The Bugzilla Guide - 2.17.7 Development Release

The Bugzilla Team

The Bugzilla Guide - 2.17.7 Development Release

by The Bugzilla Team

Published 2004-02-05

This is the documentation for Bugzilla, a bug-tracking system from mozilla.org. Bugzilla is an enterprise-class piece of software that tracks millions of bugs and issues for hundreds of organizations around the world.

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

Table of Contents

1. About This Guide **1**
1.1. Copyright Information 1
1.2. Disclaimer 1
1.3. New Versions 1
1.4. Credits 2
1.5. Document Conventions 2
2. Installing Bugzilla **4**
2.1. Installation 4
2.2. Configuration 8
2.3. Optional Additional Configuration 13
2.4. OS-Specific Installation Notes 17
2.5. Troubleshooting 20
3. Administering Bugzilla **23**
3.1. Bugzilla Configuration 23
3.2. User Administration 24
3.3. Products 26
3.4. Components 27
3.5. Versions 27
3.6. Milestones 27
3.7. Voting 28
3.8. Groups and Group Security 28
3.9. Upgrading to New Releases 29
4. Customising Bugzilla **33**
4.1. Template Customization 33
4.2. Template Hooks 36
4.3. Customizing Who Can Change What 38
4.4. Modifying Your Running System 39
4.5. MySQL Bugzilla Database Introduction 39
4.6. Integrating Bugzilla with Third-Party Tools 45
5. Using Bugzilla **46**
5.1. Introduction 46
5.2. Create a Bugzilla Account 46
5.3. Anatomy of a Bug 46
5.4. Searching for Bugs 47
5.5. Bug Lists 48
5.6. Filing Bugs 48
5.7. Patch Viewer 48
5.8. Hints and Tips 50
5.9. User Preferences 51
5.10. Reports 52
A. The Bugzilla FAQ **53**
B. Contrib **62**
B.1. Command-line Search Interface 62

C. Manual Installation of Perl Modules 63
C.1. Instructions 63
C.2. Download Locations 63
D. GNU Free Documentation License 66
0. Preamble 66
1. Applicability and Definition 66
2. Verbatim Copying 67
3. Copying in Quantity 67
4. Modifications 68
5. Combining Documents 69
6. Collections of Documents 69
7. Aggregation with Independent Works 69
8. Translation 70
9. Termination 70
10. Future Revisions of this License 70
How to use this License for your documents 70
Glossary 72

List of Examples

3-1. Upgrading using CVS....30
3-2. Upgrading using the tarball....31
3-3. Upgrading using patches....31

Chapter 1. About This Guide

1.1. Copyright Information

1.2. Disclaimer

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

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

Although the Bugzilla development team has taken great care to ensure that all exploitable bugs have been fixed, security holes surely exist in any piece of code. Great care should be taken both in the installation and usage of this software. The Bugzilla development team members assume no liability for your use of Bugzilla. You have the source code, and are responsible for auditing it yourself to ensure your security needs are met.

1.3. New Versions

This is the 2.17.7 version of The Bugzilla Guide. It is so named to match the current version of Bugzilla. This version of the guide, like its associated Bugzilla version, is a development version.

The latest version of this guide can always be found at http://www.bugzilla.org, or checked out via CVS by following the Mozilla CVS (http://www.mozilla.org/cvs.html) instructions and check out the `mozilla/webtools/bugzilla/docs/` subtree. However, you should read the version which came with the Bugzilla release you are using.

The Bugzilla Guide, or a section of it, is also available in the following languages: German (http://bugzilla-de.sourceforge.net/docs/html/).

In addition, there are Bugzilla template localisation projects in the following languages. They may have translated documentation available: Belarusian (http://sourceforge.net/projects/bugzilla-be/), Brazilian Portuguese (http://sourceforge.net/projects/bugzilla-br/), Chinese (http://sourceforge.net/projects/bugzilla-cn/), French (http://sourceforge.net/projects/bugzilla-fr/), German (http://sourceforge.net/projects/bugzilla-de/), Korean (http://sourceforge.net/projects/bugzilla-kr/), Russian (http://sourceforge.net/projects/bugzilla-ru/) and Spanish (http://sourceforge.net/projects/bugzilla-es/).

If you would like to volunteer to translate the Guide into additional languages, please contact Dave Miller (mailto:justdave@syndicomm.com).

1.4. Credits

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

Matthew P. Barnson, Kevin Brannen, Dawn Endico, Ben FrantzDale, Eric Hanson, Tara Hernandez, Dave Lawrence, Zach Lipton, Gervase Markham, Andrew Pearson, Joe Robins, Spencer Smith, Jacob Steenhagen, Ron Teitelbaum, Terry Weissman, Martin Wulffeld.

Also, thanks are due to the members of the netscape.public.mozilla.webtools (news://news.mozilla.org/netscape.public.mozilla.webtools) newsgroup. Without your discussions, insight, suggestions, and patches, this could never have happened.

1.5. Document Conventions

This document uses the following conventions:

Descriptions	Appearance
Warning	**Caution** Don’t run with scissors!
Hint	**Tip:** Would you like a breath mint?
Note	**Note:** Dear John...
Information requiring special attention	**Warning** Read this or the cat gets it.
File or directory name	`filename`
Command to be typed	**command**
Application name	application
Normal user’s prompt under bash shell	bash$
Root user’s prompt under bash shell	bash#
Normal user’s prompt under tcsh shell	tcsh$
Environment variables	VARIABLE
Term found in the glossary	*Bugzilla*
Code example	`<para> Beginning and end of paragraph </para>`

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

Chapter 2. Installing Bugzilla

2.1. Installation

> **Note:** If you just want to *use* Bugzilla, you do not need to install it. None of this chapter is relevant to you. Ask your Bugzilla administrator for the URL to access it over the web.

The Bugzilla server software is usually installed on Linux or Solaris. If you are installing on another OS, check Section 2.4 before you start your installation to see if there are any special instructions.

As an alternative to following these instructions, you may wish to try Arne Schirmacher's unofficial and unsupported Bugzilla Installer (http://www.softwaretesting.de/article/view/33/1/8/), which installs Bugzilla and all its prerequisites on Linux or Solaris systems.

This guide assumes that you have administrative access to the Bugzilla machine. It not possible to install and run Bugzilla itself without administrative access except in the very unlikely event that every single prerequisite is already installed.

> **Warning**
>
> The installation process may make your machine insecure for short periods of time. Make sure there is a firewall between you and the Internet.

You are strongly recommended to make a backup of your system before installing Bugzilla (and at regular intervals thereafter :-).

In outline, the installation proceeds as follows:

1. Install Perl (5.6.0 or above)
2. Install MySQL (3.23.41 or above)
3. Install a Webserver
4. Install Bugzilla
5. Install Perl modules
6. Configure all of the above.

2.1.1. Perl

Installed Version Test: `perl -v`

Any machine that doesn't have Perl on it is a sad machine indeed. If you don't have it and your OS doesn't provide official packages, visit http://www.perl.com. Although Bugzilla runs with Perl 5.6.0, it's a good idea to be using the latest stable version. As of this writing, that is Perl 5.8.2.

2.1.2. MySQL

Installed Version Test: `mysql -V`

If you don't have it and your OS doesn't provide official packages, visit http://www.mysql.com. You need MySQL version 3.23.41 or higher.

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

If you install from something other than a packaging/installation system (such as .rpm, .dep, .exe, or .msi) make sure the MySQL server is started when the machine boots.

2.1.3. Web Server

Installed Version Test: view the default welcome page at http://<your-machine>/

You have freedom of choice here, pretty much any web server that is capable of running *CGI* scripts will work. However, we strongly recommend using the Apache web server (either 1.3.x or 2.x), and the installation instructions usually assume you are using it. If you have got Bugzilla working using another webserver, please share your experiences with us by filing a bug in Bugzilla Documentation (http://bugzilla.mozilla.org/enter_bug.cgi?product=Bugzilla&component=Documentation).

If you don't have Apache and your OS doesn't provide official packages, visit http://httpd.apache.org/.

2.1.4. Bugzilla

Download a Bugzilla tarball (or check it out from CVS) and place it in a suitable directory, writable by the default web server user (probably "nobody"). Good locations are either directly in the main web space for your web server or perhaps in `/usr/local` with a symbolic link from the web space.

> **Caution**
>
> The default Bugzilla distribution is not designed to be placed in a `cgi-bin` directory. This includes any directory which is configured using the `ScriptAlias` directive of Apache.

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 `checksetup.pl` script, which locks down your installation.

2.1.5. Perl Modules

Bugzilla's installation process is based on a script called `checksetup.pl`. The first thing it checks is whether you have appropriate versions of all the required Perl modules. The aim of this section is to pass this check. When it passes, *do not run it again*, but proceed to Section 2.2.

At this point, you need to `su` to root. You should remain as root until the end of the install. Then run:

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

`checksetup.pl` will print out a list of the required and optional Perl modules, together with the versions (if any) installed on your machine. The list of required modules is reasonably long; however, you may already have several of them installed.

There is a meta-module called Bundle::Bugzilla, which installs all the other modules with a single command. You should use this if you are running Perl 5.6.1 or above.

The preferred way of installing Perl modules is via CPAN on Unix, or PPM on Windows (see Section 2.4.1.2). These instructions assume you are using CPAN; if for some reason you need to install the Perl modules manually, see Appendix C.

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

If you using Bundle::Bugzilla, invoke the magic CPAN command on it. Otherwise, you need to work down the list of modules that `checksetup.pl` says are required, in the order given, invoking the command on each.

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

Here is a complete list of modules and their minimum versions. Some modules have special installation notes, which follow.

Required Perl modules:

1. AppConfig (1.52)
2. CGI (2.93)
3. Data::Dumper (any)
4. Date::Format (2.21)
5. DBI (1.32)
6. DBD::mysql (2.1010)
7. File::Spec (0.82)
8. File::Temp (any)
9. Template (2.08)
10. Text::Wrap (2001.0131)

Optional Perl modules:

1. GD (1.20) for bug charting
2. Chart::Base (0.99c) for bug charting
3. GD::Graph (any) for bug charting

4. GD::Text::Align (any) for bug charting
5. XML::Parser (any) for the XML interface
6. PatchReader (0.9.1) for pretty HTML view of patches
7. MIME::Parser (any) for the optional email interface

2.1.5.1. DBD::mysql

The installation process will ask you a few questions about the desired compilation target and your MySQL installation. For most of the questions the provided default will be adequate, but when asked if your desired target is the MySQL or mSQL packages, you should select the MySQL-related ones. Later you will be asked if you wish to provide backwards compatibility with the older MySQL packages; you should answer YES to this question. The default is NO.

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

2.1.5.2. Template Toolkit (2.08)

When you install Template Toolkit, you'll get asked various questions about features to enable. The defaults are fine, except that it is recommended you use the high speed XS Stash of the Template Toolkit, in order to achieve best performance.

2.1.5.3. GD (1.20)

The GD module is only required if you want graphical reports.

> **Note:** The Perl GD module 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 module README. If compiling GD fails, it's probably because you're missing a required library.

> **Tip:** The version of the GD module you need is very closely tied to the `libgd` version installed on your system. If you have a version 1.x of `libgd` the 2.x versions of the GD module won't work for you.

2.1.5.4. Chart::Base (0.99c)

The Chart::Base module is only required if you want graphical reports. Note that earlier versions that 0.99c used GIFs, which are no longer supported by the latest versions of GD.

2.1.5.5. GD::Graph (any)

The GD::Graph module is only required if you want graphical reports.

2.1.5.6. GD::Text::Align (any)

The GD::Text::Align module is only required if you want graphical reports.

2.1.5.7. XML::Parser (any)

The XML::Parser module is only required if you want to import XML bugs using the `importxml.pl` script. This is required to use Bugzilla's "move bugs" feature; you may also want to use it for migrating from another bug database. XML::Parser requires that the `expat` library is already installed on your machine.

2.1.5.8. MIME::Parser (any)

The MIME::Parser module is only required if you want to use the email interface located in the `contrib` directory.

2.1.5.9. PatchReader (0.9.1)

The PatchReader module is only required if you want to use Patch Viewer, a Bugzilla feature to show code patches in your web browser in a more readable form.

2.2. Configuration

> **Warning**
>
> Poorly-configured MySQL and Bugzilla installations have given attackers full access to systems in the past. Please take the security parts of these guidelines seriously, even for Bugzilla machines hidden away behind your firewall.

2.2.1. localconfig

Once you run `checksetup.pl` with all the correct modules installed, it displays a message about, and write out a file called, `localconfig`. This file contains the default settings for a number of Bugzilla parameters.

Load this file in your editor. The only value you *need* to change is $db_pass, the password for the user you will create for your database. Pick a strong password (for simplicity, it should not contain single quote characters) and put it here.

The other options in the `localconfig` file are documented by their accompanying comments. If you have a slightly non-standard MySQL setup, you may wish to change one or more of the other "$db_*" parameters.

You may also wish to change the names of the priorities, severities, operating systems and platforms for your installation. However, you can always change these after installation has finished; if you then re-run `checksetup.pl`, the changes will get picked up.

2.2.2. MySQL

2.2.2.1. Security

MySQL ships as insecure by default. It allows anybody to on the local machine full administrative capabilities without requiring a password; the special MySQL root account (note: this is *not* the same as the system root) also has no password. Also, many installations default to running mysqld as the system root.

1. To disable the anonymous user account and set a password for the root user, execute the following. The root user password should be different to the bugs user password you set in `localconfig` in the previous section, and also different to the password for the system root account on your machine.

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

   From this point forward, to run the `mysql` command-line client, you will need to type **mysql -u root -p** and enter *`new_password`* when prompted.

2. If you run MySQL on the same machine as your web server, you should disable remote access to MySQL by adding the following to your `/etc/my.conf`:

   ```
   [myslqd]
   # Prevent network access to MySQL.
   skip-networking
   ```

3. Consult the documentation that came with your system for information on making mysqld run as an unprivileged user.
4. For added security, you could also run MySQL, or even all of Bugzilla in a chroot jail; however, instructions for doing that are beyond the scope of this document.

2.2.2.2. Allow large attachments

You need to configure MySQL to accept large packets, if you want to have attachments larger than 64K. Add the text below to your `/etc/my.conf`. There is also a parameter in Bugzilla for setting the maximum allowable attachment size, (default 1MB). Bugzilla will only accept attachments up to the lower of these two sizes.

```
[mysqld]
# Allow packets up to 1M
set-variable = max_allowed_packet=1M
```

2.2.2.3. Add a user to MySQL

You need to add a new MySQL user for Bugzilla to use. (It's not safe to have Bugzilla use the MySQL root account.) The following instructions assume the defaults in `localconfig`; if you changed those, you need to modify the SQL command appropriately. You will need the *`$db_pass`* password you set in `localconfig` in Section 2.2.1.

We use an SQL **GRANT** command to create a "bugs" user. 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.

Run the `mysql` command-line client and enter:

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,INDEX,ALTER,CREATE,
       DROP,REFERENCES ON bugs.* TO bugs@localhost
       IDENTIFIED BY '$db_pass';
mysql> FLUSH PRIVILEGES;
```

Note: If you are using MySQL 4, you need to add the `LOCK TABLES` and `CREATE TEMPORARY TABLES` permissions to the list.

2.2.3. checksetup.pl

Next, rerun `checksetup.pl`. It reconfirms that all the modules are present, and notices the altered localconfig file, which it assumes you have edited to your satisfaction. It compiles the UI templates, connects to the database using the 'bugs' user you created and the password you defined, and creates the 'bugs' database and the tables therein.

After that, it asks for details of an administrator account. Bugzilla can have multiple administrators - you can create more later - but it needs one to start off with. Enter the email address of an administrator, his or her full name, and a suitable Bugzilla password.

`checksetup.pl` will then finish. You may rerun `checksetup.pl` at any time if you wish.

2.2.4. Web server

Configure your web server according to the instructions in the appropriate section. The Bugzilla Team recommends Apache.

2.2.4.1. Apache httpd

Load `httpd.conf` in your editor.

Uncomment (or add) the following line. This configures Apache to run .cgi files outside the `cgi-bin` directory.

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

Apache uses `<Directory>` directives to permit fine-grained permission setting. Add the following two lines to a `<Directory>` directive that applies either to the Bugzilla directory or one of its parents (e.g. the `<Directory`

`/var/www/html>` directive). This allows Bugzilla's `.htaccess` files to override global permissions, and allows .cgi files to run in the Bugzilla directory.

```
Options +ExecCGI +FollowSymLinks
AllowOverride Limit
```

Add `index.cgi` to the end of the `DirectoryIndex` line.

`checksetup.pl` can set tighter permissions on Bugzilla's files and directories if it knows what user the webserver runs as. Look for the `User` line in `httpd.conf`, and place that value in the *`$webservergroup`* variable in `localconfig`. Then rerun `checksetup.pl`.

2.2.4.2. Microsoft Internet Information Services

If you need, or for some reason even want, to use Microsoft's Internet Information Services or Personal Web Server you should be able to. You will need to configure them to know how to run CGI scripts. This is described in Microsoft Knowledge Base article Q245225 (http://support.microsoft.com/support/kb/articles/Q245/2/25.asp) for Internet Information Services and Q231998 (http://support.microsoft.com/support/kb/articles/Q231/9/98.asp) for Personal Web Server.

Also, and this can't be stressed enough, make sure that files such as `localconfig` and your `data` directory are secured as described in Section 2.2.4.4.

2.2.4.3. AOL Server

Ben FrantzDale reported success using AOL Server with Bugzilla. He reported his experience and what appears below is based on that.

AOL Server will have to be configured to run *CGI* scripts, please consult the documentation that came with your server for more information on how to do this.

Because AOL Server doesn't support `.htaccess` files, you'll have to create a *TCL* script. You should create an `aolserver/modules/tcl/filter.tcl` file (the filename shouldn't matter) with the following contents (change `/bugzilla/` to the web-based path to your Bugzilla installation):

```
ns_register_filter preauth GET /bugzilla/localconfig filter_deny
ns_register_filter preauth GET /bugzilla/localconfig~ filter_deny
ns_register_filter preauth GET /bugzilla/\#localconfig\# filter_deny
ns_register_filter preauth GET /bugzilla/*.pl filter_deny
ns_register_filter preauth GET /bugzilla/syncshadowdb filter_deny
ns_register_filter preauth GET /bugzilla/runtests.sh filter_deny
ns_register_filter preauth GET /bugzilla/data/* filter_deny
ns_register_filter preauth GET /bugzilla/template/* filter_deny

proc filter_deny { why } {
    ns_log Notice "filter_deny"
    return "filter_return"
}
```

> **Warning**
>
> This probably doesn't account for all possible editor backup files so you may wish to add some additional variations of `localconfig`. For more information, see bug 186383 (http://bugzilla.mozilla.org/show_bug.cgi?id=186383) or Bugtraq ID 6501 (http://online.securityfocus.com/bid/6501).

Note: If you are using webdot from research.att.com (the default configuration for the `webdotbase` paramater), you will need to allow access to `data/webdot/*.dot` for the reasearch.att.com machine.

If you are using a local installation of GraphViz (http://www.graphviz.org), you will need to allow everybody to access `*.png`, `*.gif`, `*.jpg`, and `*.map` in the `data/webdot` directory.

2.2.4.4. Web Server Access Controls

Users of Apache can skip this section because Bugzilla ships with `.htaccess` files which restrict access in the manner required. Users of other webservers, read on.

There are several files in the Bugzilla directory that should not be accessible from the web. You need to configure your webserver so they they aren't. Not doing this may reveal sensitive information such as database passwords.

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

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

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

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

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

- In `Bugzilla`:

  - Block everything

- In `template`:
  - Block everything

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

2.2.5. Bugzilla

Your Bugzilla should now be working. Access `http://<your-bugzilla-server>/` - you should see the Bugzilla front page. If not, consult the Troubleshooting section, Section 2.5.

Log in with the administrator account you defined in the last `checksetup.pl` run. You should go through the parameters on the Edit Parameters page (see link in the footer) and see if there are any you wish to change. They key parameters are documented in Section 3.1; you should certainly alter **maintainer** and **urlbase**; you may also want to alter **cookiepath** or **requirelogin**.

This would also be a good time to revisit the `localconfig` file and make sure that the names of the priorities, severities, platforms and operating systems are those you wish to use when you start creating bugs. Remember to rerun `checksetup.pl` if you change it.

Bugzilla has several optional features which require extra configuration. You can read about those in Section 2.3.

2.3. Optional Additional Configuration

Bugzilla has a number of optional features. This section describes how to configure or enable them.

2.3.1. Bug Graphs

If you have installed the necessary Perl modules you can start collecting statistics for the nifty Bugzilla graphs.

```
bash# crontab -e
```

This should bring up the crontab file in your editor. Add a cron entry like this to run `collectstats.pl` daily at 5 after midnight:

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

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

2.3.2. Dependency Charts

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

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

The easiest way to get this working is to install GraphViz (http://www.graphviz.org/). If you do that, you need to enable server-side image maps (http://httpd.apache.org/docs/mod/mod_imap.html) in Apache. Alternatively, you could set up a webdot server, or use the AT&T public webdot server. This is the default for the webdotbase param, but it's often overloaded and slow. Note that AT&T's server won't work if Bugzilla is only accessible using HARTS. *Editor's note: What the heck is HARTS? Google doesn't know...*

2.3.3. The Whining Cron

What good are bugs if they're not annoying? To help make them more so you can set up Bugzilla's automatic whining system to complain at engineers which leave their bugs in the NEW or REOPENED state without triaging them.

This can be done by adding the following command as a daily crontab entry, in the same manner as explained above for bug graphs. This example runs it at 12.55am.

```
55 0 * * * cd <your-bugzilla-directory> ; ./whineatnews.pl
```

2.3.4. Patch Viewer

Patch Viewer is the engine behind Bugzilla's graphical display of code patches. You can integrate this with copies of the `cvs`, `lxr` and `bonsai` tools if you have them, by giving the locations of your installation of these tools in `editparams.cgi`.

Patch Viewer also optionally will use the `cvs`, `diff` and `interdiff` command-line utilities if they exist on the system. Interdiff can be obtained from http://cyberelk.net/tim/patchutils/. If these programs are not in the system path, you can configure their locations in `localconfig`.

2.3.5. LDAP Authentication

LDAP authentication is a module for Bugzilla's plugin authentication architecture.

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.

> **Caution**
>
> Because the Bugzilla account is not created until the first time a user logs in, a user who has not yet logged is unknown to Bugzilla. This means they cannot be used as an assignee or QA contact (default or otherwise), added to any cc list, or any other such operation. One possible workaround is the `bugzilla_ldapsync.rb` script in the *`contrib`* directory. Another possible solution is fixing bug 201069 (http://bugzilla.mozilla.org/show_bug.cgi?id=201069).

Parameters required to use LDAP Authentication:

loginmethod

This parameter should be set to "LDAP" *only* if you will be using an LDAP directory for authentication. If you set this param to "LDAP" but fail to set up the other parameters listed below you will not be able to log back in to Bugzilla one you log out. If this happens to you, you will need to manually edit `data/params` and set loginmethod to "DB".

LDAPserver

This parameter should be set to the name (and optionally the port) of your LDAP server. If no port is specified, it assumes the default LDAP port of 389.

Ex. "ldap.company.com" or "ldap.company.com:3268"

LDAPbinddn [Optional]

Some LDAP servers will not allow an anonymous bind to search the directory. If this is the case with your configuration you should set the LDAPbinddn parameter to the user account Bugzilla should use instead of the anonymous bind.

Ex. "cn=default,cn=user:password"

LDAPBaseDN

The LDAPBaseDN parameter should be set to the location in your LDAP tree that you would like to search for email addresses. Your uids should be unique under the DN specified here.

Ex. "ou=People,o=Company"

LDAPuidattribute

The LDAPuidattribute parameter should be set to the attribute which contains the unique UID of your users. The value retrieved from this attribute will be used when attempting to bind as the user to confirm their password.

Ex. “uid”

LDAPmailattribute

The LDAPmailattribute parameter should be the name of the attribute which contains the email address your users will enter into the Bugzilla login boxes.

Ex. “mail”

2.3.6. Prevent users injecting malicious Javascript

It is possible for a Bugzilla user to take advantage of character set encoding ambiguities to inject HTML into Bugzilla comments. This could include malicious scripts. Due to internationalization concerns, we are unable to incorporate by default the code changes suggested by the CERT advisory (http://www.cert.org/tech_tips/malicious_code_mitigation.html#3) on this issue. If your installation is for an English speaking audience only, making the change below will prevent this problem.

Simply locate the following line in `Bugzilla/CGI.pm`:

```
$self->charset(”);
```

and change it to:

```
$self->charset(’ISO-8859-1’);
```

2.3.7. `mod_throttle`

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

2.3.8. TCP/IP Ports

A single-box Bugzilla only requires port 80, plus port 25 if you are using the optional email interface. You should firewall all other ports and/or disable services listening on them.

2.3.9. Daemon Accounts

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

2.4. OS-Specific Installation Notes

Many aspects of the Bugzilla installation can be affected by the the operating system you choose to install it on. Sometimes it can be made easier and others more difficult. This section will attempt to help you understand both the difficulties of running on specific operating systems and the utilities available to make it easier.

If you have anything to add or notes for an operating system not covered, please file a bug in Bugzilla Documentation (http://bugzilla.mozilla.org/enter_bug.cgi?product=Bugzilla&component=Documentation).

2.4.1. Microsoft Windows

Making Bugzilla work on Windows is still a painful processes. The Bugzilla Team is working to make it easier, but that goal is not considered a top priority. If you wish to run Bugzilla, we still recommend doing so on a Unix based system such as GNU/Linux. As of this writing, all members of the Bugzilla team and all known large installations run on Unix based systems.

If after hearing all that, you have enough pain tolerance to attempt installing Bugzilla on Win32, here are some pointers. Because this is a development version of the guide, these instructions are subject to change without notice. In fact, the Bugzilla Team hopes to have Bugzilla reasonably close to "out of the box" compatibility with Windows by the 2.18 release.

2.4.1.1. Win32 Perl

Perl for Windows can be obtained from ActiveState (http://www.activestate.com/). You should be able to find a compiled binary at http://aspn.activestate.com/ASPN/Downloads/ActivePerl/.

2.4.1.2. Perl Modules on Win32

Bugzilla on Windows requires the same perl modules found in Section 2.1.5. The main difference is that windows uses *PPM* instead of CPAN.

```
C:\perl> ppm <module name>
```

> **Note:** The above syntax should work for all modules with the exception of Template Toolkit. The Template Toolkit website (http://tt2.org/download.html#win32) suggests using the instructions on OpenInteract's website (http://openinteract.sourceforge.net/).

2.4.1.3. Code changes required to run on win32

As Bugzilla still doesn't run "out of the box" on Windows, code has to be modified. This section lists the required changes.

2.4.1.3.1. Changes to `checksetup.pl`

In `checksetup.pl`, the line reading:

```
my $mysql_binaries = 'which mysql';
```

to

```
my $mysql_binaries = "D:\\mysql\\bin\\mysql";
```

And you'll also need to change:

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

to

```
my $webservergid = '8'
```

2.4.1.3.2. Changes to `BugMail.pm`

To make bug email work on Win32 (until bug 84876 (http://bugzilla.mozilla.org/show_bug.cgi?id=84876) lands), the simplest way is to have the Net::SMTP Perl module installed and change this:

```
open(SENDMAIL, "|/usr/lib/sendmail $sendmailparam -t -i") ||
  die "Can't open sendmail";

print SENDMAIL trim($msg) . "\n";
close SENDMAIL;
```

to

```
use Net::SMTP;
my $smtp_server = 'smtp.mycompany.com';  # change this

# Use die on error, so that the mail will be in the 'unsent mails' and
# can be sent from the sanity check page.
my $smtp = Net::SMTP->new($smtp_server) ||
  die 'Cannot connect to server \'$smtp_server\";

$smtp->mail('bugzilla-daemon@mycompany.com');  # change this
$smtp->to($person);
$smtp->data();
$smtp->datasend($msg);
```

```
$smtp->dataend();
$smtp->quit;
```

Don't forget to change the name of your SMTP server and the domain of the sending email address (after the '@') in the above lines of code.

2.4.1.4. Serving the web pages

As is the case on Unix based systems, any web server should be able to handle Bugzilla; however, the Bugzilla Team still recommends Apache whenever asked. No matter what web server you choose, be sure to pay attention to the security notes in Section 2.2.4.4. More information on configuring specific web servers can be found in Section 2.2.4.

> **Note:** If using Apache on windows, you can set the ScriptInterpreterSource (http://httpd.apache.org/docs-2.0/mod/core.html#scriptinterpretersource) directive in your Apache config to avoid having to modify the first line of every script to contain your path to perl instead of `/usr/bin/perl`.

2.4.2. Mac OS X

Apple did not include the GD library with Mac OS X. Bugzilla needs this for bug graphs.

You can install it using 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 use it to install the `gd2` package.

It will prompt you for a number of dependencies, type 'y' and hit enter to install all of the dependencies and then watch it work. You will then be able to use *CPAN* to install the GD Perl module.

> **Note:** 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 will be at `/sw/lib` and `/sw/include` instead of `/usr/lib` and `/usr/include`. When the Perl module config script asks where your `libgdi` is, be sure to tell it `/sw/lib`.

Also available via Fink is `expat`. After using fink to install the expat package you will be able to install XML::Parser using CPAN. There is one caveat. Unlike recent versions of the GD module, XML::Parser doesn't prompt for the location of the required libraries. When using CPAN, you will need to use the following command sequence:

```
# perl -MCPAN -e'look XML::Parser'        ❶
# perl Makefile.PL EXPATLIBPATH=/sw/lib EXPATINCPATH=/sw/include
# make; make test; make install           ❷
# exit                                    ❸
```

❶❸ The look command will download the module and spawn a new shell with the extracted files as the current working directory. The exit command will return you to your original shell.

❷ You should watch the output from these make commands, especially “make test” as errors may prevent XML::Parser from functioning correctly with Bugzilla.

2.4.3. Linux-Mandrake 8.0

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

```
bash# urpmi perl-mysql
bash# urpmi perl-chart
bash# urpmi perl-gd
bash# urpmi perl-MailTools             ❶
bash# urpmi apache-modules
```

❶ for Bugzilla email integration

2.5. Troubleshooting

This section gives solutions to common Bugzilla installation problems. If none of the section headings seems to match your problem, read the general advice.

2.5.1. General Advice

If you can’t get `checksetup.pl` to run to completion, it normally explains what’s wrong and how to fix it. If you can’t work it out, or if it’s being uncommunicative, post the errors in the netscape.public.mozilla.webtools (news://news.mozilla.org/netscape.public.mozilla.webtools) newsgroup.

If you have made it all the way through Section 2.1 (Installation) and Section 2.2 (Configuration) but accessing the Bugzilla URL doesn’t work, the first thing to do is to check your webserver error log. For Apache, this is often located at `/etc/logs/httpd/error_log`. The error messages you see may be self-explanatory enough to enable you to diagnose and fix the problem. If not, see below for some commonly-encountered errors. If that doesn’t help, post the errors to the newsgroup.

2.5.2. I installed a Perl module, but `checksetup.pl` claims it’s not installed!

You have two versions of Perl on your machine. You are installing modules into one, and Bugzilla is using the other. Rerun the CPAN commands (or manual compile) using the full path to Perl from the top of `checksetup.pl`. This will make sure you are installing the modules in the right place.

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

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

Certain older versions of the CPAN toolset were somewhat naive about how to upgrade Perl modules. When a couple of modules got rolled into the core Perl distribution for 5.6.1, CPAN thought that the best way to get those modules up to date was to haul down the Perl distribution itself and build it. Needless to say, this has caused headaches for just about everybody. Upgrading to a newer version of CPAN with the commandline above should fix things.

2.5.4. DBD::Sponge::db prepare failed

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

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

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

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

by

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

(note the S added to NAME.)

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

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

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

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

2.5.6. Your vendor has not defined Fcntl macro O_NOINHERIT

This is caused by a bug in the version of File::Temp that is distributed with perl 5.6.0. Many minor variations of this error have been reported:

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

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

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

Numerous people have reported that upgrading to version 5.6.1 or higher solved the problem for them. A less involved fix is to apply the following patch, which is also available as a patch file (../xml/filetemp.patch).

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

Chapter 3. Administering Bugzilla

3.1. Bugzilla Configuration

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

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

   For example, if your Bugzilla query page is `http://www.foo.com/bugzilla/query.cgi`, set your "urlbase" to `http://www.foo.com/bugzilla/`.
3. **makeproductgroups**: This dictates whether or not to automatically create groups when new products are created.
4. **useentrygroupdefault**: Bugzilla products can have a group associated with them, so that certain users can only see bugs in certain products. When this parameter is set to "on", this causes the initial group controls on newly created products to place all newly-created bugs in the group having the same name as the product immediately. After a product is initially created, the group controls can be further adjusted without interference by this mechanism.
5. **shadowdb**: You run into an interesting problem when Bugzilla reaches a high level of continuous activity. MySQL supports only table-level write locking. What this means is that if someone needs to make a change to a bug, they will lock the entire table until the operation is complete. Locking for write also blocks reads until the write is complete. Note that more recent versions of mysql support row level locking using different table types. These types are slower than the standard type, and Bugzilla does not yet take advantage of features such as transactions which would justify this speed decrease. The Bugzilla team are, however, happy to hear about any experiences with row level locking and Bugzilla.

   The "shadowdb" parameter was designed to get around this limitation. While only a single user is allowed to write to a table at a time, reads can continue unimpeded on a read-only shadow copy of the database. Although your database size will double, a shadow database can cause an enormous performance improvement when implemented on extremely high-traffic Bugzilla databases.

   As a guide, on reasonably old hardware, mozilla.org began needing "shadowdb" when they reached around 40,000 Bugzilla users with several hundred Bugzilla bug changes and comments per day.

   The value of the parameter defines the name of the shadow bug database. You will need to set the host and port settings from the params page, and set up replication in your database server so that updates reach this readonly mirror. Consult your database documentation for more detail.
6. **shutdownhtml**: If you need to shut down Bugzilla to perform administration, enter some descriptive HTML here and anyone who tries to use Bugzilla will receive a page to that effect. Obviously, editparams.cgi will still be accessible so you can remove the HTML and re-enable Bugzilla. :-)
7. **passwordmail**: Every time a user creates an account, the text of this parameter (with substitutions) is sent to the new user along with their password message.

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

8. **movebugs**: This option is an undocumented feature to allow moving bugs between separate Bugzilla installations. You will need to understand the source code in order to use this feature. Please consult `movebugs.pl` in your Bugzilla source tree for further documentation, such as it is.
9. **useqacontact**: This allows you to define an email address for each component, in addition to that of the default owner, who will be sent carbon copies of incoming bugs.
10. **usestatuswhiteboard**: This defines whether you wish to have a free-form, overwritable field associated with each bug. The advantage of the Status Whiteboard is that it can be deleted or modified with ease, and provides an easily-searchable field for indexing some bugs that have some trait in common.
11. **whinedays**: Set this to the number of days you want to let bugs go in the NEW or REOPENED state before notifying people they have untouched new bugs. If you do not plan to use this feature, simply do not set up the whining cron job described in the installation instructions, or set this value to "0" (never whine).
12. **commenton***: All these fields allow you to dictate what changes can pass without comment, and which must have a comment from the person who changed them. Often, administrators will allow users to add themselves to the CC list, accept bugs, or change the Status Whiteboard without adding a comment as to their reasons for the change, yet require that most other changes come with an explanation.

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

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

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

3.2. User Administration

3.2.1. Creating the Default User

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

> **Tip:** If you wish to add more administrative users, add them to the "admin" group and, optionally, add edit the tweakparams, editusers, creategroups, editcomponents, and editkeywords groups to add the entire admin group to those groups.

3.2.2. Managing Other Users

3.2.2.1. Creating new users

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

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

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

3.2.2.2. Modifying Users

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

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

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

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

Warning

Don't disable all the administrator accounts!

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

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

3.3. Products

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

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

To create a new product:

1. Select "products" from the footer
2. Select the "Add" link in the bottom right

3. Enter the name of the product and a description. The Description field may contain HTML.

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

3.4. Components

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

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

To create a new Component:

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

3.5. Versions

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

To create and edit Versions:

1. From the "Edit product" screen, select "Edit Versions"
2. You will notice that the product already has the default version "undefined". Click the "Add" link in the bottom right.
3. Enter the name of the Version. This field takes text only. Then click the "Add" button.

3.6. Milestones

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

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

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

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

3.7. Voting

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

To modify Voting settings:

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

3.8. Groups and Group Security

Groups allow the administrator to isolate bugs or products that should only be seen by certain people. The association between products and groups is controlled from the product edit page under "Edit Group Controls."

If the makeproductgroups param is on, a new group will be automatically created for every new product.

On the product edit page, there is a page to edit the "Group Controls" for a product and determine which groups are applicable, default, and mandatory for each product as well as controlling entry for each product and being able to set bugs in a product to be totally read-only unless some group restrictions are met.

For each group, it is possible to specify if membership in that group is...

1. required for bug entry,
2. Not applicable to this product(NA), a possible restriction for a member of the group to place on a bug in this product(Shown), a default restriction for a member of the group to place on a bug in this product(Default), or a mandatory restriction to be placed on bugs in this product(Mandatory).
3. Not applicable by non-members to this product(NA), a possible restriction for a non-member of the group to place on a bug in this product(Shown), a default restriction for a non-member of the group to place on a bug in this product(Default), or a mandatory restriction to be placed on bugs in this product when entered by a non-member(Mandatory).
4. required in order to make *any* change to bugs in this product *including comments.*

To create Groups:

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

> **Warning**
>
> If specifying a domain in the regexp, make sure you end the regexp with a $. Otherwise, when granting access to "@mycompany\.com", you will allow access to ’badperson@mycompany.com.cracker.net’. You need to use ’@mycompany\.com$’ as the regexp.

4. After you add your new group, edit the new group. On the edit page, you can specify other groups that should be included in this group and which groups should be permitted to add and delete users from this group.

Note that group permissions are such that you need to be a member of *all* the groups a bug is in, for whatever reason, to see that bug. Similarly, you must be a member of *all* of the entry groups for a product to add bugs to a product and you must be a member of *all* of the canedit groups for a product in order to make *any* change to bugs in that product.

3.9. Upgrading to New Releases

> **Warning**
>
> Upgrading is a one-way process. You should backup your database and current Bugzilla directory before attempting the upgrade. If you wish to revert to the old Bugzilla version for any reason, you will have to restore from these backups.

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

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

There are also three different methods to upgrade your installation.

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

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

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

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

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

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

Example 3-1. Upgrading using CVS

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

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

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

Caution

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

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

Example 3-2. Upgrading using the tarball

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

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

Warning

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

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

Example 3-3. Upgrading using patches

The Bugzilla team will normally make a patch file available for revisions to go from the most recent revision to the new one. You could also read the release notes and grab the patches attached to the mentioned bug, but it is safer to use the released patch file as sometimes patches get changed before they get checked in. It is also theoretically possible to scour the fixed bug list and pick and choose which patches to apply from a point release, but this is not recommended either as what you'll end up with is a hodge podge Bugzilla that isn't really any version. This would also make it more difficult to upgrade in the future.

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

Caution

If you do this, beware that this doesn't change the entires in your CVS directory so it may make updates using CVS (Example 3-1) more difficult in the future.

Chapter 4. Customising Bugzilla

4.1. Template Customization

Administrators can configure the look and feel of Bugzilla without having to edit Perl files or face the nightmare of massive merge conflicts when they upgrade to a newer version in the future.

Templatization also makes localized versions of Bugzilla possible, for the first time. It's possible to have Bugzilla's UI language determined by the user's browser. More information is available in Section 4.1.5.

4.1.1. What to Edit

The template directory structure is that there's a top level directory, `template`, which contains a directory for each installed localization. The default English templates are therefore in `en`. Underneath that, there is the `default` directory and optionally the `custom` directory. The `default` directory contains all the templates shipped with Bugzilla, whereas the `custom` directory does not exist at first and must be created if you want to use it.

There are two different ways of editing Bugzilla's templates, and which you use depends mainly on the method you plan to use to upgrade Bugzilla. The first method of making customizations is to directly edit the templates in `template/en/default`. This is probably the best method for small changes if you are going to use the CVS method of upgrading, because if you then execute a **cvs update**, any template fixes will get automagically merged into your modified versions.

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

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

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

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

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

4.1.2. How To Edit Templates

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

The syntax of the Template Toolkit language is beyond the scope of this guide. It's reasonably easy to pick up by looking at the current templates; or, you can read the manual, available on the Template Toolkit home page (http://www.template-toolkit.org).

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

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

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

4.1.3. Template Formats

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

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

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

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

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

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

4.1.4. Particular Templates

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

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

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

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

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

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

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

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

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

For example, if your enter_bug template had a field

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

and then your comment.txt.tmpl had

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

then

```
BuildID: 20020303
```

would appear in the initial checkin comment.

4.1.5. Configuring Bugzilla to Detect the User's Language

Bugzilla honours the user's Accept: HTTP header. You can install templates in other languages, and Bugzilla will pick the most appropriate according to a priority order defined by you. Many language templates can be obtained from http://www.bugzilla.org/download.html#localizations. Instructions for submitting new languages are also available from that location.

After untarring the localizations (or creating your own) in the `BUGZILLA_ROOT/template` directory, you must update the `languages` parameter to contain any localizations you'd like to permit. You may also wish to set the `defaultlanguage` parameter to something other than "en" if you don't want Engish to be the default language.

4.2. Template Hooks

Template hooks are a way for extensions to Bugzilla to insert code into the standard Bugzilla templates without modifying the template files themselves. The hooks mechanism defines a consistent API for extending the standard templates in a way that cleanly separates standard code from extension code. Hooks reduce merge conflicts and make it easier to write extensions that work across multiple versions of Bugzilla, making upgrading a Bugzilla installation with installed extensions easier.

A template hook is just a named place in a standard template file where extension template files for that hook get processed. Each hook has a corresponding directory in the Bugzilla directory tree. Hooking an extension template to a hook is as simple as putting the extension file into the hook's directory. When Bugzilla processes the standard template and reaches the hook, it will process all extension templates in the hook's directory. The hooks themselves can be added into any standard template upon request by extension authors.

To use hooks to extend a Bugzilla template, first make sure there is a hook at the appropriate place within the template you want to extend. Hooks appear in the standard Bugzilla templates as a single directive in the format `[% Hook.process("name") %]`, where `name` is the unique (within that template) name of the hook.

If you aren't sure which template you want to extend or just want to browse the available hooks, either use your favorite multi-file search tool (e.g. **grep**) to search the standard templates for occurrences of `Hook.process` or browse the directory tree in `BUGZILLA_ROOT/template/en/extension/hook/`, which contains a directory for each hook in the following location:

`BUGZILLA_ROOT/template/en/extension/hook/PATH_TO_STANDARD_TEMPLATE/STANDARD_TEMPLATE_NAME/HOOK_NAME/`

If there is no hook at the appropriate place within the Bugzilla template you want to extend, file a bug requesting one (http://bugzilla.mozilla.org/enter_bug.cgi?product=Bugzilla&component=User%20Interface), specifying:

the template for which you are requesting a hook;
where in the template you would like the hook to be placed (line number/position for latest version of template in CVS or description
the purpose of the hook;
a link to information about your extension, if any.

The Bugzilla reviewers will promptly review each hook request, name the hook, add it to the template, check the new version of the template into CVS, and create the corresponding directory in `BUGZILLA_ROOT/template/en/extension/hook/`.

You may optionally attach a patch to the bug which implements the hook and check it in yourself after receiving approval from a Bugzilla reviewer. The developers may suggest changes to the location of the hook based on their analysis of your needs or so the hook can satisfy the needs of multiple extensions, but the process of getting hooks approved and checked in is not as stringent as the process for general changes to Bugzilla, and any extension, whether released or still in development, can have hooks added to meet their needs.

After making sure the hook you need exists (or getting it added if not), add your extension template to the directory within the Bugzilla directory tree corresponding to the hook.

That's it! Now, when the standard template containing the hook is processed, your extension template will be

processed at the point where the hook appears.

For example, let's say you have an extension named Projman that adds project management capabilities to Bugzilla. Projman has an administration interface `edit-projects.cgi`, and you want to add a link to it into the navigation bar at the bottom of every Bugzilla page for those users who are authorized to administer projects.

The navigation bar is generated by the template file `useful-links.html.tmpl`, which is located in the `global/` subdirectory on the standard Bugzilla template path `BUGZILLA_ROOT/template/en/default/`. Looking in `useful-links.html.tmpl`, you find the following hook at the end of the list of standard Bugzilla administration links:

```
...
    [% ', <a href="editkeywords.cgi">keywords</a>'
                                              IF user.groups.editkeywords %]
    [% Hook.process("edit") %]
...
```

The corresponding directory for this hook is `BUGZILLA_ROOT/template/en/extension/hook/global/useful-links.html.tmpl/edit/`.

You put a template named `projman-edit-projects.html.tmpl` into that directory with the following content:

```
...[% ', <a href="edit-projects.cgi">projects</a>' IF user.groups.projman_admins %]
```

Voila! The link now appears after the other administration links in the navigation bar for users in the `projman_admins` group.

Notes:

- You may want to prefix your extension template names with the name of your extension, e.g. `projman-foo.html.tmpl`, so they do not conflict with the names of templates installed by other extensions.
- If your extension includes entirely new templates in addition to extensions of standard templates, it should install those new templates into an extension-specific subdirectory of the `BUGZILLA_ROOT/template/en/extension/` directory. The `extension/` directory, like the `default/` and `custom/` directories, is part of the template search path, so putting templates there enables them to be found by the template processor.

  The template processor looks for templates first in the `custom/` directory (i.e. templates added by the specific installation), then in the `extension/` directory (i.e. templates added by extensions), and finally in the `default/` directory (i.e. the standard Bugzilla templates). Thus extension templates can override standard templates, but installation-specific templates override both.

  Note that overriding standard templates with extension templates gives you great power but also makes upgrading an installation harder. As with custom templates, we recommend using this functionality sparingly and only when absolutely necessary.

- Installation customizers can also take advantage of hooks when adding code to a Bugzilla template. To do so, create directories in `BUGZILLA_ROOT/template/en/custom/hook/` equivalent to the directories in `BUGZILLA_ROOT/template/en/extension/hook/` for the hooks you want to use, then place your customization templates into those directories.

Obviously this method of customizing Bugzilla only lets you add code to the standard templates; you cannot change the existing code. Nevertheless, for those customizations that only add code, this method can reduce conflicts when merging changes, making upgrading your customized Bugzilla installation easier.

4.3. Customizing Who Can Change What

> **Warning**
>
> This feature should be considered experimental; the Bugzilla code you will be changing is not stable, and could change or move between versions. Be aware that if you make modifications as outlined here, you may have to re-make them or port them if Bugzilla changes internally between versions, and you upgrade.

Companies often have rules about which employees, or classes of employees, are allowed to change certain things in the bug system. For example, only the bug's designated QA Contact may be allowed to VERIFY the bug. Bugzilla has been designed to make it easy for you to write your own custom rules to define who is allowed to make what sorts of value transition.

For maximum flexibility, customizing this means editing Bugzilla's Perl code. This gives the administrator complete control over exactly who is allowed to do what. The relevant function is called `CheckCanChangeField()`, and is found in `process_bug.cgi` in your Bugzilla directory. If you open that file and search for "sub CheckCanChangeField", you'll find it.

This function has been carefully commented to allow you to see exactly how it works, and give you an idea of how to make changes to it. Certain marked sections should not be changed - these are the "plumbing" which makes the rest of the function work. In between those sections, you'll find snippets of code like:

```
# Allow the owner to change anything.
if ($ownerid eq $whoid) {
    return 1;
}
```

It's fairly obvious what this piece of code does.

So, how does one go about changing this function? Well, simple changes can be made just be removing pieces - for example, if you wanted to prevent any user adding a comment to a bug, just remove the lines marked "Allow anyone to change comments." And if you want the reporter to have no special rights on bugs they have filed, just remove the entire section which refers to him.

More complex customizations are not much harder. Basically, you add a check in the right place in the function, i.e. after all the variables you are using have been set up. So, don't look at $ownerid before $ownerid has been obtained from the database. You can either add a positive check, which returns 1 (allow) if certain conditions are true, or a negative check, which returns 0 (deny.) E.g.:

```
if ($field eq "qacontact") {
    if (Bugzilla->user->groups("quality_assurance")) {
        return 1;
    }
    else {
```

```
            return 0;
        }
    }
```

This says that only users in the group "quality_assurance" can change the QA Contact field of a bug. Getting more weird:

```
    if (($field eq "priority") &&
        (Bugzilla->user->email =~ /.*\@example\.com$/))
    {
        if ($oldvalue eq "P1") {
            return 1;
        }
        else {
            return 0;
        }
    }
```

This says that if the user is trying to change the priority field, and their email address is @example.com, they can only do so if the old value of the field was "P1". Not very useful, but illustrative.

For a list of possible field names, look in `data/versioncache` for the list called `@::log_columns`. If you need help writing custom rules for your organization, ask in the newsgroup.

4.4. 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.

`versioncache` 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.

4.5. 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!

4.5.1. 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 (http://www.mysql.com/documentation/) . 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 (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;**

4.5.1.1. 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 the names of all the "spreadsheets" (tables) in your database.

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 |
| 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.

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 | UNCONF
```

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?

4.6. Integrating Bugzilla with Third-Party Tools

4.6.1. Bonsai

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

4.6.2. CVS

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

Follow the instructions in this Guide for enabling Bugzilla e-mail integration. Ensure that your check-in script sends an email to your Bugzilla e-mail gateway with the subject of “[Bug XXXX]”, and you can have CVS check-in comments append to your Bugzilla bug. If you want to have the bug be closed automatically, you’ll have to modify the `contrib/bugzilla_email_append.pl` script.

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

4.6.3. 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.

4.6.4. Tinderbox/Tinderbox2

Tinderbox is a continuous-build system which can integrate with Bugzilla - see http://www.mozilla.org/projects/tinderbox for details of Tinderbox, and http://tinderbox.mozilla.org/showbuilds.cgi to see it in action.

Chapter 5. Using Bugzilla

5.1. Introduction

This section contains information for end-users of Bugzilla. There is a Bugzilla test installation, called Landfill (http://landfill.bugzilla.org/bugzilla-tip/), which you are welcome to play with (if it's up.) However, it does not necessarily have all Bugzilla features enabled, and runs an up-to-the-minute version, so some things may not quite work as this document describes.

5.2. Create a Bugzilla Account

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

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

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

5.3. Anatomy of a Bug

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

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

   Administration: Administration of a Bugzilla installation.
   Bugzilla-General: Anything that doesn't fit in the other components, or spans multiple components.
   Creating/Changing Bugs: Creating, changing, and viewing bugs.
   Documentation: The Bugzilla documentation, including The Bugzilla Guide.
   Email: Anything to do with email sent by Bugzilla.
   Installation: The installation process of Bugzilla.
   Query/Buglist: Anything to do with searching for bugs and viewing the buglists.

Reporting/Charting: Getting reports from Bugzilla.
User Accounts: Anything about managing a user account from the user's perspective. Saved queries, creating accounts, changing
User Interface: General issues having to do with the user interface cosmetics (not functionality) including cosmetic issues, HTM

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

5.4. Searching for Bugs

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

The Search page has controls for selecting different possible values for all of the fields in a bug, as described above. For some fields, multiple values can be selected. In those cases, Bugzilla returns bugs where the content of the field matches any one of the selected values. If none is selected, then the field can take any value.

Once you've run a search, you can save it as a Saved Search, which appears in the page footer.

Highly advanced querying is done using Boolean Charts. See the Boolean Charts help link on the Search page for more information.

5.5. Bug Lists

If you run a search, a list of matching bugs will be returned.

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

Long Format: this gives you a large page with a non-editable summary of the fields of each bug.
CSV: get the buglist as comma-separated values, for import into e.g. a spreadsheet.
Change Columns: change the bug attributes which appear in the list.
Change several bugs at once: If your account is sufficiently empowered, you can make the same change to all the bugs in the list - for
Send mail to bug owners: Sends mail to the owners of all bugs on the list.
Edit Search: If you didn't get exactly the results you were looking for, you can return to the Query page through this link and make sm
Remember Search As: You can give a search a name and remember it; a link will appear in your page footer giving you quick access t

5.6. Filing Bugs

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

The procedure for filing a test bug is as follows:

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

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

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

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

5.7. Patch Viewer

Viewing and reviewing patches in Bugzilla is often difficult due to lack of context, improper format and the inherent readability issues that raw patches present. Patch Viewer is an enhancement to Bugzilla designed to fix that by offering increased context, linking to sections, and integrating with Bonsai, LXR and CVS.

Patch viewer allows you to:

View patches in color, with side-by-side view rather than trying to interpret the contents of the patch.
See the difference between two patches.
Get more context in a patch.
Collapse and expand sections of a patch for easy reading.
Link to a particular section of a patch for discussion or review
Go to Bonsai or LXR to see more context, blame, and cross-references for the part of the patch you are looking at
Create a rawtext unified format diff out of any patch, no matter what format it came from

5.7.1. Viewing Patches in Patch Viewer

The main way to view a patch in patch viewer is to click on the "Diff" link next to a patch in the Attachments list on a bug. You may also do this within the edit window by clicking the "View Attachment As Diff" button in the Edit Attachment screen.

5.7.2. Seeing the Difference Between Two Patches

To see the difference between two patches, you must first view the newer patch in Patch Viewer. Then select the older patch from the dropdown at the top of the page ("Differences between [dropdown] and this patch") and click the "Diff" button. This will show you what is new or changed in the newer patch.

5.7.3. Getting More Context in a Patch

To get more context in a patch, you put a number in the textbox at the top of Patch Viewer ("Patch / File / [textbox]") and hit enter. This will give you that many lines of context before and after each change. Alternatively, you can click on the "File" link there and it will show each change in the full context of the file. This feature only works against files that were diffed using "cvs diff".

5.7.4. Collapsing and Expanding Sections of a Patch

To view only a certain set of files in a patch (for example, if a patch is absolutely huge and you want to only review part of it at a time), you can click the "(+)" and "(-)" links next to each file (to expand it or collapse it). If you want to collapse all files or expand all files, you can click the "Collapse All" and "Expand All" links at the top of the page.

5.7.5. Linking to a Section of a Patch

To link to a section of a patch (for example, if you want to be able to give someone a URL to show them which part you are talking about) you simply click the "Link Here" link on the section header. The resulting URL can be copied

and used in discussion. (Copy Link Location in Mozilla works as well.)

5.7.6. Going to Bonsai and LXR

To go to Bonsai to get blame for the lines you are interested in, you can click the "Lines XX-YY" link on the section header you are interested in. This works even if the patch is against an old version of the file, since Bonsai stores all versions of the file.

To go to LXR, you click on the filename on the file header (unfortunately, since LXR only does the most recent version, line numbers are likely to rot).

5.7.7. Creating a Unified Diff

If the patch is not in a format that you like, you can turn it into a unified diff format by clicking the "Raw Unified" link at the top of the page.

5.8. Hints and Tips

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

5.8.1. Autolinkification

Bugzilla comments are plain text - so typing <U> will produce less-than, U, greater-than rather than underlined text. However, Bugzilla will automatically make hyperlinks out of certain sorts of text in comments. For example, the text "http://www.bugzilla.org" will be turned into a link: http://www.bugzilla.org. Other strings which get linkified in the obvious manner are:

bug 12345
comment 7
bug 23456, comment 53
attachment 4321
mailto:george@example.com
george@example.com
ftp://ftp.mozilla.org
Most other sorts of URL

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

5.8.2. Quicksearch

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

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

5.8.3. Comments

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

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

5.8.4. Attachments

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

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

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

Bugzilla stores and uses a Content-Type for each attachment (e.g. text/html). To download an attachment as a different Content-Type (e.g. application/xhtml+xml), you can override this using a 'content-type' parameter on the URL, e.g. `&content-type=text/plain`.

5.9. User Preferences

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

5.9.1. Account Settings

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

5.9.2. Email Settings

On this tab you can reduce or increase the amount of email sent you from Bugzilla, opting in our out depending on your relationship to the bug and the change that was made to it.

You can also do further filtering on the client side by using the X-Bugzilla-Reason mail header which Bugzilla adds to all bugmail. This tells you what relationship you have to the bug in question, and can be any of Owner, Reporter, QAcontact, CClist, Voter and WatchingComponent.

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

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

5.9.3. Permissions

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

5.10. Reports

To be written

Appendix A. The Bugzilla FAQ

This FAQ includes questions not covered elsewhere in the Guide.

1. General Questions

1.1. What license is Bugzilla distributed under?

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

1.2. How do I get commercial support for Bugzilla?

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

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

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

There are *dozens* of major companies with public Bugzilla sites to track bugs in their products. We have a fairly complete list available on our website at http://bugzilla.org/installation-list/. If you have an installation of Bugzilla and would like to be added to the list, whether it's a public install or not, simply e-mail Gerv <gerv@mozilla.org>.

1.4. Who maintains Bugzilla?

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

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

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

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

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

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

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

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

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

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

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

1.8. What is `/usr/bonsaitools/bin/perl`?

Bugzilla used to have the path to perl on the shebang line set to `/usr/bonsaitools/bin/perl` because when Terry first started writing the code for mozilla.org he needed a version of Perl and other tools that were completely under his control. This location was abandoned for the 2.18 release in favor of the more sensible `/usr/bin/perl`. If you installed an older verion of Bugzilla and created the symlink we suggested, you can remove it now (provided that you don't have anything else, such as Bonsai, using it and you don't intend to reinstall an older version of Bugzilla).

1.9. My perl is not located at `/usr/bin/perl`, is there an easy way to change it everywhere it needs to be changed?

Yes, the following bit of perl magic will change all the shebang lines. Be sure to change `/usr/local/bin/perl` to your path to the perl binary.

```
perl -pi -e 's@#\!/usr/bin/perl@#\!/usr/local/bin/perl@' *cgi *pl
```

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

At present, no.

1.11. Does bugzilla run under `mod_perl`?

At present, no. This is being worked on.

2. Managerial Questions

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

It is web and e-mail based.

2.2. 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 in bug 91037 (http://bugzilla.mozilla.org/show_bug.cgi?id=91037)

2.3. 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/report.cgi for samples of what Bugzilla can do in reporting and graphing.

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

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

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

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

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

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

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

Bugzilla can output buglists as HTML (the default), CSV or RDF. The link for CSV can be found at the bottom of the buglist in HTML format. This CSV format can easily be imported into MS Excel or other spreadsheet applications.

To use the RDF format of the buglist it is necessary to append a `&ctype=rdf` to the URL. RDF is meant to be machine readable and thus it is assumed that the URL would be generated programatically so there is no user visible link to this format.

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

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

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

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

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

Yes. No. Yes (using the CSV format).

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

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

2.10. 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.

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

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

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

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

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

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

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

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

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

3. Bugzilla Security

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

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

3.2. Are there any security problems with Bugzilla?

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

4. Bugzilla Email

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

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

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

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

4.3. I want whineatnews.pl to whine at something other than new and reopened bugs. How do I do it?

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

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

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

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

If you are using sendmail, try enabling `sendmailnow` in `editparams.cgi`.

If you are using an alternate *MTA*, make sure the options given in `Bugzilla/BugMail.pm` and any other place where sendmail is called from are correct for your MTA. You should also ensure that the `sendmailnow` param is set to `on`.

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

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

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

5. Bugzilla Database

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

Red Hat's old version of Bugzilla (based on 2.8) worked on Oracle, but it is now so old as to be obsolete, and is totally unsupported. Red Hat's newer version (based on 2.17.1 and soon to be merged into the main distribution) runs on PostgreSQL. At this time we know of no recent ports of Bugzilla to Oracle; to be honest, Bugzilla doesn't need what Oracle offers.

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

Run the "sanity check" utility (`sanitycheck.cgi`) from your web browser to see! If it finishes without errors, you're *probably* OK. If it doesn't come back OK (i.e. any red letters), there are certain things Bugzilla can recover from and certain things it can't. If it can't auto-recover, I hope you're familiar with mysqladmin commands or have installed another way to manage your database. Sanity Check, although it is a good basic check on your database integrity, by no means is a substitute for competent database administration and avoiding deletion of data. It is not exhaustive, and was created to do a basic check for the most common problems in Bugzilla databases.

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

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

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

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

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

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

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

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

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

6. Bugzilla and Win32

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

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

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

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

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

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

Microsoft has some advice on this matter, as well:

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

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

Your modules may be outdated or inaccurate. Try:

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

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

7. Bugzilla Usage

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8. Bugzilla Hacking

8.1. What kind of style should I use for templatization?

Gerv and Myk suggest a 2-space indent, with embedded code sections on their own line, in line with outer tags. Like this:

```
<fred>
[% IF foo %]
  <bar>
  [% FOREACH x = barney %]
    <tr>
      <td>
        [% x %]
      </td>
    <tr>
  [% END %]
[% END %]
</fred>
```

Myk also recommends you turn on PRE_CHOMP in the template initialization to prevent bloating of HTML with unnecessary whitespace.

Please note that many have differing opinions on this subject, and the existing templates in Bugzilla espouse both this and a 4-space style. Either is acceptable; the above is preferred.

8.2. What bugs are in Bugzilla right now?

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

You can view bugs marked for 2.18 release here (http://bugzilla.mozilla.org/buglist.cgi?product=Bugzilla&target_milestone=Bugzilla+2.18). This list includes bugs for the 2.18 release that have already been fixed and checked into CVS. Please consult the Bugzilla Project Page (http://www.bugzilla.org/) for details on how to check current sources out of CVS so you can have these bug fixes early!

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

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

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

Appendix B. Contrib

There are a number of unofficial Bugzilla add-ons in the `$BUGZILLA_ROOT/contrib/` directory. This section documents them.

B.1. Command-line Search Interface

There are a suite of Unix utilities for searching Bugzilla from the command line. They live in the `contrib/cmdline` directory. However, they have not yet been updated to work with 2.16 (post-templatisation.). There are three files - `query.conf`, `buglist` and `bugs`.

`query.conf` 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 column list 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, grep for COLUMNLIST in your cookies file 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 Peck says she has good results piping `buglist` output through **w3m -T text/html -dump**

Appendix C. Manual Installation of Perl Modules

C.1. Instructions

If you need to install Perl modules manually, here's how it's done. Download the module using the link given in the next section, and then apply this magic incantation, as root:

```
bash# tar -xzvf <module>.tar.gz
bash# cd <module>
bash# perl Makefile.PL
bash# make
bash# make test
bash# make install
```

C.2. Download Locations

Note: some modules are in the core distribution of ActiveState Perl for Windows. Others are not available. No PPM links have been provided in either of these two cases.

CGI:

CPAN Download Page: http://search.cpan.org/dist/CGI.pm/
PPM Download Link: http://ppm.activestate.com/PPMPackages/zips/6xx-builds-only/CGI.zip
Documentation: http://www.perldoc.com/perl5.8.0/lib/CGI.html

TimeDate:

CPAN Download Page: http://search.cpan.org/dist/TimeDate/
PPM Download Link: http://ppm.activestate.com/PPMPackages/zips/6xx-builds-only/TimeDate.zip
Documentation: http://search.cpan.org/dist/TimeDate/lib/Date/Format.pm

DBI:

CPAN Download Page: http://search.cpan.org/dist/DBI/
PPM Download Link: http://ppm.activestate.com/PPMPackages/zips/6xx-builds-only/DBI.zip
Documentation: http://dbi.perl.org/docs/

DBD::mysql:

CPAN Download Page: http://search.cpan.org/dist/DBD-mysql/
PPM Download Link: http://ppm.activestate.com/PPMPackages/zips/6xx-builds-only/DBD-Mysql.zip
Documentation: http://search.cpan.org/dist/DBD-mysql/lib/DBD/mysql.pm

File::Spec:

CPAN Download Page: http://search.cpan.org/dist/File-Spec/
PPM Download Page: http://ppm.activestate.com/PPMPackages/zips/6xx-builds-only/File-Spec.zip
Documentation: http://www.perldoc.com/perl5.8.0/lib/File/Spec.html

File::Temp:

CPAN Download Page: http://search.cpan.org/dist/File-Temp/
Documentation: http://www.perldoc.com/perl5.8.0/lib/File/Temp.html

Template Toolkit:

CPAN Download Page: http://search.cpan.org/dist/Template-Toolkit/
PPM Download Link: http://openinteract.sourceforge.net/ppmpackages/5.6/Template-Toolkit.tar.gz
Documentation: http://www.template-toolkit.org/docs.html

Text::Wrap:

CPAN Download Page: http://search.cpan.org/dist/Text-Tabs+Wrap/
Documentation: http://www.perldoc.com/perl5.8.0/lib/Text/Wrap.html

GD:

CPAN Download Page: http://search.cpan.org/dist/GD/
PPM Download Link: http://ppm.activestate.com/PPMPackages/zips/6xx-builds-only/GD.zip
Documentation: http://stein.cshl.org/WWW/software/GD/

Chart::Base:

CPAN Download Page: http://search.cpan.org/dist/Chart/

GD::Graph:

CPAN Download Page: http://search.cpan.org/dist/GDGraph/
PPM Download Link: http://ppm.activestate.com/PPMPackages/zips/6xx-builds-only/GDGraph.zip
Documentation: http://search.cpan.org/dist/GDGraph/Graph.pm

GD::Text::Align:

CPAN Download Page: http://search.cpan.org/dist/GDTextUtil/
PPM Download Page: http://ppm.activestate.com/PPMPackages/zips/6xx-builds-only/GDTextUtil.zip
Documentation: http://search.cpan.org/dist/GDTextUtil/Text/Align.pm

MIME::Parser:

CPAN Download Page: http://search.cpan.org/dist/MIME-tools/
PPM Download Link: http://ppm.activestate.com/PPMPackages/zips/6xx-builds-only/MIME-tools.zip
Documentation: http://search.cpan.org/dist/MIME-tools/lib/MIME/Parser.pm

XML::Parser:

CPAN Download Page: http://search.cpan.org/dist/XML-Parser/
Documentation: http://www.perldoc.com/perl5.6.1/lib/XML/Parser.html

PatchReader:

CPAN Download Page: http://search.cpan.org/author/JKEISER/PatchReader/
Documentation: http://www.johnkeiser.com/mozilla/Patch_Viewer.html

Appendix D. 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 Definition

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` to restrict access to certain files. In Bugzilla, they are used to keep secret files which would otherwise compromise your installation - e.g. the `localconfig` file contains the password to your database. 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.

Useful Directives when configuring Bugzilla

`AddHandler (http://httpd.apache.org/docs-2.0/mod/core.html#addhandler)`

Tell Apache that it’s OK to run CGI scripts.

`AllowOverride (http://httpd.apache.org/docs-2.0/mod/core.html#allowoverride)`
`Options (http://httpd.apache.org/docs-2.0/mod/core.html#options)`

These directives are used to tell Apache many things about the directory they apply to. For Bugzilla’s purposes, we need them to allow script execution and `.htaccess` overrides.

`DirectoryIndex (http://httpd.apache.org/docs-2.0/mod/mod_dir.html#directoryindex)`

Used to tell Apache what files are indexes. If you can not add `index.cgi` to the list of valid files, you’ll need to set `$index_html` to 1 in `localconfig` so **./checksetup.pl** will create an `index.html` that redirects to `index.cgi`.

`ScriptInterpreterSource`
`(http://httpd.apache.org/docs-2.0/mod/core.html#scriptinterpretersource)`

Used when running Apache on windows so the shebang line doesn’t have to be changed in every Bugzilla script.

For more information about how to configure Apache for Bugzilla, see Section 2.2.4.1.

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.

Bugzilla

Bugzilla is the world-leading free software bug tracking system.

C

Common Gateway Interface

CGI is an acronym for Common Gateway Interface. This is a standard for interfacing an external application with a web server. Bugzilla is an example of a CGI application.

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).

Comprehensive Perl Archive Network

CPAN stands for the “Comprehensive Perl Archive Network”. CPAN maintains a large number of extremely useful *Perl* modules - encapsulated chunks of code for performing a particular task.

`contrib`

The `contrib` directory is a location to put scripts that have been contributed to Bugzilla but are not a part of the official distribution. These scripts are written by third parties and may be in languages other than perl. For those that are in perl, there may be additional modules or other requirements than those of the offical distribution.

> **Note:** Scripts in the `contrib` directory are not offically supported by the Bugzilla team and may break in between versions.

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.

G

Groups

The word “Groups” has a very special meaning to Bugzilla. Bugzilla’s main security mechanism comes by placing users in groups, and assigning those groups certain privileges to view bugs in particular *Products* in the *Bugzilla* database.

J

JavaScript

JavaScript is cool, we should talk about it.

M

Message Transport Agent

A Message Transport Agent is used to control the flow of email on a system. Many unix based systems use sendmail (http://www.sendmail.org) which is what Bugzilla expects to find by default at `/usr/sbin/sendmail`. Many other MTA's will work, but they all require that the `sendmailnow` param be set to `on`.

MySQL

MySQL is currently the required *RDBMS* for Bugzilla. MySQL can be downloaded from http://www.mysql.com. While you should familiarize yourself with all of the documentation, some high points are:

Backup (http://www.mysql.com/doc/en/Backup.html)

Methods for backing up your Bugzilla database.

Option Files (http://www.mysql.com/doc/en/Option_files.html)

Information about how to configure MySQL using `my.cnf`.

Privilege System (http://www.mysql.com/doc/en/Privilege_system.html)

Much more detailed information about the suggestions in Section 2.2.2.1.

P

Perl Package Manager

http://aspn.activestate.com/ASPN/Downloads/ActivePerl/PPM/

Product

A Product is a broad category of types of bugs, normally representing a single piece of software or entity. In general, there are several Components to a Product. A Product may define a group (used for security) for all bugs entered into its Components.

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

Relational DataBase Managment System

A relational database management system is a database system that stores information in tables that are related to each other.

Regular Expression

A regular expression is an expression used for pattern matching. Documentation (http://perldoc.com/perl5.6/pod/perlre.html#Regular-Expressions)

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.

Tool Command Language

TCL is an open source scripting language available for Windows, Macintosh, and Unix based systems. Bugzilla 1.0 was written in TCL but never released. The first release of Bugzilla was 2.0, which was when it was ported to perl.

Z

Zarro Boogs Found

This is just a goofy way of saying that there were no bugs found matching your query. When asked to explain this message, Terry had the following to say:

> I’ve been asked to explain this ... way back when, when Netscape released version 4.0 of its browser, we had a release party. Naturally, there had been a big push to try and fix every known bug before the release. Naturally, that hadn’t actually happened. (This is not unique to Netscape or to 4.0; the same thing has happened with every software project I’ve ever seen.) Anyway, at the release party, T-shirts were handed out that said something like "Netscape 4.0: Zarro Boogs". Just like the software, the T-shirt had no known bugs. Uh-huh.
>
> So, when you query for a list of bugs, and it gets no results, you can think of this as a friendly reminder. Of *course* there are bugs matching your query, they just aren’t in the bugsystem yet...
>
> —Terry Weissman