How to release Preparation

Ongoing work

Communicate

All progress should be relayed on chat channel: https://gitter.im/tiki-org/community and/or Dev Mailing List. Communication should be used at different levels; Specific team and all the community. It should be used to announce but also to implicate members and to get help (mainly for testing purposes).

Update relevant pages about progress, reports and regressions.

Coordinate with other teams

- Review regularly the regressions and bugs with the wish list team
- Check that a doc page for the version has been created

1.1. Preparatory work

Some steps here are not necessary if releasing only minor version but this is actually the most time-consuming part!

Create the release pages and component related to the new version

Using the content for the existing page from the previous version (copy/paste) create a set of page for the new version.

- Tiki n (n being the version number)
- Regressions in n.x (n being the version number)
- Create a new item with the new version number in the category "Version" on dev.t.o

Create a clear roadmap and schedule milestones.

Notice:

1. Don't forget to adapt and improve.
2. Add the new version page in the "About Development" menu
3. Update https://tiki.org/All+Releases (you can add the Star chosen name later)

1.1.1. Start consultation process for star name

Each major version has a star name. This should be picked before releasing the alpha.

1.1.2. Fix major bugs and regressions

- current
- Do some tests thanks to the Pre-Dogfood Server

1.1.2.1. Acknowledgements

- Ask bug submitters (including security vulnerability reporters) how they would like to be credited in the Tiki release announcement.

1.1.3. Check database

Note: all database check steps, from Tiki 18, are done automatically by the release script by checking the last CI run for that branch. https://sourceforge.net/p/tikiwiki/code/HEAD/tree/trunk/doc/devtools/release.php

Which itself uses
https://gitlab.com/tikiwiki/tiki/-/blob/master/.gitlab-ci.yml

1.1.3.1. Check _tiki.sql suffixes

- Done in release script via https://sourceforge.net/p/tikiwiki/code/HEAD/tree/trunk/doc/devtools/check_schema_naming_convention.php

1.1.3.2. Structure

Does a fresh install with tiki.sql have the same structure as an upgraded Tiki from the previous version?

1. ~~Checkout a fresh Tiki 6.0, upgrade to Tiki 7.0~~
2. ~~Checkout a fresh Tiki 7.0~~
3. ~~Export SQL (ex.: mysqldump)~~
4. ~~Compare the two files and resolve any differences~~

Use the new script

Related:
http://www.maatkit.org/doc/mk-table-sync.html

1.1.3.3. Drop Table

In tiki.sql (for your version), make sure all table creation starts with
DROP TABLE IF EXISTS
This avoids that when people re-install, they get an error "table already exists"

You can use the following command line to scan tiki.sql:

```
perl -le 'while (<>) { $line++; $drop=1 if /^DROP/; die("DROP missing at line $line") if
/^CREATE TABLE/ &&!$drop; $drop = 0 unless /^DROP/; }' db/tiki.sql
```

Ricardo and @Roberto: Please test new script https://sourceforge.net/p/tikiwiki/code/65638 and add to release script

1.1.3.4. MyISAM

In tiki.sql (for your version), make sure all table creations are consistent. As of Tiki7, that means ending with ENGINE=MyISAM; (but it could change in the future). This is also required in the new schema updates. Reference

Ricardo and @Roberto: Please review https://sourceforge.net/p/tikiwiki/code/65625/ and incorporate to release script

1.1.3.5. InnoDB

The InnoDB installation depends on the table definition having ENGINE=MyISAM added to it (because a text conversion is done by the installer). Also check that the script tiki_convert_myisam_to_innodb.sql (for your version) includes all the new tables.
Like this: https://sourceforge.net/p/tikiwiki/code/43274/

1.1.4. Check SEFURLs

Make sure _htaccess and web_config are in sync.
For more info, go the Clean URLs or ask Arildb

See also: https://sourceforge.net/p/tikiwiki/code/HEAD/log/?path=/trunk/web_config (instructions at the top of the file)

1.1.5. Review all external links

For security, privacy, future-proofness, and site-deaths, the Tiki code should as seldom as possible link to outside of the control of the site admin or the Tiki community. A link to *.tiki.org/* with PluginRedirect is ideal.

quick script to check all http links in templates

```
grep -R http: templates | grep -v svn
```

Ideally, it would be done for the whole code base.

1.1.5.1. Make sure URLs are still active.

It's better to link to tiki.org/Something from which we can put a short page and a link, or even a Redirect

1.1.5.2. No direct calling of JavaScript

- Remove any link to JavaScript files via http like this one, which are not controlled by the site admin. Here is an example of the fix: http://sourceforge.net/p/tikiwiki/code/34348
    - If there is a way to store than JavaScript on a reputable CDN, this is great, but it should be optional, default to off. And it should be using https.

1.1.6. Check JSLint

inside Eclipse/Apatana: http://www.jslint.com/

1.1.7. Check PHP syntax is correct for each branch

Notably, 12.x has a minimum requirement of PHP 5.3 but after that 5.5 is used and some 5.5 required uses get backported to 12.x by mistake, especially the short array syntax (using [] instead of array() for instance). Searching for these regular expressions on all *.php files, (but with the help of phpStorm's "context" in multi-file search set to "accept comments and string literals") seems to find them.

- `[, \()\[` *(should find arrays declared using the short syntax - finds a few false positives in code with strangely positioned linefeeds)*
- `(?:empty|isset)\([^\)\[]*-\>[^\)\[]*\(` *(should find instances of "return value in write context" where a function is called inside an empty or isset test). Two found in vendor files, might need looking into for 12.5 release*

1.1.8. compile themes to css (from .scss or .less files) (optional)

- Tiki19+: php console.php scss:compile
- If you know how to check this easily: Some contributors could have made changes to css or less files and forgot to change the other. A sync should be made, and any missing commits added to the branch. Compiling Less files with *php console.php less:compile* (Tiki13-18) in the PhpStorm terminal produces results consistent among current developers.
    - This tool (or a version of it) might be able to decompile CSS into LESS which we could then compare somehow, but it wouldn't be trivial

1.1.9. Check the README file for manual commits

- The "function update_readme_file" of doc/devtools/release.php will output to the top-level README:
    - Check if anyone has committed anything manually to README that needs to be brought back into this script

1.1.10. Remove any out of sync English strings

- lang/en/language.php and lang/en/language.js may contain some tweaks to the English versions that were made after the string freeze. This is useful to improve the English text without breaking any translations. But this workaround should not be used forever and at each major release, this should be cleaned up. Strings should be removed from lang/en/language.php and incorporated in the relevant tpl and php files. Some judgment calls must be made to decide if the translation is invalidated or not. If you fix a typo in English, the translations are still good. However, if the meaning of the English string changes, the translations should be invalidated and translators need to review. This being said, if the meaning changes, it probably should not have been put in lang/en/language.php and lang/en/language.js in the first place.
    - Here is an example: https://sourceforge.net/p/tikiwiki/code/50704
    - Related: Mass spelling correction

```
grep -v '^//' lang/en/language.php
```

For all the strings that are to be updated, once you replace in Tiki files (source and templates), you look up the language files which have a translation for the old string, and you make that translation the translation of the new string. Since they mean the same thing. Then you can delete the line in lang/en/language.php

1.1.11. Delete secdb files from previous versions

Ex.: When releasing 5.2, you may see db/tiki-secdb_5.1_mysql.sql lying around

1.1.12. Check that all PHP files have a feature check

Each PHP file in Tiki should start with a feature check. So if the feature is de-activated, the file is dead. If a particular file is discovered to be insecure, users can deactivate the feature until they upgrade to the release which contains a fix. To avoid forgetting to add this feature check on new files, a feature check script has been created. Some files, by design, can't have a feature check and these files should be audited manually.

Run doc/devtools/securitycheck.php and check each "potentially unsafe" file.

> Script must be improved

1.1.13. Check external software library dependencies

This applies to all External Libraries

- coming from Composer : https://sourceforge.net/p/tikiwiki/code/HEAD/tree/trunk/vendor_bundled/composer.json When possible use notation to get minor updates with no backward compatible changes
- some known to need a Cleanup
- (and perhaps there are still some elsewhere in the code).

Background reading: Why your software project will slowly die without continuous updating

1.1.13.1. Integrity

- We are getting some files from Packagist, and they are not maintained by the official project. A check is required to make sure the files are the same. If the package is coming directly from the publisher, no need for an additional check.

1.1.13.2. Security

- Are there any dependencies that have security releases? If so, the update is mandatory
    - https://security.sensiolabs.org

1.1.13.3. General up-to-date-ness

- Generally, a Tiki release should contain the latest stable version of the library. However, if there is a major update to a lib (which has a higher chance of breaking in the upgrade), it can be decided to hold back. If this is the case, indicate it below and specify the reason (and any relevant link to a discussion). If you feel it's too risky for the stable branch, then, choose instead to do in trunk, and there could be a backport

later.

List of external libraries that we have decided to hold back on the upgrade.

Zend Framework 1	We are staying at ZF1 because it's a big job to upgrade, some components are not yet (or no longer?) available in ZF2, and thus, we will wait for now. NOTE that fix in r54095 is a workaround for a zf1 pbm fixed in zf2. It can be removed after replacement with zf2.

LTS policy

To be discussed: what is our policy for LTS versions? When do we stop updating the libs? (except for security releases)

1.1.14. Prevent directory browsing

- Add index.php (see others for examples) to all directories in which it has been forgotten
- Note while it is extremely important to prevent browsing in folders where there might be Tiki created content, like files, it is unclear if it provides any more benefit for folders that are open source code anyway, esp as I think we need route.php for tiki to work nowadays anyway. Hence we should consider revisiting this release task.

```
find . -not -path \*\.svn\* -not -path \*vendor\* -type d -exec ls {}/index.php \; | grep "No such file"
```

1.1.15. Prefs

1.1.15.1. Generate preference report

As per Preferences report to make sure this still runs and see if there any obvious issues. And attach the file to the version page (ex.: doc.tiki.org/Tiki12)

1.1.16. Security patches

In coordination with the Security Team.

This step should be done just before the actual packaging, to avoid disclosing a vulnerability. In some cases, this will be done after the alpha and beta stages.