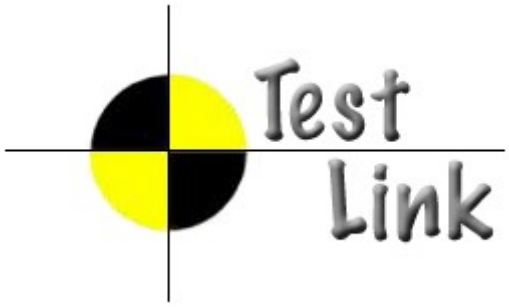


Installation & Configuration Manual

TestLink version 1.8

Version: 2.10

Status: Reviewed

1. Scope

This document serves as a reference and knowledge base for the installation and configuration of tool **TestLink 1.8**. The first part includes the installation procedure and second part the configuration explanation.

The latest documentation is available on TestLink homepage. You can also ask for help to solve your problems in an appropriate section of TestLink forum.

Summary of installation process:

1. Install background services
2. Transfer and uncompress files into web directory
3. Generate database tables and add data (create default or transfer from previous DB)
4. Edit configuration files
5. PHP File extensions
6. Login

TestLink includes installation scripts that helps you easily set-up all required configuration and database structure.

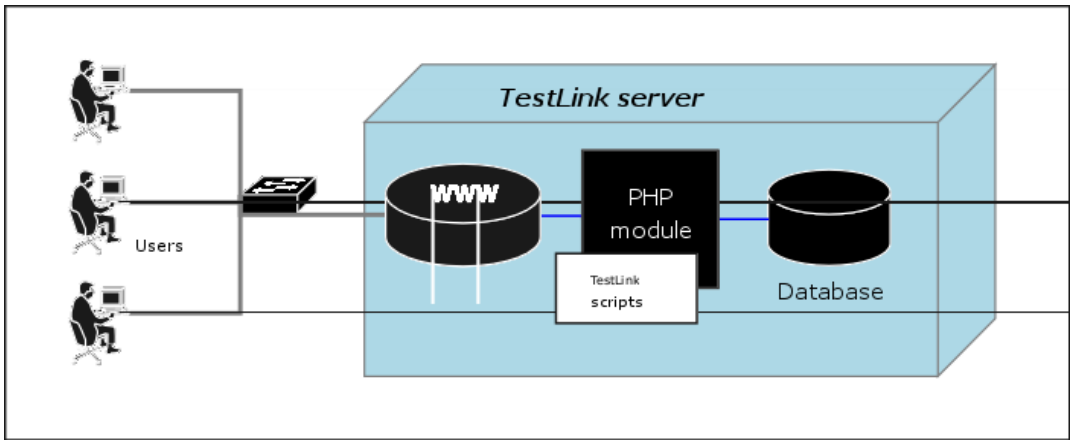
Table of Contents

2. System Requirements

2.1. Client side

We support WWW browsers **Firefox** 1.0 (and higher) and **Internet Explorer 6**. There are some issues with IE 7 because macrosoft doesn't satisfy standards. Generally any other browser should work if it supports JavaScript, XHTML and CSS.

2.2. Server side



TestLink server requires these applications as background:

- **Database** You can run also your database on both the same or different server than TestLink php scripts.
    - MySQL 4.1.x and higher (4.0.x doesn't support UTF-8) [1][2]
    - Postgres 8.x and higher
    - MS SQL 2000 and higher (experimental)
    - Any other - TestLink satisfies SQL standard. So you can use any well known database. However you need to set-up database schema yourself (we don't support it yet).
- **Web server** (Apache 1.3.x or 2.x and higher, IIS 3 and higher, etc.). See `<php_root>/install.txt` for more information.
- **php** 5.x and higher (version 5.2 is recommended)
- **Bug tracking system** (optional collaboration)
    - Bugzilla 0.19.1 and higher

1 Migration from TestLink 1.6 requires MySQL5
2 Database type MyISAM (default) and InnoDB are supported

    - Mantis 1.0.1 and higher
    - JIRA 3.1.1 and higher
    - TrackPlus 3.3 and higher
    - Eventum 2.0 and higher
    - Trac 0.10 and higher
- There is no requirement about your operating system (tested on Linux and MS Win32).

3. Installation

You can use automatic scripted installation or manual steps. If you are upgrading from a previous version of TestLink look at the Upgrading section.

3.1. Pre-installation steps

Do the next steps before installation:

1. Install environment: Web server **with php5** and **database**. Refer to documentation of these products and TestLink System requirements (2.2 section). You can also find installations package of all these products and install it together; for example XAMPP, EasyPHP, Uniform Server, etc.

   PHP4 is not supported from TL 1.8 version

2. Transfer the TestLink installation file to your web server using whatever method you like best (ftp, scp, etc.). You will need to telnet/SSH access into the server machine for the next steps (if not localhost).

3. Decompress the package

   - **Linux:** untar/gunzip it to the directory that you want. The usual command is (1 step):

     ```
     # tar zxvf <filename.tar.gz>
     ```

   - **Microsoft:** Total Commander, Winzip, Stuffit, and other programs should also be able to handle decompression of the archive.

4. At this point you may want to rename the directory to something simpler like 'testlink'. You will use the mv command to rename a directory (Windows users substitute the "ren" command or use explorer).

   ```
   # mv <directory_name> testlink
   ```

5. Continue Installation of database structure and configuration or Upgrade.

3.2. AUTOMATIC Installation

TestLink includes installation scripts that help you set-up all mandatory configuration and database structure. The following details the basic steps for installation on any system. The instructions may seem Unix-centric but should work fine on Windows systems. Barring complications, it should take you about 5-30 minutes to install, configure, and be using TestLink.

Next we will create the necessary database tables and a configuration file for database access.

1. From your web browser access **`http://<yoursite>/testlink/install/index.php`**.

2. This page will walk through the following steps:

- check basic parameters for the web server, php config and DB version.
- prompt for the database type and location, and a database user/password pair. For installation, an administrative user/password pair can also be provided. The operating user requires ALTER, SELECT, INSERT, and UPDATE privileges. For installation, INDEX, CREATE, DELETE, and DROP privileges are also required.
- create the database and tables.

  Important: *A DEFAULT ADMINISTRATOR level account is created.*

  The account name and password are: **admin / admin**. Use this when you first login to TestLink. Immediately go to Manage and create at least one administrator level account. You can recreate it but you should delete the account to prevent the cookie_string from being used to trick the package. It would be even better to rename the account or delete it permanently.

  SECURITY: *Remove the default admin account is good practice*
- perform some post installation checks on the system.

3. After a successful upgrade you should remove the `<testlinkwebdir>/install/` directory for security reasons.
4. The next part involves configuring the installation to work with your specific setup. See configuration section for description of configurable parameters.

3.3. MANUAL Installation

If you want to perform a manual installation here are the steps needed for a successful installation.

Warning: *We recommend using the automated script as there are some undocumented tweaks. You can read the code of installation script if you need to do it manually and have a difficulty.*

The next description is written for MySQL, but these actions are applicable (with different tools and syntax) for other databases as well. For installing the DB you can either choose the command line tools available in your MySQL installation or any MySQL Database Client (e.g. phpMyAdmin).

- Prepare MySQL via command line tools:
  - Create a new empty MySQL database.

    for MySQL >= 4.1 (with UTF8) do

    ```
    CREATE DATABASE testlink CHARACTER SET utf8 COLLATE utf8_general_ci
    ```

    By choosing UTF8 you should also change the value of DB_SUPPORTS_UTF8 to TRUE in your `<testlinkdir>/config.inc.php` See Configuration for more.
  - Create tables for the newly created database.

```
# mysql -u <user> -p<password> <dbname> < <testlinkdir>/install/sql/
testlink_create_tables.sql
E.g. # mysql -u testlink -ppass testlink <
/var/www/html/testlink/install/sql/testlink_create_tables.sql
```

- Populate initial data for the newly created database (admin account, default roles).

```
# mysql -u <user> -p<password> <dbname> < <testlinkdir>/install/sql/
testlink_create_default_data.sql
```

- Alternatively you can use phpMyAdmin:
  - Create new database from main page (UTF-8 character set).
  - Optionally create a new user and assign him correct rights for the created database.
  - Select the created database in the left pane.
  - Navigate to SQL window.
  - Upload SQL request from files `/install/sql/testlink_create_tables.sql` and run the script.
  - Upload SQL request from files `/install/sql/testlink_create_default_data.sql` and run the script.
- Create a <testlinkdir>/config_db.inc.php file with the following data (example):

```
<?php // Automatically Generated by TestLink Installer
define('DB_TYPE', 'mysql');
define('DB_USER', 'testlinker');
define('DB_PASS', 'testlink_pass');
define('DB_HOST', 'localhost');
define('DB_NAME', 'tl_master');
?>
```

- (Optional) Create a DB user for connection from TestLink. Don't forget to assign a correct rights (at least SELECT, INSERT, UPDATE, DELETE) for the created database. The user must be defined in config_db.inc.php. Otherwise you can use any other user available in MySQL database with correct rights.
- You must allow write access to directories where TestLink expect to write. Change the permissions of the `templates_c`, `upload_area` and `logs` directory to be writeable by the web server. Linux/UNIX run from the TestLink root directory

```
# chmod 777 gui/templates_c
# chmod 777 logs
# chmod 777 upload_area
```

IIS users also needs to have allow it in dependence to IIS global configuration.

Note: *You can configure testlink to use another writable directories for security reason. Modify configuration parameters to point another directories (see configuration section).*

- Log into TestLink! Default credentials are:
- user: admin; pass: admin
- Changing this password is a good security practice. TestLink notifies if you don't do it.
- After a successful upgrade you should remove the `<testlinkwebdir>/install/` directory for security reasons.
- The next part involves configuring the installation to work with your specific set-up. See configuration section for description of configurable parameter.
- Report any issues or feedback to TestLink Bug tracking system page.

3.4. Post installation steps

3.4.1. Configure TestLink

There is amount of settings that helps to tool for your case. There is configuration section in this document. In addition we recommend to read config.inc.php and cfg/const.inc.php. You find commented parameters here.

3.4.2. Back-up TestLink installation

It's useful to have the last working configuration in extra back-up. Minimal list contains: custom_config.inc.php, config_db.inc.php and configuration for connection to you bug tracker (if any).

You should make some modification and patches. We suggest to backup all the directory after any change then.

3.4.3. Database back-up

Important: save your time – spend a half of hour today with arranging a simple script than boring days later!

There are two things to **back-up daily**: database and attachments. You could use sophisticated tool to store back-up files, store to back-up files to another server or use raid to suppress a storage failure.

Create a script that collect data from database and attachment folder. An example for Linux environment follows.

```
#!/bin/sh
# Script for backup TestLink service


# create filenames
```

```
mydate=`date +%y%m%d`
backup_folder="/home/gat/backup"
upload_folder="/home/gat/web/testlink/upload_area/"
filename1="$backup_folder/testlink_db_$mydate.bck.sql"
filename2="$backup_folder/testlink_upload_$mydate.bck.tgz"

# dump data
mysqldump -uroot -pyour_password testlink_17 > $filename1
# compress
gzip -f9 $filename1
# backup attachments
tar -cvzf $filename2 $upload_folder

# save to backuped area on server titan
# mount titan:/export/gat /home/gat/backup/titan/ -o
soft,nosuid,rw,noauto,user,noexec
#cp $filename.gz /home/gat/backup/titan/
```

The script create two compressed files with date in file name. There is also commented possibility to mount another server and copy back-up files to another server.

Now you have a script that can do back-up. You need to set-up system to run it every day. Linux/Unix offers cron service to do it. There is example of settings (root execute the script as user gat):

```
europa> crontab -l
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.20829 installed on Fri Aug  6 11:56:56 2004)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
15 2 * * * su - gat -c /home/gat/backup/db_backup.sh >/dev/null 2>&1
```

Do not postpone back-up to never!

4. Upgrading

Major version upgrade: You can upgrade either automatically (via script) or manually. There are typically new functionality and several changes in database against older TL main releases. I.e. you are not able to use directly your original database. (for example 1.6 -> 1.7)

Hot-Fix version is bug fixing release.

4.1.1. Hot-Fix release update

Maintenance (Bug fixing) release is for example 1.6.0 -> 1.6.1. Database schema shouldn't changed in this case.

- Backup all files of the previous version in testlink directory and database.
- Remove the all files from directory.
- Copy a new version to the same directory.
- Copy **config_db.inc.php** and **custom_config.inc.php** file to the new structure and modify other configuration parameters and changes according your previous settings (for example in config.inc.php file).
- Upgrade of DB is not required.
- Now, it should work.

4.1.2. Automatic upgrading major version

- Follow/check preinstallation steps. Requirement changes.
- From a web browser run `http://<testlinkwebdir>/install/index.php`
- Choose '***Migration from ...***' link if your original version is available in menu. Run the scripts until you see that process is finished. For example: your version is TL 1.6.3. Run scripts under 'Migration from 1.6.2 to 1.7.x', then run scripts under 'Migration from 1.7.2 (or greater) to 1.8.0' link.
- Choose '***Upgrade Installation***' link. Run the scripts until you see that process is finished.
- After a successful upgrade you should remove the `<testlinkwebdir>/install/` directory for security reasons.
- The next part involves configuring the installation to work with your specific setup. See configuration section for description of configurable parameter.
- Report any issues or feedback to TestLink Bug tracking system page.

Note: Upgrade from version 1.6 (and older) to 1.7 needs to use the large scripted database migration because of major changes in DB schema. It is an extra action in installation script (follow the instructions of the script).

4.1.3. Manual upgrading

This chapter describes the changes from previous versions. The automatic upgrade is recommended. Use this chapter for special cases and fiddling config. You can do it of course after a study of changes in database and installation script. Good idea is to compare SQL files for create DB tables (your current version and a new one).

4.2. Backward compatibility

4.2.1. Database schema changes

New and enhanced functionality results into adaptation of

- user password is encrypted (1.5)
- A new tables for SRS feature: requirements, req_coverage, requirement_doc (1.6)
- Attachments (1.7)
- Custom fields (1.7)
- Test Suites (1.7)
- Test Case versioning (1.7)
- Events logging (1.8)

Note: See `<testlink_root>/install/sql/alter/` directory to see exact syntax of changes.

4.2.2. Changed Terminology in TL 1.7

We continuously clarify terminology to fit testing standards. You will experience the next changes:

- Product => Test Project
- Component, Category => Test Suite

4.2.3. Obsolete functionality from 1.8

- Personal metrics on main page (parameter: MAIN_PAGE_METRICS_ENABLED)

4.2.4. Test Plan relation to Test Project

Starting with version 1.6 when you create a Test Plan, it's associated to the current selected Test Project as default. TestLink 1.7 automatically offers to assign not-associated Test Plans.

Backward compatibility: The solution in TL 1.6 tables include field *TestProjectID* in the Test Plan table. Test Plans could be available over all Test projects (Products). Such Test Plan has *TestProjectID* value = 0.

Warning: unassigned Test Plans are not officially supported

Before TestLink 1.6 the Test Plans where not associated to an specific Test Project. When upgrading from 1.5.x to 1.6, it's not possible for the installer to know which Test Project relates to which test plan, so Test Project ID is set to 0. This results in a situation where you find you can't see any of your old Test Plans !!!. To solve this problem the following configuration parameter was added:

```
$g_show_tp_without_prodid = TRUE;
```

TestLink 1.7 automatically offers to assign not-associated Test Plans. You can also via DB administration assign this relation manually and use this feature for data from previous version.

Filtering Test Plans by Test Project: As stated before the default behaviour is to filter Test Plan by Test Project. Using the following configuration parameter:

```
$g_ui_show_check_filter_tp_by_testproject = TRUE;
```

Allow the user, through the user interface , to enable/disable test plan filter by Test Project. A check box is displayed over the test plan combo box. Force Test Plan filtering, without any user possibility to change it.

```
$g_ui_show_check_filter_tp_by_testproject = FALSE;
```

4.2.5. Test Plan relation to Test Project

TL 1.0.4 does not relate Test Plan to Test project (Product).

Configuration within `<testlink_root>/config.inc.php`:

```
$g_show_tp_without_prodid=1;
$g_ui_show_check_filter_tp_by_testproject = 1;.
```

4.2.6. Latin to UTF-8 conversion (upgrade from 1.5 and older)

TestLink 1.6 allows for UTF-8 encoded character rendering, therefore any extended character data that may have snuck into your database and didn't show up in 1.5 may start appearing in 1.6 UI. You can turn UTF-8 support off in testlink by modifying a value in the `<testlinkinstalldir>/config.inc.php` file, but then you will be missing out on the ability to use characters beyond ASCII.

If you have the same problem I did and see lots of extended characters appearing in your data after upgrading to 1.6 and having UTF-8 support turned on, you should read through the following instructions. Be sure to practice this exercise on a test machine before performing on your deployment system.

The instructions will help you clear out any non-ASCII characters from your database and set-up your database to support UTF-8.

- First make a backup of your current database using the mysqldump utility.

```
# /usr/bin/mysqldump -u root testlink15 -p > testlink15.backup
```

- Now edit testlink15.backup so schema definitions for EACH table has utf8 encoding specified. Change the CHARSET for each table from latin1 to utf8. For example the following line in the definition of a table which reads as follows :

```
ENGINE=MyISAM DEFAULT CHARSET=latin1 COMMENT='This table holds the bugs
filed for each result';
```

  should be changed to

```
ENGINE=MyISAM DEFAULT CHARSET=utf8 COMMENT='This table holds the bugs
filed for each result';
```

- Then ran testlink15.backup through my the perl script below as follows:

```
/replaceScript.pl < testlink15.backup > testlink15.cleaned
replaceScript.pl is as follows :
#!/usr/bin/perl
while (<>) {
        chomp;
        tr/\000-\177/\040/cs;
        print $_, "\n";
}
```

- Created an empty testlink16 database with utf8 charset as follows:

```
CREATE DATABASE testlink16 CHARACTER SET utf8;
```

- Install the tables into the new database

```
# mysql testlink16 -u root -p < testlink15.cleaned
```

- You can verify your database's "DB character set" is now set to utf8 by using the following command:

```
login to mysql
use testlink16
mysql> \s
--------------
```

```
mysql  Ver. 14.7 Distrib 4.1.11, for redhat-linux-gnu (i386)
Connection id:          26
Current database:       testlink15
Current user:           bugz@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         4.1.11
Protocol version:       10
Connection:             Localhost via UNIX socket
Server character set:    latin1
DB     character set:    utf8
Client character set:    latin1
Conn.  character set:    latin1
UNIX socket:            /var/lib/mysql/mysql.sock
Uptime:                 36 min 55 sec
```

- Run the upgrade installation provided by TestLink 1.6.

Other resources:

what the heck is UTF-8 ?

http://www.joelonsoftware.com/articles/Unicode.html

octal table (you can see octal values 000 - 177 are "normal ASCII" characters).

The perl script that is provided searches based on octal values.

http://web.cs.mun.ca/~michael/c/ascii-table.html

description of tr Perl operation

http://www.unix.org.ua/orelly/perl/learn/ch15_05.htm

4.2.7. Keyword Management

If you don't want to create the same keyword multiple times for the same Test Project:

```
$g_allow_duplicate_keywords=FALSE;
```

5. Configuration

This chapter describes the most important configuration parameters. Additional information are together with parameters definition in configuration files. Localization and API configuration are described in extra chapters below.

*Note: TestLink 1.8 introduced a new class **$tlCfg** to hold overall configuration. Not all parameters are migrated yet.*

5.1. Configuration Files overview

All configuration parameters are inside the file config.inc.php and included files. For this release these are the configuration files:

- **config.inc.php -** Main configuration file and wrapper for other configuration files. The file lists default values of configuration parameters. This file is included into nearly each page. See below for more.
- **config_db.inc.php -** Contains configuration parameters to access the database. This file is created by the installer during the installation or upgrade process. Normally you don't need to change it manually.
- **custom_config.inc.php –** serves for modification of default values of parameters in config.inc.php. The benefit is that your modification is easy to copy during upgrade procedure.
- **/cfg/<bug_tracker>.cfg.php** – set access to database of a bug tracking tool.
- **/cfg/const.inc.php** – define constants and variables that are not supposed to modify.
- **/gui/templates/input_dimensions.conf** - Instead of hard coding attributes of html inputs, like ***maxlength*** and ***size***, we have code it into this file (there are exceptions for historical reasons).
- **/cfg/tl_fckeditor_config.js** – set-up fckeditor component configuration.

5.1.1. Use custom_config.inc.php for your changes!

Instead of making changes to **`config.inc.php`**, we suggest adding your changes to file: **`<testlink_root>/custom_config.inc.php`**. This allows you to save your configuration in the case of update.

Example:

To configure mail server settings, copy following lines from `config.inc.php` into **`custom_config.inc.php`**, and make changes according to your configuration.

```
# this is your custom configuration file custom_config.inc.php


$g_tl_admin_email = 'gandalf@teamst.org';
$g_from_email = 'testlink_system@teamst.org';
```

```
$g_return_path_email  = 'no_replay@teamst.org';
$g_smtp_host = '10.20.30.40';
```

5.2. Logging

TestLink has its own logging system with two possible output channels: files and database. Records in database are visible via GUI. You can use it for troubleshooting. A log file is created for each user. Configure the next parameters in custom_config.inc.php file:

LOG LEVEL

Set this to the default level of logging (NONE, ERROR, INFO, DEBUG, EXTENDED). Note that TestLink doesn't verify the size of created files. I.e. Use DEBUG level only for development or bug investigation to save disc place. ERROR level is recommended for production. It's default settings.

```
$tlCfg->log_level = 'ERROR';
```

LOGGING OUTPUT

There are two output channels: files and database. Both ways are up by default. Set the next parameter to false if you would like to disable one or both channels.

```
$g_loggerCfg = null;  // all loggers enabled (default)
$g_loggerCfg['db']['enabled'] = FALSE;       // true/false
$g_loggerCfg['file']['enabled'] = FALSE;     // true/false
```

LOGGING PATH

The path for the logging of TestLink. E.g. */tmp/* for Linux and *c:\temp* for winxp.

```
$tlCfg->log_path = TL_ABS_PATH . 'logs' . DS;
```

Note: Using debug level for your production server can generate large files. You can consider to create a script for periodic clean-up.

PHP environment logging has Error level by default. We want php errors to show up for users. You can modify it of course. See php.net site for more.

```
error_reporting(E_ALL);
```

SMARTY DEBUG WINDOW

Developers should use this parameter to show extra window with list of all parameters and values that are sent from php script to Smarty template component. The parameter must be false for production installation.

```
$tlCfg->smarty_debug = false;
```

5.3. Configuration of Bug Tracker

To enable this feature you need to change a configuration paramenter on the configuration file (custom_config.inc.php). The interface is disabled by default (value 'NO').

The available values are: 'NO', 'BUGZILLA', 'MANTIS', 'JIRA', 'TRACKPLUS', 'EVENTUM', 'SEAPINE' or 'TRAC'. For example:

```
$g_interface_bugs = 'MANTIS';
```

See system requirements chapter for supported versions. The particular BTS configuration file could be for example:

/cfg/bugzilla.cfg.php

/cfg/mantis.cfg.php

generally,

/cfg/<tracker_name>.cfg.php

Contains configuration parameters to access to particular issue tracking system. You need to edit this file if you want to access issue information from testlink (bugtracking system integration feature). See Appendix for an example of configuration.

5.4. Generated documents

The next strings are used in front page of printed document. Left blank to disable.

```
$tlCfg->document_generator->company_name = 'Your Company';
$tlCfg->document_generator->company_copyright = '2008 (c) TestLink Community';
$tlCfg->document_generator->confidential_msg = 'GPL';
```

Generated documents has own layout template. You can modify CSS template to you own:

```
$tlCfg->document_generator->css_template = $tlCfg->theme_dir .
'css/tl_documents.css';
```

Test case version could be included in a generated document together with Test case title:

```
$tlCfg->document_generator->tc_version_enabled = FALSE;
```

5.5. Send E-MAIL

TestLink has integrated mailing support for sending reports and notification. You must set-up the next values:

SMTP server delivers a generated email. The value "localhost" is enough in the most cases.

```
$g_smtp_host        = 'localhost';
```

Email address of administrator and sender are also mandatory parameters:

```
$g_tl_admin_email     = 'your.name@your_company.com'; # for problem/error
notification
$g_from_email         = 'no_replay@testlink.test_team';  # email sender (showed
to recipient)
$g_return_path_email  = 'your.name@your_company.com';
```

Important: set-up SMTP host and email addresses are mandatory configuration.

Optionally you can set priority of email. The value “not urgent” is default.

```
# Urgent = 1, Not Urgent = 5, Disable = 0
$g_mail_priority = 5;
```

Your SMTP server should requires login to relay emails. The values remains empty in the most of cases. Configure authentication:

```
$g_smtp_username    = '';
$g_smtp_password    = '';
```

5.6. User authentication

TestLink supports two kinds of authentication

- 'MD5' - use encrypted password stored on internal database
- 'LDAP' - use password from LDAP Server

Internal password is default:

```
$tlCfg->authentication['method']      = 'MD5';
```

LDAP authentication needs a few more parameters to be set:

```
$tlCfg->authentication['ldap_server']        = 'localhost';
$tlCfg->authentication['ldap_port']          = '389';
$tlCfg->authentication['ldap_version']       = '3';
$tlCfg->authentication['ldap_root_dn']       = 'dc=mycompany,dc=com';
$tlCfg->authentication['ldap_organization'] = '';    // e.g.
'(organizationname=*Traffic)'
$tlCfg->authentication['ldap_uid_field']     = 'uid';
$tlCfg->authentication['ldap_bind_dn']       = '';
$tlCfg->authentication['ldap_bind_passwd']   = '';
```

Check LDAP documentation to understand these settings. The current last LDAP protocol

version is 3, but some organizations could use version 2 (invalid version causes connection problem).

UID field will be used to identify user against a value added via login form. The default attribute name is 'uid'. For example: Active directory uses 'sAMAccountName'.

Parameters 'ldap_bind_dn' and 'ldap_bind_passwd' holds login information for LDAP access (Left empty if you LDAP server allows anonymous binding).

More about LDAP: http://en.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol

Note: TestLink API has its own kind of authentication via keys.

5.7. GUI Customization

5.7.1. Tree menu

Testlink supports several tree menu components. Possible values are 'EXTJS', 'LAYERSMENU', 'DTREE', 'JTREE'. ***EXTJS*** is default value (we recommend it). The component EXTJS has the best performance because it uses asynchronous communication.

```
$tlCfg->treemenu_type = 'EXTJS';
```

Any type of node (Test case, Test Suite) is added with ordering number "0", when creating an node in the tree. The initial display order will be by node id.

A parent test suite has the next default values to separate child Test Suites and Test Cases. These values must be >= 0.

```
$tlCfg->treemenu_default_testsuite_order = 1;
$tlCfg->treemenu_default_testcase_order = 100;
```

Show or hide Test case unique ID on tree menu:

```
$tlCfg->treemenu_show_testcase_id = TRUE;
```

Allow Test case counters by status on tree menu:

ENABLED -> enable counters [DEFAULT VALUE]

DISABLED -> disable

```
$tlCfg->exec_cfg->enable_tree_testcase_counters = ENABLED;
```

5.7.2. GUI Layout

We have defined theme directory, that includes CSS and image files
`<testlink_root>gui/themes/default/`.

You should copy the default directory, modify content and set the next parameter to point it:

```
$tlCfg->theme_dir = 'gui/themes/your_theme/';
```

You can change TestLink appearance by writing your own CSS (Cascading Style Sheet) files. The default files within theme directory:

- testlink.css (main style definition)
- tl_print.css (specific settings for printing of pages)
- tl_documents.css (used for generated documents; for example Test Specification)
- tl_treemenu.css (specific settings for tree menu)

These filenames are defined as constants in const.inc.php file. You could be modified it if you need to.

You can set own logo instead of the default TestLink image. You must copy your image file into `gui/themes/default/images/` directory (or within you layout theme) to allow it.

```
$tlCfg->company_logo = 'company_logo.png';
```

Note: This logo is used for both GUI and generated documents.

Login page could show the informational text in html format. The value is empty by default.

```
$tlCfg->login_info = '<p>Please, contact administrator <a
href="mailto:jack@caribic.sea">G. B. Shaw</a> if you have any question.</p>';
```

You can modify the filename of bullet image. Default schema includes arrow_org.gif and slide_gripper.gif (default).

```
$tlCfg->bullet_image = 'slide_gripper.gif';
```

To specify the Test Project background colour:

```
$tlCfg->gui->testproject_coloring = 'background';
```

Open the page edit Test project to specify particular colours. Default value is 'none' (no background colour change is allowed).

Default background colour is defined:

```
$tlCfg->gui->background_color = '#9BD';
```

Set-up the next two parameters to display name and surname instead of login only. The parameter *show_realname* could be

- TRUE - build a human readable display
- FALSE - show login name

```
$tlCfg->show_realname = FALSE;
```

Specify a format of displayed name:

```
$tlCfg->username_format = '%login%';
```

Examples:

'%first% %last%' -> John Cook

'%last%, %first%' -> Cook, John

'%first% %last% %login%' -> John Cook [ux555]

Configure the default navigator frame (frmWorkArea) width on left side of window:

```
$tlCfg->frame_workarea_default_width = "30%";
```

The Test project combo-box in top menu has configurable order (value must be SQL compliant)

```
$tlCfg->gui->tprojects_combo_order_by='ORDER BY nodes_hierarchy.id DESC';
```

Examples:

'ORDER BY name'

'ORDER_BY nodes_hierarchy.id DESC' -> similar effect to order last created first

Configure round percentages on the metrics Dashboard:

```
$tlCfg->dashboard_precision = 2;
```

5.7.3. Text area editor

Text data editing is solved via Javascript editor with toolbar over text area ('fckeditor' or 'tinymce') or simple text area ('none'). *FCKeditor* component is used by default as full featured component. This is rich featured component and could be easily enhanced via configuration to allow more features and enhancements.

You can define using another editor or just simple text without formatting:

```
// 'fckeditor'
// 'tinymce'
// 'none' -> use plan html textarea input field

$tlCfg->gui->webeditor='fckeditor';
```

FCKeditor Toolbar definition allows/disables icons in text area menu. We recommend to investigate it. The default testlink toolbar definition is 'tl_default', but you can switch to

"Default" or "Basic" (defined in core files).

```
$tlCfg->fckeditor_default_toolbar = "tl_default";
```

The customizable toolbar 'tl_default' is defined in <testlink_root>/cfg/tl_fckeditor_config.js file. You can modify the content of toolbar as well as other configuration parameters (for example templates, styles, spell checker, etc.). See fckeditor homepage for more information about this component.

5.7.4. Javascript

Use EXT JS library (GUI widgets) is default value (ENABLED). You can disable id (= DISABLED) to slightly improve performance.

```
$g_use_ext_js_library = ENABLED;
```

Define table sorting library. The default value 'kryogenix.org' uses Stuart Langridge sortTable. Empty string '' disables table sorting feature.

```
$g_sort_table_engine='kryogenix.org';
```

5.7.5. Using Your own Smarty templates (GUI definition)

If You want to test a different solution for the user interface, you can develop your own Smarty Templates. At the time of this writting we have defined the following configuration array: **`$g_tpl`** with the following entries:

- $g_tpl['tcView']
- $g_tpl['tcSearchView']
- $g_tpl['tcEdit']
- $g_tpl['tcNew']
- $g_tpl['execSetResults']

This allows you to create templates with different names than the original TestLink, without the risk of overwriting them, during the next upgrade.

Note: Not all TestLink pages are ready for this kind of configuration.

The standard configuration:

```
$g_tpl['tcView'] = "tcView.tpl";
$g_tpl['tcSearchView'] = "tcSearchView.tpl";
$g_tpl['tcEdit'] = "tcEdit.tpl";
$g_tpl['tcNew'] = "tcNew.tpl";
$g_tpl['execSetResults'] = "execSetResults.tpl";
```

5.8. Test execution settings

Allow XML-RPC calls to external test automation server (the special buttons will be displayed on execution pages).

ENABLED -> enable XML-RPC calls

DISABLED -> disable

```
$tlCfg->exec_cfg->enable_test_automation = DISABLED;
```

Different layout for the attachments management on execution page (these variables are predefined in const.inc.php):

$att_model_m1 -> shows upload button and title

$att_model_m2 -> hides upload button and title [DEFAULT VALUE]

```
$tlCfg->exec_cfg->att_model = $att_model_m2;
```

Availability to delete execution result by an user:

ENABLED -> User can delete an execution result

DISABLED -> User can not. [DEFAULT VALUE]

```
$tlCfg->exec_cfg->can_delete_execution = DISABLED;
```

5.8.1. Execution history

Define order of execution history:

ASC -> Ascending (last execution at bottom)

DESC -> Descending (last execution on top) [DEFAULT VALUE]

```
$tlCfg->exec_cfg->history_order = 'DESC';
```

Define if whole execution history for the chosen build will be shown in execution window:

TRUE -> the whole execution history for the build will be shown

FALSE -> just last execution will be shown [DEFAULT VALUE]

```
$tlCfg->exec_cfg->history_on = FALSE;
```

Allow to show a results also for previous builds.

TRUE -> test case VERY LAST (i.e. in any build) execution status will be displayed

FALSE -> only last result on current build. [DEFAULT VALUE]

```
$tlCfg->exec_cfg->show_last_exec_any_build = FALSE;
```

Allow displaying of all builds execution history.

TRUE -> History for all builds will be shown

FALSE -> Only history of the current build will be shown [DEFAULT VALUE]

```
$tlCfg->exec_cfg->show_history_all_builds = FALSE;
```

5.8.2. Test execution navigator

Show test cases and test case counters coloured according to test case status:

ENABLED -> coloured test status [DEFAULT VALUE]

DISABLED -> disable

```
$tlCfg->exec_cfg->enable_tree_colouring = ENABLED;
```

Controls what happens in right frame when user clicks on a Test suite on tree menu. Disabling this setting can help to avoid performance problems.

ENABLED -> show all test cases presents on test suite and children test suite (old behaviour).

DISABLED -> nothing happens, to execute a test case you need to click on test case [DEFAULT VALUE]

```
$tlCfg->exec_cfg->show_testsuite_contents = DISABLED;
```

TBD: applicable for which tree menu component?

Allow to edit execution notes, on old executions (Attention: user must have test case execution right)

ENABLED -> user can edit execution notes, on old executions (Attention: user must have test case execution right)

DISABLED -> no edit allowed [DEFAULT VALUE]

```
$tlCfg->exec_cfg->edit_notes = DISABLED;
```

Filter Test cases a user with tester role can VIEW depending on test execution assignment.

'all' -> all test cases.

'assigned_to_me' -> test cases assigned to logged user. [DEFAULT VALUE]

'assigned_to_me_or_free' -> test cases assigned to logged user or not assigned.

```
$tlCfg->exec_cfg->view_mode->tester='assigned_to_me';
```

Filter Test cases a user with tester role can EXECUTE depending on test execution assignment.

'all' -> all test cases.

'assigned_to_me' -> test cases assigned to the current user. [DEFAULT VALUE]

'assigned_to_me_or_free' -> test cases assigned to logged user or not assigned

```
$tlCfg->exec_cfg->exec_mode->tester='assigned_to_me';
```

User filter in Test Execution navigator:

'logged_user' -> combo will be set to the current user

'none' -> no filter applied by default [DEFAULT VALUE]

```
$tlCfg->exec_cfg->user_filter_default='none';
```

5.8.3. Add a new type of Test results on execution page

You will need to work on the following files (all paths are relative to installation directory):

- `custom_config.inc.php` <-- create it if do not exist yet
- `locale/en_GB/custom_strings.txt` <-- create it instead of editing strings.txt
- `gui/themes/<your_theme>/css/testlink.css`

1. Open `cfg/const.inc.php` and search for: `$tlCfg->results['status_code']`
2. Copy following lines into `custom_config.inc.php`:

```
$tlCfg->results['status_code'] = array (
     "failed"        => 'f',
     "blocked"       => 'b',
     "passed"        => 'p',
     "not_run"       => 'n',
     "not_available" => 'x',
     "unknown"       => 'u',
     "all"           => 'all'
);

$tlCfg->results['status_label'] = array(
     "all"      => "test_status_all_status",
```

```
     "not_run"  => "test_status_not_run",
     "passed"   => "test_status_passed",
     "failed"   => "test_status_failed",
     "blocked"  => "test_status_blocked",
     "not_available" => "test_status_not_available",
     "unknown"       => "test_status_unknown"
);

$tlCfg->results['status_label_for_exec_ui'] = array(
     "passed"  => "test_status_passed",
     "failed"  => "test_status_failed",
     "blocked" => "test_status_blocked"
);

$tlCfg->results['default_status'] = "passed";
```

3. Add new statuses and save:

   tcstatus_1 -> code q

   tcstatus_2 -> code w

4. custom_config.inc.php will be:

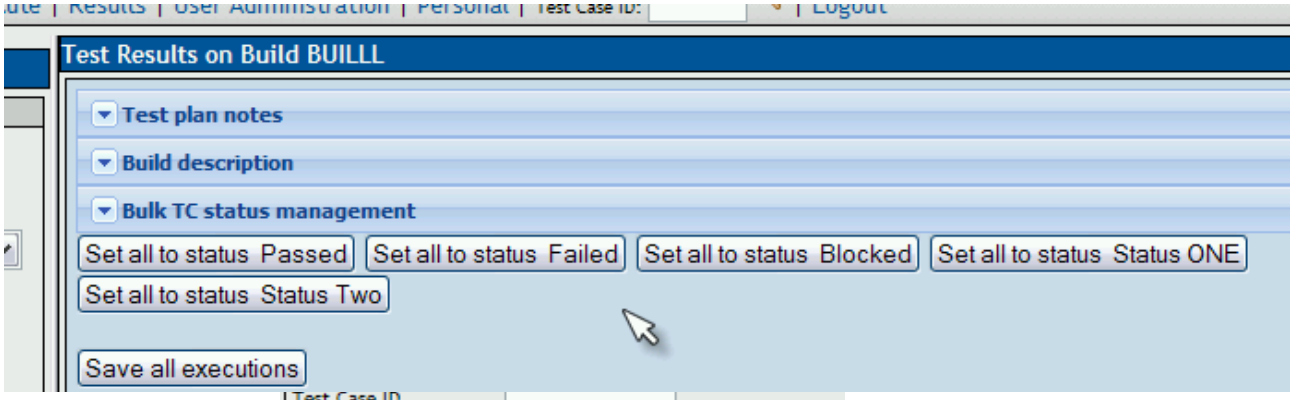
```
$tlCfg->results['status_code'] = array (
     "failed"        => 'f',
     "blocked"       => 'b',
     "passed"        => 'p',
     "not_run"       => 'n',
     "not_available" => 'x',
     "unknown"       => 'u',
     "all"           => 'all',
     "tcstatus_1"    => 'q',
     "tcstatus_2"    => 'w'
);

$tlCfg->results['status_label'] = array(
     "all"      => "test_status_all_status",
     "not_run"  => "test_status_not_run",
     "passed"   => "test_status_passed",
     "failed"   => "test_status_failed",
     "blocked"  => "test_status_blocked",
     "not_available" => "test_status_not_available",
     "unknown"       => "test_status_unknown",
     "tcstatus_1" => "test_status_new_one",
```

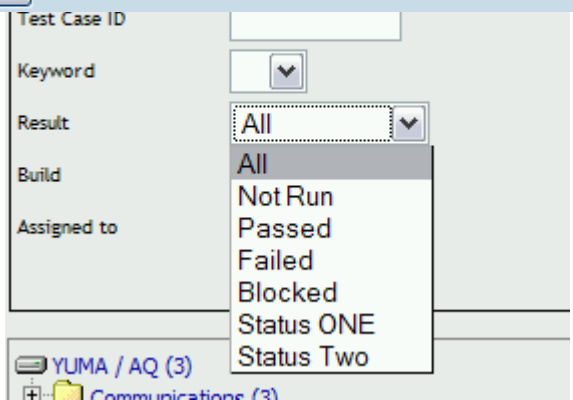
```
      "tcstatus_2" => "test_status_new_two"
);

$tlCfg->results['status_label_for_exec_ui'] = array(
      "passed"  => "test_status_passed",
      "failed"  => "test_status_failed",
      "blocked" => "test_status_blocked",
      "tcstatus_1" => "test_status_new_one",
      "tcstatus_2" => "test_status_new_two"
);

$tlCfg->results['default_status'] = "blocked";
```

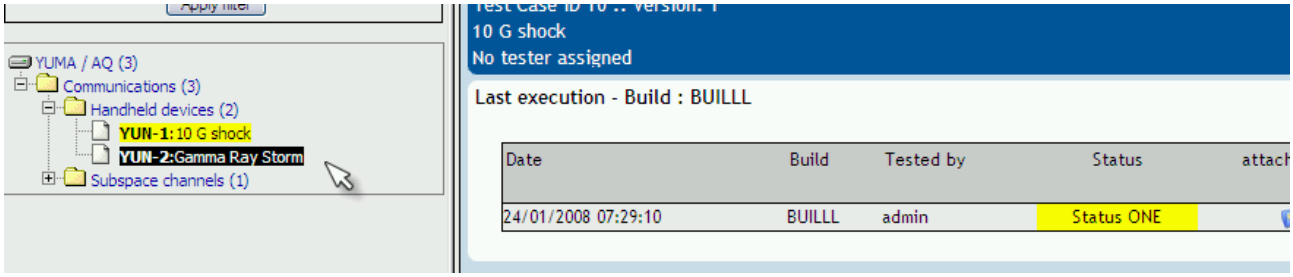
5. Modify css if you want new colours.

```
.tcstatus_1, div.tcstatus_1 {
      color:              black;
      background:         yellow;
}
.tcstatus_2, div.tcstatus_2 {
      color:                    black;
      background:         orange;
}
div.tcstatus_1, div.tcstatus_2 {
      margin:             8px;
      padding:            6px;
      text-align: center;
}
```

Effect at user interface level will be:







5.9. Test Specification

Layout of 'steps' and 'expected result' text area:

'horizontal' -> step and results on the same row

'vertical' -> steps on one row, results in the row bellow [DEFAULT VALUE]

```
$g_spec_cfg->steps_results_layout = 'vertical';
```

Availability of test suite filter in the test specification navigator:

ENABLED -> User will see a test suite filter [DEFAULT VALUE]

DISABLED -> no filter available

```
$g_spec_cfg->show_tsuite_filter = ENABLED;
```

Refreshing of Test specification navigator:

ENABLED -> every time user do an editing operation on test specification tree is updated

DISABLED -> tree will not be updated, user can update it manually.

```
$g_spec_cfg->automatic_tree_refresh = ENABLED;
```

Allow to edit an executed version of Test case.

ENABLED -> user can edit executed Test case versions

DISABLED -> editing of executed Test case versions is blocked. [DEFAULT VALUE]

```
$tlCfg->testcase_cfg->can_edit_executed = DISABLED;
```

5.9.1. Test specification templates

User can define the initial content of three text objects of a new Test case: summary, steps

and expected results. Each object configuration includes type and value. The next types are possible:

- 'none' -> template will not be used, default will be a empty text area. [DEFAULT VALUE]
- 'string' -> value of value member is assigned to FCK object
- 'string_id' -> value member is used in a lang_get() call, and return value is assigned to FCK object. Configure string_id on custom_strings.txt
- 'file' -> value member is used as file name. The file is read and it's contents assigned to a text area component as input

```
$g_testcase_template->summary->type = 'string';
$g_testcase_template->summary->value = '<p>Objective: TBD</p><p>Precondition:
N/A</p>';


$g_testcase_template->steps->type = 'none';
$g_testcase_template->steps->value = '';


$g_testcase_template->expected_results->type = 'none';
$g_testcase_template->expected_results->value = '';
```

User can define the initial content of a new Test Suite description similar way:

```
$g_testsuite_template->details->type='file';
$g_testsuite_template->details->value='D:\w3\tl\head_20080103\logs\tsuite.txt';
```

Note: wrong value for type results in no data assigned to Web Editor object.

5.10. Attachments

Attachment feature could be enabled (TRUE) /disabled (FALSE):

```
$g_attachments->enabled = TRUE;
```

The type of the repository can be database or file system:

- TL_REPOSITORY_TYPE_DB => database
- TL_REPOSITORY_TYPE_FS => filesystem

```
$g_repositoryType = TL_REPOSITORY_TYPE_FS;
```

TL_REPOSITORY_TYPE_FS: where the filesystem repository should be located

```
$g_repositoryPath = TL_ABS_PATH . "upload_area" . DS;
```

Security: We recommend to change the directory for security reason

Compression used within the repository

- TL_REPOSITORY_COMPRESSIONTYPE_NONE => no compression
- TL_REPOSITORY_COMPRESSIONTYPE_GZIP => gzip compression

```
$g_repositoryCompressionType = TL_REPOSITORY_COMPRESSIONTYPE_NONE;
```

The maximum allowed file size for each repository entry, default 1MB.

```
$tlCfg->repository_max_filesize = 1;
```

Also check your PHP settings (default is usually 2MBs)

Users should add a title for the attachment. You can leave it empty (FALSE). Default is TRUE. The actions for validation (TRUE):

- 'none' - just write on db an empty title
- 'use_filename' - use filename as title

```
$g_attachments->allow_empty_title = TRUE;
$g_attachments->action_on_save_empty_title = 'none';
```

Title is used as link description for download if title is empty:

- 'show_icon' -> the $g_attachments->access_icon will be used.
- 'show_label' -> the value of $g_attachments->access_string will be used .

```
$g_attachments->action_on_display_empty_title='show_icon';
$g_attachments->access_icon='<img src="' . TL_THEME_IMG_DIR . '/new_f2_16.png"
style="border:none">';
$g_attachments->access_string="[*]";
```

You can set own display order of uploaded files.

```
$g_attachments->order_by=" ORDER BY date_added DESC ";
```

5.11. Requirements support

Requirement functionality could be enabled / disabled per Test Project level (not via TL configuration). Navigate to the Edit Test project page as Administrator. One of the unique features of TestLink is Requirement Management.

Requirement identification string (req_doc_id) must be unique:

- TRUE -> identification is UNIQUE IN THE WHOLE DB (system_wide)
- FALSE -> identification is UNIQUE INSIDE a SRS

```
$g_req_cfg->reqdoc_id->is_system_wide=false;
```

5.11.1. Generated Test Cases from Requirements

You can choose to create test cases for every requirement after creating the Software Requirements Specifications (SRS), and populating it with requirements. A specific Test Suite is created for the purpose. You can define that the related SRS title is used (TRUE):

FALSE -> test cases are created and assigned to a test suite with name defined via `$g_req_cfg->default_testsuite_name`

TRUE -> Requirement Specification Title is used as testsuite name

```
$g_req_cfg->use_req_spec_as_testsuite_name = TRUE;
```

The next test suite title is used if you set the previous parameter as FALSE :

```
$g_req_cfg->default_testsuite_name = "Auto-created Test cases";
```

Two additional parameters (values should be html or simple text):

```
$g_req_cfg->testsuite_details = "Test Cases in the Test Suite are generated from
Requirements. A refinement of test scenario is highly recommended.";

$g_req_cfg->testcase_summary_prefix = "<b>The Test Case was generated from the
assigned requirement.</b><br />";
```

5.12. Configuration of misc functionality

5.12.1. Data import limits

Web servers have defined a maximum upload file size. PHP allows to clarify this limit and testlink uses default 204800 bytes. You could increase this value if you import a bigger file. There is also parameter limiting maximal size of one line of exported file. The value 10000 characters should be enough.

```
$tlCfg->import_max_size = '204800';

$tlCfg->import_max_row = '10000';
```

Note: attachment repository has an extra constraints.

5.12.2. Default user role

Set the default role used for new users. This values is used for users

- created from the login page.
- offered by default when using user management.
- when their original role definition is deleting from TestLink.

```
$tlCfg->default_roleid = TL_ROLES_GUEST;
```

Possible values: `TL_ROLES_TESTER`, `TL_ROLES_GUEST`, `TL_ROLES_NO_RIGHTS` and similar constants defined by you. See `const.inc.php` for more.

5.12.3. Title duplicity of Test Projects, Test Suites and Test Cases

It is possible to create one of these objects (Test Projects, Test Suites and Test Cases) doing a copy of an existing one.

The following checks will be done:

1. *Test Project* name is unique (TODO: must be unique everytime)
2. *Test Suite* Name inside *Test Project* is unique
3. *Test Case* Name inside Test Suite is unique

Note: Name of a Keyword, a requirements document, a Test Plan and Requirements identifier must be unique within Test Project.

You can configure how to proceed when the copy is done. The options are:

- `'generate_new'` : generate a new name using the value of `$g_prefix_name_for_copy` and the original object name. The prefix include timestamp by default.
- `'block'` : return with an error .
- `'allow_repeat'` : allow the name to be repeated (backward compatibility with version 1.0.4 and 1.5.x)

Example of formatting:

```
$tlCfg->name_duplicity_checking = 'generate_new';
```

Note: Obsolete parameter `$g_check_names_for_duplicates = FALSE` could disable the functionalty.

5.12.4. String checking and conversions

Allow automatic convertion of www URLs and email adresses into clickable links used by function string_display_links() for example by custom fields.

Valid values are ENABLED/DISABLED.

```
$tlCfg->html_make_links = ENABLED;
```

Define the valid html tags for "content driven" single-line and multi-line fields. Do NOT include tags with parameters (eg. <font face="arial">), <IMG> and <A HREF>. It's used by custom fields functionality for example.

```
$tlCfg->html_valid_tags = 'p, li, ul, ol, br, pre, i, b, u, em';
$tlCfg->html_valid_tags_single_line = 'i, b, u, em';
```

6. Localization

TestLink supports localization of text, date and time. There is a default value in configuration, but each user can set own language. Language code is according to common standards.

```
$g_default_language = 'en_GB';
```

6.1. String localization

A directory exists for every localization, with a standard **`strings.txt`** file inside.

```
<TL_INSTALL_DIR>/locale/de_DE/strings.txt
<TL_INSTALL_DIR>/locale/de_DE/custom_strings.txt
<TL_INSTALL_DIR>/locale/en_GB/strings.txt
...
```

To change some of the original translations without changing those provided with the original file, you can use **custom_strings.txt.** You need to place this file in the corresponding localization directory, and use the same format and rules used in the original `strings.txt`. You can redefine a value present on `strings.txt`, without need of commenting it in the original file.

Instruction and help pages have their own location: `<testlink_root>/gui/help/<language>`.

6.1.1. Date and Time Localization

For every defined locale, you can set the format for date and time presentation. This is configured using the following associative arrays: `$g_locales_date_format` and `$g_locales_timestamp_format`.

At time of this writting the configuration is :

```
$g_locales_date_format = array(
               'en_GB' => "%d/%m/%Y", 'it_IT' => "%d/%m/%Y",
               'es_AR' => "%d/%m/%Y", 'es_ES' => "%d/%m/%Y",
               'de_DE' => "%d.%m.%Y", 'fr_FR' => "%d/%m/%Y",
               'pt_BR' => "%d/%m/%Y" );
$g_locales_timestamp_format = array(
               'en_GB' => "%d/%m/%Y %H:%M:%S",
               'it_IT' => "%d/%m/%Y %H:%M:%S",
               'es_AR' => "%d/%m/%Y %H:%M:%S",
               'es_ES' => "%d/%m/%Y %H:%M:%S",
               'de_DE' => "%d.%m.%Y %H:%M:%S",
               'fr_FR' => "%d/%m/%Y %H:%M:%S",
               'pt_BR' => "%d/%m/%Y %H:%M:%S", );
```

If there is no entry in the previous arrays, the value of the following configuration variables will be used: `$g_date_format` and `$g_timestamp_format`.

Example of formatting:

```
$g_date_format ="%d/%m/%Y";
$g_timestamp_format = "%d/%m/%Y %H:%M:%S";
```

6.1.2. Charset

TestLink supports UTF-8 characters by default. The charset value is used for both data (database) and GUI (Smarty templates).

```
$tlCfg->charset =  'UTF-8'
```

We strongly recommend using unicode character set (UTF-8). ISO-8859-1 can be configured for backward compatibility. Advanced users also can use another set with their own localization.

MySQL-Versions prior to 4.1 have not utf-8 support. You can export data and convert exported file into unicode for database upgrade. Note, that MySQL 5 is reguired to migration 1.6->1.7 process.

6.1.3. GUI special characters

Separation characters used to surround some texts in the user interface (for example user role):

```
$tlCfg->gui->role_separator_open =  '[';
$tlCfg->gui->role_separator_close = ']';
```

Title separators are used when composing a title using several strings. The first one is used preferably for logic relation "object : name" (for example: *Test Plan : MyTestLink 1.0*). The second separator is used to separate "parent – child".

```
$tlCfg->gui_title_separator_1 = ' : ';
$tlCfg->gui_title_separator_2 = ' - ';
```

Each Test case has a unique identification number. You can set-up and use "external ID" as identifier instead of it. The "external ID" is composed from a Test project prefix, a separator defined below and a number related to a parent Test Suite. Define the separator:

```
$tlCfg->testcase_cfg->glue_character = '-';
```

Note: The value cannot be empty.

7. TestLink API

SOAP API is disabled by default (for security reason). Set then next parameter to TRUE if you would like to use it.

```
$tlCfg->api_enabled = FALSE;
```

Format of showing the personal API identification within GUI (View pages).

```
$tlCfg->api_id_format    = "[ID: %s ]";
```

8. FAQ

Please also check the TestLink forum.

Smarty error is shown instead of login page.

```
Smarty::include(C:\Inetpub\wwwroot\testlink\gui\templates_c\\%
%6A^6A5^6A537DD8%%login.tpl.php) [function.Smarty-include]: failed to open
stream: No such file or directory in
C:\Inetpub\wwwroot\testlink\third_party\smarty\Smarty.class.php on line
1247
```

Linux/unix users: Verify if write permissions are for temp directory (default: `<testlink_root>/gui/template_c/`). Fix by command

```
# chmod a+w <testlink_root>/gui/template_c
```

IIS users: Give the iis_user write access to the template_c directory.

Does TestLink support Secured HTTPS connection?

Yes, it's settings of your web server.

lang_api.php Error is shown instead of login page.

```
[Fri Nov 02] [error] [client xxx.xxx.xxx.xxx] PHP Fatal error: Call to
undefined function iconv() in /home/qa/site/lib/functions/lang_api.php on
line 54
```

Note to Windows® Users:
In order to enable this module on a Windows® environment, you need to put a DLL file named iconv.dll or iconv-1.3.dll (prior to 4.2.1) which is bundled with the PHP/Win32 binary package into a directory specified by the PATH environment variable or one of the system directories of your Windows® installation.

This module is part of PHP as of PHP 5 thus iconv.dll and php_iconv.dll is not needed any more.

How to upload images into text?

http://www.teamst.org/index.php?option=com_content&task=view&id=43&Itemid=2

Allocated memory problem

You can receive:

```
Fatal error: Allowed memory size of 8388608 bytes exhausted (tried to allocate
1328642 bytes)
```

For "tree menu": The fast solution is to use the simplest tree menu component "jtree" instead of default "phplayermenu". The component phplayersmenu is resource expensive. Add into

custom_config.inc.php:

```
$g_tree_type='JTREE';
```

You can also safely enhance the limit in `php.ini`.

We are aware, that PHP is not sufficient environment for difficult application. So we do refactorization continuously to save sources

I upgraded from older version and I cannot login.

Your original database could be in different charset. The default from 1.6 version is UTF-8. Try to switch DB_SUPPORTS_UTF8 to FALSE in config.inc.php.

I cannot use functionality export (XML) a test case and open a file attachment (word or excel).

Check session.save_path variable in php.ini. You need to put some directory with file write permission, for example:

```
session.save_path = c:\php5
```

This is in case you are using windows, linux would probably be:

```
session.save_path = /usr/php/sessions
```

Either way you must have write permissions on those folders.

Appendix A: Recommended configuration parameters

$g_smtp_host

$g_tl_admin_email

$g_from_email

$g_return_path_email

$g_interface_bugs

$tlCfg->document_generator->company_name

$tlCfg->document_generator->company_copyright

$tlCfg->document_generator->confidential_msg

$tlCfg->company_logo

$tlCfg->fckeditor_default_toolbar

$g_default_language

$tlCfg->api_enabled

Appendix B: Set-up Mantis bug-tracking system integration

B.I Overview

The integration between TestLink and a Bug Tracking System (BTS) has the following characteristics:

- All communication between Test Link and the BTS is done through database tables.
- TestLink (at the time of this writing) is neither able to send data to the BTS, either able to receive data from the BTS, in the traditional model of function call.

After all the configuration is up and running, from a TestLink user point of view the process will be:

1. While executing a test, it fails.
2. User saves execution result.
3. User clicks on link that opens BTS web page used for issue reporting.
4. After issue reporting, user has to take note of issue ID assigned by BTS, to input it into TestLink.
5. User returns to TestLink test execution page, and writes the issue ID in the bug input.
6. After user saves the execution, TestLink will display data taken from the BTS database.

B.II Mantis DB Configuration

Edit file **<your TestLink main directory>/cfg/mantis.cfg.php**.

Environment example: TestLink and Mantis installed on the same web server

Mantis URL	http://calypso/mantis
Test Link URL	http://calypso/testlink
Mantis Database name	mantis_bt
MySQL user/password to access Mantis DB	mantis_bt_user/mantis_bt_password

Anonymous login into mantis has to be turned on. A mantis user with viewer rights to all public projects, must be created. (anonymous account). Change/add following lines in your mantis config_inc.php (replace **dummy** with the anonymous account you will use)

```
# --- anonymous login -----------
# Allow anonymous login
$g_allow_anonymous_login = ON;
$g_anonymous_account = 'dummy';
```

B.III Enable BTS integration

Check the following lines from **config.inc.php** .

```
// ----------------------------------------------------------------------------
/** [Bug Tracking systems] */
/**
* TestLink uses bug tracking systems to check if displayed bugs resolved,
verified,
* and closed bugs. If they are it will strike through them
*
* NO        : no bug tracking system integration
* BUGZILLA  : edit configuration in TL_ABS_PATH/cfg/bugzilla.cfg.php
* MANTIS    : edit configuration in TL_ABS_PATH/cfg/mantis.cfg.php
...
*/
$g_interface_bugs='NO';
```

Copy it to **custom_config.inc.php and** change line:

```
$g_interface_bugs='NO';
```

Final result:

```
$g_interface_bugs='MANTIS';
```

B.IV Check interface

After your configuration is OK, you will find the icon to add bugs in the execute screen.

Several checks are done when you try to add the bug:

- Bug ID is present on BTS ?
- Bug ID format is valid ?

Appendix C: Revision History

#	Description	Date	Author
1.0	Initial creation of the document in DocXML	2005/03/12	A. Morsing
1.1	Corrected title, updated structure and added new sections.	2005/04/12	M. Havlat
1.2	Added some words for MySQL 4.1, UTF8 support	2005/06/27	A. Morsing
1.3	Updated automatic installation part	2005/09/12	F. Mancardi
1.4	Updated for TL 1.6.; added configuration parameters; restructured (created pre-installation steps section); corrected layout; added phpMyAdmin steps description	2005/09/13	M. Havlat
2.0	Converted to OO2 format; added DB Charset update explanation from Kevin	2005/12/04	M. Havlat
2.1	Corrected layout for export to HTML and PDF	2005/12/11	M. Havlat
2.2	Some small changes	2005/12/17	A. Morsing
2.3	Minor layout and grammar update	2006/02/14	M. Havlat
2.4	Updated for TL 1.7	2006/11/17	M. Havlát
2.5	Updated for TL 1.7; restructured; merged BTS case; layout update (prepare for 1,7,0 release)	2007/09/13	M. Havlát
2.6	Added several new parameters for 1.7, updated styles, configuration divided into logical chapters	2008/01/02	M. Havlát
2.7	Fixed: 0001347 , 0001284 , 0001331 New sections: Add a new type of Test results on execution page (drafted by Francisco), Define HTML text editor	2008/02/02	M. Havlát
2.8	Updated for TL 1.8 (include all new and missing features and refactorization of configuration files).	2008/06/06	M. Havlát
2.9	Language correction	2008/07/04	W. Pollans
2.10	Layout update, minor changes	2008/07/07	M. Havlát