How To Publish Telekom Software by a new Open Source Project

(1) Introduction

Deutsche Telekom not only encourages its employees and projects to use open source software, but also to release its software developments as Open Source Software (FOSS). Besides fulfilling the open-source license requirements, publishing or contributing (parts of) one's own work as open-source software means successfully participating in the open-source community and playing the game in accordance with the open-source rules and customs.

- This guideline describes how to release Telekom software as a new open source project (on Github) led by Deutsche Telekom under https://www.github.com/telekom
- For contributing to an existing (foreign) open-source see http://files.opensource.telekom.net/ospo-contributing-to-a-foreign-foss-project-flowchart.pdf

(2) Existing Tools

For releasing software in a GitHub repository correctly, Deutsche Telekom has designed a process documented by several files:

- The process flow chart
  - the complete process chart as PDF file
  - the main process as PNG file
  - the sub process 'selecting a license' as PNG file
- The corresponding checklist for getting a github repository
  - as editable md file preferred to use as data container for the communication with the OSPO
  - as readable pdf file
- The commented version of the checklist explaining the respective backgrounds as PDF file (this file)
- The guideline how to create an appropriate GitHub account
  - in English
  - in German

(3) The Process at a Glance

If you - on behalf of Deutsche Telekom - want to release your work as open source software, do this:

- Download and study the process flow chart
- Download the corresponding checklist, execute the steps and fill in the checklist as a form.
- Get in touch with the OSPO of Deutsche Telekom, hand over the form, and you will get the respective repository in an initialized version which already contains necessary and/or typically used compliance and health documents.

(4) Process in Detail

The following text is intended to be read by DT employees who need more (background) information. It follows the structure of the checklist mentioned above, but it additionally explains, why the single steps have to be taken. For those, who need a quick result, the documents mentioned under (3) could be more helpful.

Preparing the Publishing Work

- [ ] **S00**: Get the approval by your project and/or line manager.

> There are three reasons why you must contact your line manager:
>
> - Releasing your DT software as open-source software is encouraged by our company - if that steps supports your business needs. Whether that's true or not cannot be decided by a general unit like the DT **O**pen **S**ource **P**rogram **O**ffice, but only by the project itself. So, the responsible managers must approve the publication.
> - If you as an employee of DT publishes DT software as open source software, you release a piece of intellectual property of DT. Hence, it is a step of protecting yourself protecting to get the approval.
> - Most of us use their 'private' GitHub account also when they work as DT employee. To deal with two different accounts is very unhandy - as it has been turned out in some tests. Therefore, DT allows you to use your 'private' GitHub account. But it requires that you behave yourself in accordance with the DT Code of Conduct. To avoid misunderstandings your line manager should know that you are going to release DT work as open source software.

- [ ] **S01.***: Inaugurate an Open Source Supervisor as contact person for the Open Source Program Office (OSPO)

> To compliantly release your software as FOSS, you probably need help. The OSPO brings the respective legal and procedural knowledge into your project by training a contact person to do the right things or to support you to do so adequately. We call this contact person an *Open Source Supervisor*. He should fulfill a specific skill profile (see https://files.opensource.telekom.net/os-supervisor-profile.pdf). Feel free to either re-use / inaugurate a global OS-Supervisor for/of your department or to inaugurate your project specific OS-Supervisor. And yes, it is possible to become your own OS-Supervisor. This supervisor is trained by the OSPO to support you practically.

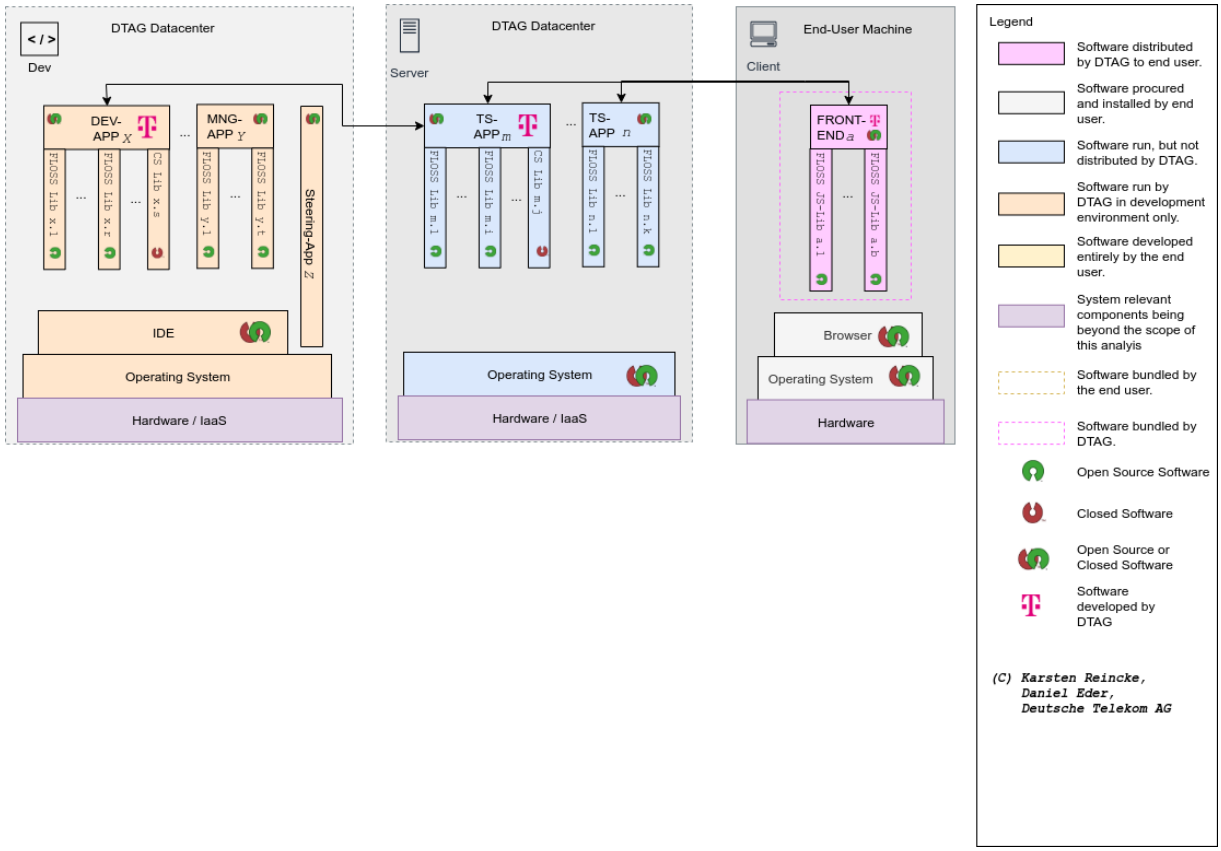
- [ ] **S02**: Create your GitHub accounts in accordance to the guideline How to create Telekom fitting GitHub accounts

> Most of us use their private GitHub accounts also if the act as DT employee on GitHub. Therefore, these accounts should especially been protected from being abused. How to protect your accounts by some methods offered by GitHub is explained by this guideline

- [ ] **S03**: Determine whether you only want to release an interface (without any reference implementation)? - (`If YES -> S7`).

> Initially defining and releasing an interface as open source software simplifies the approval process: Due to the fact, that interfaces (APIs) without any reference implementation cannot be patented and do not use open source sub components (because they are not specified by any executable code), you can skip the patent analysis and the generation of the BOM.

- [ ] **S04A**: Describe the architecture of the prerequisited software stack:



> This is the output of the drawio template DT offers for documenting software architectures.

- [ ] **S04B**: By using the software architecture create a BOM of your software (= list of all FOSS sub libs / components your software requires).

> Often the application or library itself, that you want to publish as open source software, uses 'embedded' open source sub-components. The open source licenses of these sub-components can influence the act of licensing of the application or the library. Or they have to be checked by the patent analysis too. Hence you need the BOOM for be able to do the next steps appropriately

Avoiding unwanted IPR / patent side effects

- [ ] **S05.A**: Determine whether you have applied a patent of / for your software? `If NO -> S5C`.
- [ ] **S05.B**: Decide whether you want to set up a patent based business model `If YES do not publish your code as FOSS`.

> If you have applied a patent with your code and if you want to establish a patent based business model (= getting fees for permitting the use of the patent), then you should not release your software as open source software: Their exist some open source licenses with an explicit patent clause which permits the free use of your patent in the context of your software. And for the other open source licenses it is often legally argued that they contain an implicit patent clause. Hence, if you release your patented software you are about to subvert your intended business model

- [ ] **S05.C**: Determine whether there exist patents in the DT patent pool which are necessary to use your software or one of the FOSS sub components / libs it requires? `If NO -> S06`.
- [ ] **S05.D**: Obtain the approval for the respective patents to be used free of charge in the context of your software. `If you don't get such an approval do not publish your code as FOSS`.

> It is not necessarily harmful for DT if your software touches a patent of its patent pool. DT owns a lot of patents by which it wants to protect itself against patent trolls etc. Due to the fact that an open source license (implicitly) permits the use of the patent only in the context of the licensed software, the protecting effect is not negated generally. But DT also owns

> some (telecommunication) patents it does not want to permit to be used under any circumstances. Thus, you have to make sure that the DT patents touched by your software do not belong to those DT never wants to be used for free.

Choosing a FOSS License by Avoiding copyleft conflicts

- [ ] **S06**: Determine whether elements of the BOM are distributed under a strong copyleft license?
  - [ ] `If none of them` **S07.A**: Select any FOSS license for your software : [_____] *(see ospo-selecting-a-foss-license.pdf)*.
  - [ ] `If one of them` **S07.B**: Choose the same license as license of your software: [_____] .
  - [ ] `IF several of them` **S07.C**: Manage the license conflict. `RESTART!`

> If two sub components are licensed under different licenses with strong copyleft effect, then both licenses require, that the software using the sub components must be licensed under the same license as the sub component. This condition cannot be fulfilled by the using main component. Hence, you have to solve the conflict.

Gathering the Repository Data

- [ ] **S08.A**: Select a shorter GitHub repository name (as part of the repo URL): [_____] (*CamelCase or small-letters-with-hyphens*).
- [ ] **S08.B**: Select a longer more speaking software name (as part of the README): [_____] (*CamelCase etc.*)
- [ ] **S08.C**: Verify, that the selected names do not exist (in the context of GitHub).
- [ ] **S08.D**: Briefly describe the purpose of your software

> [
>
>]

> These values are used by the OSPO to create the repository. Yes, one can modify them later on. But at least the shorter GitHub repository name should carefully be chosen, because modifying the repo identifier would brake all links of the already cloned instances.

- [ ] **S09**: List the GitHub ids of the intended repo owners / admins here:

NO.	NAME	eMail-Address	GitHub Avatar
0.)	Karsten Reincke	karsten.reincke@telekom.de	kreincke
1.)	…	…	…

Getting the Repository

- [ ] **S10**: Handover this filled in form to the *DT OSPO*.
- [ ] **S11**: Verify the created repository:
  - [ ] **S11.A**: let the admin(s) checkout the repo, modify a file, and commit and push the modification. (*Do the admins really have write access?*)
  - [ ] **S11.B**: let the admin(s) add a normal user. (*Do the admins really have admin rights?*)
- [ ] **S12**: Adjust the compliance and health documents in accordance with the file `4-REPO-OWNERS` .

Dealing with File Headers

Each source file added to the repository must be opened by a project specific file header. The repository you receive contains your version under templates/fileheader.txt. It is up to you to put this template in front of each file and to initialize it. But do not modify its structure.

Here is the explanation why we must do so:

Although since March 1, 1989, the author automatically owns all copyrights also in the USA (previously this was only true for the European family of copyright laws), placing a copyright notice nevertheless still acts as legal insurance against "innocent infringement" defenses, where defendants claim, they were unaware of the copyright owner.

Additionally, in Europe, the initial author can distribute his copyrights to other persons except the right to be named as 'the author': in Europe, an author can never lose the right to be the author.

From these aspects follows that we - as Deutsche Telekom - indeed have to insert into the copyright line (a) the names of the authors (for respecting their authorship) AND (b) the name of our company for indicating that the company owns all other rights. And we have to do so for making the sentence 'This file is distributed under the conditions of the XYZ license' legally valid and effective because only the copyright holder can license work as open-source software.