
V Skills
Skills for a secure future
Certified Bugzilla Testing Professional
Sample Material
V-Skills Certifications
A Government of India
&
Government of NCT Delhi Initiative
V-Skills
Skills for a secure future

1. INSTALLING BUGZILLA

1.1. Installation

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 from your web browser.

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

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.

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:

- ✓ Install Perl (5.8.1 or above)
- ✓ Install a Database Engine
- ✓ Install a Webserver
- ✓ Install Bugzilla
- ✓ Install Perl modules
- ✓ Install a Mail Transfer Agent (Sendmail 8.7 or above, or an MTA that is Sendmail-compatible with at least this version)
- ✓ Configure all of the above.

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.org. Although Bugzilla runs with Perl 5.8.1, it's a good idea to be using the latest stable version.

Database Engine

Bugzilla supports MySQL, PostgreSQL and Oracle as database servers. You only require one of these systems to make use of Bugzilla.

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 5.0.15 or higher.

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 (RPM Package Manager), .deb (Debian Package), .exe (Windows Executable), or .msi (Windows Installer), make sure the MySQL server is started when the machine boots.

PostgreSQL

Installed Version Test:
psql –V

If you don't have it and your OS doesn't provide official packages, visit http://www.postgresql.org/. You need PostgreSQL version 8.03.0000 or higher.

If you install from something other than a packaging/installation system, such as .rpm (RPM Package Manager), .deb (Debian Package), .exe (Windows Executable), or .msi (Windows Installer), make sure the PostgreSQL server is started when the machine boots.

Oracle

Installed Version Test:
Select * from v$version (you first have to log in into your DB)

If you don't have it and your OS doesn't provide official packages, visit http://www.oracle.com/. You need Oracle version 10.02.0 or higher.

If you install from something other than a packaging/installation system, such as .rpm (RPM Package Manager), .deb (Debian Package), .exe (Windows Executable), or .msi (Windows Installer), make sure the Oracle server is started when the machine boots.

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 web server, please share your experiences with us by filing a bug in Bugzilla Documentation.

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

Bugzilla

Download a Bugzilla tarball (or check it out from Bzr) and place it in a suitable directory, accessible by the default web server user (probably "apache" or "www"). Good locations are either directly in the web server's document directories or in /usr/local with a symbolic link to the web server's document directories or an alias in the web server's configuration.

The default Bugzilla distribution is NOT designed to be placed in a cgi-bin directory. This includes any directory which is configured using the Script Alias directive of Apache.

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

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.

At this point, you need to su to root. You should remain as root until the end of the install. To check you have the required modules, run:

```
bash# ./checksetup.pl --check-modules
```

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.

The preferred way to install missing Perl modules is to use the package manager provided by your operating system (e.g "rpm" or "yum" on Linux distros, or "ppm" on Windows if using ActivePerl). If some Perl modules are still missing or are too old, then we recommend using the install-module.pl script (doesn't work with ActivePerl on Windows). If for some reason you really need to install the Perl modules manually, see Appendix C, Manual Installation of Perl Modules. For instance, on Unix, you invoke install-module.pl as follows:

```
bash# perl install-module.pl <modulename>
```

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.

If you are using a package-based system, and attempting to install the Perl modules from CPAN, you may need to install the "development" packages for MySQL and GD before attempting to

install the related Perl modules. The names of these packages will vary depending on the specific distribution you are using, but are often called <packagename>-devel.
Here is a complete list of modules and their minimum versions. Some modules have special installation notes, which follow.

Required Perl modules:
- ✓ CGI (3.51)
- ✓ Date::Format (2.23)
- ✓ DateTime (0.28)
- ✓ DateTime::TimeZone (0.71)
- ✓ DBI (1.54)
- ✓ DBD::mysql (4.001) if using MySQL
- ✓ DBD::Pg (2.7.0) if using PostgreSQL
- ✓ DBD::Oracle (1.19) if using Oracle
- ✓ Digest::SHA (any)
- ✓ Email::Send (2.04)
- ✓ Email::MIME (1.904)
- ✓ Template (2.22)
- ✓ URI (1.37)

Optional Perl modules:
- ✓ GD (1.20) for bug charting
- ✓ Template::Plug-in::GD::Image (any) for Graphical Reports
- ✓ Chart::Lines (2.1.0) for bug charting
- ✓ GD::Graph (any) for bug charting
- ✓ GD::Text (any) for bug charting
- ✓ XML::Twig (any) for bug import/export
- ✓ MIME::Parser (5.406) for bug import/export
- ✓ LWP::User Agent (any) for Automatic Update Notifications
- ✓ Patch Reader (0.9.6) for pretty HTML view of patches
- ✓ Net::LDAP (any) for LDAP Authentication
- ✓ Authen::SASL (any) for SASL Authentication
- ✓ Authen::Radius (any) for RADIUS Authentication
- ✓ SOAP::Lite (0.712) for the web service interface
- ✓ JSON::RPC (any) for the JSON-RPC interface
- ✓ Test::Taint (any) for the web service interface
- ✓ HTML::Parser (3.40) for More HTML in Product/Group Descriptions
- ✓ HTML::Scrubber (any) for More HTML in Product/Group Descriptions
- ✓ Email::Reply (any) for Inbound Email
- ✓ The Schwartz (1.07) for Mail Queueing
- ✓ Daemon::Generic (any) for Mail Queueing
- ✓ mod_perl2 (1.999022) for mod_perl

Mail Transfer Agent (MTA)

Bugzilla is dependent on the availability of an e-mail system for its user authentication and for other tasks.

This is not entirely true. It is possible to completely disable email sending, or to have Bugzilla store email messages in a file instead of sending them. However, this is mainly intended for testing, as disabling or diverting email on a production machine would mean that users could miss important events (such as bug changes or the creation of new accounts).

On Linux, any Sendmail-compatible MTA (Mail Transfer Agent) will suffice. Sendmail, Postfix, qmail and Exim are examples of common MTAs. Sendmail is the original Unix MTA, but the others are easier to configure, and therefore many people replace Sendmail with Postfix or Exim. They are drop-in replacements, so Bugzilla will not distinguish between them.

If you are using Sendmail, version 8.7 or higher is required. If you are using a Sendmail-compatible MTA, it must be congruent with at least version 8.7 of Sendmail.

Consult the manual for the specific MTA you choose for detailed installation instructions. Each of these programs will have their own configuration files where you must configure certain parameters to ensure that the mail is delivered properly. They are implemented as services, and you should ensure that the MTA is in the auto-start list of services for the machine.

If a simple mail sent with the command-line 'mail' program succeeds, then Bugzilla should also be fine.

Installing Bugzilla on mod_perl

It is now possible to run the Bugzilla software under mod_perl on Apache. mod_perl has some additional requirements to that of running Bugzilla under mod_cgi (the standard and previous way).

Bugzilla requires mod_perl to be installed, which can be obtained from http://perl.apache.org - Bugzilla requires version 1.999022 (AKA 2.0.0-RC5) to be installed.

1.2. Configuration

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. Be certain to read Chapter 4, Bugzilla Security for some important security tips.

localconfig

You should now run checksetup.pl again, this time without the --check-modules switch.

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

This time, checksetup.pl should tell you that all the correct modules are installed and will display 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 two values you need to change are $db_driver and $db_pass, respectively the type of the database and 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. $db_driver can be either 'mysql', 'Pg', 'Oracle' or 'Sqlite'.

In Oracle, $db_name should actually be the SID name of your database (e.g. "XE" if you are using Oracle XE).

You may need to change the value of webservergroup if your web server does not run in the "apache" group. On Debian, for example, Apache runs in the "www-data" group. If you are going to run Bugzilla on a machine where you do not have root access (such as on a shared web hosting account), you will need to leave webservergroup empty, ignoring the warnings that checksetup.pl will subsequently display every time it is run.

If you are using suexec, you should use your own primary group for webservergroup rather than leaving it empty.

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

Database Server

This section deals with configuring your database server for use with Bugzilla. Currently, MySQL, PostgreSQL, Oracle and SQLite are available.

Bugzilla Database Schema

The Bugzilla database schema is available at Ravenbrook. This very valuable tool can generate a written description of the Bugzilla database schema for any version of Bugzilla. It can also generate a diff between two versions to help someone see what has changed.

MySQL

MySQL's default configuration is insecure. We highly recommend to run mysql_secure_installation on Linux or the MySQL installer on Windows, and follow the instructions. Important points to note are:

- ✓ Be sure that the root account has a secure password set.
- ✓ Do not create an anonymous account, and if it exists, say "yes" to remove it.
- ✓ If your web server and MySQL server are on the same machine, you should disable the network access.

Allow large attachments and many comments

By default, MySQL will only allow you to insert things into the database that are smaller than 1MB. Attachments may be larger than this. Also, Bugzilla combines all comments on a single bug into one field for full-text searching, and the combination of all comments on a single bug could in some cases be larger than 1MB.

To change MySQL's default, you need to edit your MySQL configuration file, which is usually /etc/my.cnf on Linux. We recommend that you allow at least 4MB packets by adding the "max_allowed_packet" parameter to your MySQL configuration in the "[mysqld]" section, like this:

```
[mysqld]
# Allow packets up to 4MB
max_allowed_packet=4M
```

Allow small words in full-text indexes

By default, words must be at least four characters in length in order to be indexed by MySQL's full-text indexes. This causes a lot of Bugzilla specific words to be missed, including "cc", "ftp" and "uri".

MySQL can be configured to index those words by setting the ft_min_word_len param to the minimum size of the words to index. This can be done by modifying the /etc/my.cnf according to the example below:

```
[mysqld]
# Allow small words in full-text indexes
ft_min_word_len=2
```

Rebuilding the indexes can be done based on documentation found at http://www.mysql.com/doc/en/Fulltext_Fine-tuning.html.

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.

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, LOCK TABLES,
    CREATE TEMPORARY TABLES, DROP, REFERENCES ON bugs.*
    TO bugs@localhost IDENTIFIED BY '$db_pass';
mysql> FLUSH PRIVILEGES;
```

Permit attachments table to grow beyond 4GB

By default, MySQL will limit the size of a table to 4GB. This limit is present even if the underlying filesystem has no such limit. To set a higher limit, follow these instructions.

After you have completed the rest of the installation (or at least the database setup parts), you should run the MySQL command-line client and enter the following, replacing $bugs_db with your Bugzilla database name (bugs by default):

```
mysql> use $bugs_db
mysql> ALTER TABLE attachments
        AVG_ROW_LENGTH=1000000, MAX_ROWS=20000;
```

The above command will change the limit to 20GB. Mysql will have to make a temporary copy of your entire table to do this. Ideally, you should do this when your attachments table is still small.

This does not affect Big Files, attachments that are stored directly on disk instead of in the database.

PostgreSQL

Add a User to PostgreSQL

You need to add a new user to PostgreSQL for the Bugzilla application to use when accessing the database. The following instructions assume the defaults in localconfig; if you changed those, you need to modify the commands appropriately. You will need the $db_pass password you set in localconfig.

On most systems, to create the user in PostgreSQL, you will need to login as the root user, and then

```
bash# su – postgres
```

As the postgres user, you then need to create a new user:

```
bash$ createuser -U postgres -dRSP bugs
```

When asked for a password, provide the password which will be set as $db_pass in localconfig. The created user will not be a superuser (-S) and will not be able to create new users (-R). He will only have the ability to create databases (-d).

Configure PostgreSQL

Now, you will need to edit pg_hba.conf which is usually located in /var/lib/pgsql/data/. In this file, you will need to add a new line to it as follows:

```
host all bugs 127.0.0.1 255.255.255.255 md5
```

This means that for TCP/IP (host) connections, allow connections from '127.0.0.1' to 'all' databases on this server from the 'bugs' user, and use password authentication (md5) for that user.

Now, you will need to restart PostgreSQL, but you will need to fully stop and start the server rather than just restarting due to the possibility of a change to postgresql.conf. After the server has restarted, you will need to edit localconfig, finding the $db_driver variable and setting it to Pg and changing the password in $db_pass to the one you picked previously, while setting up the account.

Oracle

Create a New Table space

You can use the existing tablespace or create a new one for Bugzilla. To create a new tablespace, run the following command:

```
CREATE TABLESPACE bugs
DATAFILE '$path_to_datafile' SIZE 500M
AUTOEXTEND ON NEXT 30M MAXSIZE UNLIMITED
```

Here, the name of the tablespace is 'bugs', but you can choose another name. $path_to_datafile is the path to the file containing your database, for instance /u01/oradata/bugzilla.dbf. The initial size of the database file is set in this example to 500 Mb, with an increment of 30 Mb everytime we reach the size limit of the file.

Add a User to Oracle

The user name and password must match what you set in localconfig ($db_user and $db_pass, respectively). Here, we assume that the user name is 'bugs' and the tablespace name is the same as above.

```
CREATE USER bugs
IDENTIFIED BY "$db_pass"
DEFAULT TABLESPACE bugs
TEMPORARY TABLESPACE TEMP
PROFILE DEFAULT;
-- GRANT/REVOKE ROLE PRIVILEGES
GRANT CONNECT TO bugs;
GRANT RESOURCE TO bugs;
-- GRANT/REVOKE SYSTEM PRIVILEGES
GRANT UNLIMITED TABLESPACE TO bugs;
GRANT EXECUTE ON CTXSYS.CTX_DDL TO bugs;
```

Configure the Web Server

If you use Apache, append these lines to httpd.conf to set ORACLE_HOME and LD_LIBRARY_PATH. For instance:

```
SetEnv ORACLE_HOME /u01/app/oracle/product/10.2.0/
SetEnv LD_LIBRARY_PATH /u01/app/oracle/product/10.2.0/lib/
```

When this is done, restart your web server.

SQLite

Due to SQLite's concurrency limitations we recommend SQLite only for small and development Bugzilla installations.

No special configuration is required to run Bugzilla on SQLite. The database will be stored in data/db/$db_name, where $db_name is the database name defined in localconfig.

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.

Web server

Configure your web server according to the instructions in the appropriate section. (If it makes a difference in your choice, the Bugzilla Team recommends Apache.) To check whether your web server is correctly configured, try to access testagent.cgi from your web server. If "OK" is displayed, then your configuration is successful. Regardless of which web server you are using, however, ensure that sensitive information is not remotely available by properly applying the access controls. You can run testserver.pl to check if your web server serves Bugzilla files as expected.

Bugzilla using Apache

You have two options for running Bugzilla under Apache - mod_cgi (the default) and mod_perl (new in Bugzilla 2.23)

Apache httpd™ with mod_cgi

To configure your Apache web server to work with Bugzilla while using mod_cgi, do the following:

- ✓ Load httpd.conf in your editor. In Fedora and Red Hat Linux, this file is found in /etc/httpd/conf.
- ✓ Apache uses <Directory> directives to permit fine-grained permission setting. Add the following lines to a directive that applies to the location of your Bugzilla installation. (If such a section does not exist, you'll want to add one.) In this example, Bugzilla has been installed at /var/www/html/bugzilla.

```
<Directory /var/www/html/bugzilla>
AddHandler cgi-script .cgi
Options +ExecCGI
DirectoryIndex index.cgi index.html
AllowOverride Limit FileInfo Indexes Options
</Directory>
```

These instructions: allow apache to run .cgi files found within the bugzilla directory; instructs the server to look for a file called index.cgi or, if not found, index.html if someone only types the directory name into the browser; and allows Bugzilla's .htaccess files to override some global permissions.

It is possible to make these changes globally, or to the directive controlling Bugzilla's parent directory (e.g. <Directory /var/www/html/>). Such changes would also apply to the Bugzilla directory... but they would also apply to many other places where they may or may not be appropriate. In most cases, including this one, it is better to be as restrictive as possible when granting extra access.

On Windows, you may have to also add the ScriptInterpreterSource Registry-Strict line, see Windows specific notes.

- ✓ **checksetup.pl** can set tighter permissions on Bugzilla's files and directories if it knows what group the web server runs as. Find the Group line in httpd.conf, place the value found there in the $webservergroup variable in localconfig, then rerun checksetup.pl.

- ✓ **Optional:** If Bugzilla does not actually reside in the webspace directory, but instead has been symbolically linked there, you will need to add the following to the Options line of the Bugzilla <Directory> directive (the same one as in the step above):

+FollowSymLinks

Without this directive, Apache will not follow symbolic links to places outside its own directory structure, and you will be unable to run Bugzilla.

Apache httpd™ with mod_perl

Some configuration is required to make Bugzilla work with Apache and mod_perl

- ✓ Load httpd.conf in your editor. In Fedora and Red Hat Linux, this file is found in /etc/httpd/conf.
- ✓ Add the following information to your httpd.conf file, substituting where appropriate with your own local paths.

This should be used instead of the <Directory> block shown above. This should also be above any other mod_perl directives within the httpd.conf and must be specified in the order as below.

You should also ensure that you have disabled KeepAlive support in your Apache install when utilizing Bugzilla under mod_perl

```
PerlSwitches -w -T
PerlConfigRequire /var/www/html/bugzilla/mod_perl.pl
```

- ✓ checksetup.pl can set tighter permissions on Bugzilla's files and directories if it knows what group the web server runs as. Find the Group line in httpd.conf, place the value found there in the $webservergroup variable in localconfig, then rerun checksetup.pl.

On restarting Apache, Bugzilla should now be running within the mod_perl environment. Please ensure you have run checksetup.pl to set permissions before you restart Apache.

Please bear the following points in mind when looking at using Bugzilla under mod_perl:

- ✓ mod_perl support in Bugzilla can take up a HUGE amount of RAM. You could be looking at 30MB per httpd child, easily. Basically, you just need a lot of RAM. The more RAM you can get, the better. mod_perl is basically trading RAM for speed. At least 2GB total system RAM is recommended for running Bugzilla under mod_perl.

- ✓ Under mod_perl, you have to restart Apache if you make any manual change to any Bugzilla file. You can't just reload--you have to actually restart the server (as in make sure it stops and starts again). You can change localconfig and the params file manually, if you want, because those are re-read every time you load a page.

- ✓ You must run in Apache's Prefork MPM (this is the default). The Worker MPM may not work--we haven't tested Bugzilla's mod_perl support under threads. (And, in fact, we're fairly sure it won't work.)

- ✓ Bugzilla generally expects to be the only mod_perl application running on your entire server. It may or may not work if there are other applications also running under mod_perl. It does try its best to play nice with other mod_perl applications, but it still may have conflicts.

- ✓ It is recommended that you have one Bugzilla instance running under mod_perl on your server. Bugzilla has not been tested with more than one instance running.

Microsoft Internet Information Services™

If you are running Bugzilla on Windows and choose to use Microsoft's Internet Information Services™ or Personal Web Server™ you will need to perform a number of other configuration steps as explained below. You may also want to refer to the following Microsoft Knowledge Base articles: 245225 "HOW TO: Configure and Test a PERL Script with IIS 4.0, 5.0, and 5.1" (for Internet Information Services™) and 231998 "HOW TO: FP2000: How to Use Perl with Microsoft Personal Web Server on Windows 95/98" (for Personal Web Server™).

You will need to create a virtual directory for the Bugzilla install. Put the Bugzilla files in a directory that is named something other than what you want your end-users accessing. That is, if you want your users to access your Bugzilla installation through "http://<yourdomainname>/Bugzilla", then do not put your Bugzilla files in a directory named "Bugzilla". Instead, place them in a different location, and then use the IIS Administration tool to create a Virtual Directory named "Bugzilla" that acts as an alias for the actual location of the files. When creating that virtual directory, make sure you add the "Execute (such as ISAPI applications or CGI)" access permission.

You will also need to tell IIS how to handle Bugzilla's .cgi files. Using the IIS Administration tool again, open up the properties for the new virtual directory and select the Configuration option to access the Script Mappings. Create an entry mapping .cgi to:

<full path to perl.exe >\perl.exe -x<full path to Bugzilla> -wT "%s" %s

For example:
c:\perl\bin\perl.exe -xc:\bugzilla -wT "%s" %s

The Active State install may have already created an entry for .pl files that is limited to "GET,HEAD,POST". If so, this mapping should be removed as Bugzilla's .pl files are not designed to be run via a web server.

IIS will also need to know that the index.cgi should be treated as a default document. On the Documents tab page of the virtual directory properties, you need to add index.cgi as a default document type. If you wish, you may remove the other default document types for this particular virtual directory, since Bugzilla doesn't use any of them.

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

Bugzilla

Your Bugzilla should now be working. Access http://<your-bugzilla-server>/ - you should see the Bugzilla front page.

The URL above may be incorrect if you installed Bugzilla into a subdirectory or used a symbolic link from your web site root to the Bugzilla directory.

Log in with the administrator account you defined in the last checksetup.pl run. You should go through the Parameters page and see if there are any you wish to change. They key parameters are documented, you should certainly alter maintainer and urlbase; you may also want to alter cookie path or require login.

Bugzilla has several optional features which require extra configuration.

1.3. Optional Additional Configuration

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.

Windows does not have 'cron', but it does have the Task Scheduler, which performs the same duties. There are also third-party tools that can be used to implement cron, such as nncron.

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 CONFIRMED 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
Windows does not have 'cron', but it does have the Task Scheduler, which performs the same duties. There are also third-party tools that can be used to implement cron, such as nncron.

Whining

As of Bugzilla 2.20, users can configure Bugzilla to regularly annoy them at regular intervals, by having Bugzilla execute saved searches at certain times and emailing the results to the user. This is known as "Whining". The process of configuring But for whining to work a Perl script must be executed at regular intervals.

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 every 15 minutes.

*/15 * * * * cd <your-bugzilla-directory> && ./whine.pl
Whines can be executed as often as every 15 minutes, so if you specify longer intervals between executions of whine.pl, some users may not be whined at as often as they would expect. Depending on the person, this can either be a very Good Thing or a very Bad Thing.

Windows does not have 'cron', but it does have the Task Scheduler, which performs the same duties. There are also third-party tools that can be used to implement cron, such as nncron.

Serving Alternate Formats with the right MIME type

Some Bugzilla pages have alternate formats, other than just plain HTML. In particular, a few Bugzilla pages can output their contents as either XUL (a special Mozilla format that looks like a program GUI) or RDF (a type of structured XML that can be read by various programs).

In order for your users to see these pages correctly, Apache must send them with the right MIME type. To do this, add the following lines to your Apache configuration, either in the <Virtual Host> section for your Bugzilla, or in the <Directory> section for your Bugzilla:

Add Type application/vnd.mozilla.xul+xml .xul
Add Type application/rdf+xml .rdf

1.4. Multiple Bugzilla databases with a single installation

The previous instructions referred to a standard installation, with one unique Bugzilla database. However, you may want to host several distinct installations, without having several copies of the code. This is possible by using the PROJECT environment variable. When accessed, Bugzilla checks for the existence of this variable, and if present, uses its value to check for an alternative

configuration file named localconfig.<PROJECT> in the same location as the default one (localconfig). It also checks for customized templates in a directory named <PROJECT> in the same location as the default one (template/<langcode>). By default this is template/en/default so PROJECT's templates would be located at template/en/PROJECT.

To set up an alternate installation, just export PROJECT=foo before running checksetup.pl for the first time. It will result in a file called localconfig.foo instead of localconfig. Edit this file as described above, with reference to a new database, and re-run checksetup.pl to populate it. That's all.

Now you have to configure the web server to pass this environment variable when accessed via an alternate URL, such as virtual host for instance. The following is an example of how you could do it in Apache, other Webservers may differ.

<Virtual Host 212.85.153.228:80>
  Server Name foo.bar.baz
  SetEnv PROJECT foo
  Alias /bugzilla /var/www/bugzilla
</Virtual Host>

1.5. OS-Specific Installation Notes

Many aspects of the Bugzilla installation can be affected by 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.

Microsoft Windows

Making Bugzilla work on Windows is more difficult than making it work on Unix. For that reason, we still recommend doing so on a Unix based system such as GNU/Linux. That said, if you do want to get Bugzilla running on Windows, you will need to make the following adjustments. A detailed step-by-step installation guide for Windows is also available if you need more help with your installation.

Win32 Perl

Perl for Windows can be obtained from Active State. You should be able to find a compiled binary at http://aspn.activestate.com/ASPN/Downloads/ActivePerl/. The following instructions assume that you are using version 5.8.1 of Active State.

These instructions are for 32-bit versions of Windows. If you are using a 64-bit version of Windows, you will need to install 32-bit Perl in order to install the 32-bit modules as described below.

Perl Modules on Win32

Bugzilla on Windows requires perl modules. The main difference is that windows use PPM instead of CPAN. Active State provides a GUI to manage Perl modules. We highly recommend that you use it. If you prefer to use ppm from the command-line, type:

C:\perl> ppm install <module name>

The PPM repository stores modules in 'packages' that may have a slightly different name than the module. If retrieving these modules from there, you will need to pay attention to the information provided when you run checksetup.pl as it will tell you what package you'll need to install.

If you are behind a corporate firewall, you will need to let the Active State PPM utility know how to get through it to access the repositories by setting the HTTP proxy system environmental variable.

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.

The web server looks at /usr/bin/perl to call Perl. If you are using Apache on windows, you can set the ScriptInterpreterSource directive in your Apache config file to make it look at the right place: insert the line

ScriptInterpreterSource Registry-Strict
into your httpd.conf file, and create the key

HKEY_CLASSES_ROOT\.cgi\Shell\ExecCGI\Command
with C:\Perl\bin\perl.exe -T as value (adapt to your path if needed) in the registry. When this is done, restart Apache.

Sending Email

To enable Bugzilla to send email on Windows, the server running the Bugzilla code must be able to connect to, or act as, an SMTP server.

Mac OS X™

Making Bugzilla work on Mac OS X requires the following adjustments.

Send mail

In Mac OS X 10.3 and later, Postfix is used as the built-in email server. Postfix provides an executable that mimics send mail enough to fool Bugzilla, as long as Bugzilla can find it. Bugzilla is able to find the fake send mail executable without any assistance.

Libraries & Perl Modules on Mac OS X

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

You can use MacPorts (http://www.macports.org/) or Fink (http://sourceforge.net/projects/fink/), both of which are similar in nature to the CPAN installer, but install common unix programs.

Follow the instructions for setting up MacPorts or Fink. Once you have one installed, you'll want to use it to install the gd2 package.

Fink 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.

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/includes instead of /usr/lib and /usr/include. When the Perl module config script asks where your libgd is, be sure to tell it /sw/lib.

Also available via MacPorts and Fink is expat. After installing the expat package, you will be able to install XML::Parser using CPAN. If you use fink, 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'
# perl Makefile.PL EXPATLIBPATH=/sw/lib EXPATINCPATH=/sw/include
# make; make test; make install
# exit
```

The look command will download the module and spawn a new shell with the extracted files as the current working directory.

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

The exit command will return you to your original shell.

Linux/BSD Distributions

Many Linux/BSD distributions include Bugzilla and its dependencies in their native package management systems. Installing Bugzilla with root access on any Linux/BSD system should be as simple as finding the Bugzilla package in the package management application and installing it using the normal command syntax. Several distributions also perform the proper web server configuration automatically on installation.

Please consult the documentation of your Linux/BSD distribution for instructions on how to install packages, or for specific instructions on installing Bugzilla with native package management tools. There is also a Bugzilla Wiki Page for distro-specific installation notes.

1.6. UNIX (non-root) Installation Notes

Introduction

If you are running a *NIX OS as non-root, either due to lack of access (web hosts, for example) or for security reasons, this will detail how to install Bugzilla on such a setup.

MySQL

You may have MySQL installed as root. If you're setting up an account with a web host, a MySQL account needs to be set up for you. From there, you can create the bugs account, or use the account given to you.

You may have problems trying to set up GRANT permissions to the database. If you're using a web host, chances are that you have a separate database which is already locked down (or one big database with limited/no access to the other areas), but you may want to ask your system administrator what the security settings are set to, and/or run the GRANT command for you.

Also, you will probably not be able to change the MySQL root user password (for obvious reasons), so skip that step002E

Running MySQL as Non-Root

The Custom Configuration Method

Create a file .my.cnf in your home directory (using /home/foo in this example) as follows....

```
[mysqld]
datadir=/home/foo/mymysql
socket=/home/foo/mymysql/thesock
port=8081

[mysql]
socket=/home/foo/mymysql/thesock
port=8081

[mysql.server]
user=mysql
basedir=/var/lib

[safe_mysqld]
err-log=/home/foo/mymysql/the.log
pid-file=/home/foo/mymysql/the.pid
```

The Custom Built Method

You can install MySQL as a not-root, if you really need to. Build it with PREFIX set to /home/foo/mysql, or use pre-installed executables, specifying that you want to put all of the data files in /home/foo/mysql/data. If there is another MySQL server running on the system that you do not own, use the -P option to specify a TCP port that is not in use.

Starting the Server

After your mysqld program is built and any .my.cnf file is in place, you must initialize the databases (ONCE).
bash$ mysql_install_db

Then start the daemon with
bash$ safe_mysql &

After you start mysqld the first time, you then connect to it as "root" and GRANT permissions to other users. (Again, the MySQL root account has nothing to do with the *NIX root account.)

You will need to start the daemons yourself. You can either ask your system administrator to add them to system startup files, or add a crontab entry that runs a script to check on these daemons and restart them if needed.\

Do NOT run daemons or other services on a server without first consulting your system administrator! Daemons use up system resources and running one may be in violation of your terms of service for any machine on which you are a user!

Perl

On the extremely rare chance that you don't have Perl on the machine, you will have to build the sources yourself. The following commands should get your system installed with your own personal version of Perl:

bash$ wget http://perl.org/CPAN/src/stable.tar.gz
bash$ tar zvxf stable.tar.gz
bash$ cd perl-5.8.1
bash$ sh Configure -de -Dprefix=/home/foo/perl
bash$ make && make test && make install

Once you have Perl installed into a directory (probably in ~/perl/bin), you will need to install the Perl Modules, described below.

Perl Modules

Installing the Perl modules as a non-root user is accomplished by running the install-module.pl script.

HTTP Server

Ideally, this also needs to be installed as root and run under a special web server account. As long as the web server will allow the running of *.cgi files outside of a cgi-bin, and a way of denying web access to certain files (such as an .htaccess file), you should be good in this department.

Running Apache as Non-Root

You can run Apache as a non-root user, but the port will need to be set to one above 1024. If you type httpd -V, you will get a list of the variables that your system copy of httpd uses. One of those, namely HTTPD_ROOT, tells you where that installation looks for its config information.

From there, you can copy the config files to your own home directory to start editing. When you edit those and then use the -d option to override the HTTPD_ROOT compiled into the web server, you get control of your own customized web server.

You will need to start the daemons yourself. You can either ask your system administrator to add them to system startup files, or add a crontab entry that runs a script to check on these daemons and restart them if needed.

Do NOT run daemons or other services on a server without first consulting your system administrator! Daemons use up system resources and running one may be in violation of your terms of service for any machine on which you are a user!

Bugzilla

When you run ./checksetup.pl to create the localconfig file, it will list the Perl modules it finds. If one is missing, go back and double-check the module installation, then delete the localconfig file and try again.

One option in localconfig you might have problems with is the web server group. If you can't successfully browse to the index.cgi (like a Forbidden error), you may have to relax your permissions, and blank out the web server group Of course, this may pose as a security risk. Having a properly jailed shell and/or limited access to shell accounts may lessen the security risk, but use at your own risk.

suexec or shared hosting

If you are running on a system that uses suexec (most shared hosting environments do this), you will need to set the webservergroup value in localconfig to match your primary group, rather than the one the web server runs under. You will need to run the following shell commands after running ./checksetup.pl, every time you run it (or modify checksetup.pl to do them for you via the system() command).

For i in docs graphs images js skins; do find $i -type d -exec chmod o+rx {} \; ; done
For i in jpg gif css js png html rdf xul; do find . -name *.$i -exec chmod o+r {} \; ; done
Find . -name .htaccess -exec chmod o+r {} \;
Chmod o+x . data data/webdot

Pay particular attention to the number of semicolons and dots. They are all important. A future version of Bugzilla will hopefully be able to do this for you out of the box.

1.7. Upgrading to New Releases

Upgrading to new Bugzilla releases is very simple. There is a script named checksetup.pl included with Bugzilla that will automatically do all of the database migration for you.

The following sections explain how to upgrade from one version of Bugzilla to another. Whether you are upgrading from one bug-fix version to another (such as 4.2 to 4.2.1) or from one major version to another (such as from 4.0 to 4.2), the instructions are always the same.

Any examples in the following sections are written as though the user were updating to version 4.2.1, but the procedures are the same no matter what version you're updating to. Also, in the examples, the user's Bugzilla installation is found at /var/www/html/bugzilla. If that is not the same as the location of your Bugzilla installation, simply substitute the proper paths where appropriate.

Before You Upgrade

Before you start your upgrade, there are a few important steps to take:

- ✓ Read the Release Notes of the version you're upgrading to, particularly the "Notes for Upgraders" section.
- ✓ View the Sanity Check page on your installation before upgrading. Attempt to fix all warnings that the page produces before you go any further, or you may experience problems during your upgrade.
- ✓ Shut down your Bugzilla installation by putting some HTML or text in the shutdownhtml parameter.
- ✓ Make a backup of the Bugzilla database. THIS IS VERY IMPORTANT. If anything goes wrong during the upgrade, your installation can be corrupted beyond recovery. Having a backup keeps you safe.

Upgrading is a one-way process. You cannot "downgrade" an upgraded Bugzilla. If you wish to revert to the old Bugzilla version for any reason, you will have to restore your database from this backup.

Here are some sample commands you could use to backup your database, depending on what database system you're using. You may have to modify these commands for your particular setup.

MySQL:
mysqldump --opt -u bugs -p bugs > bugs.sql

PostgreSQL:
pg_dump --no-privileges --no-owner -h localhost -U bugs > bugs.sql

Getting The New Bugzilla

There are three ways to get the new version of Bugzilla. We'll list them here briefly and then explain them more later.

Bzr - If you have bzr installed on your machine and you have Internet access, this is the easiest way to upgrade, particularly if you have made modifications to the code or templates of Bugzilla.

Download the tarball - This is a very simple way to upgrade, and good if you haven't made many (or any) modifications to the code or templates of your Bugzilla.

Patches - If you have made modifications to your Bugzilla, and you don't have Internet access or you don't want to use bzr, then this is the best way to upgrade.

You can only do minor upgrades with patches.

If you have modified your Bugzilla

If you have modified the code or templates of your Bugzilla, then upgrading requires a bit more thought and effort.

The larger the jump you are trying to make, the more difficult it is going to be to upgrade if you have made local customizations. Upgrading from 4.2 to 4.2.1 should be fairly painless even if you are heavily customized, but going from 2.18 to 4.2 is going to mean a fair bit of work re-writing your local changes to use the new files, logic, templates, etc. If you have done no local changes at all, however, then upgrading should be approximately the same amount of work regardless of how long it has been since your version was released.
Upgrading using Bzr
This requires that you have bzr installed (most Unix machines do), and requires that you are able to access bzr.mozilla.org, which may not be an option if you don't have Internet access.

The following shows the sequence of commands needed to update a Bugzilla installation via Bzr, and a typical series of results. These commands assume that you already have Bugzilla installed using Bzr.

If your installation is still using CVS, you must first convert it to Bzr. A very detailed step by step documentation can be found on wiki.mozilla.org.

```
bash$ cd /var/www/html/bugzilla
bash$ bzr switch 4.2 (only run this command when not yet running 4.2)
bash$ bzr up -r tag:bugzilla-4.2.1
+N  extensions/MoreBugUrl/
+N  extensions/MoreBugUrl/Config.pm
+N  extensions/MoreBugUrl/Extension.pm
...
 M  Bugzilla/Attachment.pm
 M  Bugzilla/Attachment/PatchReader.pm
 M  Bugzilla/Bug.pm
...
All changes applied successfully.
```

If a line in the output from bzr up mentions a conflict, then that represents a file with local changes that Bzr 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.

Upgrading using the Tarball

If you are unable (or unwilling) to use Bzr, another option that's always available is to obtain the latest tarball from the Download Page and create a new Bugzilla installation from that.

This sequence of commands shows how to get the tarball from the command-line; it is also possible to download it from the site directly in a web browser. If you go that route, save the file to the /var/www/html directory (or its equivalent, if you use something else) and omit the first three lines of the example.

```
bash$ cd /var/www/html
bash$ wget http://ftp.mozilla.org/pub/mozilla.org/webtools/bugzilla-4.2.1.tar.gz
(Output omitted)
bash$ tar xzvf bugzilla-4.2.1.tar.gz
bugzilla-4.2.1/
bugzilla-4.2.1/colchange.cgi
(Output truncated)
bash$ cd bugzilla-4.2.1
bash$ cp ../bugzilla/localconfig* .
bash$ cp -r ../bugzilla/data .
bash$ cd ..
bash$ mv bugzilla bugzilla.old
bash$ mv bugzilla-4.2.1 bugzilla
```

The cp commands both end with periods which is a very important detail--it means that the destination directory is the current working directory.

[Caution]
If you have some extensions installed, you will have to copy them to the new bugzilla directory too. Extensions are located in bugzilla/extensions/. Only copy those you installed, not those managed by the Bugzilla team.

This upgrade method will give you a clean install of Bugzilla. That's fine if you don't have any local customizations that you want to maintain. If you do have customizations, then you will need to reapply them by hand to the appropriate files.

Upgrading Using Patches

A patch is a collection of all the bug fixes that have been made since the last bug-fix release.

If you are doing a bug-fix upgrade—that is, one where only the last number of the revision changes, such as from 4.2 to 4.2.1—then you have the option of obtaining and applying a patch file from the Download Page.

As above, this example starts with obtaining the file via the command line. If you have already downloaded it, you can omit the first two commands.

```
bash$ cd /var/www/html/bugzilla
bash$ wget http://ftp.mozilla.org/pub/mozilla.org/webtools/bugzilla-4.2-to-4.2.1.diff.gz
(Output omitted)
bash$ gunzip bugzilla-4.2-to-4.2.1.diff.gz
bash$ patch -p1 < bugzilla-4.2-to-4.2.1.diff
patching file Bugzilla/Constants.pm
patching file enter_bug.cgi
(etc.)
```

Be aware that upgrading from a patch file does not change the entries in your .bzr directory. This could make it more difficult to upgrade using Bzr in the future.

Completing Your Upgrade
Now that you have the new Bugzilla code, there are a few final steps to complete your upgrade.

- ✓ If your new Bugzilla installation is in a different directory or on a different machine than your old Bugzilla installation, make sure that you have copied the data directory and the local config file from your old Bugzilla installation. (If you followed the tarball instructions above, this has already happened.)

- ✓ If this is a major update, check that the configuration for your new Bugzilla is up-to-date. Sometimes the configuration requirements change between major versions.

- ✓ If you didn't do it as part of the above configuration step, now you need to run checksetup.pl, which will do everything required to convert your existing database and settings for the new version:

```
bash$ cd /var/www/html/bugzilla
bash$ ./checksetup.pl
```

The period at the beginning of the command ./checksetup.pl is important and cannot be omitted.

If this is a major upgrade (say, 3.6 to 4.2 or similar), running checksetup.pl on a large installation (75,000 or more bugs) can take a long time, possibly several hours.

- ✓ Clear any HTML or text that you put into the shutdownhtml parameter, to re-activate Bugzilla.
- ✓ View the Sanity Check page in your upgraded Bugzilla.

It is recommended that, if possible, you fix any problems you see, immediately. Failure to do this may mean that Bugzilla will not work correctly. Be aware that if the sanity check page contains more errors after an upgrade, it doesn't necessarily mean there are more errors in your database than there were before, as additional tests are added to the sanity check over time, and it is possible that those errors weren't being checked for in the old version.

Automatic Notifications of New Releases

Bugzilla 3.0 introduced the ability to automatically notify administrators when new releases are available, based on the upgrade notification parameter. Administrators will see these notifications when they access the index.cgi page, i.e. generally when logging in. Bugzilla will check once per day for new releases, unless the parameter is set to “disabled”. If you are behind a proxy, you may have to set the proxy_url parameter accordingly. If the proxy requires authentication, use the