The Bugzilla Guide

Matthew P. Barnson

barnboy@trilobyte.net

Zach Lipton

zach AT zachlipton DOT com

Revision History

Revision v2.11	20 December 2000	Revised by: MPB

Converted the README, FAQ, and DATABASE information into SGML docbook format.

Revision 2.11.1	06 March 2001	Revised by: MPB

Took way too long to revise this for 2.12 release. Updated FAQ to use qandaset tags instead of literallayout, cleaned up administration section, added User Guide section, miscellaneous FAQ updates and third−party integration information. From this point on all new tags are lowercase in preparation for the 2.13 release of the Guide in XML format instead of SGML.

Revision 2.12.0	24 April 2001	Revised by: MPB

Things fixed this release: Elaborated on queryhelp interface, added FAQ regarding moving bugs from one keyword to another, clarified possible problems with the Landfill tutorial, fixed a boatload of typos and unclear sentence structures. Incorporated the README into the UNIX installation section, and changed the README to indicate the deprecated status. Things I know need work: Used "simplelist" a lot, where I should have used "procedure" to tag things. Need to lowercase all tags to be XML compliant.

Revision 2.14.0	07 August 2001	Revised by: MPB

Attempted to integrate relevant portions of the UNIX and Windows installation instructions, moved some data from FAQ to Install, removed references to README from text, added Mac OS X install instructions, fixed a bunch of tpyos (Mark Harig), linked text that referenced other parts of the Guide, and nuked the old MySQL permissions section.

This is the documentation for Bugzilla, the Mozilla bug−tracking system.

Bugzilla is an enterprise−class set of software utilities that, when used together, power issue−tracking for hundreds of organizations around the world, tracking millions of bugs. While it is easy to use and quite flexible, it is very difficult for a novice to install and maintain. Although we have provided step−by−step directions, Bugzilla is not always easy to get working. Please be sure the person responsible for installing and maintaining this software is a qualified professional on operating system upon which you install Bugzilla.

THIS DOCUMENTATION IS MAINTAINED IN DOCBOOK 4.1 SGML FORMAT. IF YOU WISH TO MAKE CORRECTIONS, PLEASE MAKE THEM IN PLAIN TEXT OR SGML DIFFS AGAINST THE SOURCE. I CANNOT ACCEPT ADDITIONS TO THE GUIDE WRITTEN IN HTML!

Table of Contents

Chapter 1. About This Guide.....1
1.1. Purpose and Scope of this Guide.....1
1.2. Copyright Information.....1
1.3. Disclaimer.....2
1.4. New Versions.....2
1.5. Credits.....2
1.6. Contributors.....3
1.7. Feedback.....3
1.8. Translations.....3
1.9. Document Conventions.....3

Chapter 2. Using Bugzilla.....5
2.1. What is Bugzilla?.....5
2.2. Why Should We Use Bugzilla?.....5
2.3. How do I use Bugzilla?.....6
2.3.1. Create a Bugzilla Account.....7
2.3.2. The Bugzilla Query Page.....8
2.3.3. Creating and Managing Bug Reports.....10
2.4. Where can I find my user preferences?.....12
2.4.1. Account Settings.....12
2.4.2. Email Settings.....12
2.4.3. Page Footer.....13
2.4.4. Permissions.....14
2.5. Using Bugzilla−Conclusion.....14

Chapter 3. Installation.....15
3.1. ERRATA.....15
3.2. Step−by−step Install.....16
3.2.1. Introduction.....16
3.2.2. Installing the Prerequisites.....16
3.2.3. Installing MySQL Database.....17
3.2.4. Perl (5.004 or greater).....17
3.2.5. DBI Perl Module.....18
3.2.6. Data::Dumper Perl Module.....19
3.2.7. MySQL related Perl Module Collection.....19
3.2.8. TimeDate Perl Module Collection.....20
3.2.9. GD Perl Module (1.8.3).....20
3.2.10. Chart::Base Perl Module (0.99c).....20
3.2.11. DB_File Perl Module.....20
3.2.12. HTTP Server.....20
3.2.13. Installing the Bugzilla Files.....21
3.2.14. Setting Up the MySQL Database.....22
3.2.15. Tweaking localconfig.....23
3.2.16. Setting Up Maintainers Manually (Optional).....24
3.2.17. The Whining Cron (Optional).....24
3.2.18. Bug Graphs (Optional).....25
3.2.19. Securing MySQL.....25
3.3. Mac OS X Installation Notes.....26

Table of Contents

3.4. BSD Installation Notes....28
3.5. Installation General Notes....28
3.5.1. Modifying Your Running System....28
3.5.2. Upgrading From Previous Versions....28
3.5.3. .htaccess files and security....28
3.5.4. mod_throttle and Security....29
3.5.5. Preventing untrusted Bugzilla content from executing malicious Javascript code....29
3.5.6. UNIX Installation Instructions History....29
3.6. Win32 Installation Notes....30
3.6.1. Win32 Installation: Step−by−step....30
3.6.2. Additional Windows Tips....34
3.6.3. Bugzilla LDAP Integration....35

Chapter 4. Administering Bugzilla....37
4.1. Post−Installation Checklist....37
4.2. User Administration....39
4.2.1. Creating the Default User....40
4.2.2. Managing Other Users....40
4.3. Product, Component, Milestone, and Version Administration....43
4.3.1. Products....43
4.3.2. Components....44
4.3.3. Versions....45
4.3.4. Milestones....45
4.3.5. Voting....46
4.3.6. Groups and Group Security....47
4.4. Bugzilla Security....51

Chapter 5. Integrating Bugzilla with Third−Party Tools....54
5.1. Bonsai....54
5.2. CVS....54
5.3. Perforce SCM....54
5.4. Tinderbox/Tinderbox2....54

Chapter 6. The Future of Bugzilla....55

Chapter 7. Bugzilla Variants and Competitors....67
7.1. Red Hat Bugzilla....67
7.2. Loki Bugzilla (Fenris)....67
7.3. Issuezilla....67
7.4. Scarab....67
7.5. Perforce SCM....67
7.6. SourceForge....68
Appendix A. The Bugzilla FAQ....68
Appendix B. Software Download Links....87
Appendix C. The Bugzilla Database....87
C.1. Database Schema Chart....87
C.2. MySQL Bugzilla Database Introduction....89
C.2.1. Bugzilla Database Basics....89

Table of Contents

C.3. MySQL Permissions & Grant Tables....94
Appendix D. Useful Patches and Utilities for Bugzilla....98
D.1. Apache mod_rewrite magic....98
D.2. The setperl.csh Utility....98
D.3. Command−line Bugzilla Queries....99
D.4. The Quicksearch Utility....100
D.5. Hacking Bugzilla....100
Appendix E. GNU Free Documentation License....102
0. PREAMBLE....102
1. APPLICABILITY AND DEFINITIONS....103
2. VERBATIM COPYING....103
3. COPYING IN QUANTITY....104
4. MODIFICATIONS....104
5. COMBINING DOCUMENTS....105
6. COLLECTIONS OF DOCUMENTS....106
7. AGGREGATION WITH INDEPENDENT WORKS....106
8. TRANSLATION....106
9. TERMINATION....107
10. FUTURE REVISIONS OF THIS LICENSE....107
How to use this License for your documents....107
Glossary....107
0−9, high ascii....108
A....108
B....108
D....109
I....109
M....109
P....109
Q....110
R....110
S....110
T....110
Z....111

Chapter 1. About This Guide

1.1. Purpose and Scope of this Guide

This document was started on September 17, 2000 by Matthew P. Barnson after a great deal of procrastination updating the Bugzilla FAQ, which I left untouched for nearly half a year. After numerous complete rewrites and reformatting, it is the document you see today.

Bugzilla is simply the best piece of bug−tracking software the world has ever seen. This document is intended to be the comprehensive guide to the installation, administration, maintenance, and use of the Bugzilla bug−tracking system.

This release of the Bugzilla Guide is the *2.14* release. It is so named that it may match the current version of Bugzilla. The numbering tradition stems from that used for many free software projects, in which *even−numbered* point releases (1.2, 1.14, etc.) are considered "stable releases", intended for public consumption; on the other hand, *odd−numbered* point releases (1.3, 2.09, etc.) are considered unstable *development* releases intended for advanced users, systems administrators, developers, and those who enjoy a lot of pain.

Newer revisions of the Bugzilla Guide follow the numbering conventions of the main−tree Bugzilla releases, available at http://www.mozilla.org/projects/bugzilla. Intermediate releases will have a minor revision number following a period. The current version of Bugzilla, as of this writing (August 10, 2001) is 2.14; if something were seriously wrong with that edition of the Guide, subsequent releases would receive an additional dotted−decimal digit to indicate the update (2.14.1, 2.14.2, etc.). Got it? Good.

I wrote this in response to the enormous demand for decent Bugzilla documentation. I have incorporated instructions from the Bugzilla README, Frequently Asked Questions, Database Schema Document, and various mailing lists to create it. Chances are, there are glaring errors in this documentation; please contact `<barnboy@trilobyte.net>` to correct them.

1.2. Copyright Information

If you have any questions regarding this document, its copyright, or publishing this document in non−electronic form, please contact Matthew P. Barnson.

1.3. Disclaimer

No liability for the contents of this document can be accepted. Use the concepts, examples, and other content at your own risk. As this is a new edition of this document, there may be errors and inaccuracies that may damage your system. Use of this document may cause your girlfriend to leave you, your cats to pee on your furniture and clothing, your computer to cease functioning, your boss to fire you, 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". I 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. Heaven knows it's saved my bacon time after time; 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 insure your security needs are met.

1.4. New Versions

This is the 2.14 version of The Bugzilla Guide. 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.

This document can be found in the following places:

- TriloBYTE
- Mozilla.org
- The Linux Documentation Project

The latest version of this document can be checked out via CVS. Please follow the instructions available at the Mozilla CVS page, and check out the mozilla/webtools/bugzilla/docs/ branch.

1.5. Credits

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

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

Tara Hernandez for keeping Bugzilla development going strong after Terry left Mozilla.org

Dave Lawrence for providing insight into the key differences between Red Hat's customized Bugzilla, and being largely responsible for the "Red Hat Bugzilla" appendix

Dawn Endico for being a hacker extraordinaire and putting up with my incessant questions and arguments on irc.mozilla.org in #mozwebtools

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

1.6. Contributors

Thanks go to these people for significant contributions to this documentation (in no particular order):

Andrew Pearson, Spencer Smith, Eric Hanson, Kevin Brannen, Ron Teitelbaum, Jacob Steenhagen, Joe Robins

1.7. Feedback

I welcome feedback on this document. Without your submissions and input, this Guide cannot continue to exist. Please mail additions, comments, criticisms, etc. to `<barnboy@trilobyte.net>`. Please send flames to `<devnull@localhost>`

1.8. Translations

The Bugzilla Guide needs translators! Please volunteer your translation into the language of your choice. If you will translate this Guide, please notify the members of the mozilla−webtools mailing list at `<mozilla-webtools@mozilla.org>`, and arrange with Matt Barnson to check it into CVS.

1.9. Document Conventions

This document uses the following conventions

Descriptions	Appearance
Warnings	Warnings.

Hint	 Hint.
Notes	 Note.
Information requiring special attention	 Warning.
File Names	`file.extension`
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#
Prompt of user command under tcsh shell	tcsh$
Environment Variables	`VARIABLE`
Emphasized word	*word*
Code Example	<para>Beginning and end of paragraph</para>

Chapter 2. Using Bugzilla

What, Why, How, & Where?

2.1. What is Bugzilla?

Bugzilla is one example of a class of programs called "Defect Tracking Systems", or, more commonly, "Bug−Tracking Systems". Defect Tracking Systems allow individual or groups of developers to keep track of outstanding bugs in their product effectively. Bugzilla was originally written by Terry Weissman in a programming language called "TCL", to replace a crappy bug−tracking database used internally for 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 has matured immensely, and now boasts many advanced features. These include:

- integrated, product−based granular security schema
- inter−bug dependencies and dependency graphing
- advanced reporting capabilities
- a robust, stable RDBMS back−end
- extensive configurability
- a very well−understood and well−thought−out natural bug resolution protocol
- email, XML, console, and HTTP APIs
- available integration with automated software configuration management systems, including Perforce and CVS (through the Bugzilla email interface and checkin/checkout scripts)
- too many more features to list

Despite its current robustness and popularity, Bugzilla faces some near−term challenges, such as reliance on a single database, a lack of abstraction of the user interface and program logic, verbose email bug notifications, a powerful but daunting query interface, little reporting configurability, problems with extremely large queries, some unsupportable bug resolution options, little internationalization (although non−US character sets are accepted for comments), and dependence on some nonstandard libraries.

Some recent headway has been made on the query front, however. If you are using the latest version of Bugzilla, you should see a "simple search" form on the default front page of your Bugzilla install. Type in two or three search terms and you should pull up some relevant information. This is also available as "queryhelp.cgi".

Despite these small problems, Bugzilla is very hard to beat. It is under *very* active development to address the current issues, and continually gains new features.

2.2. Why Should We Use Bugzilla?

No, Who's on first...

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, Loki software, Linux−Mandrake, and VA Systems. Combined with systems such as CVS, Bonsai, or Perforce SCM, 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 by using Bugzilla's e−mail integration features be able to 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.

2.3. How do I use Bugzilla?

Hey! I'm Woody! Howdy, Howdy, Howdy!

Bugzilla is a large, complex system. Describing how to use it requires some time. If you are only interested in installing or administering a Bugzilla installation, please consult the Installing and Administering Bugzilla portions of this Guide. This section is principally aimed towards developing end−user mastery of Bugzilla, so you may fully enjoy the benefits afforded by using this reliable open−source bug−tracking software.

Throughout this portion of the Guide, we will refer to user account options available at the Bugzilla test installation, landfill.tequilarista.org.

> Some people have run into difficulties completing this tutorial. If you run into problems, please check the updated online documentation available at http://www.trilobyte.net/barnsons. If you're still stumped, please subscribe to the newsgroup and provide details of exactly what's stumping you! If enough people complain, I'll have to fix it in the next version of this Guide. You can subscribe to

the newsgroup at news://news.mozilla.org/netscape.public.mozilla.webtools

Although Landfill serves as a great introduction to Bugzilla, it does not offer all the options you would have as a user on your own installation of Bugzilla, nor can it do more than serve as a general introduction to Bugzilla. Additionally, Landfill often runs cutting−edge versions of Bugzilla for testing, so some things may work slightly differently than mentioned here.

2.3.1. Create a Bugzilla Account

First things first! 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 the end−user Bugzilla experience, use this URL: http://landfill.tequilarista.org/bugzilla−tip/

1. Click the "Open a new Bugzilla account" link.
2. Enter your "E−mail address" and "Real Name" (or whatever name you want to call yourself) in the spaces provided, then select the "Create Account" button.
3. 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 should be changed at your nearest opportunity (we'll go into how to do it later).
4. Click the "Log In" link in the yellow area at the bottom of the page in your browser, then enter your "E−mail address" and "Password" you just received into the spaces provided, and select "Login".

If you ever forget your password, you can come back to this page, enter your "E−mail address", then select the "E−mail me a password" button to have your password mailed to you again so that you can login.

Many modern browsers include an "Auto−Complete" or "Form Fill" feature to remember the user names and passwords you type in at many sites. Unfortunately, sometimes they attempt to guess what you will put in as your password, and guess wrong. If you notice a text box is already filled out, please overwrite the contents of the text box so you can be sure to input the correct information.

Congratulations! If you followed these directions, you now are the proud owner of a user account on landfill.tequilarista.org (Landfill) or your local Bugzilla install. You should now see in your browser a page called the "Bugzilla Query Page". It may look daunting, but with this Guide to walk you through it, you will master it in no time.

2.3.2. The Bugzilla Query Page

The Bugzilla Query Page is the heart and soul of the Bugzilla user experience. It is the master interface where you can find any bug report, comment, or patch currently in the Bugzilla system. We'll go into how to create your own bug report later on.

There are efforts underway to simplify query usage. If you have a local installation of Bugzilla 2.12 or higher, you should have `quicksearch.html` available to use and simplify your searches. There is also a helper for the query interface, called `queryhelp.cgi`. Landfill tends to run the latest code, so these two utilities should be available there for your perusal.

At this point, please visit the main Bugzilla site, bugzilla.mozilla.org, to see a more fleshed−out query page.

The first thing you need to notice about the Bugzilla Query Page is that nearly every box you see on your screen has a hyperlink nearby, explaining what it is or what it does. Near the upper−left−hand corner of your browser window you should see the word "Status" underlined. Select it.

Notice the page that popped up? Every underlined word you see on your screen is a hyperlink that will take you to context−sensitive help. Click around for a while, and learn what everything here does. To return to the query interface after pulling up a help page, use the "Back" button in your browser.

I'm sure that after checking out the online help, you are now an expert on the Bugzilla Query Page. If, however, you feel you haven't mastered it yet, let me walk you through making a few successful queries to find out what there are in the Bugzilla bug−tracking system itself.

1. Ensure you are back on the "Bugzilla Query Page". Do nothing in the boxes marked "Status", "Resolution", "Platform", "OpSys", "Priority", or "Severity". The default query for "Status" is to find all bugs that are NEW, ASSIGNED, or REOPENED, which is what we want. If you don't select anything in the other 5 scrollboxes there, then you are saying that "any of these are OK"; we're not locking ourselves into only finding bugs on the "DEC" Platform, or "Windows 95" OpSys (Operating System). You're smart, I think you have it figured out.

   Basically, selecting *anything* on the query page narrows your search down. Leaving stuff unselected, or text boxes unfilled, broadens your search.

2. You see the box immediately below the top six boxes that contains an "Email" text box, with the words "matching as", a drop−down selection box, then some checkboxes with "Assigned To" checked by default? This allows you to filter your search down based upon email address. Let's put my email address in there, and see what happens.

   Type "barnboy@trilobyte.net" in the top Email text box.

3. Let's narrow the search some more. Scroll down until you find the box with the word "Program" over the top of it. This is where we can narrow our search down to only specific products (software programs or product lines) in our Bugzilla database. Please notice the box is a *scrollbox*. Using the down arrow on the scrollbox, scroll down until you can see an entry called "Bugzilla". Select this entry.
4. Did you notice that some of the boxes to the right changed when you selected "Bugzilla"? Every Program (or Product) has different Versions, Components, and Target Milestones associated with it. A "Version" is the number of a software program.

Example 2−1. Some Famous Software Versions

Do you remember the hype in 1995 when Microsoft Windows 95(r) was released? It may have been several years ago, but Microsoft(tm) spent over $300 Million advertising this new Version of their software. Three years later, they released Microsoft Windows 98(r), another new version, to great fanfare, and then in 2000 quietly released Microsoft Windows ME(Millenium Edition)(r).

Software "Versions" help a manufacturer differentiate their current product from their previous products. Most do not identify their products by the year they were released. Instead, the "original" version of their software will often be numbered "1.0", with small bug−fix releases on subsequent tenths of a digit. In most cases, it's not a decimal number; for instance, often 1.9 is an *older* version of the software than 1.11, but is a *newer* version than 1.1.1.

In general, a "Version" in Bugzilla should refer to *released* products, not products that have not yet been released to the public. Forthcoming products are what the Target Milestone field is for.

A "Component" is a piece of a Product. It may be a standalone program, or some other logical division of a Product or Program. Normally, a Component has a single Owner, who is responsible for overseeing efforts to improve that Component.

Example 2−2. Mozilla's Bugzilla Components

Mozilla's "Bugzilla" Product is composed of several pieces (Components):

Administration, Administration of a bugzilla installation, including `editcomponents.cgi`, `editgroups.cgi`, `editkeywords.cgi`, `editparams.cgi`, `editproducts.cgi`, `editusers.cgi`, `editversions.cgi`, and `sanitycheck.cgi`.

Bugzilla−General, Anything that doesn't fit in the other components, or spans multiple components.

Creating/Changing Bugs, Creating, changing, and viewing bugs. `enter_bug.cgi`, `post_bug.cgi`, `show_bug.cgi` and `process_bug.cgi`.

Documentation, The bugzilla documentation, including anything in the `docs/` directory and The Bugzilla Guide (This document :)

Email, Anything to do with email sent by Bugzilla. `processmail`

Installation, The installation process of Bugzilla. This includes `checksetup.pl` and whatever else it evolves into.

Query/Buglist, Anything to do with searching for bugs and viewing the buglists. `query.cgi` and `buglist.cgi`

Reporting/Charting, Getting reports from Bugzilla. `reports.cgi` and `duplicates.cgi`

User Accounts, Anything about managing a user account from the user's perspective. `userprefs.cgi`, saved queries, creating accounts, changing passwords, logging in, etc.

User Interface, General issues having to do with the user interface cosmetics (not functionality) including cosmetic issues, HTML templates, etc.

A "Milestone", or "Target Milestone" is a often a planned future "Version" of a product. In many cases, though, Milestones simply represent significant dates for a developer. Having certain features in your Product is frequently tied to revenue (money) the developer will receive if the features work by the time she reaches the Target Milestone. Target Milestones are a great tool to organize your time. If someone will pay you $100,000 for incorporating certain features by a certain date, those features by that Milestone date become a very high priority. Milestones tend to be highly malleable creatures, though, that appear to be in reach but are out of reach by the time the important day arrives.

The Bugzilla Project has set up Milestones for future Bugzilla versions 2.14, 2.16, 2.18, 3.0, etc. However, a Target Milestone can just as easily be a specific date, code name, or weird alphanumeric combination, like "M19".

5. OK, now let's select the "Bugzilla" component from its scrollbox.
6. Skip down the page a bit −− do you see the "submit query" button? Select it, and let's run this query!
7. Congratulations! You've completed your first Query, and have before you the Bug List of the author of this Guide, Matthew P. Barnson (barnboy@trilobyte.net). If I'm doing well, you'll have a cryptic "Zarro Boogs Found" message on your screen. It is just a happy hacker's way of saying "Zero Bugs Found". However, I am fairly certain I will always have some bugs assigned to me that aren't done yet, so you won't often see that message!

I encourage you to click the bug numbers in the left−hand column and examine my bugs. Also notice that if you click the underlined links near the top of this page, they do not take you to context−sensitive help here, but instead sort the columns of bugs on the screen! When you need to sort your bugs by priority, severity, or the people they are assigned to, this is a tremendous timesaver.

A couple more interesting things about the Bug List page:

Change Columns: by selecting this link, you can show all kinds of information in the Bug List

Change several bugs at once: If you have sufficient rights to change all the bugs shown in the Bug List, you can mass−modify them. This is a big time−saver.

Send mail to bug owners: If you have many related bugs, you can request an update from every person who owns the bugs in the Bug List asking them the status.

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 small revisions to the query you just made so you get more accurate results.

There are many more options to the Bugzilla Query Page and the Bug List than I have shown you. But this should be enough for you to learn to get around. I encourage you to check out the Bugzilla Home Page to learn about the Anatomy and Life Cycle of a Bug before continuing.

2.3.3. Creating and Managing Bug Reports

And all this time, I thought we were taking bugs out...

2.3.3.1. Writing a Great Bug Report

Before we plunge into writing your first bug report, I encourage you to read some bug−writing guidelines. If you are reading this document as part of a Bugzilla CVS checkout or un−tarred Bugzilla distribution, you should be able to read them by clicking here. If you are reading this online, check out the Mozilla.org bug−writing guidelines at http://www.mozilla.org/quality/bug−writing−guidelines.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.

While you are at it, why not learn how to find previously reported bugs? Mozilla.org has published a great tutorial on finding duplicate bugs, available at http://www.mozilla.org/quality/help/beginning−duplicate−finding.html.

I realize this was a lot to read. However, understanding the mentality of writing great bug reports will help us on the next part!

1. Go back to http://landfill.tequilarista.org/bugzilla−tip/ in your browser.
2. Select the Enter a new bug report link.
3. Select a product.
4. Now you should be at the "Enter Bug" form. The "reporter" should have been automatically filled out for you (or else Bugzilla prompted you to Log In again −− you did keep the email with your username and password, didn't you?).
5. Select a Component in the scrollbox.
6. Bugzilla should have made reasonable guesses, based upon your browser, for the "Platform" and "OS" drop−down boxes. If those are wrong, change them −− if you're on an SGI box running IRIX, we want to know!
7. Fill in the "Assigned To" box with the email address you provided earlier. This way you don't end up sending copies of your bug to lots of other people, since it's just a test bug.
8. Leave the "CC" text box blank. Fill in the "URL" box with "http://www.mozilla.org".
9. Enter "The Bugzilla Guide" in the Summary text box, and place any comments you have on this tutorial, or the Guide in general, into the Description box.

Voila! Select "Commit" and send in your bug report! Next we'll look at resolving bugs.

2.3.3.2. Managing your Bug Reports

OK, you should have a link to the bug you just created near the top of your page. It should say "Bug XXXX posted", with a link to the right saying "Back to BUG# XXXX". Select this link.

1. Scroll down a bit on the subsequent page, until you see the "Resolve bug, changing resolution to (dropdown box). Normally, you would "Accept bug (change status to ASSIGNED)", fix it, and then resolve. But in this case, we're going to short−circuit the process because this wasn't a real bug. Change the dropdown next to "Resolve Bug" to "INVALID", make sure the radio button is marked next to "Resolve Bug", then click "Commit".
2. Hey! It said it couldn't take the change in a big red box! That's right, you must specify a Comment in order to make this change. Select the "Back" button in your browser, add a Comment, then try Resolving the bug with INVALID status again. This time it should work.

You have now learned the basics of Bugzilla navigation, entering a bug, and bug maintenance. I encourage you to explore these features, and see what you can do with them! We'll spend no more time on individual Bugs or Queries from this point on, so you are on your own there.

But I'll give a few last hints!

There is a CLUE on the Query page that will teach you more how to use the form.

If you click the hyperlink on the Component box of the Query page, you will be presented a form that will describe what all the components are.

Possibly the most powerful feature of the Query page is the Boolean Chart section. It's a bit confusing to use the first time, but can provide unparalleled flexibility in your queries, allowing you to build extremely powerful requests.

Finally, you can build some nifty Reports using the "Bug Reports" link near the bottom of the query page, and also available via the "Reports" link at the footer of each page.

2.4. Where can I find my user preferences?

Indiana, it feels like we walking on fortune cookies!

These ain't fortune cookies, kid...

Customized User Preferences offer tremendous versatility to your individual Bugzilla experience. Let's plunge into what you can do! The first step is to click the "Edit prefs" link at the footer of each page once you have logged in to Landfill.

2.4.1. Account Settings

On this page, you can change your basic Account Settings, including your password and full name. For security reasons, in order to change anything on this page you must type your *current* password into the "Old Password" field. If you wish to change your password, type the new password you want into the "New Password" field and again into the "Re–enter new password" field to ensure you typed your new password correctly. Select the "Submit" button and you are done.

2.4.2. Email Settings

2.4.2.1. Email Notification

Here you can reduce or increase the amount of email sent you from Bugzilla. Although this is referred to as "Advanced Email Filtering Options", they are, in fact, the standard email filter set. All of them are self–explanatory, but you can use the filters in interesting ways. For instance, some people (notably Quality Assurance personnel) often only care to receive updates regarding a bug when the bug changes state, so they can track bugs on their flow charts and know when it is time to pull the bug onto a quality assurance platform for inspection. Other people set up email gateways to Bonsai, the Mozilla automated CVS management system or Tinderbox, the Mozilla automated build management system, and restrict which types of Bugzilla information are fed to these systems..

2.4.2.2. New Email Technology

> This option may not be available in all Bugzilla installations, depending upon the preferences of the systems administrator responsible for the setup of your Bugzilla. However, if you really want this functionality, ask her to "enable newemailtech in Params" and "make it the default for all new users", referring her to the Administration section of this Guide.

Disregard the warnings about "experimental and bleeding edge"; the code to handle email in a cleaner manner than that historically used for Bugzilla is quite robust and well–tested now.

I recommend you enable the option, "Click here to sign up (and risk any bugs)". Your email–box will thank you for it. The fundamental shift in "newemailtech" is away from standard UNIX "diff" output, which is quite ugly, to a prettier, better laid–out email.

2.4.2.3. "Watching" Users

> This option may not be available in all Bugzilla installations, depending upon the preferences of the systems administrator responsible for the setup of your Bugzilla. However, if you really want this functionality, ask her to "enable watchers in Params".

By entering user email names into the "Users to watch" text entry box, delineated by commas, you can watch bugs of other users. This powerful functionality enables seamless transitions as developers change projects, managers wish to get in touch with the issues faced by their direct reports, or users go on vacation. If any of these three situations apply to you, you will undoubtedly find this feature quite convenient.

2.4.3. Page Footer

> By default, this page is quite barren. However, go explore the Query Page some more; you will find that you can store numerous queries on the server, so if you regularly run a particular query it is just a drop–down menu away. On this page of Preferences, if you have many stored queries you can elect to have them always one–click away!

If you have many stored queries on the server, here you will find individual drop−downs for each stored query. Each drop−down gives you the option of that query appearing on the footer of every page in Bugzilla! This gives you powerful one−click access to any complex searches you may set up, and is an excellent way to impress your boss...

By default, the "My Bugs" link appears at the bottom of each page. However, this query gives you both the bugs you have reported, as well as those you are assigned. One of the most common uses for this page is to remove the "My Bugs" link, replacing it with two other queries, commonly called "My Bug Reports" and "My Bugs" (but only referencing bugs assigned to you). This allows you to distinguish those bugs you have reported from those you are assigned. I commonly set up complex Boolean queries in the Query page and link them to my footer in this page. When they are significantly complex, a one−click reference can save hours of work.

2.4.4. Permissions

This is a purely informative page which outlines your current permissions on this installation of Bugzilla. If you have permissions to grant certain permissions to other users, the "other users" link appears on this page as well as the footer. For more information regarding user administration, please consult the Administration section of this Guide.

2.5. Using Bugzilla−Conclusion

Thank you for reading through this portion of the Bugzilla Guide. I anticipate it may not yet meet the needs of all readers. If you have additional comments or corrections to make, please submit your contributions to the mozilla−webtools mailing list/newsgroup. The mailing list is mirrored to the netscape.public.mozilla.webtools newsgroup, and the newsgroup is mirrored to mozilla−webtools@mozilla.org

Chapter 3. Installation

These installation instructions are presented assuming you are installing on a UNIX or completely POSIX−compliant system. If you are installing on Microsoft Windows or another oddball operating system, please consult the appropriate sections in this installation guide for notes on how to be successful.

3.1. ERRATA

Here are some miscellaneous notes about possible issues you main run into when you begin your Bugzilla installation. Reference platforms for Bugzilla installation are Redhat Linux 7.2, Linux−Mandrake 8.0, and Solaris 8.

If you are installing Bugzilla on S.u.S.e. 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.

Bugzilla may be installed on Macintosh OS X (10), which is a unix−based (BSD) operating system. Everything required for Bugzilla on OS X will install cleanly, but the optional GD perl module which is used for bug charting requires some additional setup for installation. Please see the Mac OS X installation section below for details

Release Notes for Bugzilla 2.14 are available at `docs/rel_notes.txt` in your Bugzilla source distribution.

The preferred documentation for Bugzilla is available in docs/, with a variety of document types available. Please refer to these documents when installing, configuring, and maintaining your Bugzilla installation.

Bugzilla is not a package where you can just plop it in a directory, twiddle a few things, and you're off. Installing Bugzilla assumes you know your variant of UNIX or Microsoft Windows well, are familiar with the command line, and are comfortable compiling and installing a plethora of third−party utilities. To install Bugzilla on Win32 requires fair Perl proficiency, and if you use a webserver other than Apache you should be intimately familiar with the security mechanisms and CGI environment thereof.

Bugzilla has not undergone a complete security review. Security holes may exist in the code. 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.

3.2. Step−by−step Install

3.2.1. Introduction

Installation of bugzilla is pretty straightforward, particularly if your machine already has MySQL and the MySQL−related perl packages installed. If those aren't installed yet, then that's the first order of business. The other necessary ingredient is a web server set up to run cgi scripts. While using Apache for your webserver is not required, it is recommended.

Bugzilla has been successfully installed under Solaris, Linux, and Win32. The peculiarities of installing on Win32 (Microsoft Windows) are not included in this section of the Guide; please check out the Win32 Installation Notes for further advice on getting Bugzilla to work on Microsoft Windows.

The Bugzilla Guide is contained in the "docs/" folder in your Bugzilla distribution. It is available in plain text (docs/txt), HTML (docs/html), or SGML source (docs/sgml).

3.2.2. Installing the Prerequisites

If you want to skip these manual installation steps for the CPAN dependencies listed below, and are running the very most recent version of Perl and MySQL (both the executables and development libraries) on your system, check out Bundle::Bugzilla in Using Bundle::Bugzilla instead of manually installing Perl modules

The software packages necessary for the proper running of bugzilla are:

1. MySQL database server and the mysql client (3.22.5 or greater)
2. Perl (5.004 or greater, 5.6.1 is recommended if you wish to use Bundle::Bugzilla)
3. DBI Perl module
4. Data::Dumper Perl module
5. Bundle::Mysql Perl module collection
6. TimeDate Perl module collection
7. GD perl module (1.8.3) (optional, for bug charting)
8. Chart::Base Perl module (0.99c) (optional, for bug charting)
9. DB_File Perl module (optional, for bug charting)
10. The web server of your choice. Apache is recommended.
11. MIME::Parser Perl module (optional, for contrib/bug_email.pl interface)

It is a good idea, while installing Bugzilla, to ensure it is not *accessible* by other machines on the Internet. Your machine may be vulnerable to attacks while you are installing. In other words, ensure there is some kind of firewall between you and the rest of the Internet. 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.

Linux−Mandrake 8.0, the author's test system, 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**

3.2.3. Installing MySQL Database

Visit MySQL homepage at www.mysql.com and grab the latest stable release of the server. Many of the binary versions of MySQL store their data files in `/var` which is often part of a smaller root partition. If you decide to build from sources you can easily set the dataDir as an option to `configure`.

If you install from source or non−package (RPM, deb, etc.) binaries you 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.

You should have your init script 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.

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.

3.2.4. Perl (5.004 or greater)

Any machine that doesn't have perl on it is a sad machine indeed. Perl for *nix systems can be gotten in source form from http://www.perl.com. Although Bugzilla runs with most post−5.004 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.

Perl is now a far cry from the the single compiler/interpreter binary it once was. It includes a great many required modules and quite a few other support files. If you're not up to or not inclined to build perl from source, you'll want to install it on your machine using some sort of packaging system (be it RPM, deb, or what have you) to ensure a sane install. In the subsequent sections you'll be installing quite a few perl modules; this can be quite ornery if your perl installation isn't up to snuff.

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 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.

You can skip the following Perl module installation steps by installing Bundle::Bugzilla from *CPAN*, which includes them. All Perl module installation steps require you have an active Internet connection. If you wish to use Bundle::Bugzilla, however, you must be using the latest version of Perl (at this writing, version 5.6.1)

```
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.

3.2.5. DBI Perl Module

The DBI module is a generic Perl module used by other database related Perl modules. For our purposes it's required by 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.

Like almost all Perl modules DBI can be found on the Comprehensive Perl Archive Network (CPAN) at http://www.cpan.org. The CPAN servers have a real tendency to bog down, so please use mirrors. The current location at the time of this writing can be found in Appendix B.

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 DBI:

```
bash# perl -MCPAN -e 'install "DBI"'
```

Replace "DBI" with the name of whichever module you wish to install, such as Data::Dumper, TimeDate, GD, etc.

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`**

If everything went ok that should be all it takes. For the vast majority of perl modules this is all that's required.

3.2.6. Data::Dumper Perl Module

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.

Data::Dumper is used by the MySQL−related Perl modules. It can be found on CPAN (see Appendix B) and can be installed by following the same four step make sequence used for the DBI module.

3.2.7. MySQL related Perl Module Collection

The Perl/MySQL interface requires a few mutually−dependent perl modules. These modules are grouped together into the the Msql−Mysql−modules package. This package can be found at CPAN. After the archive file has been downloaded it should be untarred.

The MySQL modules are all built using one make file which is generated by running: `bash#` **perl Makefile.pl**

The MakeMaker process will ask you a few questions about the desired compilation target and your MySQL installation. For many of the questions the provided default will be adequate.

When asked if your desired target is the MySQL or mSQL packages, 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' and a null password should find itself with sufficient access to run tests on the 'test' database which MySQL created upon installation. If 'make test' and 'make install' go through without errors you should be ready to go as far as database connectivity is concerned.

3.2.8. TimeDate Perl Module Collection

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 (see link: Appendix B). The component module we're most interested in is the Date::Format module, but installing all of them is probably a good idea anyway. The standard Perl module installation instructions should work perfectly for this simple package.

3.2.9. GD Perl Module (1.8.3)

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.

Actually bugzilla uses the Graph module which relies on GD itself. Isn't that always the way with object−oriented programming? At any rate, you can find the GD library on CPAN in Appendix B.

> 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. Just realize that if compiling GD fails, it's probably because you're missing a required library.

3.2.10. Chart::Base Perl Module (0.99c)

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 where it is found as the Chart−x.x... tarball, linked in Appendix B. Note that as with the GD perl module, only the version listed above, or newer, will work. Earlier versions used GIF's, which are no longer supported by the latest versions of GD.

3.2.11. DB_File Perl Module

DB_File is a module which allows Perl programs to make use of the facilities provided by Berkeley DB version 1.x. This module is required by collectstats.pl which is used for bug charting. If you plan to make use of bug charting, you must install this module.

3.2.12. HTTP Server

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

I strongly recommend Apache as the web server to use. The Bugzilla Guide installation instructions, in general, assume you are using Apache. As more users use different webservers and send me information on the peculiarities of installing using their favorite webserver, I will provide notes for them.

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 srm.conf file:

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

With apache you'll also want to make sure that within the access.conf file the line:

```
Options ExecCGI
```

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

Users of newer versions of Apache will generally find both of the above lines will be in the httpd.conf file, rather than srm.conf or access.conf.

There are important files and directories that should not be a served by the HTTP server. These are most files in the "data" and "shadow" directories and the "localconfig" file. You should configure your HTTP server to not serve content from 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. I appreciate notes on how to get this same functionality using other webservers.

3.2.13. Installing the Bugzilla Files

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 off of the main web space for your web server or perhaps off of `/usr/local` with a symbolic link in the web space that points to the Bugzilla directory. At any rate, just dump all the files in the same place, and make sure you can access the files in that directory through your web server.

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.

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, or use The setperl.csh Utility, found in Useful Patches and Utilities for Bugzilla. I suggest using the symlink approach for future release compatability.

Example 3−1. Setting up bonsaitools symlink

Here's how you set up the Perl symlink on Linux to make Bugzilla work. Your mileage may vary. For some UNIX operating systems, you probably need to subsitute "/usr/local/bin/perl" for "/usr/bin/perl" below; if on certain other UNIX systems, Perl may live in weird places like "/opt/perl". As root, run these commands:

```
bash# mkdir /usr/bonsaitools
bash# mkdir /usr/bonsaitools/bin
bash# ln −s /usr/bin/perl /usr/bosaitools/bin/perl
```

Alternately, you can simply run this perl one−liner to change your path to perl in all the files in your Bugzilla installation:

```
perl −pi −e 's@#!/usr/bonsaitools/bin/perl@#!/usr/bin/perl@' *cgi *pl Bug.pm
```

Change the second path to perl to match your installation.

If you don't have root access to set this symlink up, check out the The setperl.csh Utility, listed in Useful Patches and Utilities for Bugzilla. It will change the path to perl in all your Bugzilla files for you.

3.2.14. 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 a 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.

Bugzilla has not undergone a thorough security audit. It may be possible for a system cracker to somehow trick Bugzilla into executing a command such as **DROP DATABASE mysql**.

That would be bad.

Give 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 your new_password. Remember that MySQL user names have nothing to do with Unix user names (login names).

Next, we create the "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@localhost IDENTIFIED BY
'bugs_password';
mysql> FLUSH PRIVILEGES;
```

Next, run the magic checksetup.pl script. (Many thanks to Holger Schurig <holgerschurig@nikocity.de> for writing this script!) 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`.

3.2.15. Tweaking `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 above

You should also install .htaccess files that the Apache webserver will use to restrict access to Bugzilla data files. See .htaccess files and security.

Once you are happy with the settings, re−run `checksetup.pl`. On this second run, it will create the database and an administrator account for which you will be prompted to provide information.

When logged into an administrator account once Bugzilla is running, if you go to the query page (off of the Bugzilla main menu), you'll find an "edit parameters" option that is filled with editable treats.

Should everything work, you will have a nearly empty Bugzilla database and a newly−created `localconfig` file in your Bugzilla root directory.

> The second time you run checksetup.pl, you should become the user your web server runs as, and that you ensure that you set the "webservergroup" parameter in localconfig to match the web server's group name, if any. I believe, for the next release of Bugzilla, this will be fixed so that Bugzilla supports a "webserveruser" parameter in localconfig as well.
>
> **Example 3−2. Running checksetup.pl as the web user**
>
> Assuming your web server runs as user "apache", and Bugzilla is installed in "/usr/local/bugzilla", here's one way to run checksetup.pl as the web server user. As root, for the *second run* of checksetup.pl, do this:
>
> ```
> bash# chown −R apache:apache /usr/local/bugzilla
> bash# su − apache
> bash# cd /usr/local/bugzilla
> bash# ./checksetup.pl
> ```
>
> > 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.

3.2.16. Setting Up Maintainers Manually (Optional)

If you want to add someone else to every group by hand, you can do it by typing the appropriate MySQL commands. Run **mysql −u root −p bugs** You may need different parameters, depending on your security settings. Then:

`mysql>` **`update profiles set groupset=0x7fffffffffffffff where login_name = 'XXX';`** (yes, that's *fifteen*"f"'s.

replacing XXX with the Bugzilla email address.

3.2.17. The Whining Cron (Optional)

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. 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
```

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
```

3.2.18. Bug Graphs (Optional)

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 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.

3.2.19. Securing MySQL

If you followed the installation instructions for setting up your "bugs" and "root" user in MySQL, much of this should not apply to you. If you are upgrading an existing installation of Bugzilla, you should pay close attention to this section.

Most MySQL installs have "interesting" default security parameters:

mysqld defaults to running as root
it defaults to allowing external network connections
it has a known port number, and is easy to detect
it defaults to no passwords whatsoever
it defaults to allowing "File_Priv"

This means anyone from anywhere on the internet can not only drop the database with one SQL command, and they can write as root to the system.

To see your permissions do:

```
bash# mysql -u root -p
mysql> use mysql;
```

```
mysql> show tables;
mysql> select * from user;
mysql> select * from db;
```

To fix the gaping holes:

```
DELETE FROM user WHERE User='';
UPDATE user SET Password=PASSWORD('new_password') WHERE user='root';
FLUSH PRIVILEGES;
```

If you're not running "mit−pthreads" you can use:

```
GRANT USAGE ON *.* TO bugs@localhost;
GRANT ALL ON bugs.* TO bugs@localhost;
REVOKE DROP ON bugs.* FROM bugs@localhost;
FLUSH PRIVILEGES;
```

With "mit−pthreads" you'll need to modify the "globals.pl" Mysql−>Connect line to specify a specific host name instead of "localhost", and accept external connections:

```
GRANT USAGE ON *.* TO bugs@bounce.hop.com;
GRANT ALL ON bugs.* TO bugs@bounce.hop.com;
REVOKE DROP ON bugs.* FROM bugs@bounce.hop.com;
FLUSH PRIVILEGES;
```

Use .htaccess files with the Apache webserver to secure your bugzilla install. See .htaccess files and security

Consider also:

1. Turning off external networking with "−−skip−networking", unless you have "mit−pthreads", in which case you can't. Without networking, MySQL connects with a Unix domain socket.
2. using the −−user= option to mysqld to run it as an unprivileged user.
3. starting MySQL in a chroot jail
4. running the httpd in a "chrooted" jail
5. making sure the MySQL passwords are different from the OS passwords (MySQL "root" has nothing to do with system "root").
6. running MySQL on a separate untrusted machine
7. making backups ;−)

3.3. 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 (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 the following patch to the Makefile.PL file (save the patch into a file and use the command **patch < patchfile**:

```
--- GD-1.33/Makefile.PL Fri Aug  4 16:59:22 2000
+++ GD-1.33-darwin/Makefile.PL  Tue Jun 26 01:29:32 2001
@@ -3,8 +3,8 @@
 warn "NOTICE: This module requires libgd 1.8.3 or higher (shared library version 4.X).\n";

 # =====> PATHS: CHECK AND ADJUST <=====
-my @INC     = qw(-I/usr/local/include -I/usr/local/include/gd);
-my @LIBPATH = qw(-L/usr/lib/X11 -L/usr/X11R6/lib -L/usr/X11/lib -L/usr/local/lib );
+my @INC     = qw(-I/sw/include -I/sw/include/gd -I/usr/local/include -I/usr/local/include/gd);
+my @LIBPATH = qw(-L/usr/lib/X11 -L/usr/X11R6/lib -L/usr/X11/lib -L/sw/lib -L/usr/local/lib);
 my @LIBS    = qw(-lgd -lpng -lz);

 # FEATURE FLAGS
@@ -23,7 +23,7 @@

 push @LIBS,'-lttf' if $TTF;
 push @LIBS,'-ljpeg' if $JPEG;
-push @LIBS, '-lm' unless $^O eq 'MSWin32';
+push @LIBS, '-lm' unless ($^O =~ /^MSWin32|darwin$/);

 # FreeBSD 3.3 with libgd built from ports croaks if -lXpm is specified
 if ($^O ne 'freebsd' && $^O ne 'MSWin32') {
```

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

perl Makefile.PL

make

make test

make install

And don't forget to run **exit** to get back to cpan.

Happy Hacking!

3.4. BSD Installation Notes

For instructions on how to set up Bugzilla on FreeBSD, NetBSD, OpenBSD, BSDi, etc. please consult Section 3.3.

3.5. Installation General Notes

3.5.1. Modifying Your Running System

Bugzilla optimizes database lookups by storing all relatively static information in the versioncache file, located in the data/ subdirectory under your installation directory.

If you make a change to the structural data in your database (the versions table for example), or to the "constants" encoded in defparams.pl, you will need to remove the cached content from the data directory (by doing a "rm data/versioncache"), or your changes won't show up.

That file gets automatically regenerated whenever it's more than an hour old, so Bugzilla will eventually notice your changes by itself, but generally you want it to notice right away, so that you can test things.

3.5.2. Upgrading From Previous Versions

The developers of Bugzilla are constantly adding new tables, columns and fields. You'll get SQL errors if you just update the code. The strategy to update is to simply always run the checksetup.pl script whenever you upgrade your installation of Bugzilla. If you want to see what has changed, you can read the comments in that file, starting from the end.

If you are running Bugzilla version 2.8 or lower, and wish to upgrade to the latest version, please consult the file, "UPGRADING−pre−2.8" in the Bugzilla root directory after untarring the archive.

3.5.3. `.htaccess` files and security

To enhance the security of your Bugzilla installation, Bugzilla will generate *`.htaccess`* files which the Apache webserver can use to restrict access to the bugzilla data files. The checksetup script will generate the `.htaccess` files.

> 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.

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

3.5.4. `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/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 for more information.

3.5.5. 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. 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# cd $BUGZILLA_HOME; for i in `ls *.cgi`; \
          do cat $i | sed 's/Content-type\: text\/html/Content-Type: text\/html\; charset=ISO-8859-1/' >$i.tmp; \
          mv $i.tmp $i; done
```

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".

3.5.6. UNIX Installation Instructions History

This document was originally adapted from the Bonsai installation instructions by Terry Weissman <terry@mozilla.org>.

The February 25, 1999 re−write of this page was done by Ry4an Brase <ry4an@ry4an.org>, with some edits by Terry Weissman, Bryce Nesbitt, Martin Pool, & Dan Mosedale (But don't send bug reports to them; report them using bugzilla, at http://bugzilla.mozilla.org/enter_bug.cgi?product=Bugzilla).

This document was heavily modified again Wednesday, March 07 2001 to reflect changes for Bugzilla 2.12 release by Matthew P. Barnson. The securing MySQL section should be changed to become standard procedure for Bugzilla installations.

Finally, the README in its entirety was marked up in SGML and included into the Guide on April 24, 2001 by Matt Barnson. Since that time, it's undergone extensive modification as Bugzilla grew.

Comments from people using this Guide for the first time are particularly welcome.

3.6. Win32 Installation Notes

This section covers installation on Microsoft Windows 95, 98, ME, NT, and 2000. Bugzilla works fine on Win32 platforms, but please remember that the Bugzilla team and the author of the Guide neither endorse nor support installation on Microsoft Windows. Bugzilla installs and runs *best* and *easiest* on UNIX−like operating systems, and that is the way it will stay for the foreseeable future. The Bugzilla team is considering supporting Win32 for the 2.16 release and later.

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.

3.6.1. Win32 Installation: Step−by−step

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 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".

   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.

- Install ActivePerl for Windows. Check 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, and Perl on Win32 FAQ

- Use ppm from your perl\bin directory to install the following packs: DBI, DBD−Mysql, TimeDate, Chart, Date−Calc, Date−Manip, and GD. You may need to extract them from .zip format using Winzip or other unzip program first. These additional ppm modules can be downloaded from ActiveState.

  > You can find a list of modules at http://www.activestate.com/PPMPackages/zips/5xx−builds−only/

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

Example 3−3. Installing ActivePerl ppd Modules on Microsoft Windows

`C:>`**`ppm DBD−Mysql`**

Watch your capitalization!

You can find ActiveState ppm modules at http://www.activestate.com/PPMPackages/5.6plus

- Install MySQL for NT.

  > You can download MySQL for Windows NT from MySQL.com. Some find it helpful to use the WinMySqlAdmin utility, included with the download, to set up the database.

- 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.
-

`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.

- `mysql>` **`FLUSH PRIVILEGES;`**
- `mysql>` **`create database bugs;`**
- `mysql>` **`exit;`**
- `C:>` **`C:\mysql\bin\mysqladmin −u root −p reload`**
- 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'
```

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

  Not sure on the "8" for `$webservergroup` above. If it's wrong, please send corrections.
- Edit `defparams.pl` to suit your requirements. Particularly, set `DefParam("maintainer")` and `DefParam("urlbase") to match your install.`

  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.
- 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, Windmail, Mercury Sendmail, 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. 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;
     ```

     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);
```

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.

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

  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.

  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;
```

- 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.

- Modify the invocation of all system() calls in all perl scripts in your Bugzilla directory. For instance, change this line in processmail:

```
system ("./processmail.pl",@ARGLIST);
```

to

```
system ("perl processmail.pl",@ARGLIST);
```

- Add `binmode()` calls so attachments will work (bug 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);
```

According to bug 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 aribtrarily putting `binmode()` at the begining of the attachment files, there should be logic to determine if `binmode()` is needed or not.

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.

3.6.2. Additional Windows Tips

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

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.

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 2.14.

Example 3−4. 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.

3.6.3. Bugzilla LDAP Integration

What follows is some late−breaking information on using the LDAP authentication options with Bugzilla. The author has not tested these (nor even formatted this section!) so please contribute feedback to the newsgroup.

```
Mozilla::LDAP module

The Mozilla::LDAP module allows you to use LDAP for authentication to
the Bugzilla system.  This module is not required if you are not using
LDAP.

Mozilla::LDAP (aka PerLDAP) is available for download from
http://www.mozilla.org/directory.

NOTE: The Mozilla::LDAP module requires Netscape's Directory SDK.
Follow the link for "Directory SDK for C" on that same page to
download the SDK first.  After you have installed this SDK, then
install the PerLDAP module.
----------------------------------------------------------------------

Post-Installation Checklist
```

```
----------------------------------------------------------------------
Set 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.
----------------------------------------------------------------------

(Not sure where this bit should go, but it's important that it be in
there somewhere...)
----------------------------------------------------------------------
Using LDAP authentication for Bugzilla:

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.
----------------------------------------------------------------------
```

Chapter 4. Administering Bugzilla

Or, I just got this cool thing installed. Now what the heck do I do with it?

So you followed "Bugzilla Installation" to the letter, and logged into Bugzilla for the very first time with your super−duper god account. You sit, contentedly staring at the Bugzilla Query Screen, the worst of the whole mad business of installing this terrific program behind you. It seems, though, you have nothing yet to query! Your first act of business should be to setup the operating parameters for Bugzilla so you can get busy getting data into your bug tracker.

4.1. Post−Installation Checklist

After installation, follow the checklist below to help ensure that you have a successful installation. If you do not see a recommended setting for a parameter, consider leaving it at the default while you perform your initial tests on your Bugzilla setup.

1. Bring up `editparams.cgi` in your web browser. This should be available as the "edit parameters" link from any Bugzilla screen once you have logged in.
2. The "maintainer" is the email address of the person responsible for maintaining this Bugzilla installation. The maintainer need not be a valid Bugzilla user. Error pages, error emails, and administrative mail will be sent with the maintainer as the return email address.

   Set "maintainer" to *your* email address. This allows Bugzilla's error messages to display your email address and allow people to contact you for help.
3. The "urlbase" 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" is http://www.foo.com/bugzilla/.
4. "usebuggroups" dictates whether or not to implement group−based security for Bugzilla. If set, Bugzilla bugs can have an associated groupmask defining which groups of users are allowed to see and edit the bug.

   Set "usebuggroups" to "on" *only* if you may wish to restrict access to products. I suggest leaving this parameter *off* while initially testing your Bugzilla.
5. "usebuggroupsentry", when set to "on", requires that all bugs have an associated groupmask when submitted. This parameter is made for those installations where product isolation is a necessity.

   Set "usebuggroupsentry" to "on" if you absolutely need to restrict access to bugs from the moment they are submitted through resolution. Once again, if you are simply testing your installation, I suggest against turning this parameter on; the strict security checking may stop you from being able to modify your new entries.
6. 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.

Set "shadowdb" to "bug_shadowdb" if you will be running a *very* large installation of Bugzilla. The shadow database enables many simultaneous users to read and write to the database without interfering with one another.

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".

Once again, in testing you should avoid this option −− use it if or when you *need* to use it, and have repeatedly run into the problem it was designed to solve −− very long wait times while attempting to commit a change to the database. Mozilla.org began needing "shadowdb" when they reached around 40,000 Bugzilla users with several hundred Bugzilla bug changes and comments per day.

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!

- "headerhtml", "footerhtml", "errorhtml", "bannerhtml", and "blurbhtml" are all templates which control display of headers, footers, errors, banners, and additional data. We could go into some detail regarding the usage of these, but it is really best just to monkey around with them a bit to see what they do. I strongly recommend you copy your `data/params` file somewhere safe before playing with these values, though. If they are changed dramatically, it may make it impossible for you to display Bugzilla pages to fix the problem until you have restored your `data/params` file.

If you have custom logos or HTML you must put in place to fit within your site design guidelines, place the code in the "headerhtml", "footerhtml", "errorhtml", "bannerhtml", or "blurbhtml" text boxes.

The "headerhtml" text box is the HTML printed out *before* any other code on the page, except the CONTENT−TYPE header sent by the Bugzilla engine. If you have a special banner, put the code for it in "bannerhtml". You may want to leave these settings at the defaults initially.

- "passwordmail" is rather simple. Every time a user creates an account, the text of this parameter is read as the text to send 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.

- "useqacontact" 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. The critical difference between a QA Contact and an Owner is that the QA Contact follows the component. If you reassign a bug from component A to component B, the QA Contact for that bug will change with the reassignment, regardless of owner.

"usestatuswhiteboard" 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. Many people will put "help wanted", "stalled", or "waiting on reply from somebody" messages into the Status Whiteboard field so those who peruse the bugs are aware of their status even more than that which can be indicated by the Resolution fields.

Do you want to use the QA Contact ("useqacontact") and status whiteboard ("usestatuswhiteboard") fields? These fields are useful because they allow for more flexibility, particularly when you have an existing Quality Assurance and/or Release Engineering team, but they may not be needed for many smaller installations.

- Set "whinedays" to the amount 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).
- "commenton" 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.

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!)

- The "supportwatchers" option can be an exceptionally powerful tool in the hands of a power Bugzilla user. By enabling this option, you allow users to receive email updates whenever other users receive email updates. 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 priveleges. She would still only receive email updates for those bugs she could normally view.

For Bugzilla sites which require strong inter−Product security to prevent snooping, watchers are not a good idea.

However, for most sites you should set "supportwatchers" to "On". This feature is helpful for team leads to monitor progress in their respective areas, and can offer many other benefits, such as allowing a developer to pick up a former engineer's bugs without requiring her to change all the information in the bug.

4.2. User Administration

User administration is one of the easiest parts of Bugzilla. Keeping it from getting out of hand, however, can become a challenge.

4.2.1. 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 were to delete the "super user" account, re−running checksetup.pl will again prompt you for this username and password.

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>" denotes the mysql prompt, not something you should type in): **`mysql>` use bugs; `mysql>` update profiles set groupset=0x7ffffffffffffff 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.

4.2.2. Managing Other Users

4.2.2.1. Logging In

1. Open the index.html page for your Bugzilla installation in your browser window.
2. Click the "Query Existing Bug Reports" link.
3. Click the "Log In" link at the foot of the page.
4. Type your email address, and the password which was emailed to you when you created your Bugzilla account, into the spaces provided.

Congratulations, you are logged in!

4.2.2.2. Creating new users

Your users can create their own user accounts by clicking the "New Account" link at the bottom of each page. 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.
2. To see a specific user, type a portion of their login name in the box provided and click "submit". To see all users, simply click the "submit" button. You must click "submit" here to be able to add a new user.

More functionality is available via the list on the right−hand side of the text entry box. You can match what you type as a case−insensitive substring (the default) of all users on your system, a case−sensitive regular expression (please see the **man regexp** manual page for details on regular expression syntax), or a *reverse* regular expression match, where every user name which does NOT match the regular expression is selected.

- Click the "Add New User" link at the bottom of the user list
- Fill out the form presented. This page is self–explanatory. When done, click "submit".

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.

4.2.2.3. Disabling Users

I bet you noticed that big "Disabled Text" entry box available from the "Add New User" screen, when you edit an account? By entering any text in this box and selecting "submit", you have prevented the user from using Bugzilla via the web interface. Your explanation, written in this text box, will be presented to the user the next time she attempts to use the system.

Don't disable your own administrative account, or you will hate life!

At this time, "Disabled Text" does not prevent a user from using the email interface. If you have the email interface enabled, they can still continue to submit bugs and comments that way. We need a patch to fix this.

4.2.2.4. Modifying Users

Here I will attempt to describe the function of each option on the Edit User screen.

- *Login Name*: This is generally the user's email address. However, if you have edited your system parameters, this may just be the user's login name or some other identifier.

For compatability reasons, you should probably stick with email addresses as user login names. It will make your life easier.

- *Real Name*: Duh!
- *Password*: You can change the user password here. It is normal to only see asterisks.
- *Email Notification*: You may choose from one of three options:

1. All qualifying bugs except those which I change: The user will be notified of any change to any bug for which she is the reporter, assignee, QA Contact, CC recipient, or "watcher".
2. Only those bugs which I am listed on the CC line: The user will not be notified of changes to bugs where she is the assignee, reporter, or QA Contact, but will receive them if she is on the CC list.

> She will still receive whining cron emails if you set up the "whinemail" feature.

* *All Qualifying Bugs*: This user is a glutton for punishment. If her name is in the reporter, QA Contact, CC, assignee, or is a "watcher", she will get email updates regarding the bug.

Disable Text: If you type anything in this box, including just a space, the user account is disabled from making any changes to bugs via the web interface, and what you type in this box is presented as the reason.

> Don't disable the administrator account!

> As of this writing, the user can still submit bugs via the e−mail gateway, if you set it up, despite the disabled text field. The e−mail gateway should *not* be enabled for secure installations of Bugzilla.

* *CanConfirm*: This field is only used if you have enabled "unconfirmed" status in your parameters screen. If you enable this for a user, that user can then move bugs from "Unconfirmed" to "Confirmed" status (e.g.: "New" status). Be judicious about allowing users to turn this bit on for other users.
* *Creategroups*: This option will allow a user to create and destroy groups in Bugzilla. Unless you are using the Bugzilla GroupSentry security option "usebuggroupsentry" in your parameters, this setting has no effect.
* *Editbugs*: Unless a user has this bit set, they can only edit those bugs for which they are the assignee or the reporter.

> Leaving this option unchecked does not prevent users from adding comments to a bug! They simply cannot change a bug priority, severity, etc. unless they are the assignee or reporter.

* *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. The name of a product or component can be changed without affecting the associated bugs, but it tends to annoy the hell out of your users when these change a lot.
* *Editkeywords*: If you use Bugzilla's keyword functionality, enabling this feature allows a user can 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. You must be very careful about creating too many new keywords if you run a very large Bugzilla installation; keywords are global variables across products, and you can often run into a phenomenon called "keyword bloat". This confuses users, and then the feature goes unused.
* *Editusers*: This flag allows a user do what you're doing right now: edit other users. This will allow those with the right to do so to remove administrator priveleges from other users or grant them to themselves. Enable with care.

- *PRODUCT*: PRODUCT bugs access. This allows an administrator, with product−level granularity, to specify in which products a user can edit bugs. The user must still have the "editbugs" privelege to edit bugs in this area; this simply restricts them from even seeing bugs outside these boundaries if the administrator has enabled the group sentry parameter "usebuggroupsentry". Unless you are using bug groups, this option has no effect.

4.3. Product, Component, Milestone, and Version Administration

Dear Lord, we have to get our users to do WHAT?

4.3.1. Products

Formerly, and in some spots still, called "Programs"

Products are the broadest category in Bugzilla, and you should have the least of these. If your company makes computer games, you should have one product per game, and possibly a few special products (website, meetings...)

A Product (formerly called "Program", and still referred to that way in some portions of the source code) controls some very important functions. The number of "votes" available for users to vote for the most important bugs is set per−product, as is the number of votes required to move a bug automatically from the UNCONFIRMED status to the NEW status. One can close a Product for further bug entry and define various Versions available from the Edit product screen.

To create a new product:

1. Select "components" from the yellow footer

It may seem counterintuitive to click "components" when you want to edit the properties associated with Products. This is one of a long list of things we want in Bugzilla 3.0...

- Select the "Add" link to the right of "Add a new product".
- Enter the name of the product and a description. The Description field is free−form.

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.

4.3.2. Components

Components are subsections of a Product.

Example 4−1. Creating some Components

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*; the Owner and QA Contact fields in a bug are otherwise unrelated to the Component.

To create a new Component:

1. Select the "Edit components" link from the "Edit product" page
2. Select the "Add" link to the right of the "Add a new component" text on the "Select Component" page.
3. Fill out the "Component" field, a short "Description", and the "Initial Owner". The Component and Description fields are free−form; the "Initial Owner" field must be that of a user ID already existing in the database. If the initial owner does not exist, Bugzilla will refuse to create the component.

> Is your "Default Owner" a user who is not yet in the database? No problem.
>
> a. Select the "Log out" link on the footer of the page.
> b. Select the "New Account" link on the footer of the "Relogin" page
> c. Type in the email address of the default owner you want to create in the "E−mail address" field, and her full name in the "Real name" field, then select the "Submit Query" button.
> d. Now select "Log in" again, type in your login information, and you can modify the product to use the Default Owner information you require.

- Either Edit more components or return to the Bugzilla Query Page. To return to the Product you were editing, you must select the Components link as before.

4.3.3. Versions

Versions are the revisions of the product, such as "Flinders 3.1", "Flinders 95", and "Flinders 2000". Using Versions helps you isolate code changes and are an aid in reporting.

Example 4−2. Common Use of Versions

A user reports a bug against Version "Beta 2.0" of your product. The current Version of your software is "Release Candidate 1", and no longer has the bug. This will help you triage and classify bugs according to their relevance. It is also possible people may report bugs against bleeding−edge beta versions that are not evident in older versions of the software. This can help isolate code changes that caused the bug

Example 4−3. A Different Use of Versions

This field has been used to good effect by an online service provider in a slightly different way. They had three versions of the product: "Production", "QA", and "Dev". Although it may be the same product, a bug in the development environment is not normally as critical as a Production bug, nor does it need to be reported publicly. When used in conjunction with Target Milestones, one can easily specify the environment where a bug can be reproduced, and the Milestone by which it will be fixed.

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". If your product doesn't use version numbers, you may want to leave this as it is or edit it so that it is "−−−". You can then go back to the edit versions page and add new versions to your product.

   Otherwise, click the "Add" button to the right of the "Add a new version" text.
3. Enter the name of the Version. This can be free−form characters up to the limit of the text box. Then select the "Add" button.
4. At this point you can select "Edit" to edit more Versions, or return to the "Query" page, from which you can navigate back to the product through the "components" link at the foot of the Query page.

4.3.4. 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. Or, you have a bug that you plan to fix for 2.8, this would have a milestone of 2.8.

Milestone options will only appear for a Product if you turned the "usetargetmilestone" field in the "Edit Parameters" screen "On".

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

1. Select "edit milestones"
2. Select "Add" to the right of the "Add a new milestone" 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. Select "Add".

Example 4−4. Using SortKey with Target Milestone

Let's say you create a target milestone called "Release 1.0", with Sortkey set to "0". Later, you realize that you will have a public beta, called "Beta1". You can create a Milestone called "Beta1", with a Sortkey of "−1" in order to ensure people will see the Target Milestone of "Beta1" earlier on the list than "Release 1.0"

* If you want to add more milestones, select the "Edit" link. If you don't, well shoot, you have to go back to the "query" page and select "components" again, and make your way back to the Product you were editing.

  This is another in the list of unusual user interface decisions that we'd like to get cleaned up. Shouldn't there be a link to the effect of "edit the Product I was editing when I ended up here"? In any case, clicking "components" in the footer takes you back to the "Select product" screen, from which you can begin editing your product again.
* From the Edit product screen again (once you've made your way back), enter the URL for a description of what your milestones are for this product in the "Milestone URL" field. It should be of the format "http://www.foo.com/bugzilla/product_milestones.html"

Some common uses of this field include product descriptions, product roadmaps, and of course a simple description of the meaning of each milestone.

* If you're using Target Milestones, the "Default Milestone" field must have some kind of entry. If you really don't care if people set coherent Target Milestones, simply leave this at the default, "−−−". However, controlling and regularly updating the Default Milestone field is a powerful tool when reporting the status of projects.

Select the "Update" button when you are done.

4.3.5. Voting

The concept of "voting" is a poorly understood, yet powerful feature for the management of open−source projects. Each user is assigned so many Votes per product, which they can freely reassign (or assign multiple votes to a single bug). 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.

The daunting challenge of Votes is deciding where you draw the line for a "vocal majority". If you only have a user base of 100 users, setting a low threshold for bugs to move from UNCONFIRMED to NEW makes sense. As the Bugzilla user base expands, however, these thresholds must be re−evaluated. You should gauge whether this feature is worth the time and close monitoring involved, and perhaps forego implementation until you have a critical mass of users who demand it.

To modify Voting settings:

1. Navigate to the "Edit product" screen for the Product you wish to modify
2. Set "Maximum Votes per person" to your calculated value. Setting this field to "0" disables voting.
3. Set "Maximum Votes a person can put on a single bug" to your calculated value. It should probably be some number lower than the "Maximum votes per person". Setting this field to "0" disables voting, but leaves the voting options open to the user. This is confusing.
4. Set "Number of votes a bug in this product needs to automatically get out of the UNCONFIRMED state" to your calculated number. Setting this field to "0" disables the automatic move of bugs from UNCONFIRMED to NEW. Some people advocate leaving this at "0", but of what use are Votes if your Bugzilla user base is unable to affect which bugs appear on Development radar?

You should probably set this number to higher than a small coalition of Bugzilla users can influence it. Most sites use this as a "referendum" mechanism −− if users are able to vote a bug out of UNCONFIRMED, it is a *really* bad bug!

- Once you have adjusted the values to your preference, select the "Update" button.

4.3.6. Groups and Group Security

Groups can be very useful in bugzilla, because they allow users to isolate bugs or products that should only be seen by certain people. Groups can also be a complicated minefield of interdependencies and weirdness if mismanaged.

Example 4−5. When to Use Group Security

Many Bugzilla sites isolate "Security−related" bugs from all other bugs. This way, they can have a fix ready before the security vulnerability is announced to the world. You can create a "Security" product which, by default, has no members, and only add members to the group (in their individual User page, as described under User Administration) who should have priveleged access to "Security" bugs. Alternately, you may create a Group independently of any Product, and change the Group mask on individual bugs to restrict access to members only of certain Groups.

Groups only work if you enable the "usebuggroups" paramater. In addition, if the "usebuggroupsentry" parameter is "On", one can restrict access to products by groups, so that only members of a product group are able to view bugs within that product. Group security in Bugzilla can be divided into two categories: Generic and Product−Based.

> Groups in Bugzilla are a complicated beast that evolved out of very simple user permission bitmasks, apparently itself derived from common concepts in UNIX access controls. A "bitmask" is a fixed−length number whose value can describe one, and only one, set of states. For instance, UNIX file permissions are assigned bitmask values: "execute" has a value of 1, "write" has a value of 2, and "read" has a value of 4. Add them together, and a file can be read, written to, and executed if it has a bitmask of "7". (This is a simplified example −− anybody who knows UNIX security knows there is much more to it than this. Please bear with me for the purpose of this note.) The only way a bitmask scheme can work is by doubling the bit count for each value. Thus if UNIX wanted to offer another file permission, the next would have to be a value of 8, then the next 16, the next 32, etc.
>
> Similarly, Bugzilla offers a bitmask to define group permissions, with an internal limit of 64. Several are already occupied by built−in permissions. The way around this limitation is to avoid assigning groups to products if you have many products, avoid bloating of group lists, and religiously prune irrelevant groups. In reality, most installations of Bugzilla support far fewer than 64 groups, so this limitation has not hit for most sites, but it is on the table to be revised for Bugzilla 3.0 because it interferes with the security schemes of some administrators.

To enable Generic Group Security ("usebuggroups"):

1. Turn "On" "usebuggroups" in the "Edit Parameters" screen.
2. You will generally have no groups set up. Select the "groups" link in the footer.
3. Take a moment to understand the instructions on the "Edit Groups" screen. Once you feel confident you understand what is expected of you, select the "Add Group" link.
4. Fill out the "New Name" (remember, no spaces!), "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.

Example 4−6. Creating a New Group

I created a group called DefaultGroup with a description of "This is simply a group to play with", and a New User RegExp of ".*@mydomain.tld". This new group automatically includes all Bugzilla users with "@mydomain.tld" at the end of their user id. When I finished, my new group was assigned bit #128.

When you have finished, select the Add button.

To enable Product−Based Group Security (usebuggroupsentry):

Don't forget that you only have 64 groups masks available, total, for your installation of Bugzilla! If you plan on having more than 50 products in your individual Bugzilla installation, and require group security for your products, you should consider either running multiple Bugzillas or using Generic Group Security instead of Product−Based ("usebuggroupsentry") Group Security.

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

"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.

- You will generally have no Groups set up, unless you enabled "usebuggroupsentry" prior to creating any Products. To create "Generic Group Security" groups, follow the instructions given above. To create Product−Based Group security, simply follow the instructions for creating a new Product. If you need to add users to these new groups as you create them, you will find the option to add them to the group available under the "Edit User" screens.

You may find this example illustrative for how bug groups work.

Example 4−7. Bugzilla Groups

```
Bugzilla Groups example
-----------------------

For this example, let us suppose we have four groups, call them
Group1, Group2, Group3, and Group4.

We have 5 users, User1, User2, User3, User4, User5.

We have 8 bugs, Bug1, ..., Bug8.

Group membership is defined by this chart:
(X denotes that user is in that group.)
(I apologize for the nasty formatting of this table.  Try viewing
it in a text-based browser or something for now. -MPB)

      G G G G
      r r r r
      o o o o
      u u u u
      p p p p
```

```
      1 2 3 4
     +-+-+-+-+
User1|X| | | |
     +-+-+-+-+
User2| |X| | |
     +-+-+-+-+
User3|X| |X| |
     +-+-+-+-+
User4|X|X|X| |
     +-+-+-+-+
User5| | | | |
     +-+-+-+-+

Bug restrictions are defined by this chart:
(X denotes that bug is restricted to that group.)

     G G G G
     r r r r
     o o o o
     u u u u
     p p p p
     1 2 3 4
    +-+-+-+-+
Bug1| | | | |
    +-+-+-+-+
Bug2| |X| | |
    +-+-+-+-+
Bug3| | |X| |
    +-+-+-+-+
Bug4| | | |X|
    +-+-+-+-+
Bug5|X|X| | |
    +-+-+-+-+
Bug6|X| |X| |
    +-+-+-+-+
Bug7|X|X|X| |
    +-+-+-+-+
Bug8|X|X|X|X|
    +-+-+-+-+

Who can see each bug?

Bug1 has no group restrictions.  Therefore, Bug1 can be seen by any
user, whatever their group membership.  This is going to be the only
bug that User5 can see, because User5 isn't in any groups.

Bug2 can be seen by anyone in Group2, that is User2 and User4.

Bug3 can be seen by anyone in Group3, that is User3 and User4.

Bug4 can be seen by anyone in Group4.  Nobody is in Group4, so none of
```

```
these users can see Bug4.

Bug5 can be seen by anyone who is in _both_ Group1 and Group2.  This
is only User4.  User1 cannot see it because he is not in Group2, and
User2 cannot see it because she is not in Group1.

Bug6 can be seen by anyone who is in both Group1 and Group3.  This
would include User3 and User4.  Similar to Bug5, User1 cannot see Bug6
because he is not in Group3.

Bug7 can be seen by anyone who is in Group1, Group2, and Group3.  This
is only User4.  All of the others are missing at least one of those
group priveleges, and thus cannot see the bug.

Bug8 can be seen by anyone who is in Group1, Group2, Group3, and
Group4.  There is nobody in all four of these groups, so nobody can
see Bug8.  It doesn't matter that User4 is in Group1, Group2, and
Group3, since he isn't in Group4.
```

4.4. Bugzilla Security

Putting your money in a wall safe is better protection than depending on the fact that no one knows that you hide your money in a mayonnaise jar in your fridge.

Poorly−configured MySQL, Bugzilla, and FTP 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.

Secure your installation.

These instructions must, of necessity, be somewhat vague since Bugzilla runs on so many different platforms. If you have refinements of these directions for specific platforms, please submit them to mozilla−webtools@mozilla.org

1. Ensure you are running at least MysQL version 3.22.32 or newer. Earlier versions had notable security holes and poorly secured default configuration choices.
2. *There is no substitute for understanding the tools on your system!* Read The MySQL Privilege System until you can recite it from memory!

   At the very least, ensure you password the "mysql −u root" account and the "bugs" account, establish grant table rights (consult the Keystone guide in Appendix C: The Bugzilla Database for some easy−to−use details) that do not allow CREATE, DROP, RELOAD, SHUTDOWN, and PROCESS for user "bugs". I wrote up the Keystone advice back when I knew far less about security than I do now :)

3. Lock down /etc/inetd.conf. Heck, disable inet entirely on this box. It should only listen to port 25 for Sendmail and port 80 for Apache.
4. Do not run Apache as "nobody". This will require very lax permissions in your Bugzilla directories. Run it, instead, as a user with a name, set via your httpd.conf file.

   "nobody" is a real user on UNIX systems. Having a process run as user id "nobody" is absolutely no protection against system crackers versus using any other user account. As a general security measure, I recommend you create unique user ID's for each daemon running on your system and, if possible, use "chroot" to jail that process away from the rest of your system.

- Ensure you have adequate access controls for the $BUGZILLA_HOME/data/ and $BUGZILLA_HOME/shadow/ directories, as well as the $BUGZILLA_HOME/localconfig and $BUGZILLA_HOME/globals.pl files. The localconfig file stores your "bugs" user password, which would be terrible to have in the hands of a criminal, while the "globals.pl" stores some default information regarding your installation which could aid a system cracker. In addition, some files under $BUGZILLA_HOME/data/ store sensitive information, and $BUGZILLA_HOME/shadow/ stores bug information for faster retrieval. If you fail to secure these directories and this file, you will expose bug information to those who may not be allowed to see it.

  Bugzilla provides default .htaccess files to protect the most common Apache installations. However, you should verify these are adequate according to the site−wide security policy of your web server, and ensure that the .htaccess files are allowed to "override" default permissions set in your Apache configuration files. Covering Apache security is beyond the scope of this Guide; please consult the Apache documentation for details.

  If you are using a web server that does not support the .htaccess control method, *you are at risk!* After installing, check to see if you can view the file "localconfig" in your web browser (e.g.: http://bugzilla.mozilla.org/localconfig). If you can read the contents of this file, your web server has not secured your bugzilla directory properly and you must fix this problem before deploying Bugzilla. If, however, it gives you a "Forbidden" error, then it probably respects the .htaccess conventions and you are good to go.

On Apache, you can use .htaccess files to protect access to these directories, as outlined in Bug 57161 for the localconfig file, and Bug 65572 for adequate protection in your data/ and shadow/ directories.

Note the instructions which follow are Apache−specific. If you use IIS, Netscape, or other non−Apache web servers, please consult your system documentation for how to secure these files from being transmitted to curious users.

Place the following text into a file named ".htaccess", readable by your web server, in your $BUGZILLA_HOME/data directory.

```
<Files comments> allow
      from all </Files> deny from all
```

Place the following text into a file named ".htaccess", readable by your web server, in your $BUGZILLA_HOME/ directory.

```
<Files localconfig> deny
      from all </Files> allow from all
```

Place the following text into a file named ".htaccess", readable by your web server, in your $BUGZILLA_HOME/shadow directory.

```
deny from all
```

Chapter 5. Integrating Bugzilla with Third−Party Tools

5.1. 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. These kinds of changes cause the engineer responsible to be "on the hook" (include cool URL link here for Hook policies at mozilla.org). Bonsai also includes gateways to Tinderbox, the Mozilla automated build management system and Bugzilla

5.2. CVS

CVS integration is best accomplished, at this point, using the Bugzilla Email Gateway. There have been some files submitted to allow greater CVS integration, but we need to make certain that Bugzilla is not tied into one particular software management package.

Follow the instructions in the FAQ 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 have your check−in script include an @resolution field, you can even change the Bugzilla bug state.

There is also a 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/, under the "cvszilla" link.

5.3. Perforce SCM

You can find the project page for Bugzilla and Teamtrack Perforce integration (p4dti) at: 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.

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.

5.4. Tinderbox/Tinderbox2

We need Tinderbox integration information.

Chapter 6. The Future of Bugzilla

```
Bugzilla's Future.  Much of this is the present, now.
```

Bugzilla's future is a constantly−changing thing, as various developers "scratch an itch" when it comes to functionality. Thus this section is very malleable, subject to change without notice, etc. You'll probably also notice the lack of formatting. I apologize that it's not quite as readable as the rest of the Guide.

```
                              Bugzilla Blue Sky

Customisability

   One of the major stumbling blocks of Bugzilla has been that it is too
   rigid and does not adapt itself well enough to the needs of an
   organisation.  This has led to organisations making changes to the
   Bugzilla code that need to be redone each new version of Bugzilla.
   Bugzilla should attempt to move away from this to a world where this
   doesn't need to occur.

   Most of the subsections in this section are currently explicit design
   goals for the "Bugzilla 3" rewrite.  This does not necessarily mean
   that they will not occur before them in Bugzilla 2, but most are
   significant undertakings.

  Field Customisation

   Many installations wish to customise the fields that appear on bug
   reports.   Current versions of Bugzilla offer limited
   customisability.  In particular, some fields can be turned off.

   However, many administrators wish to add their own fields, and rename
   or otherwise modify existing fields.  An architecture that supports
   this would be extraordinarily useful.

   Indeed, many fields work similarly and could be abstracted into "field
   types", so that an administrator need write little or no code to
   support the new fields they desire.

   Possible field types include text (eg status whiteboard), numbers,
   dates (eg report time), accounts (eg reporter, qa, cc), inter-bug
   relationships (dependencies, duplicates), option groups (platform, os,
   severity, priority, target milestone, version) etc.

   Ideally an administrator could configure their fields through a
   Bugzilla interface that requires no code to be added.  However, it is
   highly unlikely this ideal will never be met, and in a similar way
   that office applications have scripting languages, Bugzilla should
   allow new field types to be written.

   Similarly, a common desire is for resolutions to be added or removed.
```

```
  Allocations

 ?

  Option Groups

 ?

  Relations

 ?

Database Integrity

 Furthermore, it is desirable for administrators to be able to specify
 rules that must or should apply between the fields on a bug report.

 For example, you might wish to specify that a bug with status ASSIGNED
 must have a target milestone field that that is not untargetted.  Or
 that a bug with a certain number of votes should get ASSIGNED.  Or
 that the QA contact must be different from the assignee.

 "Must" relationships could be implemented by refusing to make changes
 that violate the relationships, or alternatively, automatically
 updating certain fields in order to satisfy the criteria.  Which
 occurs should be up to the administrator.

 "Should" relationships could be implemented by a combination of
 emitting warnings on the process bug page, the same on notification
 mails, or emitting periodic whine mails about the situation.  Again,
 which occurs should be up to the administrator.

 It should also be possible for whine mails to be emitted for "must"
 relationships, as they might become violated through direct database
 access, Bugzilla bugs, or because they were there before the
 relationship was enforced.

 As well as implementing intra-bug constraints, it would be useful to
 create inter-bug constraints.  For example, a bug that is dependent on
 another bug should not have an earlier milestone or greater priority
 than that bug.

Database Adaptability

 Often an administrator desires that fields adapt to the values of
 other fields.  For example, the value of a field might determine the
 possible values of another field or even whether it appears (whether
 it is "applicable").

 Limited adaptability is present in Bugzilla 2, and only on the
```

"Product" field:
* The possible values of the target milestone, version and component fields depend on the product.
* UNCONFIRMED can be turned off for specific products.
* Voting can be configured differently or turned off for different products, and there is a separate user vote limits for each product.

It would be good if more adaptability was present, both in terms of all fields relying on the product, as well as the ability to adapt based on the value of all fields.

Example ???

General adaptability raises the issue of circular references between fields causing problems. One possible solution to this is to place the fields in a total ordering and require a field refer only to the previous fields.

In Bugzilla 2, changing the product of a bug meant a second page would appear that allowed you to choose a new milestone, component and version, as those fields adapted themselves to the new product. This page could be generalised to support all instances where:
* a field value must or might be changed because the possible values have changed
* is going to drop off because it it is no longer applicable, and this should be confirmed
* must be specified because it is suddenly applicable, and the default value, if one exists, might not be acceptable

Database Independence

Currently Bugzilla only runs on the MySQL database. It would be desirable for Bugzilla to run on other databases, because:
* Organisations may have existing database products they use and would prefer to run a homogenous environment.
* Databases each have their own shortcomings, including MySQL. An administrator might choose a database that would work better with their Bugzilla.

This raises the possibility that we could use features that are only present in some databases, by appropriately falling back. For example, in the MySQL world, we live without:
* record-level locking, instead we use table-level locking
* referential and record constraints, instead we checking code
* subselects, instead we use multiple queries and redundant "caches"

Multiple Front Ends

Currently Bugzilla is manipulated via the Web, and notifies via E-Mail. It would be desirable for Bugzilla to easily support various front ends.

There is no reason that Bugzilla could not be controlled via a whole range of front ends, including Web, E-Mail, IRC, ICQ, etc, and similarly for how it notifies. It's also possible that we could introduce a special Bugzilla client that uses its own protocol, for maximum user productivity.

Indeed a request reply might be returned via a totally different transport method than was use to submit the request.

Internationalisation

Bugzilla currently supports only English. All of the field names, user instructions, etc are written in English. It would be desirable to allow "language packs" so Bugzilla can be easily used in non-English speaking locales.

To a degree field customisation supports this, because administrators could specify their own fields names anyway. However, there will always be some basic facilities not covered by this, and it is desirable that the administrator's interface also is internationalisable.

Better Searching

General Summary Reports

Sometimes, the normal querying page leaves a lot to be desired. There are other facilities already in place or which people have asked for:

Most Doomed Reports - All Bugs or All Bugs In A Product, Categorised On Assignee, Shows and Counts Number of Bugs For Each Assignee
Most Voted For Bugs - All Bugs, Categorised On Product, Shows Top Ten Bugs Voters Most Want Fixed
Number of Open Bugs For An Assignee - Bug List, Categorised On Developers, Counts Number of Bugs In Category

The important thing to realise is that people want categorised reports on all sorts of things - a general summary report.

In a categorised report, you choose the subset of bugs you wish to operate on (similar to how you would specify a query), and then categorise them on one or more fields.

For each category you display the count of the number of things in that category. You can optionally display the bugs themselves, or leave them out, just showing the counts. And you can optionally limit the number of things (bugs or subcategories) that display in each category.

Such a mechanism would let you do all of the above and more. Applications of this mechanism would only be recognised once it was implemented.

Related Bugs

It would be nice to have a field where you could enter other bugs related to the current bug. It would be handy for navigation and possibly even finding duplicates.

Column Specification Support

Currently bug lists use the columns that you last used. This doesn't work well for "prepackaged queries", where you followed a link. You can probably add a column by specifying a sort column, but this is difficult and suboptimal.

Furthermore, I find that when I want to add a column to a bug list, it's usually a one off and I would prefer it to go away for the next query. Hence, it would be nice to specify the columns that appear on the bug list (and general summary report) pages. The default query mechanism should be able to let you specify your default columns.

Advanced Querying Redesign

?

Keywords

People have a need to apply tags to bugs. In the beginning, people placed designators in the summary and status whiteboard. However, these fields were not designed for that, and so there were many flaws with this system:

* They pollute the field with information that was never intended to be present.
* Removing them with a bulk change is a difficult problem that has too many pitfalls to implement.
* You can easily get the capitalisation wrong.

Then dependencies were introduced (when?), and people realised that they could use them for "tracking bugs". Again, dependencies were not designed for that, and so there were more flaws, albeit different ones, including:

* They aren't really bugs, so it's difficult to distinguish issues from bugs.
* They can pollute bugs counts, and you must somehow exclude them from queries.
* There is a whole lot of useless information on them. They have an assignee but there is nothing to fix, and that person can get whined at by Bugzilla. They have target milestones which must be manually maintained. And so on.

Finally, keywords were introduced (when?) for this purpose to remove the need for these two systems. Unfortunately, the simple keywords implementation was itself lacking in certain features provided by the two previous systems, and has remained almost unchanged since its inception. Furthermore, it could not be forseen that in large installations, the sheer number of keywords could become unwieldly and could lead to a movement back to the other systems.

The keywords system was the right idea, however, and it remains so. Fixing the keywords system is one of the most important Bugzilla issues.

Bringing Keywords Up To Par

For the most part, keywords are very good at what they do. It is easy to add and remove them (unlike summary/whiteboard designators), we can simply see what issues are present on a bug (unlike tracking bugs), and we do not confuse bugs with issues (unlike tracking bugs).

However, there are still some "regressions" in the keyword system over previous systems:

* Users wish to view the "dependency forest" of a keyword. While a dependency tree is of one bug, a dependency forest is of a bug list, and consists of a dependency tree for each member of the bug list. Users can work around this with tracking bugs by creating a tracking bug and viewing the dependency tree of that tracking bug.
* Users wish to specify the keywords that initially apply to a bug, but instead they must edit the bug once it has already been submitted. They can work around this with summary designators, since they specify the summary at reporting time.
* Users wish to store or share a bug list that contains a keywords column. Hence they wish to be able to specify what columns appear in the bug list URL, as mentioned earlier. They can work around this using summary designators, since almost all bug lists have a summary column.
* Users wish to be able to view keywords on a bug list. However often they are only interested in a small number of keywords. Having a bug list with a keywords column means that all keywords will appear on a bug list. This can take a substantial amount of space where a bug has a lot of keywords, since the table columns in Bugzilla adjust to the largest cell in that column. Hence users wish to be able to specify which keywords should appear in the bug list. In a very real sense, each keyword is a field unto itself. Users can work around this by using summary designators, since they keywords will share the space in the summary column.
* Users wish to know when bugs with a specific issue are resolved. Hence they wish to be able to receive notifications on all the bugs with a specific keyword. The introduction a generic watching facility (also for things like watching all bugs in a component) would achieve this. Users can work around this by using tracking

bugs, as dependencies have an existing way of detecting fixes to bug a bug was blocked by.

Dealing With The Keyword Overload

At the time of writing, the mozilla.org installation has approximately 100 keywords, and many more would be in use if the keywords system didn't have the problems it does.

Such a large number of keywords introduces logistical problems:

* It must be easy for someone to learn what a keyword means. If a keyword is buried within a lot of other keywords, it can be difficult to find.
* It must be easy to see what keywords are on a bug. If the number of keywords is large, then this can be difficult.

These lead some people to feel that there are "too many keywords".

These problems are not without solutions however. It is harder to find a list of designators or tracking bugs than it is a list of keywords.

The essential problem is it needs to be easy to find the keywords we're interested in through the mass of keywords.

Keyword Applicability

As has been previously mentioned, it is desirable for fields to be able to adapt to the values of other fields. This is certainly true for keywords. Many keywords are simply not relevant because of the bugs product, component, etc.

Hence, by introducing keyword applicability, and not displaying keywords that are not relevant to the current bug, or clearly separating them, we can make the keyword overload problem less significant.

Currently when you click on "keywords" on a bug, you get a list of all bugs. It would be desirable to introduce a list of keywords tailored to a specific bug, that reports, in order:

* the keywords currently on the bug
* the keywords not currently on the bug, but applicable to the bug
* optionally, the keywords not applicable to the bug

This essentially orders the keywords into three groups, where each group is more important than the previous, and therefore appears closer to the top.

Keyword Grouping & Ordering

We could further enhance both the global and bug specific keyword list by grouping keywords. We should always have a "flat" view of keywords, but other ways of viewing the keywords would be useful too.

If keyword applicability was implemented, we could group keywords based on their "applicability condition". Keywords that apply to all bugs could be separated from keywords that apply to a specific product, both on the global keyword list and the keyword list of a bug that is in that product.

We could specify groups of our own. For example, many keywords are in a mutually exclusive group, essentially like radio buttons in a user interface. This creates a natural grouping, although other groupings occur (which depends on your keywords).

It is possible that we could use collapsing/expanding operations on "twisties" to only should the groups we are interested in.

And instead of grouping keywords, we could order them on some metric of usefulness, such as:

* when the keyword was last added to a bug
* how many bugs the keyword is on
* how many open bugs the keyword is on

Opting Out Of Keywords

Not all people are going to care about all keywords. Therefore it makes sense that you may wish to specify which keywords you are interested in, either on the bug page, or on notifications.

Other keywords will therefore not bother users who are not interested in them.

Keyword Security

Currently all keywords are available and editable to all people with edit bugs access. This situation is clearly suboptimal.

Although relying on good behaviour for people to not do what they shouldn't works reasonably well on the mozilla.org, it is better to enforce that behaviour - it can be breached through malice, accident or ignorance.

And in the situation where it is desirable for the presence or absence of a keyword not to be revealed, organisations either need to be content with the divulgence, or not use keywords at all.

In the situation where they choose to divulge, introducing the ability to restrict who can see the keyword would also reduce keyword overload.

Personal Keywords

Keywords join together a set of bugs which would otherwise be unrelated in the bug system.

We allow users to store their own queries. However we don't allow them to store their own keywords on a bug. This reduces the usefulness of personal queries, since you cannot join a set of unrelated bugs together in a way that you wish. Lists of bug numbers can work, by they can only be used for small lists, and it is impossible to share a list between multiple queries.

Personal keywords are necessary to replace personal tracking bugs, as they would not pollute the keyword space. Indeed, on many installations this could remove some keywords out of the global keyword space.

In a similar vein and with similar effects, group keywords could be introduced that are only available to members of a specific group.

Keyword Restrictions

Keywords are not islands unto themselves. Along with their potential to be involved in the inter-field relationships mentioned earlier, keywords can also be related to other keywords.

Essentially, there are two possibilities:
* a set of keywords are mutually exclusive
* the presence of a keyword implies another keyword must be present

Introduction of the ability to specify these restrictions would have benefits.

If mutually exclusive keywords were present on a bug, their removal would fix up the database, as well as reducing the number of keywords on that bug.

In the situation where a keyword implies another keyword, there are two possiblities as to how to handle the situation.

The first is automatically add the keyword. This would fix up the database, but it would increase the number of keywords on a bug.

The second is to automatically remove the keyword, and alter queries so they pick up the first keyword as well as the removed keyword. This would fix up the database and reduce the number of keywords on a bug, but it might confuse users who don't see the keyword. Alternatively, the implied keywords could be listed separately.

Notifications

Every time a bug gets changed notifications get sent out to people letting them know about what changes have been made. This is a significant feature, and all sorts of questions can be raised, but they mainly boil down to when they should be sent and what they should look like.

Changes You're Interested In

As of version 2.12 users can specify what sort of changes they are interested in receiving notifications for. However, this is still limited. As yet there is no facility to specify which keywords you care about, and whether you care about changes to fields such as the QA contact changes.
Furthermore, often an unnecessary comment will go along with a change, either because it is required, or the commenter is ignorant of how the new system works. While explaining why you did something is useful, merely commenting on what you did is not because that information is already accessible view "Bug Activity".

Because of this unnecessary comment, a lot of changes that would otherwise not generate notifications for certain people do so, because few people are willing to turn off comments. One way to deal with this problem is to allow people to specify that their comments are purely explanatory, and that anyone who is not interested in the change will not be interested in the comment.

Furthermore, one possible rationale for unnecessary comments is that the bug activity does not display on the normal page and hence it is difficult to cross reference comments and actions. Hence, it would be beneficial to be able to do this.

Bugs You're Watching

Currently to receive a notification about a bug you need to have your name on it. This is suboptimal because you need to know about a bug before you can receive notifications on it. Often you are interested in any bug with a field set to a specific value. For example, you might be interested in all bugs with a specific product, component or keyword.

If someone could automatically receive notifications about these bugs, it would make everyone's lives easier. Currently the default assignee and QA contact for a component will automatically receive notifications for

Question: This moves half way to a BCC.

Bulk Changes

A very useful feature of Bugzilla is the ability to perform an action on multiple bugs at once. However, this means that similar notifications are currently generated for each bug modified.

This can result in a torrent of notifications that can annoy.

Furthermore, since the bugs are all changed close to each other in time, it is easy for someone to mass delete all the notifications generated by a bulk change and miss an unrelated notification in the middle.

These factors can lead to a tendency for people to delay bulk changes, or avoid them entirely. This is suboptimal.

It would be better if a bulk change generated only one notification mail. This would vastly reduce the annoyance factor, and prevent accidental deletion of notifications.

One problem with this change is that some people separate out notifications using filtering. This means that they would no longer be match parts of a bulk change under different filtering rules.

One possibility to resolve this is to allow people to specify groups of bugs. All bugs within a group would go into the same notification. The filters could then distinguish the different bug groups.

In any case, it is likely there would need to be a transition period to allow people to alter their filters.

Nominations

?

Linking Bugzilla Installations

The first example of linking Bugzilla installations together has is the introduction of bug moving in version 2.12. However, it would be useful to be able to link installations in more ways.

* Dependencies and other relationships between bugs in other installations. This is difficult because dependencies are synchronised on both bugs, so the installation that changes dependencies would need to communicate the new state to the other installation. It would also mean that relationships and notifications that refer to other bugs would need to communicate with the other installation.
* References to bugs in other installations. Currently if you type "bug XXX" or "bug #XXX" where XXX is a number, you get an automatic hyperlink to that bug. It would be useful if you could say "YYY bug #XXX" where YYY is the name of another installation.

Retirement

?

```
Whiny Reports

   ?

  Group Redesign

   ?

  Hard Wrapping Comments

   Currently Bugzilla "hard wraps" its comments to a specific line size,
   similar to E-Mail.  This has various problems:
     * The way it currently works, wrapping is done in the browser at
       submission time using a non-standard HTML extension not supported
       by some (uncommon) browsers.  These browsers generate comments
       that scroll off the right side of the screen.
     * Because comments are of fixed width, when you expand your browser
       window, the comments do not expand to fit available space.

   It would be much better to move to a world of soft wrapping, where the
   browser wraps the text at display time, similar to a world processor.
    And as in a word processor, soft wrapping does not preclude the
   insertion of newlines.

   Hard wrapping is too entrenched into text E-Mail to fix, but we can
   fix Bugzilla without causing any problems.  The old content will still
   be wrapped too early, but at least new content will work.
```

Chapter 7. Bugzilla Variants and Competitors

I created this section to answer questions about Bugzilla competitors and variants, then found a wonderful site which covers an awful lot of what I wanted to discuss. Rather than quote it in its entirety, I'll simply refer you here: http://linas.org/linux/pm.html

7.1. Red Hat Bugzilla

Red Hat Bugzilla is probably the most popular Bugzilla variant on the planet. One of the major benefits of Red Hat Bugzilla is the ability to work with Oracle, MySQL, and PostGreSQL databases serving as the back−end, instead of just MySQL. Dave Lawrence has worked very hard to keep Red Hat Bugzilla up−to−date, and many people prefer the snappier−looking page layout of Red Hat Bugzilla to the default Mozilla−standard formatting.

URL: http://bugzilla.redhat.com/bugzilla/

7.2. Loki Bugzilla (Fenris)

Fenris can be found at http://fenris.lokigames.com. It is a fork from Bugzilla.

7.3. Issuezilla

Issuezilla is another fork from Bugzilla, and seems nearly as popular as the Red Hat Bugzilla fork. Some Issuezilla team members are regular contributors to the Bugzilla mailing list/newsgroup. Issuezilla is not the primary focus of bug−tracking at tigris.org, however. Their Java−based bug−tracker, Scarab, a newfangled Java−based issue tracker, is under heavy development and looks promising!

URL: http://issuezilla.tigris.org/servlets/ProjectHome

7.4. Scarab

Scarab is a promising new bug−tracking system built using Java Serlet technology. As of this writing, no source code has been released as a package, but you can obtain the code from CVS.

URL: http://scarab.tigris.org

7.5. Perforce SCM

Although Perforce isn't really a bug tracker, it can be used as such through the "jobs" functionality.

http://www.perforce.com/perforce/technotes/note052.htmlhttp://www.perforce.com/perforce/technotes/note052.html

7.6. SourceForge

SourceForge is more of a way of coordinating geographically distributed free software and open source projects over the Internet than strictly a bug tracker, but if you're hunting for bug−tracking for your open project, it may be just what the software engineer ordered!

URL: http://www.sourceforge.net

Appendix A. The Bugzilla FAQ

1. General Questions
- *A.1.1. Where can I find information about Bugzilla?*
- *A.1.2. What license is Bugzilla distributed under?*
- *A.1.3. How do I get commercial support for Bugzilla?*
- *A.1.4. What major companies or projects are currently using Bugzilla for bug−tracking?*
- *A.1.5. Who maintains Bugzilla?*
- *A.1.6. How does Bugzilla stack up against other bug−tracking databases?*
- *A.1.7. How do I change my user name in Bugzilla?*
- *A.1.8. Why doesn't Bugzilla offer this or that feature or compatability with this other tracking software?*
- *A.1.9. Why MySQL? I'm interested in seeing Bugzilla run on Oracle/Sybase/Msql/PostgreSQL/MSSQL?*
- *A.1.10. Why do the scripts say "/usr/bonsaitools/bin/perl" instead of "/usr/bin/perl" or something else?*

2. Red Hat Bugzilla
- *A.2.1. What about Red Hat Bugzilla?*
- *A.2.2. What are the primary benefits of Red Hat Bugzilla?*
- *A.2.3. What's the current status of Red Hat Bugzilla?*

3. Loki Bugzilla (AKA Fenris)
- *A.3.1. What is Loki Bugzilla (Fenris)?*

4. Pointy−Haired−Boss Questions
- *A.4.1. Is Bugzilla web−based or do you have to have specific software or specific operating system on your machine?*
- *A.4.2. Has anyone you know of already done any Bugzilla integration with Perforce (SCM software)?*
- *A.4.3. Does Bugzilla allow the user to track multiple projects?*
- *A.4.4. 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?*
- *A.4.5. Does Bugzilla allow attachments (text, screenshots, urls etc)? If yes, are there any that are NOT allowed?*
- *A.4.6. 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?*
- *A.4.7. The index.html page doesn't show the footer. It's really annoying to have to go to the querypage just to check my "my bugs" link. How do I get a footer on static HTML pages?*
- *A.4.8. Does Bugzilla provide any reporting features, metrics, graphs, etc? You know, the type of stuff that management likes to see. :)*
- *A.4.9. Is there email notification and if so, what do you see when you get an email? Do you see bug number and title or is it only the number?*
- *A.4.10. Can email notification be set up to send to multiple people, some on the To List, CC List, BCC List etc?*

A.4.11. If there is email notification, do users have to have any particular type of email application?
A.4.12. If I just wanted to track certain bugs, as they go through life, can I set it up to alert me via email whenever that bug changes, whether it be owner, status or description etc.?
A.4.13. 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?
A.4.14. Has anyone converted Bugzilla to another language to be used in other countries? Is it localizable?
A.4.15. Can a user create and save reports? Can they do this in Word format? Excel format?
A.4.16. Can a user re−run a report with a new project, same query?
A.4.17. Can a user modify an existing report and then save it into another name?
A.4.18. Does Bugzilla have the ability to search by word, phrase, compound search?
A.4.19. Can the admin person establish separate group and individual user privileges?
A.4.20. 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?
A.4.21. Are there any backup features provided?
A.4.22. Can users be on the system while a backup is in progress?
A.4.23. 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.
A.4.24. 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?
A.4.25. Is there any licensing fee or other fees for using Bugzilla? Any out−of−pocket cost other than the bodies needed as identified above?

5. Bugzilla Installation
A.5.1. How do I download and install Bugzilla?
A.5.2. How do I install Bugzilla on Windows NT?
A.5.3. Is there an easy way to change the Bugzilla cookie name?

6. Bugzilla Security
A.6.1. 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!)?
A.6.2. Are there any security problems with Bugzilla?
A.6.3. 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.

7. Bugzilla Email
A.7.1. I have a user who doesn't want to receive any more email from Bugzilla. How do I stop it entirely for this user?
A.7.2. I'm evaluating/testing Bugzilla, and don't want it to send email to anyone but me. How do I do it?
A.7.3. I want whineatnews.pl to whine at something more, or other than, only new bugs. How do I do it?
A.7.4. I don't like/want to use Procmail to hand mail off to bug_email.pl. What alternatives do I have?
A.7.5. How do I set up the email interface to submit/change bugs via email?
A.7.6. Email takes FOREVER to reach me from bugzilla −− it's extremely slow. What gives?
A.7.7. How come email never reaches me from bugzilla changes?

8. Bugzilla Database

A.8.1. I've heard Bugzilla can be used with Oracle?
A.8.2. Bugs are missing from queries, but exist in the database (and I can pull them up by specifying the bug ID). What's wrong?
A.8.3. I think my database might be corrupted, or contain invalid entries. What do I do?
A.8.4. I want to manually edit some entries in my database. How?
A.8.5. I try to add myself as a user, but Bugzilla always tells me my password is wrong.
A.8.6. I think I've set up MySQL permissions correctly, but bugzilla still can't connect.
A.8.7. How do I synchronize bug information among multiple different Bugzilla databases?
A.8.8. Why do I get bizarre errors when trying to submit data, particularly problems with "groupset"?
A.8.9. How come even after I delete bugs, the long descriptions show up?
9. Bugzilla and Win32
A.9.1. What is the easiest way to run Bugzilla on Win32 (Win98+/NT/2K)?
A.9.2. Is there a "Bundle::Bugzilla" equivalent for Win32?
A.9.3. CGI's are failing with a "something.cgi is not a valid Windows NT application" error. Why?
A.9.4. Can I have some general instructions on how to make Bugzilla on Win32 work?
A.9.5. I'm having trouble with the perl modules for NT not being able to talk to to the database.
10. Bugzilla Usage
A.10.1. The query page is very confusing. Isn't there a simpler way to query?
A.10.2. 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?
A.10.3. I can't upload anything into the database via the "Create Attachment" link. What am I doing wrong?
A.10.4. Email submissions to Bugzilla that have attachments end up asking me to save it as a "cgi" file.
A.10.5. How do I change a keyword in Bugzilla, once some bugs are using it?
11. Bugzilla Hacking
A.11.1. What bugs are in Bugzilla right now?
A.11.2. How can I change the default priority to a null value? For instance, have the default priority be "−−−" instead of "P2"?
A.11.3. What's the best way to submit patches? What guidelines should I follow?

1. General Questions

A.1.1. Where can I find information about Bugzilla?

You can stay up−to−date with the latest Bugzilla information at http://www.mozilla.org/projects/bugzilla/

A.1.2. What license is Bugzilla distributed under?

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

A.1.3. How do I get commercial support for Bugzilla?

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 whore themselves out for generous compensation. Try sending a message to the mailing list asking for a volunteer.

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

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

Netscape/AOL
Mozilla.org
AtHome Corporation
Red Hat Software
Loki Entertainment Software
SuSe Corp
The Horde Project
The Eazel Project
AbiSource
Real Time Enterprises, Inc
Eggheads.org
Strata Software
RockLinux
Creative Labs (makers of SoundBlaster)
The Apache Foundation
The Gnome Foundation
Linux−Mandrake

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

A.1.5. Who maintains Bugzilla?

Bugzilla maintenance has been in a state of flux recently. Please check the Bugzilla Project Page for the latest details.

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

A year has gone by, and I *still* can't find any head−to−head comparisons of Bugzilla against other defect−tracking software. However, from my 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 rebuttal so I can include it in the FAQ. We're not in pursuit of Bugzilla ueber alles; we simply love having a powerful, open−source tool to get our jobs done.

A.1.7. How do I change my user name in Bugzilla?

You can't. However, the administrative account can, by simply opening your user account in editusers.cgi and changing the login name.

A.1.8. Why doesn't Bugzilla offer this or that feature or compatability 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.

A.1.9. Why MySQL? I'm interested in seeing Bugzilla run on Oracle/Sybase/Msql/PostgreSQL/MSSQL?

Terry Weissman answers,

> You're not the only one. But *I* am not very interested. I'm not a real SQL or database person. I just wanted to make a useful tool, and build it on top of free software. So, I picked MySQL, and learned SQL by staring at the MySQL manual and some code lying around here, and wrote Bugzilla. I didn't know that Enum's were non−standard SQL. I'm not sure if I would have cared, but I didn't even know. So, to me, things are "portable" because it uses MySQL, and MySQL is portable enough. I fully understand (now) that people want to be portable to other databases, but that's never been a real concern of mine.

Things aren't quite that grim these days, however. Terry pretty much sums up much of the thinking many of us have for Bugzilla, but there is light on the horizon for database−independence! Here are some options:

Red Hat Bugzilla: Runs a modified Bugzilla 2.8 atop an Oracle database.
Interzilla: A project to run Bugzilla on Interbase. No code released yet, however.
Bugzilla 3.0: One of the primary stated goals is multiple database support.

A.1.10. Why do the scripts say "/usr/bonsaitools/bin/perl" instead of "/usr/bin/perl" or something else?

Mozilla.org uses /usr/bonsaitools/bin/perl. The prime rule in making submissions is "don't break bugzilla.mozilla.org". If it breaks it, your patch will be reverted faster than you can do a diff.

Here's Terry Weissman's comment, for some historical context:

> [This was] purely my own convention. I wanted a place to put a version of Perl and other tools that was strictly under my control for the various webtools, and not subject to anyone else. Edit it to point to whatever you like.

> We always recommend that, if possible, you keep the path as /usr/bonsaitools/bin/perl, and simply add a /usr/bonsaitools and /usr/bonsaitools/bin directory, then symlink your version of perl to /usr/bonsaitools/bin/perl. This will make upgrading your Bugzilla much easier in the future.
>
> Obviously, if you do not have root access to your Bugzilla box, our suggestion is irrelevant.

2. Red Hat Bugzilla

This section is no longer up−to−date. Please see the section on "Red Hat Bugzilla" under "Variants" in The Bugzilla Guide.

A.2.1. What about Red Hat Bugzilla?

Red Hat Bugzilla is arguably more user−friendly, customizable, and scalable than stock Bugzilla. Check it out at http://bugzilla.redhat.com and the sources at ftp://people.redhat.com/dkl/. They've set their Bugzilla up to work with Oracle out of the box. Note that Redhat Bugzilla is based upon the 2.8 Bugzilla tree; Bugzilla has made some tremendous advances since the 2.8 release. Why not download both Bugzillas to check out the differences for yourself?

Dave Lawrence, the original Red Hat Bugzilla maintainer, mentions:

> Somebody needs to take the ball and run with it. I'm the only maintainer and am very pressed for time.

If you, or someone you know, has the time and expertise to do the integration work so main−tree Bugzilla 2.12 and higher integrates the Red Hat Bugzilla Oracle modifications, please donate your time to supporting the Bugzilla project.

A.2.2. What are the primary benefits of Red Hat Bugzilla?

Dave Lawrence:

> For the record, we are not using any template type implementation for the cosmetic changes maded to Bugzilla. It is just alot of html changes in the code itself. I admit I may have gotten a little carried away with it but the corporate types asked for a more standardized interface to match up with other projects relating to Red Hat web sites. A lot of other web based internal tools I am working on also look like Bugzilla.
>
> I do want to land the changes that I have made to Bugzilla but I may have to back out a good deal and make a different version of Red Hat's Bugzilla for checking in to CVS. Especially the cosmetic changes because it seems they may not fit the general public. I will do that as soon as I can. I also still do my regular QA responsibilities along with Bugzilla so time is difficult sometimes to come by.
>
> There are also a good deal of other changes that were requested by management for things like support contracts and different permission groups for making bugs private. Here is a short list of the major changes that have been made:
>
> 1. No enum types. All old enum types are now separate smaller tables.
> 2. No bit wise operations. Not all databases support this so they were changed to a more generic way of doing this task
> 3. Bug reports can only be altered by the reporter, assignee, or a privileged bugzilla user. The rest of the world can see the bug but in a non−changeable format (unless the bug has been marked private). They can however add comments, add and remove

themselves from the CC list
4. Different group scheme. Each group has an id number related to it. There is a user_group table which contains userid to groupid mappings to determine which groups each user belongs to. Additionally there is a bug_group table that has bugid to groupid mappings to show which groups can see a particular bug. If there are no entries for a bug in this table then the bug is public.
5. Product groups. product_table created to only allow certain products to be visible for certain groups in both bug entry and query. This was particulary helpful for support contracts.
6. Of course many (too many) changes to Bugzilla code itself to allow use with Oracle and still allow operation with Mysql if so desired. Currently if you use Mysql it is set to use Mysql's old permission scheme to keep breakage to a minimum. Hopefully one day this will standardize on one style which may of course be something completely different.
7. Uses Text::Template perl module for rendering of the dynamic HTML pages such as enter_bug.cgi, query.cgi, bug_form.pl, and for the header and footer parts of the page. This allows the html to be separate from the perl code for customizing the look and feel of the page to one's preference.
8. There are many other smaller changes. There is also a port to Oracle that I have been working on as time permits but is not completely finished but somewhat usable. I will merge it into our standard code base when it becomes production quality. Unfortunately there will have to be some conditionals in the code to make it work with other than Oracle due to some differences between Oracle and Mysql.

Both the Mysql and Oracle versions of our current code base are available from ftp://people.redhat.com/dkl. If Terry/Tara wants I can submit patch files for all of the changes I have made and he can determine what is suitable for addition to the main bugzilla cade base. But for me to commit changes to the actual CVS I will need to back out alot of things that are not suitable for the rest of the Bugzilla community. I am open to suggestions.

A.2.3. What's the current status of Red Hat Bugzilla?

> This information is somewhat dated; I last updated it 7 June 2000. Please see the "Variants" section of "The Bugzilla Guide" for more up−to−date information regarding Red Hat Bugzilla.

Dave Lawrence:

I suppose the current thread warrants an update on the status of Oracle and bugzilla ;) We have now been running Bugzilla 2.8 on Oracle for the last two days in our production environment. I tried to do as much testing as possible with it before going live which is some of the reason for the long delay. I did not get enough feedback as I would have liked from internal developers to help weed out any bugs still left so I said "Fine, i will take it live and then I will get the feedback I want :)" So it is now starting to stabilize and it running quite well after working feverishly the last two days fixing problems as soon as they came in from the outside world. The current branch in cvs is up2date if anyone would like to grab it and try it out. The oracle _setup.pl is broken right now due to some last minute changes but I will update that soon. Therefore you would probably need to create the database tables the old fashioned way using the supplied sql creation scripts located in the ./oracle directory. We have heavy optimizations in the database it self thanks to the in−house DBA here at Red Hat so it is running quite fast. The database itself is located on a dual PII450 with 1GB ram and 14 high voltage differential raided scsi drives. The tables and indexes are partitioned in 4 chuncks across the raided drive which is nice because when ever you need to do a full table scan, it is actually starting in 4 different locations on 4 different drives simultaneously. And the indexes of course are on separate drives from the data so that speeds things up tremendously. When I can find the time I will document all that we have done to get this thing going to help others that may need it.

As Matt has mentioned it is still using out−dated code and with a little help I would like to bring everything up to date for eventual incorporation with the main cvs tree. Due to other duties I have with the company any help with this wiould be appreciated. What we are using now is what I call a best first effort. It definitely can be improved on and may even need complete rewrites in a lot of areas. A lot of changes may have to be made in the way Bugzilla does things currently to make this transition to a more generic database interface. Fortunately when making the Oracle changes I made sure I didn't do anything that I would consider Oracle specific and could not be easily done with other databases. Alot of the sql statements need to be broken up into smaller utilities that themselves would need to make decisions on what database they are using but the majority of the code can be made database neutral.

3. Loki Bugzilla (AKA Fenris)

A.3.1. What is Loki Bugzilla (Fenris)?

Loki Games has a customized version of Bugzilla available at http://fenris.lokigames.com. There are some advantages to using Fenris, chief being separation of comments based upon user privacy level, data hiding, forced login for any data retrieval, and some additional fields. Loki has mainted their code, originally a fork from the Bugzilla 2.8 code base, and it is quite a bit different than stock Bugzilla at this point. I recommend you stick with official Bugzilla version 2.14 rather than using a fork, but it's up to you.

4. Pointy−Haired−Boss Questions

The title of this section doesn't mean you're a PHB −− it just means you probably HAVE a PHB who wants to know this :)

A.4.1. Is Bugzilla web−based or do you have to have specific software or 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.

A.4.2. Has anyone you know of already done any Bugzilla integration with Perforce (SCM software)?

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

A.4.3. Does Bugzilla allow the user to track multiple projects?

Absolutely! You can track up to a "soft−limit" of around 64 individual "Products", that can each be composed of as many "Components" as you want. Check the Administration section of the Bugzilla Guide for more information regarding setting up Products and Components.

A.4.4. 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.

A.4.5. Does Bugzilla allow attachments (text, screenshots, urls etc)? If yes, are there any that are NOT allowed?

Yes. There are many specific MIME−types that are pre−defined by Bugzilla, but you may specify any arbitrary MIME−type you need when you upload the file. Since all attachments are stored in the database, however, I recommend storing large binary attachments elsewhere in the web server's file system and providing a hyperlink as a comment, or in the provided "URL" field in the bug report.

A.4.6. 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

A.4.7. The index.html page doesn't show the footer. It's really annoying to have to go to the querypage just to check my "my bugs" link. How do I get a footer on static HTML pages?

It's possible to get the footer on the static index page using Server Side Includes (SSI). The trick to doing this is making sure that your web server is set up to allow SSI and specifically, the #exec directive. You should also rename `index.html` to `index.shtml`.

After you've done all that, you can add the following line to `index.shtml`:

```
<!--#exec cmd="/usr/bin/perl −e &quot;require 'CGI.pl'; PutFooter();&quot;" -->
```

This line will be replaced with the actual HTML for the footer when the page is requested, so you should put this line where you want the footer to appear.

Because this method depends on being able to use a #exec directive, and most ISP's will not allow that, there is an alternative method. You could have a small script (such as `api.cgi`) that basically looks like:

```
#!/usr/bonsaitools/bin/perl −w

require 'globals.pl';

if ($::FORM{sub} eq 'PutFooter') {
    PutFooter();
} else {
```

```
    die 'api.cgi was incorrectly called';
}
```

and then put this line in `index.shtml`.

```
<!--#include virtual="api.cgi?sub=PutFooter"-->
```

This still requires being able to use Server Side Includes, if this simply will not work for you, see bug 80183 for a third option.

A.4.8. 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 basic reporting facilities.

For more advanced reporting, I recommend hooking up a professional reporting package, such as Crystal Reports, and use ODBC to access the MySQL database. You can do a lot through the Query page of Bugzilla as well, but right now Advanced Reporting is much better accomplished through third−party utilities that can interface with the database directly.

Advanced Reporting is a Bugzilla 3.X proposed feature.

A.4.9. Is there email notification and if so, what do you see when you get an email? Do you see bug number and title or is it only the number?

Email notification is user−configurable. The bug id and Topic of the bug report accompany each email notification, along with a list of the changes made.

A.4.10. Can email notification be set up to send to multiple people, some on the To List, CC List, BCC List etc?

Yes.

A.4.11. If there is email notification, 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.

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.

A.4.12. If I just wanted to track certain bugs, as they go through life, can I set it up to alert me via email whenever that bug changes, whether it be owner, status or description etc.?

Yes. Place yourself in the "cc" field of the bug you wish to monitor. Then change your "Notify me of changes to" field in the Email Settings tab of the User Preferences screen in Bugzilla to the "Only those bugs which I am listed on the CC line" option.

A.4.13. 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?

Mozilla allows data export through a custom DTD in XML format. It does not, however, export to specific formats other than the XML Mozilla DTD. Importing the data into Excel or any other application is left as an exercise for the reader.

If you create import filters to other applications from Mozilla's XML, please submit your modifications for inclusion in future Bugzilla distributions.

As for data import, any application can send data to Bugzilla through the HTTP protocol, or through Mozilla's XML API. However, it seems kind of silly to put another front−end in front of Bugzilla; it makes more sense to create a simplified bug submission form in HTML. You can find an excellent example at http://www.mozilla.org/quality/help/bugzilla−helper.html

A.4.14. Has anyone converted Bugzilla to another language to be used in other countries? Is it localizable?

Currently, no. Internationalization support for Perl did not exist in a robust fashion until the recent release of version 5.6.0; Bugzilla is, and likely will remain (until 3.X) completely non−localized.

A.4.15. Can a user create and save reports? Can they do this in Word format? Excel format?

Yes. No. No.

A.4.16. Can a user re−run a report with a new project, same query?

Yes.

A.4.17. Can a user modify an existing report and then save it into another name?

You can save an unlimited number of queries in Bugzilla. You are free to modify them and rename them to your heart's desire.

A.4.18. 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.

A.4.19. Can the admin person establish separate group and individual user privileges?

Yes.

A.4.20. 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.

A.4.21. 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

A.4.22. 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.

A.4.23. 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 completed by unskilled labor. Things like rotate backup tapes and check log files for the word "error".

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.

As an example, as of this writing I typically charge $115 for the first hour, and $89 each hour thereafter for consulting work. It takes me three to five hours to make Bugzilla happy on a Development installation of Linux−Mandrake.

A.4.24. 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.

A.4.25. 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.

5. Bugzilla Installation

A.5.1. How do I download and install Bugzilla?

Check http://www.mozilla.org/projects/bugzilla/ for details. Once you download it, untar it, read the Bugzilla Guide.

A.5.2. How do I install Bugzilla on Windows NT?

Installation on Windows NT has its own section in "The Bugzilla Guide".

A.5.3. Is there an easy way to change the Bugzilla cookie name?

At present, no.

6. Bugzilla Security

A.6.1. 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.* Please read the Security section of the Administration chapter of "The Bugzilla Guide" before proceeding.

A.6.2. Are there any security problems with Bugzilla?

The Bugzilla code has not undergone a complete security audit. It is recommended that you closely examine permissions on your Bugzilla installation, and follow the recommended security guidelines found in The Bugzilla Guide.

A.6.3. 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.

7. Bugzilla Email

A.7.1. I have a user who doesn't want to receive any more email from Bugzilla. How do I stop it entirely for this user?

With the email changes to 2.12, the user should be able to set this in user email preferences.

A.7.2. I'm evaluating/testing Bugzilla, and don't want it to send email to anyone but me. How do I do it?

Edit the param for the mail text. Replace "To:" with "X−Real−To:", replace "Cc:" with "X−Real−CC:", and add a "To: (myemailaddress)".

A.7.3. 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.

A.7.4. 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.

A.7.5. 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.

A.7.6. 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" script 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".

A.7.7. How come email never reaches me from bugzilla changes?

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".

8. Bugzilla Database

A.8.1. I've heard Bugzilla can be used with Oracle?

Red Hat Bugzilla, mentioned above, works with Oracle. The current version from Mozilla.org does not have this capability. Unfortunately, though you will sacrifice a lot of the really great features available in Bugzilla 2.10 and 2.12 if you go with the 2.8−based Redhat version.

A.8.2. Bugs are missing from queries, but exist in the database (and I can pull them up by specifying the bug ID). What's wrong?

You've almost certainly enabled the "shadow database", but for some reason it hasn't been updated for all your bugs. This is the database against which queries are run, so that really complex or slow queries won't lock up portions of the database for other users. You can turn off the shadow database in editparams.cgi. If you wish to continue using the shadow database, then as your "bugs" user run "./syncshadowdb −syncall" from the command line in the bugzilla installation directory to recreate your shadow database. After it finishes, be sure to check the params and make sure that "queryagainstshadowdb" is still turned on. The syncshadowdb program turns it off if it was on, and is supposed to turn it back on when completed; that way, if it crashes in the middle of recreating the database, it will stay off forever until someone turns it back on by hand. Apparently, it doesn't always do that yet.

A.8.3. 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.

A.8.4. 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 mysqladmin utility to manually insert, delete, and modify table information. Personally, I use "phpMyAdmin". You have to compile a PHP module with MySQL support to make it work, but it's very clean and easy to use.

A.8.5. I try to add myself as a user, but Bugzilla always tells me my password is wrong.

Certain version of MySQL (notably, 3.23.29 and 3.23.30) accidentally disabled the "crypt()" function. This prevented MySQL from storing encrypted passwords. Upgrade to the "3.23 stable" version of MySQL and you should be good to go.

A.8.6. 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. However, I do not recommend you run it this way on a regular basis, unless you really want your web site defaced and your machine cracked.

A.8.7. 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.

A.8.8. Why do I get bizarre errors when trying to submit data, particularly problems with "groupset"?

If you're sure your MySQL parameters are correct, you might want turn "strictvaluechecks" OFF in editparams.cgi. If you have "usebugsentry" set "On", you also cannot submit a bug as readable by more than one group with "strictvaluechecks" ON.

A.8.9. How come even after I delete bugs, the long descriptions show up?

This should only happen with Bugzilla 2.14 if you are using the "shadow database" feature, and your shadow database is out of sync. Try running `syncshadowdb −syncall` to make sure your shadow database is in synch with your primary database.

9. Bugzilla and Win32

A.9.1. 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.

A.9.2. 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.

A.9.3. 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"

A.9.4. Can I have some general instructions on how to make Bugzilla on Win32 work?

The following couple entries are deprecated in favor of the Windows installation instructions available in the "Administration" portion of "The Bugzilla Guide". However, they are provided here for historical interest and insight.

```
1. #!C:/perl/bin/perl had to be added to every perl file.
2. Converted to Net::SMTP to handle mail messages instead of
   /usr/bin/sendmail.
3. The crypt function isn't available on Windows NT (at least none that I
   am aware), so I made encrypted passwords = plaintext passwords.
4. The system call to diff had to be changed to the Cygwin diff.
5. This was just to get a demo running under NT, it seems to be working
   good, and I have inserted almost 100 bugs from another bug tracking
   system. Since this work was done just to get an in-house demo, I am NOT
   planning on making a patch for submission to Bugzilla. If you would
   like a zip file, let me know.

Q: Hmm, couldn't figure it out from the general instructions above.  How
about step-by-step?
A: Sure! Here ya go!

1. Install IIS 4.0 from the NT Option Pack #4.
2. Download and install Active Perl.
```

3. Install the Windows GNU tools from Cygwin. Make sure to add the bin directory to your system path. (Everyone should have these, whether they decide to use Bugzilla or not. :-))
4. Download relevant packages from ActiveState at http://www.activestate.com/packages/zips/. + DBD-Mysql.zip
5. Extract each zip file with WinZip, and install each ppd file using the notation: ppm install <module>.ppd
6. Install Mysql. *Note: If you move the default install from c:\mysql, you must add the appropriate startup parameters to the NT service. (ex. -b e:\\programs\\mysql)
7. Download any Mysql client. http://www.mysql.com/download_win.html
8. Setup MySql. (These are the commands that I used.)

```
I. Cleanup default database settings.
 C:\mysql\bin\mysql -u root mysql
 mysql> DELETE FROM user WHERE Host='localhost' AND User='';
 mysql> quit
C:\mysql\bin\mysqladmin reload

II. Set password for root.
 C:\mysql\bin\mysql -u root mysql
 mysql> UPDATE user SET Password=PASSWORD('new_password')
 WHERE user='root';
 mysql> FLUSH PRIVILEGES;
 mysql> quit
 C:\mysql\bin\mysqladmin -u root reload

III. Create bugs user.
 C:\mysql\bin\mysql -u root -p
 mysql> insert into user (host,user,password)
values('localhost','bugs','');
 mysql> quit
 C:\mysql\bin\mysqladmin -u root reload

IV. Create the bugs database.
 C:\mysql\bin\mysql -u root -p
 mysql> create database bugs;

V. Give the bugs user access to the bugs database.
 mysql> insert into db
(host,db,user,select_priv,insert_priv,update_priv,delete_priv,create_priv,drop_priv)
values('localhost','bugs','bugs','Y','Y','Y','Y','Y','N')
 mysql> quit
 C:\mysql\bin\mysqladmin -u root reload
```

9. Run the table scripts to setup the bugs database.
10. Change CGI.pm to use the following regular expression because of differing backslashes in NT versus UNIX.
    - `$0 =~ m:[^\\]*$:;`
11. Had to make the crypt password = plain text password in the database. (Thanks to Andrew Lahser" <andrew_lahser@merck.com>" on this one.) The files that I changed were:

```
        o globals.pl
        o CGI.pl
        o alternately, you can try commenting all references to 'crypt'
          string and replace them with similar lines but without encrypt()
          or crypr() functions insida all files.
 12. Replaced sendmail with Windmail. Basically, you have to come up with a
     sendmail substitute for NT. Someone said that they used a Perl module
     (Net::SMTP), but I was trying to save time and do as little Perl coding
     as possible.
 13. Added "perl" to the beginning of all Perl system calls that use a perl
     script as an argument and renamed processmail to processmail.pl.
 14. In processmail.pl, I added binmode(HANDLE) before all read() calls. I'm
     not sure about this one, but the read() under NT wasn't counting the
     EOLs without the binary read."
```

A.9.5. 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/

10. Bugzilla Usage

A.10.1. The query page is very confusing. Isn't there a simpler way to query?

We are developing in that direction. You can follow progress on this at http://bugzilla.mozilla.org/show_bug.cgi?id=16775. Some functionality is available in Bugzilla 2.12, and is available as "quicksearch.html"

A.10.2. 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. I personally don't like it. You have your choice of patches to change this behavior, however.

Add a "and accept bug" radio button

"Accept" button automatically assigns to you

Note that these patches are somewhat dated. You will need to do the find and replace manually to apply them. They are very small, though. It is easy.

A.10.3. 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.

A.10.4. Email submissions to Bugzilla that have attachments end up asking me to save it as a "cgi" file.

Yup. Just rename it once you download it, or save it under a different filename. This will not be fixed anytime too soon, because it would cripple some other functionality.

A.10.5. 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.

11. Bugzilla Hacking

A.11.1. What bugs are in Bugzilla right now?

Try this link to view current bugs or requests for enhancement for Bugzilla.

You can view bugs marked for 2.16 release here. This list includes bugs for the 2.16 release that have already been fixed and checked into CVS. Please consult the Bugzilla Project Page for details on how to check current sources out of CVS so you can have these bug fixes early!

A.11.2. 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. 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". Hmm, now that I think about it, that is kind of a klunky way to handle it, but for now it's what we have! Although the bug has been closed "resolved wontfix", there may be a better way to handle this...

A.11.3. What's the best way to submit patches? What guidelines should I follow?

1. Enter a bug into bugzilla.mozilla.org for the "Bugzilla" 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" radio button 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=XXXX) 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 :)

Appendix B. Software Download Links

All of these sites are current as of April, 2001. Hopefully they'll stay current for a while.

Apache Web Server: http://www.apache.org Optional web server for Bugzilla, but recommended because of broad user base and support.

Bugzilla: http://www.mozilla.org/projects/bugzilla/

MySQL: http://www.mysql.com/

Perl: http://www.perl.org/

CPAN: http://www.cpan.org/

DBI Perl module: ftp://ftp.cpan.org/pub/perl/CPAN/modules/by−module/DBI/

Data::Dumper module: ftp://ftp.cpan.org/pub/perl/CPAN/modules/by−module/Data/

MySQL related Perl modules: ftp://ftp.cpan.org/pub/perl/CPAN/modules/by−module/Mysql/

TimeDate Perl module collection: ftp://ftp.cpan.org/pub/perl/CPAN/modules/by−module/Date/

GD Perl module: ftp://ftp.cpan.org/pub/perl/CPAN/modules/by−module/GD/ Alternately, you should be able to find the latest version of GD at http://www.boutell.com/gd/

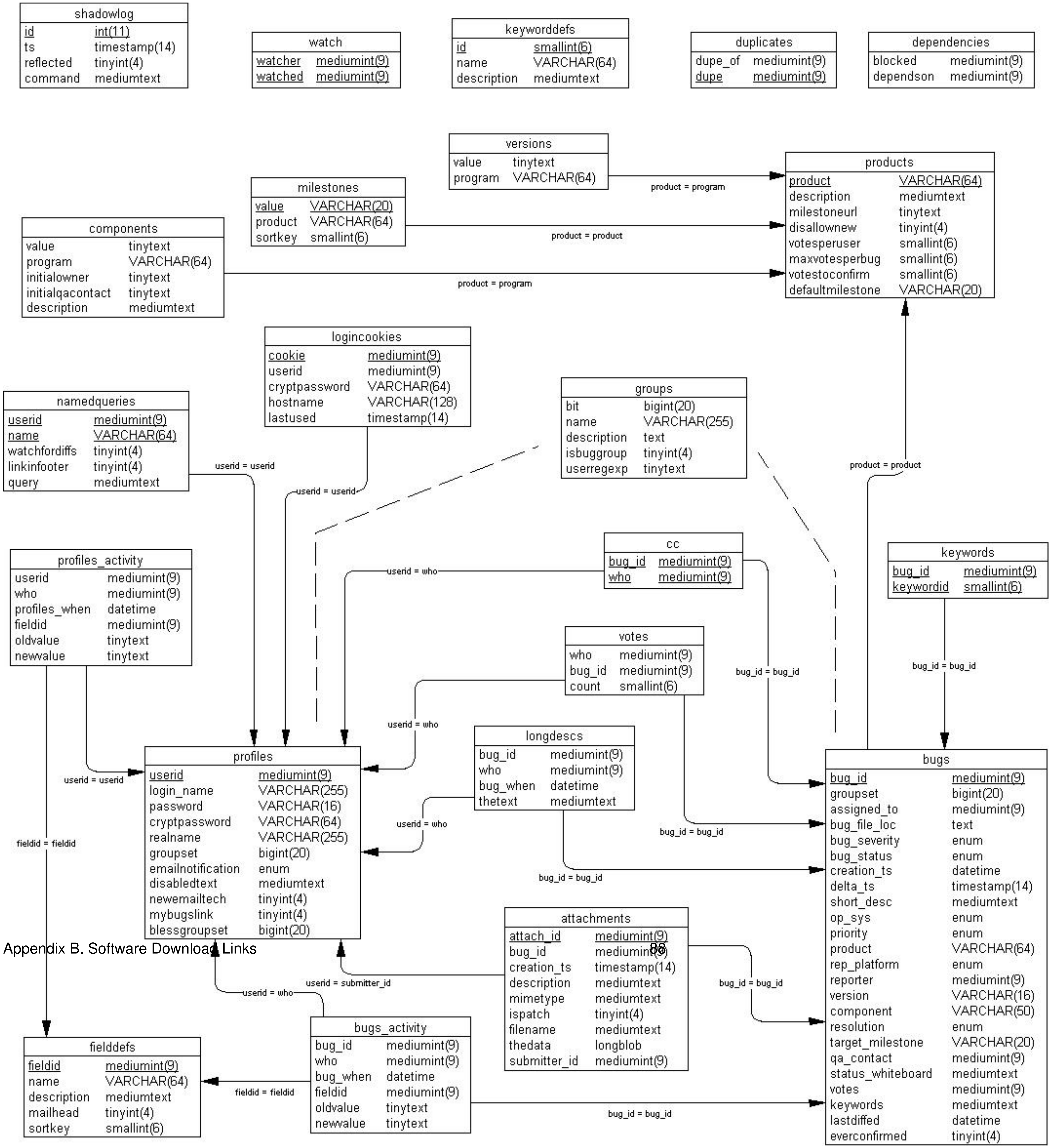
Chart::Base module: ftp://ftp.cpan.org/pub/perl/CPAN/modules/by−module/Chart/

LinuxDoc Software: http://www.linuxdoc.org/ (for documentation maintenance)

Appendix C. The Bugzilla Database

This document really needs to be updated with more fleshed out information about primary keys, interrelationships, and maybe some nifty tables to document dependencies. Any takers?

C.1. Database Schema Chart


shadowlog
id int(11)
ts timestamp(14)
reflected tinyint(4)
command mediumtext
watch
watcher mediumint(9)
watched mediumint(9)
keyworddefs
id smallint(6)
name VARCHAR(64)
description mediumtext
duplicates
dupe_of mediumint(9)
dupe mediumint(9)
dependencies
blocked mediumint(9)
dependson mediumint(9)
versions
value tinytext
program VARCHAR(64)
product = program
products
product VARCHAR(64)
description mediumtext
milestoneurl tinytext
disallownew tinyint(4)
votesperuser smallint(6)
maxvotesperbug smallint(6)
votestoconfirm smallint(6)
defaultmilestone VARCHAR(20)
milestones
value VARCHAR(20)
product VARCHAR(64)
sortkey smallint(6)
product = product
components
value tinytext
program VARCHAR(64)
initialowner tinytext
initialqacontact tinytext
description mediumtext
product = program
logincookies
cookie mediumint(9)
userid mediumint(9)
cryptpassword VARCHAR(64)
hostname VARCHAR(128)
lastused timestamp(14)
groups
bit bigint(20)
name VARCHAR(255)
description text
isbuggroup tinyint(4)
userregexp tinytext
namedqueries
userid mediumint(9)
name VARCHAR(64)
watchfordiffs tinyint(4)
linkinfooter tinyint(4)
query mediumtext
userid = userid
userid = userid
product = product
profiles_activity
userid mediumint(9)
who mediumint(9)
profiles_when datetime
fieldid mediumint(9)
oldvalue tinytext
newvalue tinytext
userid = who
cc
bug_id mediumint(9)
who mediumint(9)
keywords
bug_id mediumint(9)
keywordid smallint(6)
votes
who mediumint(9)
bug_id mediumint(9)
count smallint(6)
bug_id = bug_id
bug_id = bug_id
userid = who
longdescs
bug_id mediumint(9)
who mediumint(9)
bug_when datetime
thetext mediumtext
userid = userid
profiles
userid mediumint(9)
login_name VARCHAR(255)
password VARCHAR(16)
cryptpassword VARCHAR(64)
realname VARCHAR(255)
groupset bigint(20)
emailnotification enum
disabledtext mediumtext
newemailtech tinyint(4)
mybugslink tinyint(4)
blessgroupset bigint(20)
userid = who
fieldid = fieldid
bug_id = bug_id
bug_id = bug_id
bugs
bug_id mediumint(9)
groupset bigint(20)
assigned_to mediumint(9)
bug_file_loc text
bug_severity enum
bug_status enum
creation_ts datetime
delta_ts timestamp(14)
short_desc mediumtext
op_sys enum
priority enum
product VARCHAR(64)
rep_platform enum
reporter mediumint(9)
version VARCHAR(16)
component VARCHAR(50)
resolution enum
target_milestone VARCHAR(20)
qa_contact mediumint(9)
status_whiteboard mediumtext
votes mediumint(9)
keywords mediumtext
lastdiffed datetime
everconfirmed tinyint(4)
attachments
attach_id mediumint(9)
bug_id mediumint(9)
creation_ts timestamp(14)
description mediumtext
mimetype mediumtext
ispatch tinyint(4)
filename mediumtext
thedata longblob
submitter_id mediumint(9)
userid = submitter_id
userid = who
bug_id = bug_id
bugs_activity
bug_id mediumint(9)
who mediumint(9)
bug_when datetime
fieldid mediumint(9)
oldvalue tinytext
newvalue tinytext
fielddefs
fieldid mediumint(9)
name VARCHAR(64)
description mediumtext
mailhead tinyint(4)
sortkey smallint(6)
fieldid = fieldid
bug_id = bug_id

Bugzilla database relationships chart

C.2. MySQL Bugzilla Database Introduction

This information comes straight from my life. I was forced to learn how Bugzilla organizes database because of nitpicky requests from users for tiny changes in wording, rather than having people re−educate themselves or figure out how to work our procedures around the tool. It sucks, but it can and will happen to you, so learn how the schema works and deal with it when it comes.

So, here you are with your brand−new installation of Bugzilla. You've got MySQL set up, Apache working right, Perl DBI and DBD talking to the database flawlessly. Maybe you've even entered a few test bugs to make sure email's working; people seem to be notified of new bugs and changes, and you can enter and edit bugs to your heart's content. Perhaps you've gone through the trouble of setting up a gateway for people to submit bugs to your database via email, have had a few people test it, and received rave reviews from your beta testers.

What's the next thing you do? Outline a training strategy for your development team, of course, and bring them up to speed on the new tool you've labored over for hours.

Your first training session starts off very well! You have a captive audience which seems enraptured by the efficiency embodied in this thing called "Bugzilla". You are caught up describing the nifty features, how people can save favorite queries in the database, set them up as headers and footers on their pages, customize their layouts, generate reports, track status with greater efficiency than ever before, leap tall buildings with a single bound and rescue Jane from the clutches of Certain Death!

But Certain Death speaks up −− a tiny voice, from the dark corners of the conference room. "I have a concern," the voice hisses from the darkness, "about the use of the word 'verified'.

The room, previously filled with happy chatter, lapses into reverential silence as Certain Death (better known as the Vice President of Software Engineering) continues. "You see, for two years we've used the word 'verified' to indicate that a developer or quality assurance engineer has confirmed that, in fact, a bug is valid. I don't want to lose two years of training to a new software product. You need to change the bug status of 'verified' to 'approved' as soon as possible. To avoid confusion, of course."

Oh no! Terror strikes your heart, as you find yourself mumbling "yes, yes, I don't think that would be a problem," You review the changes with Certain Death, and continue to jabber on, "no, it's not too big a change. I mean, we have the source code, right? You know, 'Use the Source, Luke' and all that... no problem," All the while you quiver inside like a beached jellyfish bubbling, burbling, and boiling on a hot Jamaican sand dune...

Thus begins your adventure into the heart of Bugzilla. You've been forced to learn about non−portable enum() fields, varchar columns, and tinyint definitions. The Adventure Awaits You!

C.2.1. Bugzilla Database Basics

If you were like me, at this point you're totally clueless about the internals of MySQL, and if it weren't for this executive order from the Vice President you couldn't care less about the difference between a "bigint" and a "tinyint" entry in MySQL. I recommend you refer to the MySQL documentation, available at

MySQL.com. Below are the basics you need to know about the Bugzilla database. Check the chart above for more details.

1. To connect to your database:

   `bash#`**mysql**`-`*`u root`*

   If this works without asking you for a password, *shame on you*! You should have locked your security down like the installation instructions told you to. You can find details on locking down your database in the Bugzilla FAQ in this directory (under "Security"), or more robust security generalities in the MySQL searchable documentation at http://www.mysql.com/php/manual.php3?section=Privilege_system .

2. You should now be at a prompt that looks like this:

   `mysql>`

   At the prompt, if "bugs" is the name you chose in the`localconfig` file for your Bugzilla database, type:

   `mysql`**use bugs;**

   Don't forget the ";" at the end of each line, or you'll be kicking yourself later.

C.2.1.1. Bugzilla Database Tables

Imagine your MySQL database as a series of spreadsheets, and you won't be too far off. If you use this command:

`mysql>`**show tables from bugs;**

you'll be able to see all the "spreadsheets" (tables) in your database. It is similar to a file system, only faster and more robust for certain types of operations.

From the command issued above, ou should have some output that looks like this:

```
+-------------------+
| Tables in bugs    |
+-------------------+
| attachments       |
| bugs              |
| bugs_activity     |
| cc                |
| components        |
| dependencies      |
| fielddefs         |
| groups            |
| keyworddefs       |
| keywords          |
| logincookies      |
```

```
| longdescs         |
| milestones        |
| namedqueries      |
| products          |
| profiles          |
| profiles_activity |
| shadowlog         |
| tokens            |
| versions          |
| votes             |
| watch             |
+-------------------+
```

Here's an overview of what each table does. Most columns in each table have descriptive names that make it fairly trivial to figure out their jobs.

attachments: This table stores all attachments to bugs. It tends to be your largest table, yet also generally has the fewest entries because file attachments are so (relatively) large.

bugs: This is the core of your system. The bugs table stores most of the current information about a bug, with the exception of the info stored in the other tables.

bugs_activity: This stores information regarding what changes are made to bugs when -- a history file.

cc: This tiny table simply stores all the CC information for any bug which has any entries in the CC field of the bug. Note that, like most other tables in Bugzilla, it does not refer to users by their user names, but by their unique userid, stored as a primary key in the profiles table.

components: This stores the programs and components (or products and components, in newer Bugzilla parlance) for Bugzilla. Curiously, the "program" (product) field is the full name of the product, rather than some other unique identifier, like bug_id and user_id are elsewhere in the database.

dependencies: Stores data about those cool dependency trees.

fielddefs: A nifty table that defines other tables. For instance, when you submit a form that changes the value of "AssignedTo" this table allows translation to the actual field name "assigned_to" for entry into MySQL.

groups: defines bitmasks for groups. A bitmask is a number that can uniquely identify group memberships. For instance, say the group that is allowed to tweak parameters is assigned a value of "1", the group that is allowed to edit users is assigned a "2", and the group that is allowed to create new groups is assigned the bitmask of "4". By uniquely combining the group bitmasks (much like the chmod command in UNIX,) you can identify a user is allowed to tweak parameters and create groups, but not edit users, by giving him a bitmask of "5", or a user allowed to edit users and create groups, but not tweak

```
parameters, by giving him a bitmask of "6" Simple, huh?
  If this makes no sense to you, try this at the mysql prompt:
mysql> select * from groups;
  You'll see the list, it makes much more sense that way.

keyworddefs:  Definitions of keywords to be used

keywords: Unlike what you'd think, this table holds which keywords are
associated with which bug id's.

logincookies: This stores every login cookie ever assigned to you for every
machine you've ever logged into Bugzilla from.  Curiously, it never does any
housecleaning -- I see cookies in this file I've not used for months.  However,
since Bugzilla never expires your cookie (for convenience' sake), it makes
sense.

longdescs:  The meat of bugzilla -- here is where all user comments are stored!
You've only got 2^24 bytes per comment (it's a mediumtext field), so speak
sparingly -- that's only the amount of space the Old Testament from the Bible
would take (uncompressed, 16 megabytes).  Each comment is keyed to the
bug_id to which it's attached, so the order is necessarily chronological, for
comments are played back in the order in which they are received.

milestones:  Interesting that milestones are associated with a specific product
in this table, but Bugzilla does not yet support differing milestones by
product through the standard configuration interfaces.

namedqueries:  This is where everybody stores their "custom queries".  Very
cool feature; it beats the tar out of having to bookmark each cool query you
construct.

products:  What products you have, whether new bug entries are allowed for the
product, what milestone you're working toward on that product, votes, etc.  It
will be nice when the components table supports these same features, so you
could close a particular component for bug entry without having to close an
entire product...

profiles:  Ahh, so you were wondering where your precious user information was
stored?  Here it is!  With the passwords in plain text for all to see! (but
sshh... don't tell your users!)

profiles_activity:  Need to know who did what when to who's profile?  This'll
tell you, it's a pretty complete history.

shadowlog:  I could be mistaken here, but I believe this table tells you when
your shadow database is updated and what commands were used to update it.  We
don't use a shadow database at our site yet, so it's pretty empty for us.

versions:  Version information for every product

votes:  Who voted for what when
```

```
watch:  Who (according to userid) is watching who's bugs (according to their
userid).


===
THE DETAILS
===

  Ahh, so you're wondering just what to do with the information above?  At the
mysql prompt, you can view any information about the columns in a table with
this command (where "table" is the name of the table you wish to view):

mysql> show columns from table;

  You can also view all the data in a table with this command:

mysql> select * from table;

  -- note: this is a very bad idea to do on, for instance, the "bugs" table if
you have 50,000 bugs.  You'll be sitting there a while until you ctrl-c or
50,000 bugs play across your screen.

  You can limit the display from above a little with the command, where
"column" is the name of the column for which you wish to restrict information:

mysql> select * from table where (column = "some info");

  -- or the reverse of this

mysql> select * from table where (column != "some info");

  Let's take our example from the introduction, and assume you need to change
the word "verified" to "approved" in the resolution field.  We know from the
above information that the resolution is likely to be stored in the "bugs"
table. Note we'll need to change a little perl code as well as this database
change, but I won't plunge into that in this document. Let's verify the
information is stored in the "bugs" table:

mysql> show columns from bugs

  (exceedingly long output truncated here)
| bug_status| enum('UNCONFIRMED','NEW','ASSIGNED','REOPENED','RESOLVED','VERIFIED','CLOSED')||MUL | UNCONFIRMED||

  Sorry about that long line.  We see from this that the "bug status" column is
an "enum field", which is a MySQL peculiarity where a string type field can
only have certain types of entries.  While I think this is very cool, it's not
standard SQL.  Anyway, we need to add the possible enum field entry
'APPROVED' by altering the "bugs" table.

mysql> ALTER table bugs CHANGE bug_status bug_status
```

```
    -> enum("UNCONFIRMED", "NEW", "ASSIGNED", "REOPENED", "RESOLVED",
    -> "VERIFIED", "APPROVED", "CLOSED") not null;

    (note we can take three lines or more -- whatever you put in before the
semicolon is evaluated as a single expression)

Now if you do this:

mysql> show columns from bugs;

  you'll see that the bug_status field has an extra "APPROVED" enum that's
available!  Cool thing, too, is that this is reflected on your query page as
well -- you can query by the new status.  But how's it fit into the existing
scheme of things?
  Looks like you need to go back and look for instances of the word "verified"
in the perl code for Bugzilla -- wherever you find "verified", change it to
"approved" and you're in business (make sure that's a case-insensitive search).
Although you can query by the enum field, you can't give something a status
of "APPROVED" until you make the perl changes.   Note that this change I
mentioned can also be done by editing checksetup.pl, which automates a lot of
this.  But you need to know this stuff anyway, right?

  I hope this database tutorial has been useful for you.  If you have comments
to add, questions, concerns, etc. please direct them to
mbarnson@excitehome.net.  Please direct flames to /dev/null :)  Have a nice
day!



===
LINKS
===

Great MySQL tutorial site:
http://www.devshed.com/Server_Side/MySQL/
```

C.3. MySQL Permissions & Grant Tables

The following portion of documentation comes from my answer to an old discussion of Keystone, a cool product that does trouble−ticket tracking for IT departments. I wrote this post to the Keystone support group regarding MySQL grant table permissions, and how to use them effectively. It is badly in need of updating, as I believe MySQL has added a field or two to the grant tables since this time, but it serves as a decent introduction and troubleshooting document for grant table issues. I used Keynote to track my troubles until I discovered Bugzilla, which gave me a whole new set of troubles to work on :) Although it is of limited use, it still has SOME use, thus it's still included.

Please note, however, that I was a relatively new user to MySQL at the time. Some of my suggestions, particularly in how to set up security, showed a terrible lack of security−related database experience.

```
From matt_barnson@singletrac.com Wed Jul  7 09:00:07 1999
Date: Mon, 1 Mar 1999 21:37:04 -0700
From: Matthew Barnson matt_barnson@singletrac.com
To: keystone-users@homeport.org
Subject: [keystone-users] Grant Tables FAQ

    [The following text is in the "iso-8859-1" character set]
    [Your display is set for the "US-ASCII" character set]
    [Some characters may be displayed incorrectly]

Maybe we can include this rambling message in the Keystone FAQ?  It gets
asked a lot, and the only option current listed in the FAQ is
"--skip-grant-tables".

Really, you can't go wrong by reading section 6 of the MySQL manual, at
http://www.mysql.com/Manual/manual.html.  I am sure their description is
better than mine.

MySQL runs fine without permissions set up correctly if you run the mysql
daemon with the "--skip-grant-tables" option.  Running this way denies
access to nobody.  Unfortunately, unless you've got yourself firewalled it
also opens the potential for abuse if someone knows you're running it.

Additionally, the default permissions for MySQL allow anyone at localhost
access to the database if the database name begins with "test_" or is named
"test" (i.e. "test_keystone").  You can change the name of your database in
the keystone.conf file ($sys_dbname).  This is the way I am doing it for
some of my databases, and it works fine.

The methods described below assume you're running MySQL on the same box as
your webserver, and that you don't mind if your $sys_dbuser for Keystone has
superuser access.  See near the bottom of this message for a description of
what each field does.

Method #1:

1.  cd /var/lib
#location where you'll want to run /usr/bin/mysql_install_db shell
script from to get it to work.

2.  ln -s mysql data
```

```
# soft links the "mysql" directory to "data", which is what
mysql_install_db expects.  Alternately, you can edit mysql_install_db and
change all the "./data" references to "./mysql".

3.  Edit /usr/bin/mysql_install_db with your favorite text editor (vi,
emacs, jot, pico, etc.)
A)  Copy the "INSERT INTO db VALUES
('%','test\_%','','Y','Y','Y','Y','Y','Y');" and paste it immediately after
itself.  Chage the 'test\_%' value to 'keystone', or the value of
$sys_dbname in keystone.conf.
B)  If you are running your keystone database with any user, you'll need to
copy the "INSERT INTO user VALUES
('localhost','root','','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');" line after
itself and change 'root' to the name of the keystone database user
($sys_dbuser) in keystone.conf.

# adds entries to the script to create grant tables for specific
hosts and users.  The user you set up has super-user access ($sys_dbuser) --
you may or may not want this.  The layout of mysql_install_db is really very
uncomplicated.

4.  /usr/bin/mysqladmin shutdown
# ya gotta shut it down before you can reinstall the grant tables!

5.  rm -i /var/lib/mysql/mysql/*.IS?' and answer 'Y' to the deletion
questions.
# nuke your current grant tables.  This WILL NOT delete any other
databases than your grant tables.

6.  /usr/bin/mysql_install_db
# run the script you just edited to install your new grant tables.

7.  mysqladmin -u root password (new_password)
# change the root MySQL password, or else anyone on localhost can
login to MySQL as root and make changes.  You can skip this step if you want
keystone to connect as root with no password.

8.  mysqladmin -u (webserver_user_name) password (new_password)
# change the password of the $sys_dbuser.  Note that you will need
to change the password in the keystone.conf file as well in $sys_dbpasswd,
and if your permissions are set up incorrectly anybody can type the URL to
your keystone.conf file and get the password.  Not that this will help them
much if your permissions are set to @localhost.



Method #2:  easier, but a pain reproducing if you have to delete your grant
tables.  This is the "recommended" method for altering grant tables in
MySQL.  I don't use it because I like the other way :)

shell> mysql --user=root keystone
```

```
mysql> GRANT
SELECT,INSERT,UPDATE,DELETE,INDEX,ALTER,CREATE,DROP,RELOAD,SHUTDOWN,PROCESS,
FILE,
           ON keystone.*
           TO <$sys_dbuser name>@localhost
           IDENTIFIED BY '(password)'
     WITH GRANT OPTION;

OR

mysql> GRANT ALL PRIVELEGES
ON keystone.*
TO <$sys_dbuser name>@localhost
IDENTIFIED BY '(password)'
WITH GRANT OPTION;

# this grants the required permissions to the keystone ($sys_dbuser)
account defined in keystone.conf.  However, if you are runnning many
different MySQL-based apps, as we are, it's generally better to edit the
mysql_install_db script to be able to quickly reproduce your permissions
structure again.  Note that the FILE privelege and WITH GRANT OPTION may not
be in your best interest to include.


GRANT TABLE FIELDS EXPLANATION:
Quick syntax summary:  "%" in MySQL is a wildcard.  I.E., if you are
defining your DB table and in the 'host' field and enter '%', that means
that any host can access that database.  Of course, that host must also have
a valid db user in order to do anything useful.  'db'=name of database.  In
our case, it should be "keystone".  "user" should be your "$sys_dbuser"
defined in keystone.conf.  Note that you CANNOT add or change a password by
using the "INSERT INTO db (X)" command -- you must change it with the mysql
-u command as defined above.  Passwords are stored encrypted in the MySQL
database, and if you try to enter it directly into the table they will not
match.

TABLE:  USER.  Everything after "password" is a privelege granted (Y/N).
This table controls individual user global access rights.

'host','user','password','select','insert','update','delete','index','alter'
,'create','drop','grant','reload','shutdown','process','file'

TABLE:  DB.  This controls access of USERS to databases.

'host','db','user','select','insert','update','delete','index','alter','crea
te','drop','grant'

TABLE:  HOST.  This controls which HOSTS are allowed what global access
rights.  Note that the HOST table, USER table, and DB table are very closely
connected -- if an authorized USER attempts an SQL request from an
```

```
unauthorized HOST, she's denied.  If a request from an authorized HOST is
not an authorized USER, it is denied.  If a globally authorized USER does
not have rights to a certain DB, she's denied.  Get the picture?

'host','db','select','insert','update','delete','index','alter','create','dr
op','grant'


You should now have a working knowledge of MySQL grant tables.  If there is
anything I've left out of this answer that you feel is pertinent, or if my
instructions don't work for you, please let me know and I'll re-post this
letter again, corrected.  I threw it together one night out of exasperation
for all the newbies who don't know squat about MySQL yet, so it is almost
guaranteed to have errors.

Once again, you can't go wrong by reading section 6 of the MySQL manual.  It
is more detailed than I!
http://www.mysql.com/Manual/manual.html.
```

Appendix D. Useful Patches and Utilities for Bugzilla

Are you looking for a way to put your Bugzilla into overdrive? Catch some of the niftiest tricks here in this section.

D.1. Apache `mod_rewrite` magic

Apache's `mod_rewrite` module lets you do some truly amazing things with URL rewriting. Here are a couple of examples of what you can do.

1. Make it so if someone types `http://www.foo.com/12345`, Bugzilla spits back http://www.foo.com/show_bug.cgi?id=12345. Try setting up your VirtualHost section for Bugzilla with a rule like this:
   ```
   <VirtualHost 12.34.56.78>
   RewriteEngine On
   RewriteRule ^/([0-9]+)$ http://foo.bar.com/show_bug.cgi?id=$1 [L,R]
   </VirtualHost>
   ```
2. There are many, many more things you can do with mod_rewrite. As time goes on, I will include many more in the Guide. For now, though, please refer to the mod_rewrite documentation at http://www.apache.org

D.2. The setperl.csh Utility

You can use the "setperl.csh" utility to quickly and easily change the path to perl on all your Bugzilla files. This is a C−shell script; if you do not have "csh" or "tcsh" in the search path on your system, it will not work!

1. Download the "setperl.csh" utility to your Bugzilla directory and make it executable.
    a. bash# **cd /your/path/to/bugzilla**
    b. bash# **wget -O setperl.csh 'http://bugzilla.mozilla.org/showattachment.cgi?attach_id=10795'**
    c. bash# **chmod u+x setperl.csh**
2. Prepare (and fix) Bugzilla file permissions.
    a. bash# **chmod u+w ***
    b. bash# **chmod u+x duplicates.cgi**
    c. bash# **chmod a-x bug_status.html**
3. Run the script:

    bash# **./setperl.csh /your/path/to/perl**

    Example D-1. Using Setperl to set your perl path

    bash# **./setperl.csh /usr/bin/perl**

D.3. Command-line Bugzilla Queries

Users can query Bugzilla from the command line using this suite of utilities.

The query.conf file contains the mapping from options to field names and comparison types. Quoted option names are "grepped" for, so it should be easy to edit this file. Comments (#) have no effect; you must make sure these lines do not contain any quoted "option"

buglist is a shell script which submits a Bugzilla query and writes the resulting HTML page to stdout. It supports both short options, (such as "-Afoo" or "-Rbar") and long options (such as "--assignedto=foo" or "--reporter=bar"). If the first character of an option is not "-", it is treated as if it were prefixed with "--default=".

The columlist is taken from the COLUMNLIST environment variable. This is equivalent to the "Change Columns" option when you list bugs in buglist.cgi. If you have already used Bugzilla, use **grep COLUMLIST ~/.netscape/cookies** to see your current COLUMNLIST setting.

bugs is a simple shell script which calls buglist and extracts the bug numbers from the output. Adding the prefix "http://bugzilla.mozilla.org/buglist.cgi?bug_id=" turns the bug list into a working link if any bugs are found. Counting bugs is easy. Pipe the results through **sed -e 's/,/ /g' | wc | awk '{printf $2 "\n"}'**

Akkana says she has good results piping buglist output through **w3m -T text/html -dump**

1. Download three files:
    a. bash$ **wget -O query.conf 'http://bugzilla.mozilla.org/showattachment.cgi?attach_id=26157'**
    b. bash$ **wget -O buglist 'http://bugzilla.mozilla.org/showattachment.cgi?attach_id=26944'**
    c. bash# **wget -O bugs 'http://bugzilla.mozilla.org/showattachment.cgi?attach_id=26215'**
2. Make your utilities executable: bash$ **chmod u+x buglist bugs**

D.4. The Quicksearch Utility

Quicksearch is a new, experimental feature of the 2.12 release. It consist of two Javascript files, "quicksearch.js" and "localconfig.js", and two documentation files, "quicksearch.html" and "quicksearchhack.html"

The index.html page has been updated to include the QuickSearch text box.

To take full advantage of the query power, the Bugzilla maintainer must edit "localconfig.js" according to the value sets used in the local installation.

Currently, keywords must be hard−coded in localconfig.js. If they are not, keywords are not automatically recognized. This means, if localconfig.js is left unconfigured, that searching for a bug with the "foo" keyword will only find bugs with "foo" in the summary, status whiteboard, product or component name, but not those with the keyword "foo".

Workarounds for Bugzilla users:

search for '!foo' (this will find only bugs with the keyword "foo"
search 'foo,!foo' (equivalent to 'foo OR keyword:foo')

When this tool is ported from client−side JavaScript to server−side Perl, the requirement for hard−coding keywords can be fixed. This bug has details.

D.5. Hacking Bugzilla

What follows are some general guidelines for changing Bugzilla, and adhering to good coding practice while doing so. We've had some checkins in the past which ruined Bugzilla installations because of disregard for these conventions. Sorry for the lack of formatting; I got this info into the Guide on the day of 2.14 release and haven't formatted it yet.

```
The following is a guide for reviewers when checking code into Bugzilla's
CVS repostory at mozilla.org.  If you wish to submit patches to Bugzilla,
you should follow the rules and style conventions below.  Any code that
does not adhere to these basic rules will not be added to Bugzilla's
codebase.

 1. Usage of variables in Regular Expressions

    It is very important that you don't use a variable in a regular
    expression unless that variable is supposed to contain an expression.
    This especially applies when using grep.  You should use:

    grep ($_ eq $value, @array);

    - NOT -

    grep (/$value/, @array);
```

If you need to use a non-expression variable inside of an expression, be sure to quote it properly (using \Q..\E).

Coding Style for Bugzilla

While it's true that not all of the code currently in Bugzilla adheres to this styleguide, it is something that is being worked toward. Therefore, we ask that all new code (submitted patches and new files) follow this guide as closely as possible (if you're only changing 1 or 2 lines, you don't have to reformat the entire file :).

1. Whitespace

   Bugzilla's prefered indentation is 4 spaces (no tabs, please).

2. Curly braces.

   The opening brace of a block should be on the same line as the statement that is causing the block and the closing brace should be at the same indentation level as that statement, for example:

```
if ($var) {
    print "The variable is true";
} else {
    print "Try again";
}

- NOT -

if ($var)
{
    print "The variable is true";
}
else
{
    print "Try again";
}
```

3. File Names

   File names for bugzilla code and support documention should be legal across multiple platforms. \ / : * ? " < > and | are all illegal characters for filenames on various platforms. Also, file names should not have spaces in them as they can cause confusion in CVS and other mozilla.org utilities.

4. Variable Names

   If a variable is scoped globally ($::variable) its name should be descriptive of what it contains. Local variables can be named a bit looser, provided the

```
   context makes their content obvious.  For example, $ret could be used as a
   staging variable for a routine's return value as the line |return $ret;| will
   make it blatently obvious what the variable holds and most likely be shown
   on the same screen as |my $ret = "";|.

5. Cross Database Compatability

   Bugzilla was originally written to work with MySQL and therefore took advantage
   of some of its features that aren't contained in other RDBMS software.  These
   should be avoided in all new code.  Examples of these features are enums and
   encrypt().

6. Cross Platform Compatability

   While Bugzilla was written to be used on Unix based systems (and Unix/Linux is
   still the only officially supported platform) there are many who desire/need to
   run Bugzilla on Microsoft Windows boxes.  Whenever possible, we should strive
   not to make the lives of these people any more complicated and avoid doing things
   that break Bugzilla's ability to run on multiple operating systems.
```

Appendix E. GNU Free Documentation License

Version 1.1, March 2000

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front−matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front−Cover Texts or Back−Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine−readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard−conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine−generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this

License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front−Cover Texts on the front cover, and Back−Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine−readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly−accessible computer−network location containing a complete Transparent copy of the Document, free of added material, which the general network−using public has access to download anonymously at no charge using public−standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front−matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties−−for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front−Cover Text, and a passage of up to 25 words as a Back−Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front−Cover Text and one of Back−Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self−contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

---

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front−Cover Texts being LIST, and with the Back−Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front−Cover Texts, write "no Front−Cover Texts" instead of "Front−Cover Texts being LIST"; likewise for Back−Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Glossary

0−9, high ascii

.htaccess

Apache web server, and other NCSA−compliant web servers, observe the convention of using files in directories called `.htaccess` files. These restrict parameters of the web server. In Bugzilla, they are used to restrict access to certain files which would otherwise compromise your installation. For instance, the `localconfig` file contains the password to your database. If this information were generally available, and remote access to your database turned on, you risk corruption of your database by computer criminals or the curious.

A

Apache

In this context, Apache is the web server most commonly used for serving up *Bugzilla* pages. Contrary to popular belief, the apache web server has nothing to do with the ancient and noble Native American tribe, but instead derived its name from the fact that it was "a patchy" version of the original NCSA world−wide−web server.

B

Bug

A "Bug" in Bugzilla refers to an issue entered into the database which has an associated number, assignments, comments, etc. Some also refer to a "tickets" or "issues"; in the context of Bugzilla, they are synonymous.

Bug Number

Each Bugzilla Bug is assigned a number that uniquely identifies that Bug. The Bug associated with a Bug Number can be pulled up via a query, or easily from the very front page by typing the number in the "Find" box.

Bug Life Cycle

A Bug has stages through which it must pass before becoming a "closed bug", including acceptance, resolution, and verification. The "Bug Life Cycle" is moderately flexible according to the needs of the organization using it, though.

Bugzilla

Bugzilla is the industry−standard bug tracking system. It is quite popular among Open Source enthusiasts.

Component

A Component is a subsection of a Product. It should be a narrow category, tailored to your organization. All Products must contain at least one Component (and, as a matter of fact, creating a Product with no Components will create an error in Bugzilla).

CPAN

CPAN stands for the "Comprehensive Perl Archive Network". CPAN maintains a large number of extremely useful *Perl* modules. By themselves, Perl modules generally do nothing, but when used as part of a larger program, they provide much−needed algorithms and functionality.

D

daemon

A daemon is a computer program which runs in the background. In general, most daemons are started at boot time via System V init scripts, or through RC scripts on BSD−based systems. *mysqld*, the MySQL server, and *apache*, a web server, are generally run as daemons.

Groups

The word "Groups" has a very special meaning to Bugzilla. Bugzilla's main security mechanism comes by lumping users into groups, and assigning those groups certain privileges to *Products* and *Components* in the *Bugzilla* database.

I

Infinite Loop

A loop of information that never ends; see recursion.

M

mysqld

mysqld is the name of the *daemon* for the MySQL database. In general, it is invoked automatically through the use of the System V init scripts on GNU/Linux and AT&T System V−based systems, such as Solaris and HP/UX, or through the RC scripts on BSD−based systems.

P

Product

A Product is a broad category of types of bugs. In general, there are several Components to a Product. A Product also defines a default Group (used for Bug Security) for all bugs entered into components beneath it.

Example 1. A Sample Product

A company sells a software product called "X". They also maintain some older software called "Y", and have a secret project "Z". An effective use of Products might be to create Products "X", "Y", "Z", each with Components of User Interface, Database, and Business Logic. They might also change group permissions so that only those people who are members of Group "Z" can see components and bugs under Product "Z".

Perl

First written by Larry Wall, Perl is a remarkable program language. It has the benefits of the flexibility of an interpreted scripting language (such as shell script), combined with the speed and power of a compiled language, such as C. *Bugzilla* is maintained in Perl.

Q

QA

"QA", "Q/A", and "Q.A." are short for "Quality Assurance". In most large software development organizations, there is a team devoted to ensuring the product meets minimum standards before shipping. This team will also generally want to track the progress of bugs over their life cycle, thus the need for the "QA Contact" field in a Bug.

R

Recursion

The property of a function looking back at itself for something. "GNU", for instance, stands for "GNU's Not UNIX", thus recursing upon itself for definition. For further clarity, see Infinite Loop.

S

SGML

SGML stands for "Standard Generalized Markup Language". Created in the 1980's to provide an extensible means to maintain documentation based upon content instead of presentation, SGML has withstood the test of time as a robust, powerful language. *XML* is the "baby brother" of SGML; any valid XML document it, by definition, a valid SGML document. The document you are reading is written and maintained in SGML, and is also valid XML if you modify the Document Type Definition.

T

Target Milestone

Target Milestones are Product goals. They are configurable on a per−Product basis. Most software development houses have a concept of "milestones" where the people funding a project expect certain functionality on certain dates. Bugzilla facilitates meeting these milestones by giving you the ability to declare by which milestone a bug will be fixed, or an enhancement will be implemented.

Z

Zarro Boogs Found

This is the cryptic response sent by Bugzilla when a query returned no results. It is just a goofy way of saying "Zero Bugs Found".