INTRODUCTION
For various reasons your TYPO3 installation may over time accumulate data with integrity problems or data you wish to delete completely.
For instance, why keep old versions of published content? Keep that in your backup - don't load your running website with that overhead!
Or what about deleted records? Why not flush them - they also fill up your database and filesystem and most likely you can rely on your backups in case of an emergency recovery?
Also, relations between records and files inside TYPO3 may be lost over time for various reasons. If your website runs as it should such "integrity problems" are mostly easy to automatically repair by simply removing the references pointing to a missing record or file.
However, it might also be "soft references" from eg. typolinks (<link 123>...</link>) or a file references in a TypoScript template (something.file = fileadmin/template/miss_me.jpg) which are missing. Those cannot be automatically repaired but the cleanup script incorporates warnings that will tell you about these problems if they exist and you can manually fix them.
This script provides solutions to these problems by offering an array of tools that can analyze your TYPO3 installation for various problems and in some cases offer fixes for them. Also third party extensions can plug additional functionality into the script.

PREPARATIONS:
THERE IS ABSOLUTELY NO WARRANTY associated with this script! It is completely on your OWN RISK that you run it. It may cause accidential data loss due to software bugs or circumstances that it does not know about yet - or data loss might happen due to misuse!

ALWAYS make a complete backup of your website! That means:
* Dump the complete database to an SQL file. This can usually be done from the command line like this:
    mysqldump [database name] -u [database user] -p --add-drop-table > ./mywebsite.sql
* Save all files in the webroot of your site. I usually do this from the command line like this:
    tar czf ./mywebsite.tgz [webroot directory of your site]

Before running with the --AUTOFIX option ALWAYS make sure to add the parameter "--dryrun" to see what would be fixed.

Also, NEVER BYPASS the REFERENCE INDEX CHECK if --AUTOFIX is used for those tools which require a clean reference index.

It could be a good idea to run a myisamchk on your database just to make sure MySQL has everything pulled together right. Something like this will do:
    myisamchk [path_to_mysql_databases]/[database_name]/*.MYI -s -r

RUNNING the SCRIPT:
The "[base command]" is:
    [typo3_site_directory]/typo3/cli_dispatch.phpsh lowlevel_cleaner

Try this first. If it all works out you should see a help-screen. Otherwise there will be instructions about what to do. For instance, you will have to create a backend user, "_cli_lowlevel", with any

random password since you never need to log in with the user. Never mind permissions, they are not important since this script will force the user to run as "admin" in "Live" workspace.
You can use the script entirely by following the help screens. However, through this document you get some idea about the best order of events since they may affect each other.

For each of the tools in the test you can see a help screen by running:
	[base command] [toolkey]

Example with the tool "orphan_records":
	[typo3_site_directory]/typo3/cli_dispatch.phpsh lowlevel_cleaner orphan_records

SUGGESTED ORDER OF CLEAN UP:
The suggested order below assumes that you are interested in running all these tests. Maybe you are not! So you should check the description of each one and if there is any of the tests you wish not to run, just leave it out. It kind of gets simpler that way since the complexity mostly is when you wish to run all tests successively in which case there is an optimal order that ensures you don't have to run the tests all over again.

[base command] orphan_records -r --AUTOFIX
	- As a beginning, get all orphaned records out of the system since you probably want to. Since orphan records may keep some missing relations from being detected it's a good idea to get them out immediately.

[base command] versions -r --AUTOFIX
	- Flush all published versions now if you like. Published versions may also keep references to records which could affect other tests, hence do it now if you want to.

[base command] tx_templavoila_unusedce -r --AUTOFIX
	- (Assumes usage of "TemplaVoila" extension!)
	- This should be done AFTER flushing published versions (since versions could reference elements that might be safe to remove)
	- This should be done BEFORE flushing deleted versions (since this tool will create new deleted records), given that you want to completely flush them of course.
	- You should run it over again until there remains no more unused elements. You need to do this because deleting elements might generate new unused elements if the now-deleted elements had references.

[base command] double_files -r --AUTOFIX
	- Fix any files referenced twice or more before you delete records (which could potentially delete a file that is referenced by another file).

[base command] deleted -r --AUTOFIX
	- Flush deleted records. As a rule of thumb, tools that create deleted records should be run before this one so the deleted records they create are also flushed (if you like to of course)

[base command] missing_relations -r --AUTOFIX
	- Remove missing relations at this point.
	- If you get an error like this; "t3lib_refindex::setReferenceValue(): ERROR: No reference

record with hash="132ddb399c0b15593f0d95a58159439f" was found!" just run the test again until no errors occur. The reason is that another fixed reference in the same record and field changed the reference index hash. Running the test again will find the new hash string which will then work for you.

[base command] cleanflexform -r --AUTOFIX
- After the "deleted" tool since we cannot clean-up deleted records and to make sure nothing unimportant is cleaned up

[base command] rte_images -r --AUTOFIX
- Will be affected by flushed deleted records, versions and orphans so must be run after any of those tests.

EXECUTED ANYTIME:
These can be executed anytime, however you should wait till all deleted records and versions are flushed so you don't waste system resources on fixing deleted records.

[base command] missing_files -r --AUTOFIX
[base command] lost_files -r --AUTOFIX

NIGHTLY REPORTS OF PROBLEMS IN THE SYSTEM:
If you wish to scan your TYPO3 installations for problems with a cronjob or so, a shell script that outputs a report could look like this:

```
#!/bin/sh
/[WEBROOT_ABS_PATH]/typo3/dummy_4.0/typo3/cli_dispatch.phpsh lowlevel_cleaner orphan_records -r -v 2 -s
/[WEBROOT_ABS_PATH]/typo3/dummy_4.0/typo3/cli_dispatch.phpsh lowlevel_cleaner versions -r -v 2 -s
/[WEBROOT_ABS_PATH]/typo3/dummy_4.0/typo3/cli_dispatch.phpsh lowlevel_cleaner tx_templavoila_unusedce -r --refindex update -v 2 -s
/[WEBROOT_ABS_PATH]/typo3/dummy_4.0/typo3/cli_dispatch.phpsh lowlevel_cleaner double_files -r --refindex update -v 2 -s
/[WEBROOT_ABS_PATH]/typo3/dummy_4.0/typo3/cli_dispatch.phpsh lowlevel_cleaner deleted -r -v 1 -s
/[WEBROOT_ABS_PATH]/typo3/dummy_4.0/typo3/cli_dispatch.phpsh lowlevel_cleaner missing_relations -r --refindex update -v 2 -s
/[WEBROOT_ABS_PATH]/typo3/dummy_4.0/typo3/cli_dispatch.phpsh lowlevel_cleaner cleanflexform -r -v 2 -s
/[WEBROOT_ABS_PATH]/typo3/dummy_4.0/typo3/cli_dispatch.phpsh lowlevel_cleaner rte_images -r --refindex update -v 2 -s
/[WEBROOT_ABS_PATH]/typo3/dummy_4.0/typo3/cli_dispatch.phpsh lowlevel_cleaner missing_files -r --refindex update -v 2 -s
/[WEBROOT_ABS_PATH]/typo3/dummy_4.0/typo3/cli_dispatch.phpsh lowlevel_cleaner lost_files -r --refindex update -v 2 -s
```

You may wish to set the verbosity level (-v) to "3" instead of "2" as in the case above, depending on how important you consider the warnings.
You might also wish to disable tests like "deleted" which would report deleted records - something that might not warrant a warning, frankly speaking...
If you append "--AUTOFIX --YES" to each test it will actually perform clean up operations after checking, however it is NOT RECOMMENDED to do that as a nightly cron-job! In addition you should study what repair operations each test does to your system before using it!

ADDING YOUR OWN TOOLS TO THE TEST:
You can plug additional analysis tools into the cleaner script. All you need to do is create a class with a few specific functions and configure the cleaner to use it. You should encapsulate your class in an extension (as always).
In the steps below, substitute these strings with corresponding values:
- YOUREXTKEYNOUS = Your extension key, no underscores!
- YOUREXTKEY = Your full extension key
- CLEANERTOOL = Name prefix for your cleaner module

STEP1: Set up your class as a tool for the cleaner:
- In the "ext_localconf.php" file of your extension, add this:

```
$TYPO3_CONF_VARS['EXTCONF']['lowlevel']['cleanerModules']
['tx_YOUREXTKEYNOUS_CLEANERTOOL'] =


array('EXT:YOUREXTKEY/class.YOUREXTKEYNOUS_CLEANERTOOL.php:tx_YOUREXTKEYNOUS_CLEANERTOO
L');
```

- In your extension, create this PHP file:
YOUREXTKEY/class.YOUREXTKEYNOUS_CLEANERTOOL.php

- Finally, make sure to "Clear cache in typo3conf/" after having done this!

STEP2: Build your cleaner class:
- In the new PHP file, create a class with these basic functions:

```
class YOUREXTKEYNOUS_CLEANERTOOL extends tx_lowlevel_cleaner_core {

        /**
         * Constructor
         */
        function YOUREXTKEYNOUS_CLEANERTOOL()   {
                parent::tx_lowlevel_cleaner_core();

                        // Setting up help:
                $this->cli_options[] = array('--option1 value', 'Description...');
                $this->cli_options[] = array('--option2 value', 'Description...');

                $this->cli_help['name'] = 'YOUREXTKEYNOUS_CLEANERTOOL -- DESCRIPTION
HERE!';
```

```
                $this->cli_help['description'] = trim('LONG DESCRIPTION HERE');

                $this->cli_help['examples'] = 'EXAMPLES HERE';
        }

        /**
         * Analyze and return result
         */
        function main() {

                // Initialize result array:
                $resultArray = array(
                        'message' => $this->cli_help['name'].
                                                chr(10).chr(10).
                                                $this->cli_help['description'],
                        'headers' => array(
                                'SOME_ANALYSIS_1' =>
array('HEADER','DESCRIPTION',VERBOSITY_LEVEL 0-3),
                                'SOME_ANALYSIS_2' =>
array('HEADER','DESCRIPTION',VERBOSITY_LEVEL 0-3),
                                'SOME_ANALYSIS_...' =>
array('HEADER','DESCRIPTION',VERBOSITY_LEVEL 0-3),
                        ),
                        'SOME_ANALYSIS_1' => array(),
                        'SOME_ANALYSIS_2' => array(),
                        'SOME_ANALYSIS_...' => array(),
                );

                // HERE you run your analysis and put result into
                // $resultArray['SOME_ANALYSIS_1']
                // $resultArray['SOME_ANALYSIS_2']
                // $resultArray['SOME_ANALYSIS_...']

                return $resultArray;
        }

        /**
         * Mandatory autofix function
         */
        function main_autoFix($resultArray) {
                // HERE you traverse the result array and AUTOFIX what can be fixed
                // Make sure to use $this->cli_noExecutionCheck() - see examples from
bundled tools
        }
}
```

STEP3: Develop your tool to do something...
- You should now be able to see your tool appear in the list of tools and you should see output from it when you choose it.

- Make sure to study the bundled tools from EXT:lowlevel/clmods/. Try to deliver the same high quality of documentation and coding style from there. In particular how the constructor is used to set help-message information.
- Also, take a look at t3lib_cli which is the very base class - you can use the functions in there in your script.