences are beyond the scope of this guide.

Change your init script to start `mysqld` with the ability to accept large packets. By default, `mysqld` only accepts packets up to 64K long. This limits the size of attachments you may put on bugs. If you add `-O max_allowed_packet=1M` to the command that starts `mysqld` (or `safe_mysqld`), then you will be able to have attachments up to about 1 megabyte. There is a Bugzilla parameter for maximum attachment size; you should configure it to match the value you choose here.

If you plan on running Bugzilla and MySQL on the same machine, consider using the `--skip-networking` option in the init script. This enhances security by preventing network access to MySQL.

4.1.4. Perl

Any machine that doesn't have Perl on it is a sad machine indeed. Perl can be got in source form from perl.com (http://www.perl.com) for the rare *nix systems which don't have it. Although Bugzilla runs with all post-5.005 versions of Perl, it's a good idea to be up to the very latest version if you can when running Bugzilla. As of this writing, that is Perl version 5.6.1.

> **Tip:** You can skip the following Perl module installation steps by installing Bundle::Bugzilla from *CPAN*, which installs all required modules for you.
>
> ```
> bash# perl -MCPAN -e 'install "Bundle::Bugzilla"'
> ```
>
> Bundle::Bugzilla doesn't include GD, Chart::Base, or MIME::Parser, which are not essential to a basic Bugzilla install. If installing this bundle fails, you should install each module individually to isolate the problem.

4.1.5. Perl Modules

All Perl modules can be found on the Comprehensive Perl Archive Network (http://www.cpan.org) (CPAN). The CPAN servers have a real tendency to bog down, so please use mirrors.

Quality, general Perl module installation instructions can be found on the CPAN website, but the easy thing to do is to just use the CPAN shell which does all the hard work for you. To use the CPAN shell to install a module:

```
bash# perl -MCPAN -e 'install "<modulename>"'
```

To do it the hard way:

Untar the module tarball -- it should create its own directory

CD to the directory just created, and enter the following commands:

1. `bash# perl Makefile.PL`
2. `bash# make`
3. `bash# make test`
4. `bash# make install`

> **Warning**
>
> Many people complain that Perl modules will not install for them. Most times, the error messages complain that they are missing a file in "@INC". Virtually every time, this error is due to permissions being set too restrictively for you to compile Perl modules or not having the necessary Perl development libraries installed on your system. Consult your local UNIX systems administrator for help solving these permissions issues; if you *are* the local UNIX sysadmin, please consult the newsgroup/mailing list for further assistance or hire someone to help you out.

4.1.5.1. DBI

The DBI module is a generic Perl module used the MySQL-related modules. As long as your Perl installation was done correctly the DBI module should be a breeze. It's a mixed Perl/C module, but Perl's MakeMaker system simplifies the C compilation greatly.

4.1.5.2. Data::Dumper

The Data::Dumper module provides data structure persistence for Perl (similar to Java's serialization). It comes with later sub-releases of Perl 5.004, but a re-installation just to be sure it's available won't hurt anything.

4.1.5.3. MySQL-related modules

The Perl/MySQL interface requires a few mutually-dependent Perl modules. These modules are grouped together into the the Msql-Mysql-modules package.

The MakeMaker process will ask you a few questions about the desired compilation target and your MySQL installation. For most of the questions the provided default will be adequate, but when asked if your desired target is the MySQL or mSQL packages, you should select the MySQL related ones. Later you will be asked if you wish to

provide backwards compatibility with the older MySQL packages; you should answer YES to this question. The default is NO.

A host of 'localhost' should be fine and a testing user of 'test' with a null password should find itself with sufficient access to run tests on the 'test' database which MySQL created upon installation.

4.1.5.4. TimeDate modules

Many of the more common date/time/calendar related Perl modules have been grouped into a bundle similar to the MySQL modules bundle. This bundle is stored on the CPAN under the name TimeDate. The component module we're most interested in is the Date::Format module, but installing all of them is probably a good idea anyway.

4.1.5.5. GD (optional)

The GD library was written by Thomas Boutell a long while ago to programatically generate images in C. Since then it's become the defacto standard for programatic image construction. The Perl bindings to it found in the GD library are used on millions of web pages to generate graphs on the fly. That's what Bugzilla will be using it for so you must install it if you want any of the graphing to work.

> **Note:** The Perl GD library requires some other libraries that may or may not be installed on your system, including `libpng` and `libgd`. The full requirements are listed in the Perl GD library README. If compiling GD fails, it's probably because you're missing a required library.

4.1.5.6. Chart::Base (optional)

The Chart module provides Bugzilla with on-the-fly charting abilities. It can be installed in the usual fashion after it has been fetched from CPAN. Note that earlier versions that 0.99c used GIFs, which are no longer supported by the latest versions of GD.

4.1.5.7. Template Toolkit

When you install Template Toolkit, you'll get asked various questions about features to enable. The defaults are fine, except that it is recommended you use the high speed XS Stash of the Template Toolkit, in order to achieve best performance. However, there are known problems with XS Stash and Perl 5.005_02 and lower. If you wish to use these older versions of Perl, please use the regular stash.

4.1.6. HTTP Server

You have a freedom of choice here - Apache, Netscape or any other server on UNIX would do. You can run the web server on a different machine than MySQL, but need to adjust the MySQL “bugs” user permissions accordingly.

> **Note:** We strongly recommend Apache as the web server to use. The Bugzilla Guide installation instructions, in general, assume you are using Apache. If you have got Bugzilla working using another webserver, please share your experiences with us.

You’ll want to make sure that your web server will run any file with the .cgi extension as a CGI and not just display it. If you’re using Apache that means uncommenting the following line in the httpd.conf file:

```
AddHandler cgi-script .cgi
```

With Apache you’ll also want to make sure that within the httpd.conf file the line:

```
Options ExecCGI
AllowOverride Limit
```

is in the stanza that covers the directories into which you intend to put the bugzilla .html and .cgi files.

> **Note:** AllowOverride Limit allows the use of a Deny statement in the .htaccess file generated by checksetup.pl
>
> Users of older versions of Apache may find the above lines in the srm.conf and access.conf files, respecitvely.

Warning

There are important files and directories that should not be a served by the HTTP server - most files in the “data” and “shadow” directories and the “localconfig” file. You should configure your HTTP server to not serve these files. Failure to do so will expose critical passwords and other data. Please see .htaccess files and security for details on how to do this for Apache; the checksetup.pl script should create appropriate .htaccess files for you.

4.1.7. Bugzilla

You should untar the Bugzilla files into a directory that you're willing to make writable by the default web server user (probably "nobody"). You may decide to put the files in the main web space for your web server or perhaps in `/usr/local` with a symbolic link in the web space that points to the Bugzilla directory.

> **Tip:** If you symlink the bugzilla directory into your Apache's HTML heirarchy, you may receive Forbidden errors unless you add the "FollowSymLinks" directive to the <Directory> entry for the HTML root in httpd.conf.

Once all the files are in a web accessible directory, make that directory writable by your webserver's user. This is a temporary step until you run the post-install `checksetup.pl` script, which locks down your installation.

Lastly, you'll need to set up a symbolic link to `/usr/bonsaitools/bin/perl` for the correct location of your Perl executable (probably `/usr/bin/perl`). Otherwise you must hack all the .cgi files to change where they look for Perl. This can be done using the following Perl one-liner, but I suggest using the symlink approach to avoid upgrade hassles.

```
perl -pi -e
        's@#\!/usr/bonsaitools/bin/perl@#\!/usr/bin/perl@' *cgi *pl Bug.pm
        processmail syncshadowdb
```

Change `/usr/bin/perl` to match the location of Perl on your machine.

4.1.8. Setting Up the MySQL Database

After you've gotten all the software installed and working you're ready to start preparing the database for its life as the back end to a high quality bug tracker.

First, you'll want to fix MySQL permissions to allow access from Bugzilla. For the purpose of this Installation section, the Bugzilla username will be "bugs", and will have minimal permissions.

Begin by giving the MySQL root user a password. MySQL passwords are limited to 16 characters.

```
bash# mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('<new_password>') WHERE user='root';
mysql> FLUSH PRIVILEGES;
```

From this point on, if you need to access MySQL as the MySQL root user, you will need to use **mysql -u root -p** and enter <new_password>. Remember that MySQL user names have nothing to do with Unix user names (login names).

Next, we use an SQL **GRANT** command to create a "bugs" user, and grant sufficient permissions for checksetup.pl, which we'll use later, to work its magic. This also restricts the "bugs" user to operations within a database called "bugs", and only allows the account to connect from "localhost". Modify it to reflect your setup if you will be

connecting from another machine or as a different user.

Remember to set <bugs_password> to some unique password.

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,INDEX, ALTER,CREATE,DROP,REFERENCES ON bugs.* TO bugs@loca
mysql> FLUSH PRIVILEGES;
```

Note: If you are using MySQL 4, the bugs user also needs to be granted the LOCK TABLES and CREATE TEMPORARY TABLES permissions.

4.1.9. `checksetup.pl`

Next, run the magic checksetup.pl script. (Many thanks to Holger Schurig (mailto:holgerschurig@nikocity.de) for writing this script!) This script is designed to make sure your MySQL database and other configuration options are consistent with the Bugzilla CGI files. It will make sure Bugzilla files and directories have reasonable permissions, set up the `data` directory, and create all the MySQL tables.

```
bash# ./checksetup.pl
```

The first time you run it, it will create a file called `localconfig`.

This file contains a variety of settings you may need to tweak including how Bugzilla should connect to the MySQL database.

The connection settings include:

1. server's host: just use "localhost" if the MySQL server is local
2. database name: "bugs" if you're following these directions
3. MySQL username: "bugs" if you're following these directions
4. Password for the "bugs" MySQL account; (<bugs_password>) above

Once you are happy with the settings, `su` to the user your web server runs as, and re-run `checksetup.pl`. (Note: on some security-conscious systems, you may need to change the login shell for the webserver account before you can do this.) On this second run, it will create the database and an administrator account for which you will be prompted to provide information.

Note: The checksetup.pl script is designed so that you can run it at any time without causing harm. You should run it after any upgrade to Bugzilla.

4.1.10. Configuring Bugzilla

You should run through the parameters on the Edit Parameters page (link in the footer) and set them all to appropriate values. They key parameters are documented in Section 5.1.

4.2. Optional Additional Configuration

4.2.1. Dependency Charts

As well as the text-based dependency graphs, Bugzilla also supports dependency graphing, using a package called 'dot'. Exactly how this works is controlled by the 'webdotbase' parameter, which can have one of three values:

1. A complete file path to the command 'dot' (part of GraphViz (http://www.graphviz.org/)) will generate the graphs locally
2. A URL prefix pointing to an installation of the webdot package will generate the graphs remotely
3. A blank value will disable dependency graphing.

So, to get this working, install GraphViz (http://www.graphviz.org/). If you do that, you need to enable server-side image maps (http://httpd.apache.org/docs/mod/mod_imap.html) in Apache. Alternatively, you could set up a webdot server, or use the AT&T public webdot server (the default for the webdotbase param). Note that AT&T's server won't work if Bugzilla is only accessible using HTTPS.

4.2.2. Bug Graphs

As long as you installed the GD and Graph::Base Perl modules you might as well turn on the nifty Bugzilla bug reporting graphs.

Add a cron entry like this to run `collectstats.pl` daily at 5 after midnight:

```
  bash# crontab -e
 5 0 * * * cd <your-bugzilla-directory> ; ./collectstats.pl
```

After two days have passed you'll be able to view bug graphs from the Bug Reports page.

4.2.3. The Whining Cron

By now you have a fully functional Bugzilla, but what good are bugs if they're not annoying? To help make those bugs more annoying you can set up Bugzilla's automatic whining system to complain at engineers which leave their bugs in the NEW state without triaging them.

This can be done by adding the following command as a daily crontab entry (for help on that see that crontab man page):

```
cd <your-bugzilla-directory> ; ./whineatnews.pl
```

Tip: Depending on your system, crontab may have several manpages. The following command should lead you to the most useful page for this purpose:

```
man 5 crontab
```

4.2.4. LDAP Authentication

Warning

This information on using the LDAP authentication options with Bugzilla is old, and the authors do not know of anyone who has tested it. Approach with caution.

The existing authentication scheme for Bugzilla uses email addresses as the primary user ID, and a password to authenticate that user. All places within Bugzilla where you need to deal with user ID (e.g assigning a bug) use the email address. The LDAP authentication builds on top of this scheme, rather than replacing it. The initial log in is done with a username and password for the LDAP directory. This then fetches the email address from LDAP and authenticates seamlessly in the standard Bugzilla authentication scheme using this email address. If an account for this address already exists in your Bugzilla system, it will log in to that account. If no account for that email address exists, one is created at the time of login. (In this case, Bugzilla will attempt to use the "displayName" or "cn" attribute to determine the user's full name.) After authentication, all other user-related tasks are still handled by email address, not LDAP username. You still assign bugs by email address, query on users by email address, etc.

Using LDAP for Bugzilla authentication requires the Mozilla::LDAP (aka PerLDAP) Perl module. The Mozilla::LDAP module in turn requires Netscape's Directory SDK for C. After you have installed the SDK, then

install the PerLDAP module. Mozilla::LDAP and the Directory SDK for C are both available for download (http://www.mozilla.org/directory/) from mozilla.org.

Set the Param 'useLDAP' to "On" **only** if you will be using an LDAP directory for authentication. Be very careful when setting up this parameter; if you set LDAP authentication, but do not have a valid LDAP directory set up, you will not be able to log back in to Bugzilla once you log out. (If this happens, you can get back in by manually editing the data/params file, and setting useLDAP back to 0.)

If using LDAP, you must set the three additional parameters: Set LDAPserver to the name (and optionally port) of your LDAP server. If no port is specified, it defaults to the default port of 389. (e.g "ldap.mycompany.com" or "ldap.mycompany.com:1234") Set LDAPBaseDN to the base DN for searching for users in your LDAP directory. (e.g. "ou=People,o=MyCompany") uids must be unique under the DN specified here. Set LDAPmailattribute to the name of the attribute in your LDAP directory which contains the primary email address. On most directory servers available, this is "mail", but you may need to change this.

4.2.5. Preventing untrusted Bugzilla content from executing malicious Javascript code

It is possible for a Bugzilla to execute malicious Javascript code. Due to internationalization concerns, we are unable to incorporate the code changes necessary to fulfill the CERT advisory requirements mentioned in http://www.cet.org/tech_tips/malicious_code_mitigation.html/#3 (http://www.cert.org/tech_tips/malicious_code_mitigation.html/#3). Executing the following code snippet from a UNIX command shell will rectify the problem if your Bugzilla installation is intended for an English-speaking audience. As always, be sure your Bugzilla installation has a good backup before making changes, and I recommend you understand what the script is doing before executing it.

```
bash# perl -pi -e "s/Content-Type\: text\/html/Content-Type\: text\/html\; charset=ISO-
8859-1/i" *.cgi *.pl
```

All this one-liner command does is search for all instances of "Content-type: text/html" and replaces it with "Content-Type: text/html; charset=ISO-8859-1" . This specification prevents possible Javascript attacks on the browser, and is suggested for all English-speaking sites. For non-English-speaking Bugzilla sites, I suggest changing "ISO-8859-1", above, to "UTF-8".

Note: using <meta> tags to set the charset is not recommended, as there's a bug in Netscape 4.x which causes pages marked up in this way to load twice.

4.2.6. `.htaccess` files and security

To enhance the security of your Bugzilla installation, Bugzilla's `checksetup.pl` script will generate *`.htaccess`* files which the Apache webserver can use to restrict access to the bugzilla data files. These .htaccess files will not work with Apache 1.2.x - but this has security holes, so you shouldn't be using it anyway.

> **Note:** If you are using an alternate provider of webdot services for graphing (as described when viewing `editparams.cgi` in your web browser), you will need to change the ip address in `data/webdot/.htaccess` to the ip address of the webdot server that you are using.

The default .htaccess file may not provide adequate access restrictions, depending on your web server configuration. Be sure to check the <Directory> entries for your Bugzilla directory so that the `.htaccess` file is allowed to override web server defaults. For instance, let's assume your installation of Bugzilla is installed to `/usr/local/bugzilla` . You should have this <Directory> entry in your `httpd.conf` file:

```
  <Directory /usr/local/bugzilla/>
  Options +FollowSymLinks +Indexes +Includes +ExecCGI
  AllowOverride All
</Directory>
```

The important part above is "AllowOverride All" . Without that, the `.htaccess` file created by `checksetup.pl` will not have sufficient permissions to protect your Bugzilla installation.

If you are using Internet Information Server (IIS) or another web server which does not observe `.htaccess` conventions, you can disable their creation by editing `localconfig` and setting the `$create_htaccess` variable to *`0`*.

4.2.7. `mod_throttle` and Security

It is possible for a user, by mistake or on purpose, to access the database many times in a row which can result in very slow access speeds for other users. If your Bugzilla installation is experiencing this problem , you may install the Apache module `mod_throttle` which can limit connections by ip-address. You may download this module at http://www.snert.com/Software/mod_throttle/ Follow the instructions to install into your Apache install. *This module only functions with the Apache web server!* You may use the **ThrottleClientIP** command provided by this module to accomplish this goal. See the Module Instructions (http://www.snert.com/Software/mod_throttle/) for more information.

4.3. Win32 Installation Notes

This section covers installation on Microsoft Windows. Bugzilla has been made to work on Win32 platforms, but the Bugzilla team wish to emphasise that The easiest way to install Bugzilla on Intel-archiecture machines is to install some variant of GNU/Linux, then follow the UNIX installation instructions in this Guide. If you have any influence in the platform choice for running this system, please choose GNU/Linux instead of Microsoft Windows.

> **Warning**
>
> After that warning, here's the situation for 2.16 and Windows. It doesn't work at all out of the box. You are almost certainly better off getting the 2.17 version from CVS (after consultation with the Bugzilla Team to make sure you are pulling on a stable day) because we'll be doing a load of work to make the Win32 experience more pleasant than it is now.

If you still want to try this, to have any hope of getting it to work, you'll need to apply the mail patch () from bug 124174 (http://bugzilla.mozilla.org/show_bug.cgi?id=124174). After that, you'll need to read the (outdated) installation instructions below, some (probably a lot better) more recent ones (http://bugzilla.mozilla.org/attachment.cgi?id=84430&action=view) kindly provided by Toms Baugis and Jean-Sebastien Guay, and also check the Bugzilla 2.16 Win32 update page (http://www.bugzilla.org/releases/2.16/docs/win32.html). If we get time, we'll write some better installation instructions for 2.16 and put them up there. But no promises.

4.3.1. Win32 Installation: Step-by-step

Note: You should be familiar with, and cross-reference, the rest of the Bugzilla Installation section while performing your Win32 installation.

Making Bugzilla work on Microsoft Windows is no picnic. Support for Win32 has improved dramatically in the last few releases, but, if you choose to proceed, you should be a *very* skilled Windows Systems Administrator with strong troubleshooting abilities, a high tolerance for pain, and moderate perl skills. Bugzilla on NT requires hacking source code and implementing some advanced utilities. What follows is the recommended installation procedure for Win32; additional suggestions are provided in Appendix A .

1. Install Apache Web Server (http://www.apache.org/) for Windows, and copy the Bugzilla files somewhere Apache can serve them. Please follow all the instructions referenced in Bugzilla Installation regarding your Apache configuration, particularly instructions regarding the "AddHandler" parameter and "ExecCGI" .

> **Note:** You may also use Internet Information Server or Personal Web Server for this purpose. However, setup is quite different. If ActivePerl doesn't seem to handle your file associations correctly (for .cgi and .pl files), please consult Appendix A .
>
> If you are going to use IIS, if on Windows NT you must be updated to at least Service Pack 4. Windows 2000 ships with a sufficient version of IIS.

2. Install ActivePerl (http://www.activestate.com/) for Windows. Check http://aspn.activestate.com/ASPN/Downloads/ActivePerl (http://aspn.activestate.com/ASPN/Downloads/ActivePerl/) for a current compiled binary.

   Please also check the following links to fully understand the status of ActivePerl on Win32: Perl Porting (http://language.perl.com/newdocs/pod/perlport.html) , and Perl on Win32 FAQ (http://ftp.univie.ac.at/packages/perl/ports/nt/FAQ/perlwin32faq5.html)

3. Use ppm from your perl\bin directory to install the following packs: DBI, DBD-Mysql, TimeDate, Chart, Date-Calc, Date-Manip, GD, AppConfig, and Template. You may need to extract them from .zip format using Winzip or other unzip program first. Most of these additional ppm modules can be downloaded from ActiveState, but AppConfig and Template should be obtained from OpenInteract using the instructions on the Template Toolkit web site (http://openinteract.sourceforge.net/) .

   > **Note:** You can find a list of modules at http://www.activestate.com/PPMPackages/zips/5xx-builds-only/ (http://www.activestate.com/PPMPackages/zips/5xx-builds-only) or http://www.activestate.com/PPMPackages/5.6plus (http://www.activestate.com/PPMPackages/5.6plus)

   The syntax for ppm is: `C:> ppm <modulename>`

   Example 4-1. Installing ActivePerl ppd Modules on Microsoft Windows

   ```
   C:> ppm DBD-Mysql
   ```

   Watch your capitalization!

   ActiveState's 5.6Plus directory also contains an AppConfig ppm, so you might see the following error when trying to install the version at OpenInteract:

   ```
   Error installing package 'AppConfig': Read a PPD for 'AppConfig', but it is not
   intended for this build of Perl (MSWin32-x86-multi-thread)
   ```

   If so, download both the tarball (http://openinteract.sourceforge.net/ppmpackages/AppConfig.tar.gz) and the ppd (http://openinteract.sourceforge.net/ppmpackages/AppConfig.ppd) directly from OpenInteract, then run

ppm from within the same directory to which you downloaded those files and install the package by referencing the ppd file explicitly via in the install command, f.e.:

Example 4-2. Installing OpenInteract ppd Modules manually on Microsoft Windows

```
install C:\AppConfig.ppd
```

4. Install MySQL for NT.

   Note: You can download MySQL for Windows NT from MySQL.com (http://www.mysql.com/) . Some find it helpful to use the WinMySqlAdmin utility, included with the download, to set up the database.

5. Setup MySQL

   a. `C:> C:\mysql\bin\mysql -u root mysql`

   b. `mysql> DELETE FROM user WHERE Host='localhost' AND User="";`

   c. `mysql> UPDATE user SET Password=PASSWORD ('new_password') WHERE user='root';`

      "new_password" , above, indicates whatever password you wish to use for your "root" user.

   d. `mysql> GRANT SELECT, INSERT, UPDATE, DELETE, INDEX, ALTER, CREATE, DROP, REFERENCES ON bugs.* to bugs@localhost IDENTIFIED BY 'bugs_password';`

      "bugs_password" , above, indicates whatever password you wish to use for your "bugs" user.

   e. `mysql> FLUSH PRIVILEGES;`

   f. `mysql> create database bugs;`

   g. `mysql> exit;`

   h. `C:> C:\mysql\bin\mysqladmin -u root -p reload`

6. Edit `checksetup.pl` in your Bugzilla directory. Change this line:

```
my $webservergid =
            getgrnam($my_webservergroup);
```

   to

```
my $webservergid =
          $my_webservergroup;
```

or the name of the group you wish to own the files explicitly:

```
my $webservergid =
          'Administrators'
```

7. Run `checksetup.pl` from the Bugzilla directory.
8. Edit `localconfig` to suit your requirements. Set `$db_pass` to your "bugs_password" from step 5.d , and `$webservergroup` to "8" .

   Note: Not sure on the "8" for `$webservergroup` above. If it's wrong, please send corrections.

9. Edit `defparams.pl` to suit your requirements. Particularly, set `DefParam("maintainer")` and `DefParam("urlbase") to match your install.`

   Note: This is yet another step I'm not sure of, since the maintainer of this documentation does not maintain Bugzilla on NT. If you can confirm or deny that this step is required, please let me know.

10.

Note: There are several alternatives to Sendmail that will work on Win32. The one mentioned here is a *suggestion* , not a requirement. Some other mail packages that can work include BLAT (http://www.blat.net/) , Windmail (http://www.geocel.com/windmail/) , Mercury Sendmail (http://www.dynamicstate.com/) , and the CPAN Net::SMTP Perl module (available in .ppm). Every option requires some hacking of the Perl scripts for Bugzilla to make it work. The option here simply requires the least.

1. Download NTsendmail, available from www.ntsendmail.com (http://www.ntsendmail.com/) . You must have a "real" mail server which allows you to relay off it in your $ENV{"NTsendmail"} (which you should probably place in globals.pl)
2. Put ntsendmail.pm into your .\perl\lib directory.
3. Add to globals.pl:

   ```
   # these settings configure the NTsendmail
                process use NTsendmail;
                $ENV{"NTsendmail"}="your.smtpserver.box";
   ```

```
$ENV{"NTsendmail_debug"}=1;
$ENV{"NTsendmail_max_tries"}=5;
```

Note: Some mention to also edit `$db_pass` in `globals.pl` to be your “bugs_password” . Although this may get you around some problem authenticating to your database, since globals.pl is not normally restricted by `.htaccess` , your database password is exposed to whoever uses your web server.

4. Find and comment out all occurences of “ **open(SENDMAIL** ” in your Bugzilla directory. Then replace them with:

```
# new sendmail functionality my $mail=new
              NTsendmail; my $from="bugzilla\@your.machine.name.tld"; my
              $to=$login; my $subject=$urlbase;
              $mail->send($from,$to,$subject,$msg);
```

Note: Some have found success using the commercial product, Windmail . You could try replacing your sendmail calls with:

```
open SENDMAIL,
               "|\"C:/General/Web/tools/Windmail 4.0 Beta/windmail\" -t >
               mail.log";
```

or something to that effect.

11. Change all references in all files from `processmail` to `processmail.pl` , and rename `processmail` to `processmail.pl` .

Note: Many think this may be a change we want to make for main-tree Bugzilla. It’s painless for the UNIX folks, and will make the Win32 people happier.

Note: Some people have suggested using the Net::SMTP Perl module instead of NTsendmail or the other options listed here. You can change processmail.pl to make this work.

```
my $smtp = Net::SMTP->new('<Name of your SMTP server>');   #connect to SMTP server
$smtp->mail('<your name>@<you smpt server>');# use the sender's adress here
$smtp->to($tolist); # recipient's address
$smtp->data();  # Start the mail
```

```
$smtp->datasend($msg);
$smtp->dataend();   # Finish sending the mail
$smtp->quit;    # Close the SMTP connection
$logstr = "$logstr; mail sent to $tolist $cclist";
}
```

here is a test mail program for Net::SMTP:

```
use Net::SMTP;
 my $smtp = Net::SMTP->new('<Name of your SMTP server', Timeout => 30, Debug
=> 1, ); # connect to SMTP server
                 $smtp->auth;
                $smtp->mail('you@yourcompany.com');# use the sender's adress
here
                $smtp->to('someotherAddress@someotherdomain.com'); #
recipient's address
                $smtp->data();  # Start the mail
                $smtp->datasend('test');
                $smtp->dataend();   # Finish sending the mail
                $smtp->quit;    # Close the SMTP connection
exit;
```

12.

Note: This step is optional if you are using IIS or another web server which only decides on an interpreter based upon the file extension (.pl), rather than the "shebang" line (#/usr/bonsaitools/bin/perl)

Modify the path to perl on the first line (#!) of all files to point to your Perl installation, and add "perl" to the beginning of all Perl system calls that use a perl script as an argument. This may take you a while. There is a "setperl.csh" utility to speed part of this procedure, available in the Useful Patches and Utilities for Bugzilla section of The Bugzilla Guide. However, it requires the Cygwin GNU-compatible environment for Win32 be set up in order to work. See http://www.cygwin.com/ for details on obtaining Cygwin.

13. Modify the invocation of all system() calls in all perl scripts in your Bugzilla directory. You should specify the full path to perl for each system() call. For instance, change this line in processmail:

```
system ("./processmail",@ARGLIST);
        </programlisting> to
        <programlisting>
system ("C:\\perl\\bin\\perl", "processmail", @ARGLIST);
```

14. Add `binmode()` calls so attachments will work (bug 62000 (http://bugzilla.mozilla.org/show_bug.cgi?id=62000)).

    Because Microsoft Windows based systems handle binary files different than Unix based systems, you need to add the following lines to `createattachment.cgi` and `showattachment.cgi` before the `require 'CGI.pl';` line.

    ```
    binmode(STDIN);
    binmode(STDOUT);
    ```

    Note: According to bug 62000 (http://bugzilla.mozilla.org/show_bug.cgi?id=62000) , the perl documentation says that you should always use `binmode()` when dealing with binary files, but never when dealing with text files. That seems to suggest that rather than arbitrarily putting `binmode()` at the beginning of the attachment files, there should be logic to determine if `binmode()` is needed or not.

Tip: If you are using IIS or Personal Web Server, you must add cgi relationships to Properties -> Home directory (tab) -> Application Settings (section) -> Configuration (button), such as:

```
.cgi to: <perl install directory>\perl.exe %s
        %s .pl to: <perl install directory>\perl.exe %s %s
        GET,HEAD,POST
```

Change the path to Perl to match your install, of course.

4.3.2. Additional Windows Tips

Tip: From Andrew Pearson:

> You can make Bugzilla work with Personal Web Server for Windows 98 and higher, as well as for IIS 4.0. Microsoft has information available at http://support.microsoft.com/support/kb/articles/Q231/9/98.ASP (http://support.microsoft.com/support/kb/articles/Q231/9/98.ASP)
>
> Basically you need to add two String Keys in the registry at the following location:
>
> ```
> HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC\Parameters\ScriptMap
> ```
>
> The keys should be called ".pl" and ".cgi", and both should have a value something like: **c:/perl/bin/perl.exe "%s" "%s"**
>
> The KB article only talks about .pl, but it goes into more detail and provides a perl test script.

Tip: If attempting to run Bugzilla 2.12 or older, you will need to remove encrypt() calls from the Perl source. This is *not necessary* for Bugzilla 2.13 and later, which includes the current release, Bugzilla 2.16.4.

Example 4-3. Removing encrypt() for Windows NT Bugzilla version 2.12 or earlier

Replace this:

```
SendSQL("SELECT encrypt(" . SqlQuote($enteredpwd) .
          ", " . SQLQuote(substr($realcryptpwd, 0, 2)) . ")"); my
          $enteredcryptpwd = FetchOneColumn();
```

with this:

```
my $enteredcryptpwd = $enteredpwd
```

in cgi.pl.

4.4. Mac OS X Installation Notes

There are a lot of common libraries and utilities out there that Apple did not include with Mac OS X, but which run perfectly well on it. The GD library, which Bugzilla needs to do bug graphs, is one of these.

The easiest way to get a lot of these is with a program called Fink, which is similar in nature to the CPAN installer, but installs common GNU utilities. Fink is available from <http://sourceforge.net/projects/fink/>.

Follow the instructions for setting up Fink. Once it's installed, you'll want to run the following as root: **fink install gd**

It will prompt you for a number of dependencies, type 'y' and hit enter to install all of the dependencies. Then watch it work.

To prevent creating conflicts with the software that Apple installs by default, Fink creates its own directory tree at /sw where it installs most of the software that it installs. This means your libraries and headers for libgd will be at /sw/lib and /sw/include instead of /usr/lib and /usr/local/include. Because of these changed locations for the libraries, the Perl GD module will not install directly via CPAN, because it looks for the specific paths instead of getting them from your environment. But there's a way around that :-)

Instead of typing "install GD" at the `cpan>` prompt, type **look GD**. This should go through the motions of downloading the latest version of the GD module, then it will open a shell and drop you into the build directory. Apply this patch (../xml/gd-makefile.patch) to the Makefile.PL file (save the patch into a file and use the command **patch < patchfile**.)

Then, run these commands to finish the installation of the GD module:

perl Makefile.PL
make
make test
make install
And don't forget to run **exit** to get back to CPAN.

4.5. Troubleshooting

This section gives solutions to common Bugzilla installation problems.

4.5.1. Bundle::Bugzilla makes me upgrade to Perl 5.6.1

Try executing **perl -MCPAN -e 'install CPAN'** and then continuing.

Certain older versions of the CPAN toolset were somewhat naive about how to upgrade Perl modules. When a couple of modules got rolled into the core Perl distribution for 5.6.1, CPAN thought that the best way to get those modules

up to date was to haul down the Perl distribution itself and build it. Needless to say, this has caused headaches for just about everybody. Upgrading to a newer version of CPAN with the commandline above should fix things.

4.5.2. DBD::Sponge::db prepare failed

The following error message may appear due to a bug in DBD::mysql (over which the Bugzilla team have no control):

```
DBD::Sponge::db prepare failed: Cannot determine NUM_OF_FIELDS at D:/Perl/site/lib/DBD/mysql.pm line
 SV = NULL(0x0) at 0x20fc444
 REFCNT = 1
 FLAGS = (PADBUSY,PADMY)
```

To fix this, go to `<path-to-perl>/lib/DBD/sponge.pm` in your Perl installation and replace

```
my $numFields;
if ($attribs->{'NUM_OF_FIELDS'}) {
    $numFields = $attribs->{'NUM_OF_FIELDS'};
} elsif ($attribs->{'NAME'}) {
    $numFields = @{$attribs->{NAME}};
```

by

```
my $numFields;
if ($attribs->{'NUM_OF_FIELDS'}) {
    $numFields = $attribs->{'NUM_OF_FIELDS'};
} elsif ($attribs->{'NAMES'}) {
    $numFields = @{$attribs->{NAMES}};
```

(note the S added to NAME.)

4.5.3. cannot chdir(/var/spool/mqueue)

If you are installing Bugzilla on SuSE Linux, or some other distributions with "paranoid" security options, it is possible that the checksetup.pl script may fail with the error:

```
cannot chdir(/var/spool/mqueue): Permission denied
```

This is because your `/var/spool/mqueue` directory has a mode of "drwx------". Type **chmod 755 `/var/spool/mqueue`** as root to fix this problem.

4.5.4. Your vendor has not defined Fcntl macro O_NOINHERIT

This is caused by a bug in the version of File::Temp that is distributed with perl 5.6.0. Many minor variations of this error have been reported. Examples can be found in Figure 4-1.

Figure 4-1. Other File::Temp error messages

```
Your vendor has not defined Fcntl macro O_NOINHERIT, used
at /usr/lib/perl5/site_perl/5.6.0/File/Temp.pm line 208.

Your vendor has not defined Fcntl macro O_EXLOCK, used
at /usr/lib/perl5/site_perl/5.6.0/File/Temp.pm line 210.

Your vendor has not defined Fcntl macro O_TEMPORARY, used
at /usr/lib/perl5/site_perl/5.6.0/File/Temp.pm line 233.
```

Numerous people have reported that upgrading to version 5.6.1 or higher solved the problem for them. A less involved fix is to apply the patch in Figure 4-2. The patch is also available as a patch file (../xml/filetemp.patch).

Figure 4-2. Patch for File::Temp in Perl 5.6.0

```
--- File/Temp.pm.orig   Thu Feb  6 16:26:00 2003
+++ File/Temp.pm        Thu Feb  6 16:26:23 2003
@@ -205,6 +205,7 @@
     # eg CGI::Carp
     local $SIG{__DIE__} = sub {};
     local $SIG{__WARN__} = sub {};
+    local *CORE::GLOBAL::die = sub {};
     $bit = &$func();
     1;
   };
@@ -226,6 +227,7 @@
     # eg CGI::Carp
     local $SIG{__DIE__} = sub {};
     local $SIG{__WARN__} = sub {};
+    local *CORE::GLOBAL::die = sub {};
     $bit = &$func();
     1;
   };
```

Chapter 5. Administering Bugzilla

5.1. Bugzilla Configuration

Bugzilla is configured by changing various parameters, accessed from the "Edit parameters" link in the page footer. Here are some of the key parameters on that page. You should run down this list and set them appropriately after installing Bugzilla.

1. **maintainer**: The maintainer parameter is the email address of the person responsible for maintaining this Bugzilla installation. The address need not be that of a valid Bugzilla account.
2. **urlbase**: This parameter defines the fully qualified domain name and web server path to your Bugzilla installation.

   For example, if your Bugzilla query page is `http://www.foo.com/bugzilla/query.cgi`, set your “urlbase” to `http://www.foo.com/bugzilla/`.
3. **usebuggroups**: This dictates whether or not to implement group-based security for Bugzilla. If set, Bugzilla bugs can have an associated ’group’, defining which users are allowed to see and edit the bug.

   Set "usebuggroups" to "on" *only* if you may wish to restrict access to particular bugs to certain groups of users. I suggest leaving this parameter *off* while initially testing your Bugzilla.
4. **usebuggroupsentry**: Bugzilla Products can have a group associated with them, so that certain users can only see bugs in certain products. When this parameter is set to “on”, this places all newly-created bugs in the group for their product immediately.
5. **shadowdb**: You run into an interesting problem when Bugzilla reaches a high level of continuous activity. MySQL supports only table-level write locking. What this means is that if someone needs to make a change to a bug, they will lock the entire table until the operation is complete. Locking for write also blocks reads until the write is complete. The “shadowdb” parameter was designed to get around this limitation. While only a single user is allowed to write to a table at a time, reads can continue unimpeded on a read-only shadow copy of the database. Although your database size will double, a shadow database can cause an enormous performance improvement when implemented on extremely high-traffic Bugzilla databases.

   As a guide, mozilla.org began needing “shadowdb” when they reached around 40,000 Bugzilla users with several hundred Bugzilla bug changes and comments per day.

   The value of the parameter defines the name of the shadow bug database. Set "shadowdb" to e.g. "bug_shadowdb" if you will be running a *very* large installation of Bugzilla.

> **Note:** Enabling "shadowdb" can adversely affect the stability of your installation of Bugzilla. You should regularly check that your database is in sync. It is often advisable to force a shadow database sync nightly via “cron”.

If you use the "shadowdb" option, it is only natural that you should turn the "queryagainstshadowdb" option on as well. Otherwise you are replicating data into a shadow database for no reason!

6. **shutdownhtml**: If you need to shut down Bugzilla to perform administration, enter some descriptive HTML here and anyone who tries to use Bugzilla will receive a page to that effect. Obviously, editparams.cgi will still be accessible so you can remove the HTML and re-enable Bugzilla. :-)

7. **passwordmail**: Every time a user creates an account, the text of this parameter (with substitutions) is sent to the new user along with their password message.

   Add any text you wish to the "passwordmail" parameter box. For instance, many people choose to use this box to give a quick training blurb about how to use Bugzilla at your site.

8. **useqacontact**: This allows you to define an email address for each component, in addition to that of the default owner, who will be sent carbon copies of incoming bugs.

9. **usestatuswhiteboard**: This defines whether you wish to have a free-form, overwritable field associated with each bug. The advantage of the Status Whiteboard is that it can be deleted or modified with ease, and provides an easily-searchable field for indexing some bugs that have some trait in common.

10. **whinedays**: Set this to the number of days you want to let bugs go in the NEW or REOPENED state before notifying people they have untouched new bugs. If you do not plan to use this feature, simply do not set up the whining cron job described in the installation instructions, or set this value to "0" (never whine).

11. **commenton***: All these fields allow you to dictate what changes can pass without comment, and which must have a comment from the person who changed them. Often, administrators will allow users to add themselves to the CC list, accept bugs, or change the Status Whiteboard without adding a comment as to their reasons for the change, yet require that most other changes come with an explanation.

    Set the "commenton" options according to your site policy. It is a wise idea to require comments when users resolve, reassign, or reopen bugs at the very least.

    > **Note:** It is generally far better to require a developer comment when resolving bugs than not. Few things are more annoying to bug database users than having a developer mark a bug "fixed" without any comment as to what the fix was (or even that it was truly fixed!)

12. **supportwatchers**: Turning on this option allows users to ask to receive copies of all a particular other user's bug email. This is, of course, subject to the groupset restrictions on the bug; if the "watcher" would not normally be allowed to view a bug, the watcher cannot get around the system by setting herself up to watch the bugs of someone with bugs outside her privileges. They would still only receive email updates for those bugs she could normally view.

5.2. User Administration

5.2.1. Creating the Default User

When you first run checksetup.pl after installing Bugzilla, it will prompt you for the administrative username (email address) and password for this "super user". If for some reason you delete the "super user" account, re-running checksetup.pl will again prompt you for this username and password.

> **Tip:** If you wish to add more administrative users, you must use the MySQL interface. Run "mysql" from the command line, and use these commands:
>
> ```
> mysql> use bugs;
> mysql> update profiles set groupset=0x7fffffffffffff where login_name = "(user's login name)";
> ```
>
> Yes, that is *fourteen* "f" 's. A whole lot of f-ing going on if you want to create a new administator.

5.2.2. Managing Other Users

5.2.2.1. Creating new users

Your users can create their own user accounts by clicking the "New Account" link at the bottom of each page (assuming they aren't logged in as someone else already.) However, should you desire to create user accounts ahead of time, here is how you do it.

1. After logging in, click the "Users" link at the footer of the query page, and then click "Add a new user".
2. Fill out the form presented. This page is self-explanatory. When done, click "Submit".

Note: Adding a user this way will *not* send an email informing them of their username and password. While useful for creating dummy accounts (watchers which shuttle mail to another system, for instance, or email addresses which are a mailing list), in general it is preferable to log out and use the “New Account” button to create users, as it will pre-populate all the required fields and also notify the user of her account name and password.

5.2.2.2. Modifying Users

To see a specific user, search for their login name in the box provided on the "Edit Users" page. To see all users, leave the box blank.

You can search in different ways the listbox to the right of the text entry box. You can match by case-insensitive substring (the default), regular expression, or a *reverse* regular expression match, which finds every user name which does NOT match the regular expression. (Please see the **man regexp** manual page for details on regular expression syntax.)

Once you have found your user, you can change the following fields:

- *Login Name*: This is generally the user’s full email address. However, if you have are using the emailsuffix Param, this may just be the user’s login name. Note that users can now change their login names themselves (to any valid email address.)
- *Real Name*: The user’s real name. Note that Bugzilla does not require this to create an account.
- *Password*: You can change the user’s password here. Users can automatically request a new password, so you shouldn’t need to do this often. If you want to disable an account, see Disable Text below.
- *Disable Text*: If you type anything in this box, including just a space, the user is prevented from logging in, or making any changes to bugs via the web interface. The HTML you type in this box is presented to the user when they attempt to perform these actions, and should explain why the account was disabled.

Warning

Don’t disable the administrator account!

Note: The user can still submit bugs via the e-mail gateway, if you set it up, even if the disabled text field is filled in. The e-mail gateway should *not* be enabled for secure installations of Bugzilla.

- *<groupname>*: If you have created some groups, e.g. "securitysensitive", then checkboxes will appear here to allow you to add users to, or remove them from, these groups.
- *canconfirm*: This field is only used if you have enabled the "unconfirmed" status. If you enable this for a user, that user can then move bugs from "Unconfirmed" to a "Confirmed" status (e.g.: "New" status).
- *creategroups*: This option will allow a user to create and destroy groups in Bugzilla.
- *editbugs*: Unless a user has this bit set, they can only edit those bugs for which they are the assignee or the reporter. Even if this option is unchecked, users can still add comments to bugs.
- *editcomponents*: This flag allows a user to create new products and components, as well as modify and destroy those that have no bugs associated with them. If a product or component has bugs associated with it, those bugs must be moved to a different product or component before Bugzilla will allow them to be destroyed.
- *editkeywords*: If you use Bugzilla's keyword functionality, enabling this feature allows a user to create and destroy keywords. As always, the keywords for existing bugs containing the keyword the user wishes to destroy must be changed before Bugzilla will allow it to die.
- *editusers*: This flag allows a user to do what you're doing right now: edit other users. This will allow those with the right to do so to remove administrator privileges from other users or grant them to themselves. Enable with care.
- *tweakparams*: This flag allows a user to change Bugzilla's Params (using `editparams.cgi`.)
- *<productname>*: This allows an administrator to specify the products in which a user can see bugs. The user must still have the "editbugs" privilege to edit bugs in these products.

5.3. Product, Component, Milestone, and Version Administration

5.3.1. Products

Products are the broadest category in Bugzilla, and tend to represent real-world shipping products. E.g. if your company makes computer games, you should have one product per game, perhaps a "Common" product for units of technology used in multiple games, and maybe a few special products (Website, Administration...)

Many of Bugzilla's settings are configurable on a per-product basis. The number of "votes" available to users is set per-product, as is the number of votes required to move a bug automatically from the UNCONFIRMED status to the NEW status.

To create a new product:

1. Select "products" from the footer
2. Select the "Add" link in the bottom right
3. Enter the name of the product and a description. The Description field may contain HTML.

Don’t worry about the "Closed for bug entry", "Maximum Votes per person", "Maximum votes a person can put on a single bug", "Number of votes a bug in this Product needs to automatically get out of the UNCOMFIRMED state", and "Version" options yet. We’ll cover those in a few moments.

5.3.2. Components

Components are subsections of a Product. E.g. the computer game you are designing may have a "UI" component, an "API" component, a "Sound System" component, and a "Plugins" component, each overseen by a different programmer. It often makes sense to divide Components in Bugzilla according to the natural divisions of responsibility within your Product or company.

Each component has a owner and (if you turned it on in the parameters), a QA Contact. The owner should be the primary person who fixes bugs in that component. The QA Contact should be the person who will ensure these bugs are completely fixed. The Owner, QA Contact, and Reporter will get email when new bugs are created in this Component and when these bugs change. Default Owner and Default QA Contact fields only dictate the *default assignments*; these can be changed on bug submission, or at any later point in a bug’s life.

To create a new Component:

1. Select the "Edit components" link from the "Edit product" page
2. Select the "Add" link in the bottom right.
3. Fill out the "Component" field, a short "Description", the "Initial Owner" and "Initial QA Contact" (if enabled.) The Component and Description fields may contain HTML; the "Initial Owner" field must be a login name already existing in the database.

5.3.3. Versions

Versions are the revisions of the product, such as "Flinders 3.1", "Flinders 95", and "Flinders 2000". Version is not a multi-select field; the usual practice is to select the most recent version with the bug.

To create and edit Versions:

1. From the "Edit product" screen, select "Edit Versions"
2. You will notice that the product already has the default version "undefined". Click the "Add" link in the bottom right.

3. Enter the name of the Version. This field takes text only. Then click the "Add" button.

5.3.4. Milestones

Milestones are "targets" that you plan to get a bug fixed by. For example, you have a bug that you plan to fix for your 3.0 release, it would be assigned the milestone of 3.0.

> **Note:** Milestone options will only appear for a Product if you turned on the "usetargetmilestone" Param in the "Edit Parameters" screen.

To create new Milestones, set Default Milestones, and set Milestone URL:

1. Select "Edit milestones" from the "Edit product" page.
2. Select "Add" in the bottom right corner. text
3. Enter the name of the Milestone in the "Milestone" field. You can optionally set the "sortkey", which is a positive or negative number (-255 to 255) that defines where in the list this particular milestone appears. This is because milestones often do not occur in alphanumeric order For example, "Future" might be after "Release 1.2". Select "Add".
4. From the Edit product screen, you can enter the URL of a page which gives information about your milestones and what they mean.

> **Tip:** If you want your milestone document to be restricted so that it can only be viewed by people in a particular Bugzilla group, the best way is to attach the document to a bug in that group, and make the URL the URL of that attachment.

5.4. Voting

Voting allows users to be given a pot of votes which they can allocate to bugs, to indicate that they'd like them fixed. This allows developers to gauge user need for a particular enhancement or bugfix. By allowing bugs with a certain number of votes to automatically move from "UNCONFIRMED" to "NEW", users of the bug system can help high-priority bugs garner attention so they don't sit for a long time awaiting triage.

To modify Voting settings:

1. Navigate to the "Edit product" screen for the Product you wish to modify
2. *Maximum Votes per person*: Setting this field to "0" disables voting.
3. *Maximum Votes a person can put on a single bug"*: It should probably be some number lower than the "Maximum votes per person". Don’t set this field to "0" if "Maximum votes per person" is non-zero; that doesn’t make any sense.
4. *Number of votes a bug in this product needs to automatically get out of the UNCONFIRMED state*: Setting this field to "0" disables the automatic move of bugs from UNCONFIRMED to NEW.
5. Once you have adjusted the values to your preference, click "Update".

5.5. Groups and Group Security

Groups allow the administrator to isolate bugs or products that should only be seen by certain people. There are two types of group - Generic Groups, and Product-Based Groups.

Product-Based Groups are matched with products, and allow you to restrict access to bugs on a per-product basis. They are enabled using the usebuggroups Param. Turning on the usebuggroupsentry Param will mean bugs automatically get added to their product group when filed.

Generic Groups have no special relationship to products; you create them, and put bugs in them as required. One example of the use of Generic Groups is Mozilla’s "Security" group, into which security-sensitive bugs are placed until fixed. Only the Mozilla Security Team are members of this group.

To create Generic Groups:

1. Select the "groups" link in the footer.
2. Take a moment to understand the instructions on the "Edit Groups" screen, then select the "Add Group" link.
3. Fill out the "New Name", "New Description", and "New User RegExp" fields. "New User RegExp" allows you to automatically place all users who fulfill the Regular Expression into the new group. When you have finished, click "Add".

To use Product-Based Groups:

1. Turn on "usebuggroups" and "usebuggroupsentry" in the "Edit Parameters" screen.

> **Warning**
>
> XXX is this still true? "usebuggroupsentry" has the capacity to prevent the administrative user from directly altering bugs because of conflicting group permissions. If you plan on using "usebuggroupsentry", you should plan on restricting administrative account usage to administrative duties only. In other words, manage bugs with an unpriveleged user account, and manage users, groups, Products, etc. with the administrative account.

2. In future, when you create a Product, a matching group will be automatically created. If you need to add a Product Group to a Product which was created before you turned on usebuggroups, then simply create a new group, as outlined above, with the same name as the Product.

> **Warning**
>
> Bugzilla currently has a limit of 64 groups per installation. If you have more than about 50 products, you should consider running multiple Bugzillas. Ask in the newsgroup for other suggestions for working around this restriction.

Note that group permissions are such that you need to be a member of *all* the groups a bug is in, for whatever reason, to see that bug.

5.6. Bugzilla Security

> **Warning**
>
> Poorly-configured MySQL and Bugzilla installations have given attackers full access to systems in the past. Please take these guidelines seriously, even for Bugzilla machines hidden away behind your firewall. 80% of all computer trespassers are insiders, not anonymous crackers.

Note: These instructions must, of necessity, be somewhat vague since Bugzilla runs on so many different platforms. If you have refinements of these directions, please submit a bug to Bugzilla (http://bugzilla.mozilla.org/enter_bug.cgi?product=Bugzilla&component=Documentation).

> **Warning**
>
> This is not meant to be a comprehensive list of every possible security issue regarding the tools mentioned in this section. There is no subsitute for reading the information written by the authors of any software running on your system.

5.6.1. TCP/IP Ports

TCP/IP defines 65,000 some ports for trafic. Of those, Bugzilla only needs 1... 2 if you need to use features that require e-mail such as bug moving or the e-mail interface from contrib. You should audit your server and make sure that you aren't listening on any ports you don't need to be. You may also wish to use some kind of firewall software to be sure that trafic can only be recieved on ports you specify.

5.6.2. MySQL

MySQL ships by default with many settings that should be changed. By defaults it allows anybody to connect from localhost without a password and have full administrative capabilities. It also defaults to not have a root password (this is *not* the same as the system root). Also, many installations default to running mysqld as the system root.

1. Make sure you are running at least version 3.22.32 of MySQL as earlier versions had notable security holes.
2. Consult the documentation that came with your system for information on making mysqld run as an unprivleged user.
3. You should also be sure to disable the anonymous user account and set a password for the root user. This is accomplished using the following commands:

   ```
   bash$ mysql mysql
   mysql> DELETE FROM user WHERE user = '';
   mysql> UPDATE user SET password = password('new_password') WHERE user = 'root';
   mysql> FLUSH PRIVILEGES;
   ```

   From this point forward you will need to use **mysql -u root -p** and enter *`new_password`* when prompted when using the mysql client.

4. If you run MySQL on the same machine as your httpd server, you should consider disabling networking from within MySQL by adding the following to your `/etc/my.conf`:

   ```
   [myslqd]
   # Prevent network access to MySQL.
   ```

```
skip-networking
```

5. You may also consider running MySQL, or even all of Bugzilla in a chroot jail; however, instructions for doing that are beyond the scope of this document.

5.6.3. Daemon Accounts

Many daemons, such as Apache's httpd and MySQL's mysqld default to running as either "root" or "nobody". Running as "root" introduces obvious security problems, but the problems introduced by running everything as "nobody" may not be so obvious. Basically, if you're running every daemon as "nobody" and one of them gets comprimised, they all get comprimised. For this reason it is recommended that you create a user account for each daemon.

> **Note:** You will need to set the `webservergroup` to the group you created for your webserver to run as in `localconfig`. This will allow **./checksetup.pl** to better adjust the file permissions on your Bugzilla install so as to not require making anything world-writable.

5.6.4. Web Server Access Controls

There are many files that are placed in the Bugzilla directory area that should not be accessable from the web. Because of the way Bugzilla is currently layed out, the list of what should and should not be accessible is rather complicated. A new installation method is currently in the works which should solve this by allowing files that shouldn't be accessible from the web to be placed in directory outside the webroot. See bug 44659 (http://bugzilla.mozilla.org/show_bug.cgi?id=44659) for more information.

- In the main Bugzilla directory, you should:
  - Block: `*.pl`, `*localconfig*`, `runtests.sh`, `processmail`, `syncshadowdb`
  - But allow: `localconfig.js`, `localconfig.rdf`

- In `data`:
  - Block everything
  - But allow: `duplicates.rdf`

- In `data/webdot`:

  - If you use a remote webdot server:
    - Block everything
    - But allow `*.dot` only for the remote webdot server

  - Otherwise, if you use a local GraphViz:
    - Block everything
    - But allow: `*.png`, `*.gif`, `*.jpg`, `*.map`

  - And if you don't use any dot:
    - Block everything

- In `Bugzilla`:
  - Block everything

- In `template`:
  - Block everything

> **Tip:** Bugzilla ships with the ability to generate `.htaccess` files instructing Apache which files should and should not be accessible.

You should test to make sure that the files mentioned above are not accessible from the Internet, especially your `localconfig` file which contains your database password. To test, simply point your web browser at the file; for example, to test mozilla.org's installation, we'd try to access http://bugzilla.mozilla.org/localconfig. You should get a 403 Forbidden error.

> **Caution**
>
> Not following the instructions in this section, including testing, may result in sensitive information being globally accessible.

5.7. Template Customisation

One of the large changes for 2.16 was the templatisation of the entire user-facing UI, using the Template Toolkit (http://www.template-toolkit.org). Administrators can now configure the look and feel of Bugzilla without having to edit Perl files or face the nightmare of massive merge conflicts when they upgrade to a newer version in the future.

Templatisation also makes localised versions of Bugzilla possible, for the first time. In the future, a Bugzilla installation may have templates installed for multiple localisations, and select which ones to use based on the user's browser language setting.

5.7.1. What to Edit

There are two different ways of editing of Bugzilla's templates, and which you use depends mainly on how you upgrade Bugzilla. The template directory structure is that there's a top level directory, `template`, which contains a directory for each installed localisation. The default English templates are therefore in `en`. Underneath that, there is the `default` directory and optionally the `custom` directory. The `default` directory contains all the templates shipped with Bugzilla, whereas the `custom` directory does not exist at first and must be created if you want to use it.

The first method of making customisations is to directly edit the templates in `template/en/default`. This is probably the best method for small changes if you are going to use the CVS method of upgrading, because if you then execute a **cvs update**, any template fixes will get automagically merged into your modified versions.

If you use this method, your installation will break if CVS conflicts occur.

The other method is to copy the templates into a mirrored directory structure under `template/en/custom`. The templates in this directory automatically override those in default. This is the technique you need to use if you use the overwriting method of upgrade, because otherwise your changes will be lost. This method is also better if you are using the CVS method of upgrading and are going to make major changes, because it is guaranteed that the contents of this directory will not be touched during an upgrade, and you can then decide whether to continue using your own templates, or make the effort to merge your changes into the new versions by hand.

If you use this method, your installation may break if incompatible changes are made to the template interface. If such changes are made they will be documented in the release notes, provided you are using a stable release of Bugzilla. If you use using unstable code, you will need to deal with this one yourself, although if possible the changes will be mentioned before they occur in the deprecations section of the previous stable release's release notes.

> **Note:** Don't directly edit the compiled templates in `data/template/*` - your changes will be lost when Template Toolkit recompiles them.

> **Note:** It is recommended that you run **./checksetup.pl** after any template edits, especially if you've created a new file in the `custom` directory.

5.7.2. How To Edit Templates

The syntax of the Template Toolkit language is beyond the scope of this guide. It's reasonably easy to pick up by looking at the current templates; or, you can read the manual, available on the Template Toolkit home page (http://www.template-toolkit.org). However, you should particularly remember (for security reasons) to always HTML filter things which come from the database or user input, to prevent cross-site scripting attacks.

However, one thing you should take particular care about is the need to properly HTML filter data that has been passed into the template. This means that if the data can possibly contain special HTML characters such as <, and the data was not intended to be HTML, they need to be converted to entity form, ie <. You use the 'html' filter in the Template Toolkit to do this. If you fail to do this, you may open up your installation to cross-site scripting attacks.

Also note that Bugzilla adds a few filters of its own, that are not in standard Template Toolkit. In particular, the 'url_quote' filter can convert characters that are illegal or have special meaning in URLs, such as &, to the encoded form, ie %26. This actually encodes most characters (but not the common ones such as letters and numbers and so on), including the HTML-special characters, so there's never a need to HTML filter afterwards.

Editing templates is a good way of doing a "poor man's custom fields". For example, if you don't use the Status Whiteboard, but want to have a free-form text entry box for "Build Identifier", then you can just edit the templates to change the field labels. It's still be called status_whiteboard internally, but your users don't need to know that.

> **Note:** If you are making template changes that you intend on submitting back for inclusion in standard Bugzilla, you should read the relevant sections of the Developers' Guide (http://www.bugzilla.org/developerguide.html).

5.7.3. Template Formats

Some CGIs have the ability to use more than one template. For example, buglist.cgi can output bug lists as RDF or two different forms of HTML (complex and simple). (Try this out by appending `&format=simple` to a buglist.cgi URL on your Bugzilla installation.) This mechanism, called template 'formats', is extensible.

To see if a CGI supports multiple output formats, grep the CGI for "ValidateOutputFormat". If it's not present, adding multiple format support isn't too hard - see how it's done in other CGIs.

To make a new format template for a CGI which supports this, open a current template for that CGI and take note of the INTERFACE comment (if present.) This comment defines what variables are passed into this template. If there isn't one, I'm afraid you'll have to read the template and the code to find out what information you get.

Write your template in whatever markup or text style is appropriate.

You now need to decide what content type you want your template served as. Open up the `localconfig` file and find the `$contenttypes` variable. If your content type is not there, add it. Remember the three- or four-letter tag assigned to you content type. This tag will be part of the template filename.

Save the template as `<stubname>-<formatname>.<contenttypetag>.tmpl`. Try out the template by calling the CGI as `<cginame>.cgi?format=<formatname>.`

5.7.4. Particular Templates

There are a few templates you may be particularly interested in customising for your installation.

index.html.tmpl: This is the Bugzilla front page.

global/header.html.tmpl: This defines the header that goes on all Bugzilla pages. The header includes the banner, which is what appears to users and is probably what you want to edit instead. However the header also includes the HTML HEAD section, so you could for example add a stylesheet or META tag by editing the header.

global/banner.html.tmpl: This contains the "banner", the part of the header that appears at the top of all Bugzilla pages. The default banner is reasonably barren, so you'll probably want to customise this to give your installation a distinctive look and feel. It is recommended you preserve the Bugzilla version number in some form so the version you are running can be determined, and users know what docs to read.

global/footer.html.tmpl: This defines the footer that goes on all Bugzilla pages. Editing this is another way to quickly get a distinctive look and feel for your Bugzilla installation.

bug/create/user-message.html.tmpl: This is a message that appears near the top of the bug reporting page. By modifying this, you can tell your users how they should report bugs.

bug/create/create.html.tmpl and **bug/create/comment.txt.tmpl**: You may wish to get bug submitters to give certain bits of structured information, each in a separate input widget, for which there is not a field in the database. The bug entry system has been designed in an extensible fashion to enable you to define arbitrary fields and widgets, and have their values appear formatted in the initial Description, rather than in database fields. An example of this is the mozilla.org guided bug submission form (http://bugzilla.mozilla.org/enter_bug.cgi?format=guided).

To make this work, create a custom template for `enter_bug.cgi` (the default template, on which you could base it, is `create.html.tmpl`), and either call it `create.html.tmpl` or use a format and call it `create-<formatname>.html.tmpl`. Put it in the `custom/bug/create` directory. In it, add widgets for each piece of information you'd like collected - such as a build number, or set of steps to reproduce.

Then, create a template like `custom/bug/create/comment.txt.tmpl`, also named after your format if you are using one, which references the form fields you have created. When a bug report is submitted, the initial comment attached to the bug report will be formatted according to the layout of this template.

For example, if your enter_bug template had a field

```
<input type="text" name="buildid" size="30">
```

and then your comment.txt.tmpl had

```
BuildID: [% form.buildid %]
```

then

```
BuildID: 20020303
```

would appear in the initial checkin comment.

5.8. Upgrading to New Releases

Upgrading Bugzilla is something we all want to do from time to time, be it to get new features or pick up the latest security fix. How easy it is to update depends on a few factors.

- If the new version is a revision or a new point release
- How many, if any, local changes have been made

There are also three different methods to upgrade your installation.

1. Using CVS (Example 5-1)
2. Downloading a new tarball (Example 5-2)
3. Applying the relevant patches (Example 5-3)

Which options are available to you may depend on how large a jump you are making and/or your network configuration.

Revisions are normally released to fix security vulnerabilities and are distinguished by an increase in the third number. For example, when 2.16.2 was released, it was a revision to 2.16.1.

Point releases are normally released when the Bugzilla team feels that there has been a significant amount of progress made between the last point release and the current time. These are often proceeded by a stabilization period and release candidates, however the use of development versions or release candidates is beyond the scope of this document. Point releases can be distinguished by an increase in the second number, or minor version. For example, 2.16.2 is a newer point release than 2.14.5.

The examples in this section are written as if you were updating to version 2.16.2. The procedures are the same regardless if you are updating to a new point release or a new revision. However, the chance of running into trouble increases when upgrading to a new point release, escpecially if you’ve made local changes.

These examples also assume that your Bugzilla installation is at `/var/www/html/bugzilla`. If that is not the case, simply substitute the proper paths where appropriate.

Example 5-1. Upgrading using CVS

Every release of Bugzilla, whether it is a revision or a point release, is tagged in CVS. Also, every tarball we have distributed since version 2.12 has been primed for using CVS. This does, however, require that you are able to access cvs-mirror.mozilla.org on port 2401.

> **Tip:** If you can do this, updating using CVS is probably the most painless method, especially if you have a lot of local changes.

```
bash$ cd /var/www/html/bugzilla
bash$ cvs login
Logging in to :pserver:anonymous@cvs-mirror.mozilla.org:2401/cvsroot
CVS password: anonymous
bash$ cvs -q update -r BUGZILLA-2_16_2 -dP
P checksetup.pl
P collectstats.pl
P globals.pl
P docs/rel_notes.txt
P template/en/default/list/quips.html.tmpl
```

> **Caution**
>
> If a line in the output from **cvs update** begins with a `C` that represents a file with local changes that CVS was unable to properly merge. You need to resolve these conflicts manually before Bugzilla (or at least the portion using that file) will be usable.

> **Note:** You also need to run **./checksetup.pl** before your Bugzilla upgrade will be complete.

Example 5-2. Upgrading using the tarball

If you are unable or unwilling to use CVS, another option that's always available is to download the latest tarball. This is the most difficult option to use, especially if you have local changes.

```
bash$ cd /var/www/html
bash$ wget ftp://ftp.mozilla.org/pub/webtools/bugzilla-2.16.2.tar.gz
Output omitted
bash$ tar xzvf bugzilla-2.16.2.tar.gz
bugzilla-2.16.2/
bugzilla-2.16.2/.cvsignore
bugzilla-2.16.2/1x1.gif
Output truncated
bash$ cd bugzilla-2.16.2
bash$ cp ../bugzilla/localconfig* .
bash$ cp -r ../bugzilla/data .
bash$ cd ..
bash$ mv bugzilla bugzilla.old
bash$ mv bugzilla-2.16.2 bugzilla
bash$ cd bugzilla
bash$ ./checksetup.pl
Output omitted
```

Warning

The **cp** commands both end with periods which is a very important detail, it tells the shell that the destination directory is the current working directory. Also, the period at the beginning of the **./checksetup.pl** is important and can not be omitted.

Note: You will now have to reapply any changes you have made to your local installation manually.

Example 5-3. Upgrading using patches

The Bugzilla team will normally make a patch file available for revisions to go from the most recent revision to the new one. You could also read the release notes and grab the patches attached to the mentioned bug, but it is safer to use the released patch file as sometimes patches get changed before they get checked in (for minor spelling fixes and

the like). It is also theorectically possible to scour the fixed bug list and pick and choose which patches to apply from a point release, but this is not recommended either as what you'll end up with is a hodge podge Bugzilla that isn't really any version. This would also make it more difficult to upgrade in the future.

```
bash$ cd /var/www/html/bugzilla
bash$ wget ftp://ftp.mozilla.org/pub/webtools/bugzilla-2.16.1-to-2.16.2.diff.gz
Output omitted
bash$ gunzip bugzilla-2.16.1-to-2.16.2.diff.gz
bash$ patch -p1 < bugzilla-2.16.1-to-2.16.2.diff
patching file checksetup.pl
patching file collectstats.pl
patching file globals.pl
```

Caution

If you do this, beware that this doesn't change the entires in your CVS directory so it may make updates using CVS (Example 5-1) more difficult in the future.

5.9. Integrating Bugzilla with Third-Party Tools

5.9.1. Bonsai

Bonsai is a web-based tool for managing CVS, the Concurrent Versioning System . Using Bonsai, administrators can control open/closed status of trees, query a fast relational database back-end for change, branch, and comment information, and view changes made since the last time the tree was closed. Bonsai also integrates with Tinderbox, the Mozilla automated build management system.

5.9.2. CVS

CVS integration is best accomplished, at this point, using the Bugzilla Email Gateway.

Follow the instructions in this Guide for enabling Bugzilla e-mail integration. Ensure that your check-in script sends an email to your Bugzilla e-mail gateway with the subject of "[Bug XXXX]", and you can have CVS check-in

comments append to your Bugzilla bug. If you want to have the bug be closed automatically, you'll have to modify the `contrib/bugzilla_email_append.pl` script.

There is also a CVSZilla project, based upon somewhat dated Bugzilla code, to integrate CVS and Bugzilla through CVS' ability to email. Check it out at: http://homepages.kcbbs.gen.nz/~tonyg/ (http://homepages.kcbbs.gen.nz/~tonyg/).

5.9.3. Perforce SCM

You can find the project page for Bugzilla and Teamtrack Perforce integration (p4dti) at: http://www.ravenbrook.com/project/p4dti (http://www.ravenbrook.com/project/p4dti/) . "p4dti" is now an officially supported product from Perforce, and you can find the "Perforce Public Depot" p4dti page at http://public.perforce.com/public/perforce/p4dti/index.html (http://public.perforce.com/public/perforce/p4dti/index.html) .

Integration of Perforce with Bugzilla, once patches are applied, is seamless. Perforce replication information will appear below the comments of each bug. Be certain you have a matching set of patches for the Bugzilla version you are installing. p4dti is designed to support multiple defect trackers, and maintains its own documentation for it. Please consult the pages linked above for further information.

5.9.4. Tinderbox/Tinderbox2

We need Tinderbox integration information.

Appendix A. The Bugzilla FAQ

This FAQ includes questions not covered elsewhere in the Guide.

1. General Questions

1.1. Where can I find information about Bugzilla?

You can stay up-to-date with the latest Bugzilla information at http://www.bugzilla.org/ (http://www.bugzilla.org/)

1.2. What license is Bugzilla distributed under?

Bugzilla is covered by the Mozilla Public License. See details at http://www.mozilla.org/MPL/ (http://www.mozilla.org/MPL/)

1.3. How do I get commercial support for Bugzilla?

http://bugzilla.org/consulting.html is a list of people and companies who have asked us to list them as consultants for Bugzilla.

www.collab.net (http://www.collab.net/) offers Bugzilla as part of their standard offering to large projects. They do have some minimum fees that are pretty hefty, and generally aren't interested in small projects.

There are several experienced Bugzilla hackers on the mailing list/newsgroup who are willing to make themselves available for generous compensation. Try sending a message to the mailing list asking for a volunteer.

1.4. What major companies or projects are currently using Bugzilla for bug-tracking?

There are *dozens* of major companies with public Bugzilla sites to track bugs in their products. A few include:

Netscape/AOL
Mozilla.org
NASA
Red Hat Software
SuSe Corp
The Horde Project
AbiSource
Real Time Enterprises, Inc
Eggheads.org
Strata Software
RockLinux
Creative Labs (makers of SoundBlaster)
The Apache Foundation

The Gnome Foundation
Ximian
Linux-Mandrake

Suffice to say, there are more than enough huge projects using Bugzilla that we can safely say it's extremely popular.

1.5. Who maintains Bugzilla?

A core team (http://www.bugzilla.org/who_we_are.html), led by Dave Miller (justdave@bugzilla.org).

1.6. How does Bugzilla stack up against other bug-tracking databases?

We can't find any head-to-head comparisons of Bugzilla against other defect-tracking software. If you know of one, please get in touch. However, from the author's personal experience with other bug-trackers, Bugzilla offers superior performance on commodity hardware, better price (free!), more developer- friendly features (such as stored queries, email integration, and platform independence), improved scalability, open source code, greater flexibility, and superior ease-of-use.

If you happen to be a commercial bug-tracker vendor, please step forward with a list of advantages your product has over Bugzilla. We'd be happy to include it in the "Competitors" section.

1.7. Why doesn't Bugzilla offer this or that feature or compatibility with this other tracking software?

It may be that the support has not been built yet, or that you have not yet found it. Bugzilla is making tremendous strides in usability, customizability, scalability, and user interface. It is widely considered the most complete and popular open-source bug-tracking software in existence.

That doesn't mean it can't use improvement! You can help the project along by either hacking a patch yourself that supports the functionality you require, or else submitting a "Request for Enhancement" (RFE) using the bug submission interface at bugzilla.mozilla.org (http://bugzilla.mozilla.org/enter_bug.cgi?product=Bugzilla).

1.8. Why MySQL? I'm interested in seeing Bugzilla run on Oracle/Sybase/Msql/PostgreSQL/MSSQL.

MySQL was originally chosen because it is free, easy to install, and was available for the hardware Netscape intended to run it on.

There is currently work in progress to make Bugzilla work on PostgreSQL and Sybase in the default distribution. You can track the progress of these initiatives in bugs 98304 (http://bugzilla.mozilla.org/show_bug.cgi?id=98304) and 173130 (http://bugzilla.mozilla.org/show_bug.cgi?id=173130) respectively.

Once both of these are done, adding support for additional database servers should be trivial.

1.9. Why do the scripts say `/usr/bonsaitools/bin/perl` instead of `/usr/bin/perl` or something else?

Mozilla.org used `/usr/bonsaitools/bin/perl`, because originally Terry wanted a place to put a version of Perl and other tools that was strictly under his control.

> **Note:** This convention was abonded during the 2.17 development cycle so it will no longer be an issue when 2.18 comes out.

1.10. Is there an easy way to change the Bugzilla cookie name?

At present, no.

2. Managerial Questions

> **Note:** Questions likely to be asked by managers. :-)

2.1. Is Bugzilla web-based, or do you have to have specific software or a specific operating system on your machine?

It is web and e-mail based. You can edit bugs by sending specially formatted email to a properly configured Bugzilla, or control via the web.

2.2. Can Bugzilla integrate with Perforce (SCM software)?

Yes! You can find more information elsewhere in "The Bugzilla Guide" in the "Integration with Third-Party Products" section.

2.3. Does Bugzilla allow the user to track multiple projects?

Absolutely! You can track any number of Products that can each be composed of any number of Components.

> **Note:** There are only 55 groups available in version 2.16 of Bugzilla. If you are using product groups, this will also limit the number of products you can have. This limit does not exist in the current 2.17 development releases and will not exist in 2.18.

2.4. If I am on many projects, and search for all bugs assigned to me, will Bugzilla list them for me and allow me to sort by project, severity etc?

Yes.

2.5. Does Bugzilla allow attachments (text, screenshots, URLs etc)? If yes, are there any that are NOT allowed?

Yes - any sort of attachment is allowed, although administrators can configure a maximum size. Bugzilla gives the user the option of either using the MIME-type supplied by the browser, choosing from a pre-defined list or manually typing any arbitrary MIME-type.

2.6. Does Bugzilla allow us to define our own priorities and levels? Do we have complete freedom to change the labels of fields and format of them, and the choice of acceptable values?

Yes. However, modifying some fields, notably those related to bug progression states, also require adjusting the program logic to compensate for the change.

There is no GUI for adding fields to Bugzilla at this time. You can follow development of this feature at http://bugzilla.mozilla.org/show_bug.cgi?id=91037.

2.7. Does Bugzilla provide any reporting features, metrics, graphs, etc? You know, the type of stuff that management likes to see. :)

Yes. Look at http://bugzilla.mozilla.org/reports.cgi for samples of what Bugzilla can do in reporting and graphing.

If you can not get the reports you want from the included reporting scripts, it is possible to hook up a professional reporting package such as Crystal Reports using ODBC. If you choose to do this, beware that giving direct access to the database does contain some security implications. Even if you give read-only access to the bugs database it will bypass the secure bugs features of Bugzilla.

> **Note:** Bugzilla's current development versions can do a lot more in the way of reporting. To see examples, check out http://bugzilla.mozilla.org/report.cgi.

2.8. Is there email notification and if so, what do you see when you get an email?

Email notification is user-configurable. By default, the bug id and Summary of the bug report accompany each email notification, along with a list of the changes made.

2.9. Can email notification be set up to send to multiple people, some on the To List, CC List, BCC List etc?

Yes.

2.10. Do users have to have any particular type of email application?

Bugzilla email is sent in plain text, the most compatible mail format on the planet.

> **Note:** If you decide to use the bugzilla_email integration features to allow Bugzilla to record responses to mail with the associated bug, you may need to caution your users to set their mailer to "respond to messages in the format in which they were sent". For security reasons Bugzilla ignores HTML tags in comments, and if a user sends HTML-based email into Bugzilla the resulting comment looks downright awful.

2.11. Does Bugzilla allow data to be imported and exported? If I had outsiders write up a bug report using a MS Word bug template, could that template be imported into "matching" fields? If I wanted to take the results of a query and export that data to MS Excel, could I do that?

Bugzilla can only output buglists as HTML in version 2.16. There are other formats available (CSV and RDF) in the newer development versions.

Bugzilla can export bugs using `xml.cgi` with either a bug number or list of bug numbers.

Currently the only script included with Bugzilla that can import data is `importxml.pl` which is intended to be used for importing the data generated by `xml.cgi` in association with bug moving. Any other use is left as an exercise for the user.

There are also scripts included in the `contrib/` directory for using e-mail to import information into Bugzilla, but these scripts are not currently supported and included for educational purposes.

2.12. Has anyone converted Bugzilla to another language to be used in other countries? Is it localizable?

Yes. For more information including available translated templates, see http://www.bugzilla.org/download.html#localizations. The admin interfaces are still not included in these translated templates and is therefore still English only. Also, there may be issues with the charset not being declared. See bug 126226 (http://bugzilla.mozilla.org/show_bug.cgi?id=126266) for more information.

2.13. Can a user create and save reports? Can they do this in Word format? Excel format?

Yes. No. Not in 2.16.

2.14. Does Bugzilla have the ability to search by word, phrase, compound search?

You have no idea. Bugzilla's query interface, particularly with the advanced Boolean operators, is incredibly versatile.

2.15. Does Bugzilla provide record locking when there is simultaneous access to the same bug? Does the second person get a notice that the bug is in use or how are they notified?

Bugzilla does not lock records. It provides mid-air collision detection, and offers the offending user a choice of options to deal with the conflict.

2.16. Are there any backup features provided?

MySQL, the database back-end for Bugzilla, allows hot-backup of data. You can find strategies for dealing with backup considerations at http://www.mysql.com/doc/B/a/Backup.html (http://www.mysql.com/doc/B/a/Backup.html)

2.17. Can users be on the system while a backup is in progress?

Yes. However, commits to the database must wait until the tables are unlocked. Bugzilla databases are typically very small, and backups routinely take less than a minute.

2.18. What type of human resources are needed to be on staff to install and maintain Bugzilla? Specifically, what type of skills does the person need to have? I need to find out if we were to go with Bugzilla, what types of individuals would we need to hire and how much would that cost vs buying an "Out-of-the-Box" solution.

If Bugzilla is set up correctly from the start, continuing maintenance needs are minimal and can be done easily using the web interface.

Commercial Bug-tracking software typically costs somewhere upwards of $20,000 or more for 5-10 floating licenses. Bugzilla consultation is available from skilled members of the newsgroup. Simple questions are answered there and then.

2.19. What time frame are we looking at if we decide to hire people to install and maintain the Bugzilla? Is this something that takes hours or weeks to install and a couple of hours per week to maintain and customize or is this a multi-week install process, plus a full time job for 1 person, 2 people, etc?

It all depends on your level of commitment. Someone with much Bugzilla experience can get you up and running in less than a day, and your Bugzilla install can run untended for years. If your Bugzilla strategy is critical to your business workflow, hire somebody with reasonable UNIX or Perl skills to handle your process management and bug-tracking maintenance & customization.

2.20. Is there any licensing fee or other fees for using Bugzilla? Any out-of-pocket cost other than the bodies needed as identified above?

No. MySQL asks, if you find their product valuable, that you purchase a support contract from them that suits your needs.

3. Bugzilla Security

3.1. How do I completely disable MySQL security if it's giving me problems (I've followed the instructions in the installation section of this guide)?

Run MySQL like this: "mysqld --skip-grant-tables". Please remember *this makes MySQL as secure as taping a $100 to the floor of a football stadium bathroom for safekeeping.*

3.2. Are there any security problems with Bugzilla?

The Bugzilla code has undergone a reasonably complete security audit, and user-facing CGIs run under Perl's taint mode. However, it is recommended that you closely examine permissions on your Bugzilla installation, and follow the recommended security guidelines found in The Bugzilla Guide.

3.3. I've implemented the security fixes mentioned in Chris Yeh's security advisory of 5/10/2000 advising not to run MySQL as root, and am running into problems with MySQL no longer working correctly.

This is a common problem, related to running out of file descriptors. Simply add "ulimit -n unlimited" to the script which starts mysqld.

4. Bugzilla Email

4.1. I have a user who doesn't want to receive any more email from Bugzilla. How do I stop it entirely for this user?

The user should be able to set this in user email preferences (uncheck all boxes) or you can add their email address to the `data/nomail` file.

4.2. I'm evaluating/testing Bugzilla, and don't want it to send email to anyone but me. How do I do it?

Edit the "newchangedmail" Param. Replace "To:" with "X-Real-To:", replace "Cc:" with "X-Real-CC:", and add a "To: <youremailaddress>".

4.3. I want whineatnews.pl to whine at something more, or other than, only new bugs. How do I do it?

Try Klaas Freitag's excellent patch for "whineatassigned" functionality. You can find it at http://bugzilla.mozilla.org/show_bug.cgi?id=6679. This patch is against an older version of Bugzilla, so you must apply the diffs manually.

4.4. I don’t like/want to use Procmail to hand mail off to bug_email.pl. What alternatives do I have?

You can call bug_email.pl directly from your aliases file, with an entry like this:

```
bugzilla-daemon: "|/usr/local/bin/bugzilla/contrib/bug_email.pl"
```

However, this is fairly nasty and subject to problems; you also need to set up your smrsh (sendmail restricted shell) to allow it. In a pinch, though, it can work.

4.5. How do I set up the email interface to submit/change bugs via email?

You can find an updated README.mailif file in the contrib/ directory of your Bugzilla distribution that walks you through the setup.

4.6. Email takes FOREVER to reach me from Bugzilla -- it’s extremely slow. What gives?

If you are using an alternate Mail Transport Agent (MTA other than sendmail), make sure the options given in the "processmail" and other scripts for all instances of "sendmail" are correct for your MTA.

If you are using Sendmail, try enabling "sendmailnow" in editparams.cgi. If you are using Postfix, you will also need to enable “sendmailnow”.

4.7. How come email from Bugzilla changes never reaches me?

Double-check that you have not turned off email in your user preferences. Confirm that Bugzilla is able to send email by visiting the "Log In" link of your Bugzilla installation and clicking the "Email me a password" button after entering your email address.

If you never receive mail from Bugzilla, chances you do not have sendmail in "/usr/lib/sendmail". Ensure sendmail lives in, or is symlinked to, "/usr/lib/sendmail".

5. Bugzilla Database

5.1. I’ve heard Bugzilla can be used with Oracle?

Red Hat’s old version of Bugzilla (based on 2.8) worked on Oracle. Red Hat’s newer version (based on 2.17.1 and soon to be merged into the main distribution) runs on PostgreSQL. At this time we know of no recent ports of Bugzilla to Oracle but do intend to support it in the future (possibly the 2.20 time-frame).

5.2. I think my database might be corrupted, or contain invalid entries. What do I do?

Run the “sanity check” utility (`./sanitycheck.cgi` in the Bugzilla_home directory) from your web browser to see! If it finishes without errors, you’re *probably* OK. If it doesn’t come back OK (i.e. any red letters), there are certain things Bugzilla can recover from and certain things it can’t. If it can’t auto-recover, I hope you’re

familiar with mysqladmin commands or have installed another way to manage your database. Sanity Check, although it is a good basic check on your database integrity, by no means is a substitute for competent database administration and avoiding deletion of data. It is not exhaustive, and was created to do a basic check for the most common problems in Bugzilla databases.

5.3. I want to manually edit some entries in my database. How?

There is no facility in Bugzilla itself to do this. It's also generally not a smart thing to do if you don't know exactly what you're doing. However, if you understand SQL you can use the **mysql** command line utility to manually insert, delete and modify table information. There are also more intuitive GUI clients available. Personal favorites of the Bugzilla team are phpMyAdmin (http://www.phpmyadmin.net/) and MySQL Control Center (http://www.mysql.com/downloads/gui-mycc.html).

5.4. I think I've set up MySQL permissions correctly, but Bugzilla still can't connect.

Try running MySQL from its binary: "mysqld --skip-grant-tables". This will allow you to completely rule out grant tables as the cause of your frustration. If this Bugzilla is able to connect at this point then you need to check that you have granted proper permission to the user password combo defined in `localconfig`.

> **Warning**
>
> Running MySQL with this command line option is very insecure and should only be done when not connected to the external network as a troubleshooting step.

5.5. How do I synchronize bug information among multiple different Bugzilla databases?

Well, you can synchronize or you can move bugs. Synchronization will only work one way -- you can create a read-only copy of the database at one site, and have it regularly updated at intervals from the main database.

MySQL has some synchronization features builtin to the latest releases. It would be great if someone looked into the possibilities there and provided a report to the newsgroup on how to effectively synchronize two Bugzilla installations.

If you simply need to transfer bugs from one Bugzilla to another, checkout the "move.pl" script in the Bugzilla distribution.

6. Bugzilla and Win32

6.1. What is the easiest way to run Bugzilla on Win32 (Win98+/NT/2K)?

Remove Windows. Install Linux. Install Bugzilla. The boss will never know the difference.

6.2. Is there a "Bundle::Bugzilla" equivalent for Win32?

Not currently. Bundle::Bugzilla enormously simplifies Bugzilla installation on UNIX systems. If someone can volunteer to create a suitable PPM bundle for Win32, it would be appreciated.

6.3. CGI's are failing with a "something.cgi is not a valid Windows NT application" error. Why?

Depending on what Web server you are using, you will have to configure the Web server to treat *.cgi files as CGI scripts. In IIS, you do this by adding *.cgi to the App Mappings with the <path>\perl.exe %s %s as the executable.

Microsoft has some advice on this matter, as well:

> "Set application mappings. In the ISM, map the extension for the script file(s) to the executable for the script interpreter. For example, you might map the extension .py to Python.exe, the executable for the Python script interpreter. Note For the ActiveState Perl script interpreter, the extension .pl is associated with PerlIS.dll by default. If you want to change the association of .pl to perl.exe, you need to change the application mapping. In the mapping, you must add two percent (%) characters to the end of the pathname for perl.exe, as shown in this example: c:\perl\bin\perl.exe %s %s"

6.4. I'm having trouble with the perl modules for NT not being able to talk to to the database.

Your modules may be outdated or inaccurate. Try:

1. Hitting http://www.activestate.com/ActivePerl
2. Download ActivePerl
3. Go to your prompt
4. Type 'ppm'
5. `PPM>` **install DBI DBD-mysql GD**

I reckon TimeDate and Data::Dumper come with the activeperl. You can check the ActiveState site for packages for installation through PPM. http://www.activestate.com/Packages/ (http://www.activestate.com/Packages/)

7. Bugzilla Usage

7.1. How do I change my user name (email address) in Bugzilla?

New in 2.16 - go to the Account section of the Preferences. You will be emailed at both addresses for confirmation.

7.2. The query page is very confusing. Isn't there a simpler way to query?

The interface was simplified by a UI designer for 2.16. Further suggestions for improvement are welcome, but we won't sacrifice power for simplicity.

7.3. I'm confused by the behavior of the "accept" button in the Show Bug form. Why doesn't it assign the bug to me when I accept it?

The current behavior is acceptable to bugzilla.mozilla.org and most users. You have your choice of patches to change this behavior, however.

Add a "and accept bug" radio button (http://bugzilla.mozilla.org/showattachment.cgi?attach_id=8029)
"Accept" button automatically assigns to you (http://bugzilla.mozilla.org/showattachment.cgi?attach_id=8153)

Note that these patches are somewhat dated. You will need to apply them manually.

7.4. I can't upload anything into the database via the "Create Attachment" link. What am I doing wrong?

The most likely cause is a very old browser or a browser that is incompatible with file upload via POST. Download the latest Netscape, Microsoft, or Mozilla browser to handle uploads correctly.

7.5. How do I change a keyword in Bugzilla, once some bugs are using it?

In the Bugzilla administrator UI, edit the keyword and it will let you replace the old keyword name with a new one. This will cause a problem with the keyword cache. Run sanitycheck.cgi to fix it.

7.6. Why can't I close bugs from the "Change Several Bugs at Once" page?

The logic flow currently used is RESOLVED, then VERIFIED, then CLOSED. You *can* mass-CLOSE bugs from the change several bugs at once page. *but*, every bug listed on the page has to be in VERIFIED state before the control to do it will show up on the form. You can also mass-VERIFY, but every bug listed has to be RESOLVED in order for the control to show up on the form. The logic behind this is that if you pick one of the bugs that's not VERIFIED and try to CLOSE it, the bug change will fail miserably (thus killing any changes in the list after it while doing the bulk change) so it doesn't even give you the choice.

8. Bugzilla Hacking

8.1. What bugs are in Bugzilla right now?

Try this link (http://bugzilla.mozilla.org/buglist.cgi?bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&product=Bugzilla to view current bugs or requests for enhancement for Bugzilla.

You can view bugs marked for 2.18 release here (http://bugzilla.mozilla.org/buglist.cgi?product=Bugzilla&target_milestone=Bugzilla+2.18). This list includes bugs for the 2.18 release that have already been fixed and checked into CVS. Please consult the Bugzilla

Project Page (http://www.bugzilla.org/) for details on how to check current sources out of CVS so you can have these bug fixes early!

8.2. How can I change the default priority to a null value? For instance, have the default priority be "---" instead of "P2"?

This is well-documented here: http://bugzilla.mozilla.org/show_bug.cgi?id=49862 (http://bugzilla.mozilla.org/show_bug.cgi?id=49862). Ultimately, it's as easy as adding the "---" priority field to your localconfig file in the appropriate area, re-running checksetup.pl, and then changing the default priority in your browser using "editparams.cgi".

8.3. What's the best way to submit patches? What guidelines should I follow?

1. Enter a bug into bugzilla.mozilla.org for the "Bugzilla (http://bugzilla.mozilla.org/enter_bug.cgi?product=Bugzilla)" product.
2. Upload your patch as a unified diff (having used "diff -u" against the *current sources* checked out of CVS), or new source file by clicking "Create a new attachment" link on the bug page you've just created, and include any descriptions of database changes you may make, into the bug ID you submitted in step #1. Be sure and click the "Patch" checkbox to indicate the text you are sending is a patch!
3. Announce your patch and the associated URL (http://bugzilla.mozilla.org/show_bug.cgi?id=XXXXXX) for discussion in the newsgroup (netscape.public.mozilla.webtools). You'll get a really good, fairly immediate reaction to the implications of your patch, which will also give us an idea how well-received the change would be.
4. If it passes muster with minimal modification, the person to whom the bug is assigned in Bugzilla is responsible for seeing the patch is checked into CVS.
5. Bask in the glory of the fact that you helped write the most successful open-source bug-tracking software on the planet :)