# Em002-2 Instructions pour la publication de logiciels ouverts

## Recommandations concernant l'informatique de l'administration fédérale[1]

## Annexe indépendante du document principal Em002.

| | |
|---|---|
| Classification[2] | Non classifié |
| Effet juridique[3] | Recommandation |
| Domaine de planification[4] | Informatique de l'administration fédérale |
| Version | 2.0 |
| Version précédente | 1.0 du 24.02.2025 |
| État | Validé |
| Date de validation (cette version) | 9.12.2025 (Version française du 13.4.2026) |
| Validé par, base légale | Le délégué à la transformation numérique et à la gouvernance des TIC (D-DTI), conformément à l'article 40 de l'ordonnance du 1er mai 2025 relative aux services numériques et à la transformation numérique dans l'administration fédérale (ordonnance sur la numérisation, DigiV), SR 172.019.1 |
| Langues | allemand (original), français, italien, anglais (traduction) |
| Licence | CC0 1.0 Universal<br>Le document est publié sous la licence CC0. Il peut être réutilisé, modifié et transmis librement, même à des fins commerciales et sous n'importe quel format. |

[1] Gestion des recommandations concernant l'informatique de l'administration fédérale, conformément à [P035], *chap. 4.6*

[2] Pour les échelons de classifications INTERNE et CONFIDENTIEL, voir l'ordonnance du 8 novembre 2023 sur la sécurité de l'information (OSI, *RS 128.1*)

[3] Voir note 1

[4] Domaines de planification selon la *Stratégie informatique de la Confédération 2020-2023 du 3 avril 2020 (SB000)*

# Table des matières

**1 Condensé ..... 3**

**2 Introduction ..... 3**

**3 Procédure de clarification préalable ..... 5**

**3.1 Mise à disposition conformément à la LMETA ..... 5**
**3.2 Exception : droits de tiers ..... 7**
**3.3 Exception : raisons de sécurité ..... 10**
**3.4 Clarification de la publication ..... 12**
**3.5 Limiter le plus possible la charge de travail ..... 13**
**3.6 Choix de la langue de publication ..... 13**

**4 Procédure Analyse et préparation ..... 14**

**4.1 Analyse du code source ..... 14**
**4.2 Choix de la licence ..... 15**
**4.3 Documentation spécifique aux OSS ..... 15**

**5 Procédure Publication et diffusion ..... 17**

**5.1 Choix de la plateforme ..... 17**
**5.2 Publication du logiciel ..... 18**
**5.3 Communication ..... 18**
**5.4 Mise en place et gestion de la communauté ..... 19**

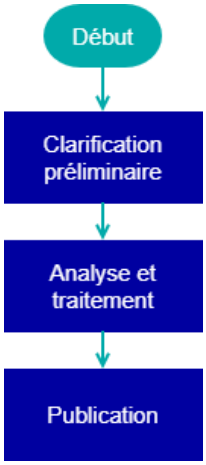
**Annexe ..... 20**

**A. Changements par rapport à la version précédente ..... 20**
**B. Références ..... 20**
**C. Abréviations ..... 20**
**D. Documentation relative au document *Em002-2.2 Liste de contrôle OSS Analyse et préparation* – Documentation ..... 21**
**E. Documentation relative au document *Em002-2.2 Liste de contrôle OSS Analyse et préparation* – Code source ..... 24**
**F. Documentation relative au document Em002-2.3 *Liste de contrôle OSS Mise à disposition et publication* ..... 27**
**G. Autres sources et licences ..... 28**

# 1 Condensé

Les instructions décrivent la procédure à suivre pour publier le code source d'un logiciel que les autorités fédérales ont développé ou fait développer pour l'exécution de leurs tâches, sous réserve que les droits de tiers ou des raisons importantes en matière de sécurité excluent ou limitent cette possibilité (art. 9 LMETA)[5]. Après la publication, toute personne est autorisée à utiliser, à développer et à distribuer le logiciel. Aucun droit de licence n'est perçu.

Les instructions sont destinées aux personnes responsables de la publication du code source et qui la réalisent.




Elles s'articulent autour de trois parties principales.

- La première partie (chap. 3) concerne l'**examen préalable** et les exceptions prévues par la LMETA.
- La deuxième partie (Chap. 4) concerne l'**analyse et la préparation** d'un code source existant.
  Elle traite aussi du choix de la licence.
- La troisième partie (chap. 5) décrit la procédure de **publication et diffusion** proprement dite et traite des autres mesures telles que la communication et la mise en place d'une communauté appropriée.

Les instructions sont assorties de **trois listes de contrôle** qui accompagnent et documentent la procédure.

Des compléments techniques sont présentés en annexe.

# 2 Introduction

Lors de la mise à disposition d'un OSS, deux cas de figure se présentent. Il peut s'agir **d'une contribution au code source et à la documentation d'un OSS** ou **de la publication d'un projet d'OSS propre**.

Dans le premier cas, il s'agit en général de corriger une erreur (*bug fix*) ou d'ajouter une fonctionnalité (*feature*). Le code source doit, ou peut, être mis à disposition sous la licence du projet d'OSS existant, en fonction du type de licence et de l'utilisation du logiciel. Le cas échéant, la mise à disposition est soumise à un accord.

Dans le second cas, celui du lancement d'un nouveau projet d'OSS, la gouvernance et la licence peuvent être choisies librement. Il faut juste faire attention à la licence à laquelle sont éventuellement soumis des éléments logiciels intégrés au nouveau projet. Pour plus de détails sur le choix des licences, voir le document *Em002-3 Guide Licences OSS*. Si une communauté est susceptible d'apporter une valeur ajoutée au projet pour l'administration fédérale et si cette dernière souhaite créer un écosystème, il faut remplir le document *Em002-4.1 Liste de contrôle Communauté OSS* conformément au document *Em002-4 Guide Communauté OSS*.

La mise à disposition d'OSS n'est pas qu'une simple mesure technique.
Elle vise aussi à mettre en place une culture de l'ouverture et à embarquer les gens. Pour être un succès, une culture OSS doit allier ouverture, excellence technique, interaction sociale solide et un engagement commun.

[5] *[LMETA2023]*

La figure ci-après indique la marche à suivre en fonction de la situation.

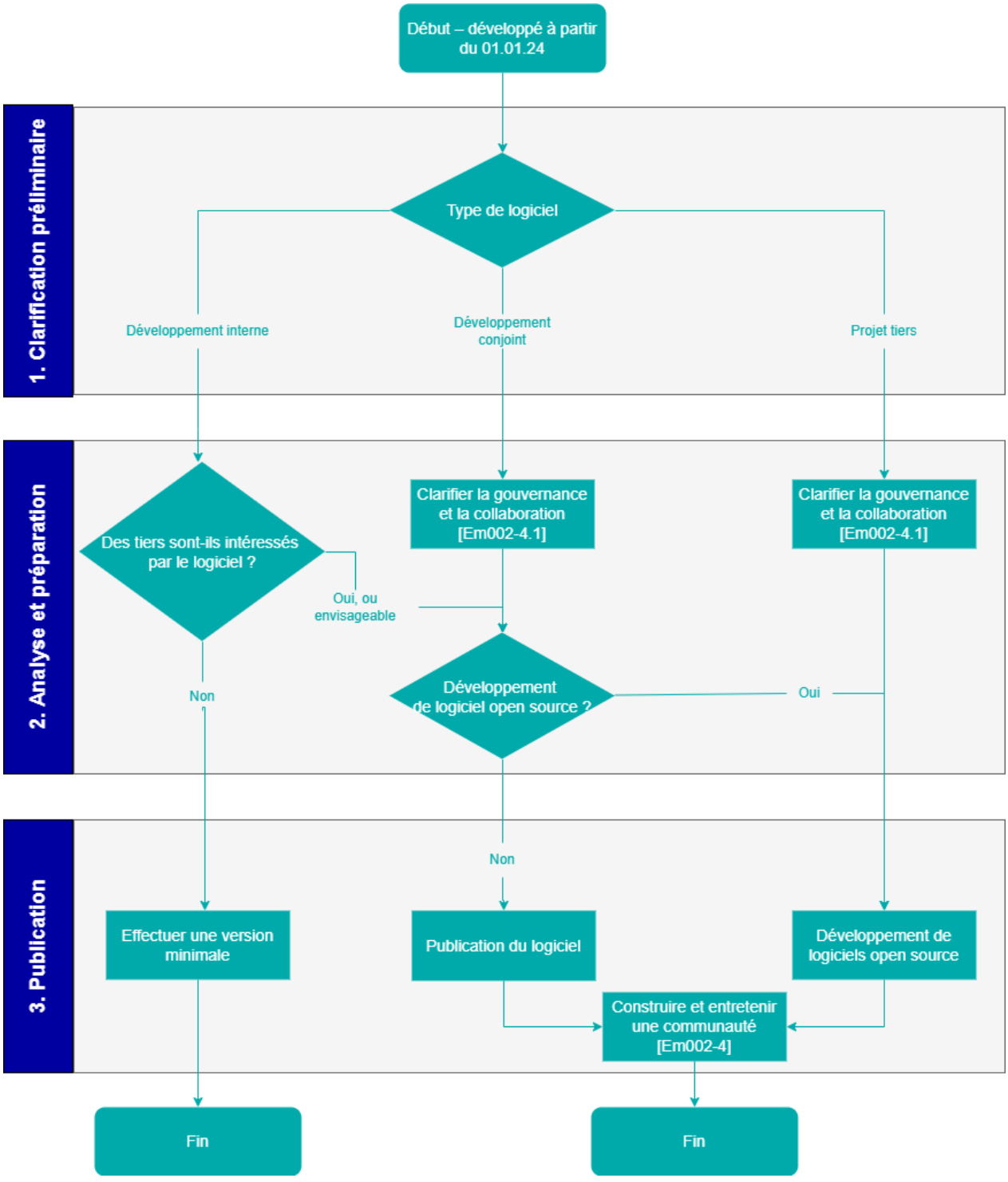



Fig. 1 prise de décision pour la mise à disposition d'un logiciel

# 3 Procédure de clarification préalable

L'**objectif** est de clarifier les avantages et les conséquences possibles de la publication. Il s'agit de décider si le logiciel peut être publié et si la mise en place d'une communauté active est judicieuse. Il faut veiller à remplir le document *Em002-2.1 Liste de contrôle OSS Examen préalable* assez tôt dans le projet, car elle peut influencer la manière de développer le logiciel.

## 3.1 Mise à disposition conformément à la LMETA

L'art. 9 LMETA dispose que l'administration fédérale doit publier le code source des logiciels qu'elle développe ou fait développer pour l'exécution de ses tâches, sous réserve que les droits de tiers ou des raisons importantes en matière de sécurité excluent ou limitent cette possibilité.

Les *[CG-OFCL]* s'appliquent dès lors qu'un logiciel individuel est créé sur mandat de la Confédération. Le ch 25.1 de celles-ci dispose que le code source et la documentation du logiciel appartiennent au maître d'ouvrage, en l'occurrence à la Confédération. Si le logiciel est une acquisition commune de plusieurs organisations ou autres institutions, il faut alors vérifier qui détient les droits du code. Par exemple, la propriété peut être détenue par une association créée à cet effet.

La simple utilisation d'un logiciel standard n'est pas soumise à l'art. 9 LMETA et la mise à disposition n'est pas possible, étant donné que les droits liés à ce logiciel sont normalement détenus par son fabricant (voir CG de la Confédération pour l'acquisition de logiciels standard). En revanche, il est tout à fait envisageable de publier des éléments ajoutés à un logiciel standard, qui appartiennent normalement à la Confédération. Les simples adaptations de configuration se prêtent moins à la publication. Ce sujet est traité dans les documents *Em002-7 Acquisition et OSS - Aspects stratégiques*, [CCMP-MB] et [OFCL-WL]. Les modifications plus conséquentes sont quant à elle toujours susceptibles d'être soumises à l'art. 9 LMETA.

### 3.1.1 Rapport entre logiciel et LMETA

La version actuelle de la norme ISO/IEC 24765 donne trois définitions de la notion de logiciel.

1. Un logiciel est un programme ou un ensemble de programmes qui servent à faire fonctionner un ordinateur.
2. Un logiciel est un programme informatique et sa documentation.
3. Un logiciel est un programme et, le cas échéant, sa documentation et d'autres données nécessaires au fonctionnement d'un ordinateur.

Sur la base de ces définitions, on peut considérer que les petits scripts, les macros et les *infrastructures as code* (IaC) sont aussi des logiciels. Comme l'art. 9, al. 1, LMETA concerne le code source, une simple publication est suffisante dans le cas de la définition 1. Toutefois, une publication sans documentation associée n'a que peu de sens, car le logiciel ne peut pas être utilisé selon la définition 2.

Pour générer des synergies avec des tiers, il faut se fonder sur la définition 3. Dans ce cas, on entend par données, les données et configurations de base (p. ex. les valeurs des énumérations et des données de base, qui devraient également être mises à disposition comme code source).

Les petits projets, scripts ou exemples de code peuvent aussi être publiés ensemble sur un dépôt si l'autorité fédérale estime que c'est judicieux. Dans ce cas, on peut utiliser ce guide

et remplir une seule fois les listes de contrôle correspondantes.

Pour savoir ce qui doit être publié et selon quelles modalités, il faut évaluer la proportionnalité et le coût de l'opération.

### 3.1.2 Type de logiciel

La Fig. 1 fait la distinction entre les développements propres, les développements en commun et les projets de tiers. Les conséquences ne sont pas les mêmes dans chaque cas. Peu importe comment le logiciel est développé, il tombe sous le coup de l'art. 9 LMETA.

Dans le cas du développement en interne, la Confédération crée elle-même le logiciel ou le mandate pour son compte. Elle choisit la gouvernance et détient les droits sur le code source.

Dans le cas du développement en commun, la Confédération crée le logiciel en collaboration avec d'autres organisations. La gouvernance et les droits sur le logiciel doivent être réglés au moyen de la forme d'organisation choisie.

Si la Confédération contribue directement à des logiciels de tiers, ce code source relève également l'art. 9 LMETA. Il faut dans ce cas veiller au respect des exigences des autorités fédérales, à ce que la contribution réponde aux exigences du projet et que l'administration fédérale se conforme à la gouvernance.

Voir à ce sujet les documents *Em002-2.1 Liste de contrôle OSS Examen préalable* et *Em002-4.1 Liste de contrôle Communauté OSS.*

### 3.1.3 Code *legacy*

Pour les vieux logiciels (**logiciels hérités**), le travail de publication à postériori est plus important que dans le cas où la mise à disposition est prévue dès le départ. Il vaut alors mieux attendre pour effectuer le travail qu'un utilisateur potentiel du logiciel se manifeste. Dans tous les cas, il faut remplir le document *Em002-2.1 Liste de contrôle OSS Examen préalable*, afin de garder une trace des décisions.

Les applications que les autorités fédérales ont commencé à développer à partir du 1er janvier 2024 ou qui ont commencé à être développées sur la base d'un contrat conclu avec la Confédération après le 1er janvier 2024 ne sont pas des systèmes hérités et doivent obligatoirement être publiées conformément à l'art. 9, al. 1, LMETA.

Cette publication n'est en aucun cas liée à l'utilité qu'elle a pour des tiers.

### 3.1.4 Bibliothèque, modules d'extension

Parfois, un logiciel n'est pas une application à part entière, mais seulement une partie d'application. Comme la LMETA et l'ordonnance ne définissent pas plus précisément le code source, les instructions s'appliquent également aux bibliothèques et aux modules d'extension qui relèvent ainsi de son champ d'application.

**Activités**

- Collecter les informations prévues dans le document *Em002-5 Aide-mémoire OSS Appliquer l'art. 9 LMETA*.
  Cela concerne à la fois des informations générales sur les OSS et des informations spécifiques sur l'application de la LMETA.
- Vérifier si les conditions de la LMETA sont remplies.
- Remplir le document *Em002-2.1 Liste de contrôle OSS Examen préalable*. En règle générale, il est judicieux de faire appel directement aux personnes de contact du

fournisseur/développeur du logiciel.
- Si nécessaire, remplir le document *Em002-4.1 Liste de contrôle Communauté OSS*.

**Décision**
Faut-il publier le logiciel et comment définir les grandes lignes de l'opération?

## 3.2 Exception : droits de tiers

Il faut renoncer à la publication lorsque celle-ci porte atteinte aux droits de tiers. Si le logiciel a été développé par des collaborateurs de la Confédération, les droits appartiennent à l'employeur[6]. Dans les contrats de location de services et de prestations informatiques, les droits sont habituellement cédés à la Confédération et doivent être réclamés.

Si le logiciel a été ou est développé sur la base des [CG-OFCL] de 2024[7], les droits de propriété intellectuelle appartiennent à l'acheteur, sauf accord contractuel contraire.

Une application est normalement constituée de nombreux éléments et composants individuels. Certains types d'éléments et de composants peuvent poser problème lorsque leur code source n'est pas ouvert ou qu'ils sont déjà soumis à une licence *open source* en tant que parties d'application.

**Bibliothèques**

Les bibliothèques (en anglais *libraries*) peuvent être directement compilées dans une application, liées à l'application ou livrées sous forme de paquets, en fonction du langage de programmation utilisé. Elles font partie intégrante de l'application.

La publication du code source est plus difficile lorsque l'application contient des bibliothèques propriétaires ou des bibliothèques soumises à une licence. Dans ce cas, le code source du reste de l'application peut certes être publié, mais les utilisateurs et développeurs potentiels doivent alors également acheter la bibliothèque concernée avant de pouvoir utiliser ou modifier l'application en question.

Si la propriété de la bibliothèque problématique est détenue par le fournisseur de l'application, il vaut alors la peine de clarifier juridiquement si la Confédération ne dispose pas d'un droit transmissible à des tiers sur ces bibliothèques.

Il faut autant que possible éviter d'utiliser des bibliothèques propriétaires et des bibliothèques soumises à une licence lors de développements propres ou, si possible, les remplacer avant la mise à disposition. Partant de l'objectif fixé à l'art. 9 LMETA, il faut publier le plus des fonctionnalités possible, dès lors qu'elles sont utilisées dans l'administration publique. Par conséquent, les bibliothèques individuelles qui empêchent l'ouverture du code source constituent une dette architecturale qui doit être documentée et éventuellement éliminée dès que possible.

**Banques de données et autres supports de stockage**

Le fait qu'une application enregistre des données dans des dépôts propriétaires ou soumis à licence (p. ex. serveur Microsoft SQL) n'empêche pas la publication du code source.

Il faut toutefois vérifier s'il serait possible d'utiliser en plus des banques de données ouvertes comme PostgreSQL. Cela permettrait d'économiser des frais d'exploitation et faciliterait l'accès à des utilisateurs potentiels.

---

[6] art. 332 CO en relation avec l'art. 6, al. 2, LPers

[7] Conditions générales de la Confédération, notamment ch. 25.1 – https://www.bkb.admin.ch/fr/conditions-generales-cg-de-la-confederation

### Serveur d'applications et systèmes d'exploitation

Le fait qu'une application fonctionne avec un système d'exploitation ou un serveur d'applications soumis à licence (p. ex. serveur Microsoft Windows ou RedHat JBoss) n'empêche pas la publication du code source.

Il est recommandé de demander au fabricant du logiciel si des composants de l'application utilisent des bibliothèques soumises à licence de fabricants tiers et si oui, lesquelles.

Si l'élaboration du logiciel est achetée en commun par plusieurs organisations ou autres institutions, le propriétaire est en général une société ou une autre personne morale créée à cet effet.

### Autre propriété intellectuelle

Les droits de tiers ne comprennent pas seulement les droits d'auteur, mais aussi d'autres droits de propriété intellectuelle (marques, brevets). Un brevet en soi n'empêche pas la publication du code source, mais il peut faire obstacle à l'utilisation ou à la modification du logiciel. Les brevets ne sont pas très courants, du moins en Suisse, mais il vaut tout de même la peine de vérifier auprès du fabricant du logiciel si un examen a déjà eu lieu.

### Acquisition des droits nécessaires

Pour répondre aux exigences de la LMETA, l'autorité fédérale peut, en sa qualité de mandant, s'assurer les droits sur les résultats du travail. Cette condition préalable à la publication peut faire l'objet d'un critère dans le cadre de l'acquisition. Il peut être plus compliqué d'acquérir les droits par la suite.

Lorsque l'autorité fédérale ne souhaite pas publier elle-même le logiciel, elle peut déléguer cette tâche, par exemple à son mandataire, ou céder les droits de publication à un tiers. Dans ce cas, les droits et les devoirs nécessaires doivent être cédés. Cela signifie que le droit est effectivement transféré à un tiers, à condition que le logiciel concerné soit mis à disposition et que le support soit assuré.

Le document [CCMP-MB] contient également des indications importantes.

## Déroulement

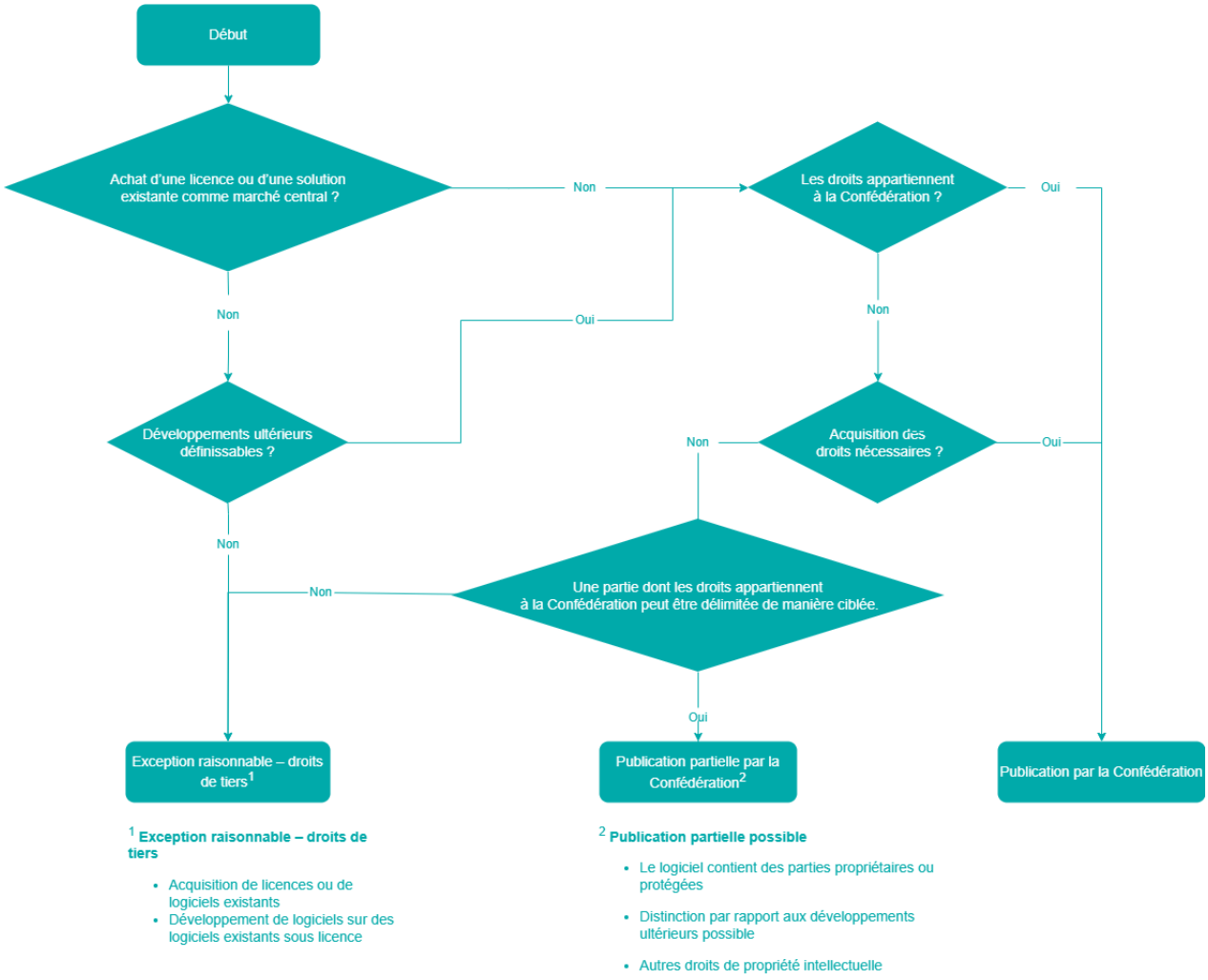



Fig. 2 exception : droits de tiers (veiller à toujours prendre en considération les documents [CCMP-MB] et *Em002-7 Acquisition et OSS - Aspects stratégiques*)

## Activités

- Vérifier si l'organisation possède les droits de propriété intellectuelle sur le logiciel.
- Vérifier s'il est possible d'acquérir les droits.
- Si nécessaire, demander l'autorisation écrite des détenteurs des droits.
- Vérifier si l'application contient des parties propriétaires ou protégées.
- Vérifier l'absence d'obstacles dus à la protection des brevets (si nécessaire et utile).
- Vérifier si la mise à disposition doit être confiée à des tiers.
- Vérifier si le logiciel doit être développé conformément à l'OSSD (avec le document *Em002-4 Guide Communauté OSS*).

## Décision

Le logiciel peut-il être publié sans porter atteinte aux droits de tiers?

## 3.3 Exception : raisons de sécurité

Les logiciels ne peuvent pas être publiés si des raisons importantes en matière de sécurité s'y opposent.

Les données proprement dites d'un logiciel ne sont pas concernées par la publication du code source. Ces données sont généralement sensibles et ne sont pas publiées. Il se peut toutefois qu'en plus des données de contenu, certains algorithmes et procédures visibles dans le code source ne doivent pas être rendus publics (p. ex. cybercapacités offensives, détails sur la détection des fraudes). La sécurité peut concerner la confidentialité, l'intégrité, la disponibilité ou la traçabilité.

### Recommandation

Il est recommandé de développer le logiciel de manière à ce que la publication du code source n'engendre pas de risque supplémentaire. Il ne faut pas partir du principe que le fait de ne pas publier protège contre les attaques (sécurité par l'obscurité).

### Déroulement

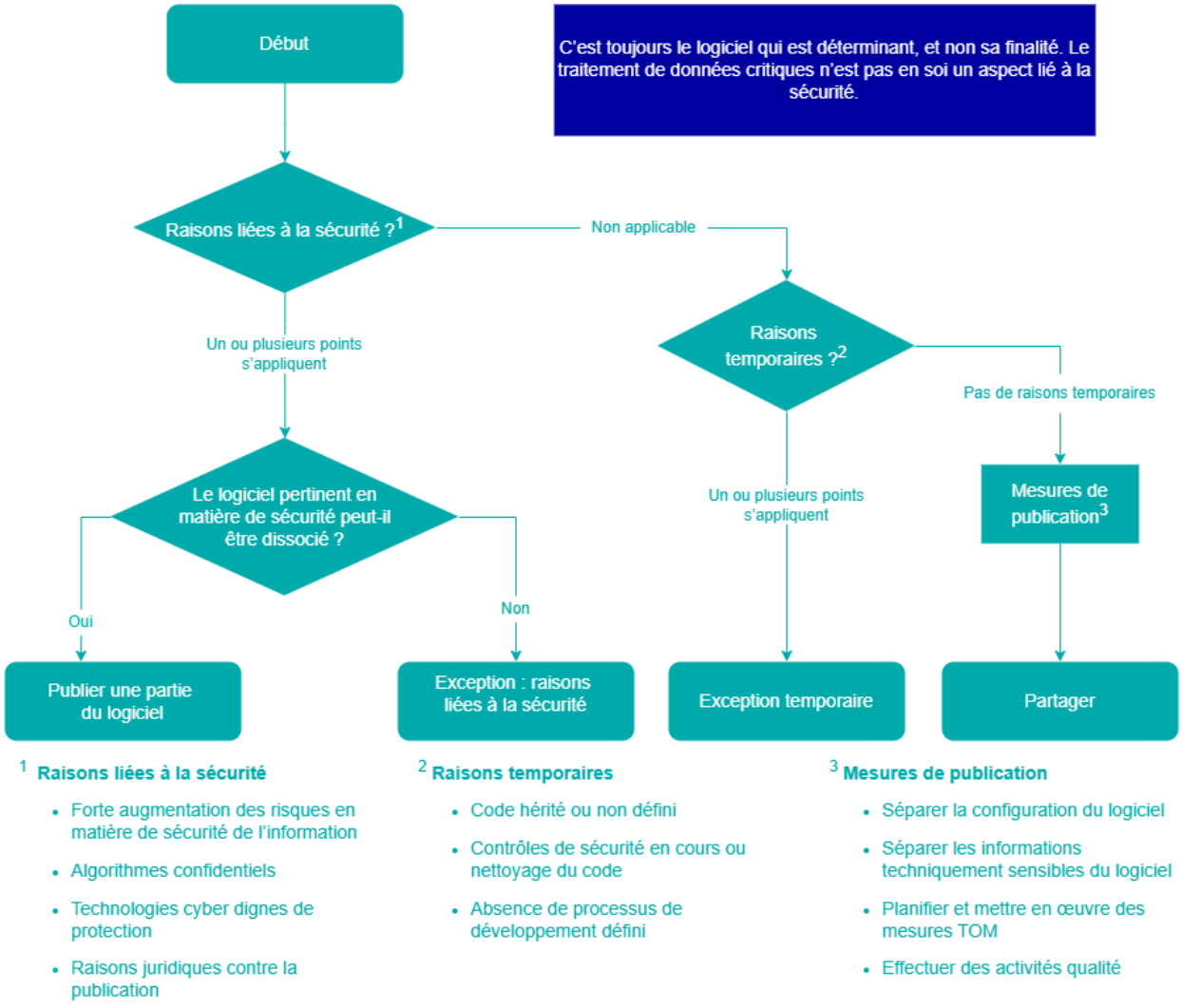



Fig. 3 Exception: raisons de sécurité

Pour chaque projet, les raisons importantes en matière de sécurité peuvent être les suivantes :

- la publication du logiciel augmente massivement le risque concernant la sécurité de l'information (art. 6 LSI[8]),
- la publication des algorithmes de lutte contre la fraude et d'autres algorithmes de ce type n'est pas forcément obligatoire,
- il s'agit de technologies de cybersécurité sensibles,
- il s'agit du logiciel de base pour l'exploitation d'infrastructures critiques,
- la publication viole d'autres dispositions légales.

Les raisons ci-après peuvent temporairement empêcher la publication, le temps de mettre en place les mesures appropriées.

- le code *legacy* n'a pas encore pu être mis au point et contient encore des éléments problématiques en matière de sécurité,
- les contrôles de sécurité du code n'ont pas encore été réalisés,
- la maturité du processus de développement n'est pas suffisante pour permettre la publication. Même si le processus de développement n'est pas suffisamment établi, il est quand même possible de publier des parties qui ne sont pas destinées à des tiers. Par exemple, les directives sur le commentaire du code et la génération de données de test doivent être adaptées en fonction de la nouvelle situation de départ après publication et fonctionner de manière fiable.

Les mesures ci-après permettent de publier des logiciels malgré l'existence de raisons importantes en matière de sécurité :

- séparer les configurations (p. ex. le type de chiffrement utilisé) et le logiciel,
- enregistrer les informations techniques sensibles (p. ex. processus, calculs) qui ne sont pas publiques dans des configurations ou des réglages externes,
- planifier des mesures organisationnelles et personnelles appropriées, les implémenter et les appliquer régulièrement,
- prévoir et exécuter des activités concernant la qualité lors de la publication. Voir le chap. 4 Analyse et préparation.

### Activités

- Vérifier que le logiciel ne présente pas de problèmes importants en matière de sécurité qui empêchent sa publication.
- Estimer les efforts nécessaires pour remédier aux causes temporaires.
- Planifier des mesures pour séparer les informations relatives à la sécurité du code source du logiciel.
- S'assurer que l'application ne contient pas de procédures qui ne doivent pas être rendues publiques.

### Décision

Y a-t-il des raisons importantes en matière de sécurité qui rendent la publication absolument impossible ?

[8] RS 128

## 3.4 Clarification de la publication

Cette étape permet de déterminer qui publie le logiciel. La publication peut être confiée à un fournisseur, lorsque celui-ci a créé le logiciel. Il faut alors déterminer où le code source sera publié et, point important, régler la gouvernance de manière appropriée.

Si le logiciel a été développé en interne et que la Confédération ne veut ou ne peut pas le publier elle-même, elle peut mandater un fournisseur externe pour le faire ou céder les droits (transmission d'obligations à des tiers).

**Mise à disposition minimale et support**

La LMETA ne fait aucune prescription concernant la publication. Ainsi, une mise à disposition minimale du code source suffit à remplir les prescriptions légales. Aucune autre activité n'est requise. Le document *Em002-4 Guide Communauté OSS* permet de répondre à la question de savoir si et dans quelle mesure un support est fourni et si le projet est géré activement.

**Développement de logiciel ouvert (OSSD)**

L'OSSD consiste à réaliser publiquement l'ensemble du processus de développement, en plus de la publication du code source. Tout est transparent, depuis les exigences (problèmes et demandes) jusqu'au code source. L'OSSD est soutenu par des outils de communication publics (liste de distribution, forums, etc.), un système de contrôle des versions (git), des listes de bogues et de fonctionnalités, une feuille de route et des outils de développeur. L'ensemble de la communauté profite de cette méthode transparente qui permet en outre de coopérer au-delà des limites entre les organisations. Cette procédure évite en outre d'avoir à réaliser des activités en lien avec la publication une fois le projet clôturé, étant donné que cette étape aura déjà été prise en compte au début du projet.

**Activités**

- Évaluer grossièrement le travail nécessaire pour la phase d'analyse. La charge de travail peut aller de trois jours à deux semaines en fonction de la taille et de la complexité de l'application.
- Libérer les moyens pour les étapes suivantes, en particulier pour l'analyse.
- Établir l'accord de principe pour la publication de l'application en tant qu'OSS.

**Décision**

La faisabilité et l'accord de principe relatifs à la publication de l'application en tant qu'OSS sont établis.

## 3.5 Limiter le plus possible la charge de travail

Les listes de contrôle et les instructions sont conçues de manière à limiter le plus possible les efforts nécessaires pour la mise à disposition. La planifier systématiquement dès le début du projet et la prévoir dans le développement permet de limiter la charge de travail. Une certaine charge administrative et des coûts sont toutefois inévitables [Le2023]. Dans certains cas, ils peuvent être compensés par les communautés qui prévoient une répartition des coûts (voir *Em002-4 Guide Communauté OSS*).

## 3.6 Choix de la langue de publication

Le choix de la langue dépend du public cible et des méthodes de travail de l'équipe. Fort heureusement, les nouveaux outils facilitent grandement la traduction (n.d.t. : sans blague). Les langues envisageables sont les langues officielles de la Confédération (art. 70 Cst.[9])et l'anglais.

Si le projet a une vocation internationale, on tendra à opter pour l'anglais. Si la publication a lieu après coup, il n'est évidemment pas utile de changer la langue.

Il faut publier au moins le fichier README.md dans plusieurs langues, de sorte que les personnes intéressées puissent savoir si le projet les intéresse vraiment.

La communication publique doit s'adresser en premier lieu au public cible. Mais il ne faut pas oublier qu'il s'agit d'une communication publique de la Confédération.

**Remarque** : la publication est un nouveau canal de l'administration fédérale pour la communication avec le public. S'il n'est pas utilisé avec soin et professionnalisme, l'image de la Confédération peut rapidement en pâtir.

[9] Cst.

# 4 Procédure Analyse et préparation

**Objectif**
Les travaux nécessaires à la publication de l'OSS ont été effectués.
La licence et le type de communauté sont choisis. À cet effet, on utilise le document *Em002-2.2 Liste de contrôle OSS Analyse et Préparation*.

## 4.1 Analyse du code source

Le code source à publier et les autres documents associés sont analysés avant la publication. L'analyse vise à s'assurer qu'aucune information sensible n'est publiée.

Si le code source à publier est volumineux, il est recommandé d'utiliser des outils automatiser pour exécuter les activités ci-après.

Pour les détails, voir l'annexeE.

**Activités**

- Vérifier comment la mesure de la qualité du code et les directives ont été appliquées (p. ex. ISO/IEC 25010:2023[10]).
- S'assurer que l'OSS ne contient pas de données ou de données de test sensibles, c'est-à-dire des données concernant par exemple des personnes ou cas réels.
- Mettre à jour la documentation du logiciel (voir annexe)
- Effacer les fichiers inutiles et les données et documents obsolètes.
- Vérifier la présence de secrets ou d'identifiants dans le code source[11].
- Réaliser des tests de sécurité ciblés et mettre en place un programme (public) de primes aux bogues[12].
- Vérifier les licences et les listes d'attribution de toutes les bibliothèques utilisées. Lorsqu'une bibliothèque est disponible sous plusieurs licences, il convient de le signaler dans les listes.
- Décrire le déploiement (pipline CI/CD) et éventuellement la mise en place d'une instance de démonstration.

Ces activités ne sont pas liées à une publication, mais elles sont indispensables pour garantir la qualité de l'OSS et assurer la conformité. Ces mesures simplifient la collaboration, augmentent la qualité des contributions externes et améliorent l'image du logiciel et de l'autorité.

[10] ISO/IEC 25010:2023 – Exigences de qualité et évaluation des systèmes et du logiciel (SQuaRE) https://www.iso.org/standard/78176.html

[11] https://www.ncsc.admin.ch/ncsc/fr/home/aktuell/im-fokus/2022/git.html

[12] https://www.ncsc.admin.ch/ncsc/fr/home/infos-fuer/infos-it-spezialisten/themen/bug-bounty-programme.html

## 4.2 Choix de la licence

**Le document *Em002-3 Guide Licences OSS* est une aide à utiliser.**

Lorsque cela est possible et judicieux, des textes de licence reconnus au niveau international seront utilisés. Toute prétention en responsabilité de la part des preneurs de licence sera exclue dans la mesure où cela est admis par le droit applicable. (art. 9, al. 4, LMETA)

**Activités**

- Vérifier les éventuelles dépendances des licences existantes.
- Vérifier la compatibilité des conditions d'utilisation avec la licence prévue en cas d'utilisation d'un code de tiers et remplacer le code si nécessaire[13].
- Choisir la licence appropriée.

La compatibilité des licences peut être vérifiée par exemple sur ORT Toolkit[14], Black Duck[15], FOSSA[16] ou FOSSology[17].
Le ToDO Group[18] propose un aperçu actuel des outils disponibles.

**Décision**

- Déterminer et documenter sous quelle licence le logiciel sera publié.

## 4.3 Documentation spécifique aux OSS

Un ensemble minimal de documents doit être fourni lors de la publication. Dans l'écosystème des OSS, certains documents se sont imposés de manière à fournir rapidement un aperçu aux visiteurs qui s'intéressent à un logiciel avec licence.

Les documents attendus sont les suivants :

| Document | Description / contenu |
|---|---|
| README(.md) | Document d'introduction, offre un aperçu du projet (but, volume, état, licence et groupes cibles) |
| LICENSE | Description de la licence choisie pour le projet |
| CONTRIBUTING(.md) | Description du processus prévu pour contribuer au projet |
| CODE_OF_CONDUCT(.md) | Définition des normes sociales inhérentes au projet |
| CHANGELOG(.md) | Liste des modifications apportées à la version du logiciel |
| THIRD-PARTY-LICENSES(.md) | Description des licences de tiers ou des composants utilisés |
| Getting started | Brèves instructions sur la manière d'installer et d'utiliser le logiciel. |

[13] Voir le document *Em002-3 Guide Licences OSS* en ce qui concerne le choix du code de tiers / des bibliothèques pour une stack technologique et la compatibilité de la licence.

[14] https://oss-review-toolkit.org/

[15] https://www.blackducksoftware.com

[16] https://www.fossa.com

[17] https://www.fossology.org

[18] https://landscape.todogroup.org/

| | |
|---|---|
| Gitignore | Description de documents qui ne sont pas publiés, comme des configurations, des données de test ou des contenus générés. Les informations sensibles et dignes de protection doivent être gérées en dehors du projet, dans des mémoires et des outils prévus à cet effet. |

Tableau 1 : documents spécifiques aux OSS

Des précisions techniques concernant ces documents sont données en annexe.

### Activités

- Établir la documentation standard.
- Ajouter à tous les fichiers les mentions de droit d'auteur et de licence ainsi que la clause de non-responsabilité.
- Ajouter d'autres descriptions en fonction de la plateforme de publication choisie.

### Décision

- Décider définitivement de publier le logiciel en tant qu'OSS.

# 5 Procédure Publication et diffusion

**Objectif**
L'application est publiée en tant qu'OSS et facile à trouver. Tous les éléments sont prêts à être publiés et la communauté est créée. À cet effet, on utilise le document *Em002-2.3 Liste de contrôle OSS Mise à disposition et publication*.
Il s'agit aussi en partie de récapituler les décisions prises.

## 5.1 Choix de la plateforme

Il existe différentes manières de publier le code source. Le tableau ci-dessous donne un aperçu des possibilités de gestion du code source (SCM) et des plateformes envisageables.

Avant la publication proprement dite, il faut choisir la plateforme appropriée. Certaines autorités fédérales sont déjà présentes sur GitHub[19]. Dans ce cas, il est judicieux d'utiliser la plateforme déjà établie.

**Recommandation**

Pour la publication de code source, nous recommandons actuellement de créer une organisation par autorité fédérale sur GitHub, d'y créer ensuite les projets (dépôts) et d'y gérer les utilisateurs et leurs autorisations. Il est prévu de créer une plateforme centralisée de la Confédération. Il est important d'utiliser des dépôts moins nombreux, mais mieux gérés afin de garantir une bonne gestion des utilisateurs et de préserver la réputation de la Confédération.

| Plateforme | Caractéristique |
|---|---|
| GitHub[20] | Plateforme en ligne de Microsoft avec une large distribution. GitHub héberge un grand nombre d'OSS et propose par ailleurs un SCM et de nombreux autres outils.<br>Disponible en version Free et Enterprise (abonnement). |
| Gitlab[21] | Plateforme en ligne de la société Gitlab, elle-même publiée sous licence *open source*. Propose un SCM et de nombreux autres outils. Peut aussi être utilisée localement, ce qui assure une certaine souveraineté.<br>Disponible en version Free et Enterprise (abonnement). |
| Bitbucket[22] | Plateforme en ligne d'Atlassian, qui est notamment intégrée dans son écosystème.<br>Disponible en version Free et Enterprise (abonnement). |
| Système SCM interne | S'il existe une plateforme SCM interne, certains dépôts et projets peuvent y être publiés et rendus accessibles. L'exploitation doit alors être assurée. |
| Plateforme fédérale | Une telle plateforme centralisée pour la publication de logiciel n'existe pas encore. Le document *Em002 Guide stratégique Administration fédérale et logiciels ouverts* propose d'en créer une. |

[19] Le site https://ossbenchmark.com donne un aperçu des organisations et des autorités présentes sur GitHub.
[20] https://github.com/about/
[21] https://about.gitlab.com/
[22] https://bitbucket.org/product/fr

| | |
|---|---|
| Site web, serveur FTP public, etc. | La LMETA ne précise pas sous quelle forme un logiciel doit être publié. La mise à disposition minimale[23] peut être effectuée sur un site web ou au moyen d'autres ressources accessibles au public. Toutefois, une telle démarche ne convient pas à une collaboration durable dans l'esprit de OSS. |

Tableau 2 plateformes de publication d'OSS

Si le logiciel est publié conjointement avec un partenaire ou une organisation externe, il faut veiller à ce que les autorisations et les accès au code source correspondants soient disponibles.

**Activités**

- Déterminer la plateforme appropriée, y compris l'organisation et le projet.
- Déterminer le libellé du projet ou des dépôts, si ce n'est déjà fait.
- Fixer et vérifier les autorisations sur le dépôt.
- Assurer une gouvernance appropriée et une éventuelle réplication du code source.

**Décision**

- La plateforme de publication est choisie.

## 5.2 Publication du logiciel

Lorsque toutes les conditions sont réunies, le logiciel peut être publié.

**Activités**

- Vérifier que les conditions sont réunies conformément au document *Em002-2.3 Liste de contrôle OSS Mise à disposition et publication*.
- Vérifier que l'accès à la plateforme est disponible et la gouvernance clarifiée.
- Le développeur publie le code source et la documentation.

## 5.3 Communication

Une fois le logiciel publié, d'autres activités de communication interne et externe peuvent être lancées au sein de l'autorité fédérale concernée, afin d'aider à distribuer le logiciel et d'apporter la plus-value souhaitée.

**Activités**

- Ajouter la description sur le site d'hébergement du dépôt, paramètres de sécurité.
- Communiquer à l'interne et, le cas échéant, à l'externe.
- Rédiger un communiqué pour la communauté.
- Faire connaître l'application au sein de l'organisation et des organes spécialisés, encourager l'utilisation et les contributions.
- Créer un fichier publiccode.yml[24] dépôt

[23] Une publication minimale comprend uniquement la publication du code source et d'autres atefacts minimaux. Ce type de publication est conforme à la LMETA, mais elle ne crée aucune valeur ajoutée.

[24] https://yml.publiccode.tools/ est un standard de métadonnées conçu pour décrire le logiciel. Il permet, grâce à un format défini, de trouver et de réutiliser plus facilement la publication. Pour plus d'informations, voir l'annexe F.

## 5.4 Mise en place et gestion de la communauté

Une communauté peut se développer à l'occasion de la publication du logiciel. Elle peut contribuer au logiciel en proposant des demandes de tirage (*pull request*), de la documentation… Pour qu'une communauté soit active, il faut fournir des efforts pour la mettre en place et la gérer sur la durée. Une communauté qui fonctionne réduit le nombre de forks non souhaités. L'objectif est d'activer la valeur ajoutée de la communauté par des activités ciblées. Le traitement des retours et des demandes concernant le logiciel doit être défini, même en cas de publication minimale.

**Activités**

- Définir le cercle des utilisateurs possibles et des personnes intéressées.
- Déterminer la forme appropriée dans le cas d'application spécifique au moyen du document *Em002-4 Guide Communauté OSS*.
- Mettre en place la communauté.
- Mener des activités de fédération de la communauté.

# Annexe

## A. Changements par rapport à la version précédente

- Chap. 5.3: ajout de la publication avec le publicode.yml
- Annexe F: ajout du publicode
- Petites modifications et améliorations rédactionnelles

## B. Références

Voir le document *Em002 Guide stratégique Administration fédérale et logiciels ouverts*

## C. Abréviations

Voir les documents *Em002 Guide stratégique Administration fédérale et logiciels ouverts* et *Em002-6 FAQ OSS*

# D. Documentation relative au document *Em002-2.2 Liste de contrôle OSS Analyse et préparation* – Documentation

## D. 1. Documentation accompagnant le code source

La documentation fait partie du code source du projet, afin que les modifications qui y sont apportées soient toujours prises en compte lors des révisions et que la documentation soit versionnée de manière analogue au projet. La documentation suit ainsi automatiquement l'évolution du projet, pour autant que son suivi soit assuré, et il est à tout moment possible de consulter la documentation des versions antérieures du projet.

Cela permet aussi à des tiers de communiquer simplement les modifications apportées au projet, sous forme de contributions.

Le format de texte Markdown est choisi comme langage de balisage, car il s'agit d'un langage facile à écrire, très répandu et lisible par une machine. De plus, toutes les plateformes courantes peuvent afficher les documents Markdown dans un format cible facile à lire, au moyen de générateurs de sites statiques, tels que Jekyll, MkDocs, Gatsby ou Sphinx, qui répondent par exemple à l'identité visuelle d'une administration.

## D. 2. Destinataires de la documentation et structure

La documentation est destinée aux spécialistes techniques, mais aussi aux utilisateurs finals de sorte qu'elle ne doit pas uniquement mettre l'accent sur la technique. Pour le choix de la langue, voir le chap. 3.6. Ce choix favorise la distribution du logiciel.

Le premier document à lire est le fichier `README.md`. Toutes les plateformes courantes reconnaissent ce fichier et le présentent comme point de départ pour la documentation[25].

Le fichier `README.md` vise à donner un aperçu du projet (but, volume, état, licence et groupes cibles). Il contient les éléments suivants :

- Nom du logiciel
- Brève description du but du logiciel Étendue des fonctions et délimitation
- Instructions d'installation
- Utilisation du logiciel
- Renvoi à une éventuelle instance de démonstration
- Support et coordonnées
- Manière de contribuer à l'OSS (CONTRIBUTING)
- Licence utilisée (LICENSE)
- Statut du projet / état du développement

[25] Plus d'informations sur la création d'un fichier readme, voir https://www.makeareadme.com/

## D. 3. Insister sur l'aspect ouvert du code source et l'indication de la licence

Directement après l'énoncé de la mission, on indique sans équivoque que le projet est un projet d'OSS. La licence utilisée est dotée d'un identificateur SPDX[26] et renvoie au texte de licence concret contenu dans le fichier LICENSE. Les modèles de licence comprennent en général une partie variable (p. ex. nom du projet, année du droit d'auteur, nom de l'auteur) qui doit être adaptée dans le fichier LICENSE.

La plupart des plateformes reconnaissent les licences dans le fichier `LICENSE` et proposent des informations claires sur lesdites licences, comme un résumé des conditions prévues.

Il est recommandé d'ajouter à chaque fichier source un entête qui comprend au moins des informations sur la licence et les droits d'auteur. Il ne faut toutefois pas modifier les entêtes qui existent déjà dans des fichiers. Il faut également respecter les standards pour les entêtes lors de la contribution à un projet qui prévoit des règles claires en la matière.

Pour plus de détails, voir : *Produire un Logiciel Libre, Préciser que le projet est libre*[27].

## D. 4. État actuel de développement

L'état actuel de développement du projet figure dans le fichier README.md afin d'être rapidement accessible. Le lecteur doit pouvoir savoir si le projet en est à ses débuts ou s'il est déjà bien avancé, s'il est géré activement et à quelle fréquence de nouvelles versions sont publiées.

Cette partie permet de décrire quel soutien de tiers est actuellement le plus important pour le projet. Il peut s'agir par exemple d'un développeur ayant des connaissances technologiques spécifiques ou d'une personne chargée de réviser la documentation du projet.

Pour plus de détails, voir *Produire un Logiciel Libre, Informer sur le statut de développement*.[28]

## D. 5. Instance de démonstration

Il est toujours recommandé de mettre à disposition une instance de démonstration des applications afin que les lecteurs puissent les tester sans gros efforts d'installation. Selon le type d'application, l'instance de démonstration peut être l'installation productive ou une étape de test. Il est important que les lecteurs puissent accéder directement à cette instance, sans intermédiaire.

Dans de nombreux projets d'OSS, il est aussi possible de mettre à disposition des outils pour aider à lancer simplement l'application. Il peut s'agir d'un conteneur, d'une application portable ou d'un script d'installation.

## D. 6. Interlocuteur et propriétaire du projet

Le fichier README.md contient des indications sur le spécialiste auquel les administrations et les organisations peuvent s'adresser directement si le projet les intéresse.

L'indication d'une adresse e-mail personnelle est à proscrire.

Le fichier `README.md` indique aussi brièvement l'organisation propriétaire du projet, en particulier lorsqu'une association a été créée pour ledit projet.

---

[26] Identificateur de licence SPDX : https://spdx.org/about

[27] https://producingoss.com/fr/getting-started.html#state-freedom

[28] https://producingoss.com/fr/getting-started.html#development-status

# D. 7. Documentation d'installation

Le fichier README.md renvoie à une documentation d'installation, qui décrit les exigences matérielles et logicielles, les composants infrastructurels (p. ex., banques de données) à utiliser et la manière d'installer et d'utiliser l'application.

# D. 8. Guide du développeur

L'accès au guide du développeur est intégré dans le fichier CONTRIBUTING.md, auquel renvoie le fichier README.md. Le guide du développeur fait partie de la documentation complémentaire destinée aux développeurs qui souhaitent contribuer au projet. Il se concentre principalement sur la coopération et la communication au sein du projet et non sur la technique.

Il contient les éléments suivants :

- Un lien vers le fichier CODE_OF_CONDUCT du projet qui définit les normes sociales inhérentes au projet.
- Il est recommandé de choisir ou de se fonder sur le code de conduite *Contributor Covenant* Conduct sous la licence CC-BY-4.0[29][30].
- Une mention indiquant que les revues du code de projet sont effectuées sous forme de demandes de tirage GitHub (*pull requests*) avec un renvoi vers la documentation correspondante.

Pour finir, on renvoie à la documentation développeurs.

# D. 9. Documentation développeurs

Tandis que le guide du développeur décrit la coopération et les normes sociales du projet, la documentation développeur se concentre sur les aspects techniques qui permettent de participer au développement du projet. Elle porte au minimum sur les points suivants :

- Les dépendances techniques du projet et les outils requis pour son développement
- Les règles formelles applicables au code source, notamment en ce qui concerne le formatage du code source
- L'organisation du code source et la manière d'effectuer le test technique des fonctions
- Une ébauche d'architecture décrivant les grandes lignes, avec si possible une délimitation du contexte et un aperçu des modules[31]
- Pour les applications : la manière de démarrer une application à des fins de développement ou de débogage et les composants d'infrastructure nécessaires dans ce contexte

[29] https://creativecommons.org/licenses/by/4.0/deed.fr

[30] https://www.contributor-covenant.org/fr/version/2/1/code_of_conduct/

[31] Architekturframework MMB (méthode de modélisation de la Confédération) ou arc42. http://arc42.org/

# E. Documentation relative au document *Em002-2.2 Liste de contrôle OSS Analyse et préparation* – Code source

## E. 1. Historique du code source

Le code source d'un projet est en général stocké dans un SCM tel que GitHub. Le SCM enregistre l'état actuel du code source, mais aussi toutes les modifications qui y ont été apportées au fil du temps (notamment les éléments effacés du code et les fichiers effacés).

Il contient de plus des méta-informations sur les modifications du code source, comme le nom et l'adresse e-mail de l'auteur ou les signatures PGP des commits et des tags.

## E. 2. Données sensibles ou confidentielles

Le code source du projet et son historique peuvent contenir des données publiques, comme un répertoire de codes postaux, des données générées aléatoirement et des données synthétiques.

Ils ne doivent contenir aucunes données ni informations sensibles ou confidentielles conformément à la loi sur la protection des données et à la loi sur la sécurité de l'information. Ces données sont en particulier :

- les noms d'utilisateur et mots de passe ou d'autres secrets comme les clés privées, les jetons d'accès et les certificats
- les noms de DNS internes, les URL, les noms d'hôte, les adresses IP, les structures de réseau et les dénominations de lecteurs
- les noms, adresses, images ou autres données concernant une personne sans son accord
- des données anonymisées ou pseudonymisées (car il existe dans ce cas un risque de pouvoir inverser l'opération)

Même si l'état actuel du code source ne contient pas de données ni d'informations sensibles ou confidentielles, il est possible de les retrouver en consultant l'historique du SCM dès lors qu'elles ont fait partie du code source ou des méta-informations SCM par le passé. Un nettoyage du code source actuel ne suffit pas à les supprimer entièrement[32].

En cas de doute après le nettoyage, il est donc recommandé de supprimer complètement l'historique avant de publier le code source.

Les noms et adresses e-mail des auteurs du code source ne sont en général pas indiqués directement dans le code source ou les méta-informations SCM (p. ex. commits GitHub ou tags GitHub annotés). Cela ne pose pas de problème sur le plan juridique étant donné que la responsabilité incombe à l'entreprise développeuses de régler la publication de ces informations. Il est toutefois recommandé de rendre l'entreprise développeuse ou les collaborateurs de la Confédération attentifs au fait que le nom et l'adresse e-mail des auteurs du code source sont susceptibles d'être publiés en même temps que le logiciel.

[32] L'outil BFG Repo-Cleaner (https://rtyley.github.io/bfg-repo-cleaner/) permet par exemple de nettoyer un dépôt.

# E. 3. Suppression des informations sur l'auteur

Il est possible de modifier l'historique du logiciel de façon à anonymiser les informations sur les auteurs. Des outils d'aide[33] permettent d'automatiser cette étape.

La suppression de ces informations entraîne la perte de l'attribution, de la traçabilité et des informations de contact. Selon la taille du code source, l'effort à fournir ne doit pas être sous-estimé. Cette manipulation n'est donc pas recommandée.
La publication des noms est autorisée uniquement si a) les personnes concernées ont donné leur accord et b) aucune raison de sécurité ne s'y oppose.

# E. 4. Prise en compte des licences des librairies

Afin de pouvoir accéder à des fonctions utilisables par tous sans avoir à les programmer, pratiquement tous les projets recourent à des bibliothèques de tiers (bibliothèques logicielles, dépendances). La plupart du temps, ces bibliothèques dépendent à leur tour d'autres bibliothèques, ce qui entraîne des dépendances à plusieurs niveaux (dépendances transitives). Selon la langue de programmation, même les plus petits projets comptent souvent des centaines de dépendances.

Le résultat de la clarification en la matière doit être pris en compte lors de choix de la licence conformément aux documents *Em00 2Guide Communauté OSS* et *Em002-2.3 Liste de contrôle OSS Mise à disposition et publication*.

Il faut s'assurer que la licence de chaque dépendance est compatible avec la licence du projet. Les dépendances particulièrement problématiques sont celles qui ne sont pas soumises à une licence *open source*, celles dont la licence n'est pas déclarée, ou celles qui utilisent une licence dite virale qui ne correspond pas à celle du projet. Certaines bibliothèques sont soumises à plus d'une licence et permettent à leur utilisateur de choisir celle dont il a besoin.

- Dans tous les langages de programmation, les dépendances du projet (y c. les dépendances transitives) et leur licence peuvent être listées automatiquement. En font partie :
- Le *project info reports plugin* du gestionnaire de paquets Apache Maven pour les projets Java, qui génère un rapport au format HTML des dépendances de toutes les bibliothèques utilisées, y compris des licences déclarées[34].
- Le *NPM license checker* pour le gestionnaire de paquets NPM des projets JavaScript, qui peut générer la liste de toutes les bibliothèques utilisées, y compris des licences déclarées, dans différentes formations (p. ex. CSV)[35].
- Le *license finder* de Pivotal, qui prend en charge différents gestionnaires de paquets et différents langages de programmation (p. ex. Ruby Gems, Python Eggs, Godeps).[36]

**Utiliser des bibliothèques avec des licences propriétaires**

Les bibliothèques qui utilisent des licences propriétaires sont problématiques pour tout projet en code source ouvert et ne peuvent en général pas être utilisées, notamment quand le code source a été publié sous une licence virale comme AGPL.

Il faut passer par le service juridique de l'office concerné pour prendre contact avec le fournisseur afin de savoir dans quelles conditions il serait possible d'utiliser la bibliothèque propriétaire dans un projet d'OSS.

---

[33] https://www.adamdehaven.com/blog/update-commit-history-author-information-for-git-repository/

[34] Plugin Info https://maven.apache.org/plugins/maven-project-info-reports-plugin/plugin-info.html

[35] NPM license checker https://github.com/davglass/license-checker

[36] License Finder de Pivotal https://github.com/pivotal/LicenseFinder

# E. 5. Utiliser les gestionnaires de paquets

Pratiquement tous les langages de programmation courants proposent des gestionnaires de paquets pour gérer leurs dépendances, par exemple Apache Maven pour les projets Java, NPM pour les projets JavaScript ou NuGet pour les projets .NET.

L'utilisation d'un gestionnaire de paquets permet de faire en sorte que les dépendances utilisées dans un projet ne fassent plus directement partie du code source, par exemple en copiant le code source de la dépendance dans le projet ou en copiant la dépendance dans le projet sous forme binaire. Au lieu de cela, le projet obtient toutes les dépendances en allant les chercher dans les déclarations correspondantes du gestionnaire de paquets.

Il est ainsi possible de lister toutes les dépendances et d'indiquer correctement leurs références. Cette méthode empêche en outre toute modification involontaire du code des dépendances. Si des modifications sont nécessaires, elles doivent être apportées directement à la source de la dépendance.

# E. 6. Déclaration de la licence de projet dans le descripteur du gestionnaire de paquets

Dans le projet, il faut prévoir de déclarer les licences dans le descripteur du gestionnaire de paquets doté d'un identifiant SPDX[37], afin que la licence du projet puisse être émise en passant par le gestionnaire de paquets. Cet élément est particulièrement important pour les bibliothèques, afin que leurs utilisateurs puissent interroger automatiquement la licence.

# E. 7. Mention des droits d'attribution et des droits d'auteur

De nombreuses bibliothèques sont soumises à une licence qui exige que les utilisateurs mentionnent les droits d'auteur (note de droits d'auteur originale) ou les droits d'attribution pour l'utilisation.

Il s'agit notamment des licences suivantes (extrait) :

Apache 2.0, ASL 1.1 (Apache 1.1), BSD, BSD 3– Clause, Creative Commons « Attribution » (CC BY), MIT, ISC

Pour des licences supplémentaires, voir la liste des licences accréditées OSI[38].

Ainsi, dans la pratique, presque toutes les bibliothèques utilisées sont concernées. Étant donné que le nombre de bibliothèques utilisées est très élevé même dans les petits projets, une mention juridiquement irréprochable du droit d'auteur ou des droits d'attribution ne sont possibles qu'au prix d'un effort très important.

Il est donc recommandé de procéder comme suit :

- Dresser une liste des bibliothèques utilisées au moyen des mécanismes appropriés du gestionnaire de paquets ou des outils tels que *NPM license checker* ou *license finder* de Pivotal.
- Préparer manuellement la liste pour qu'elle contienne les informations suivantes : nom officiel du projet de la bibliothèque, page d'accueil du projet de la bibliothèque, licence utilisée par la bibliothèque comme identifiant SPDX avec un lien vers le texte de la licence.
- Placer la liste dans le code source du projet, par exemple dans un fichier `THIRD-`

[37] Identifiant SPDX https://spdx.org/

[38] Les licences open source sont des licences qui sont conformes à la définition de l'open source – en bref, elles permettent d'utiliser, de modifier et de partager le logiciel librement.. https://opensource.org/licenses

`PARTY-LICENSES.md`.

- Pour les applications : insérer un lien dans un « À propos » ou dans la boîte de dialogue « À propos de cette application » qui renvoie au fichier `THIRD-PARTY-LICENSES.md`.
- L'utilisation de la boîte de dialogue « À propos » à cette fin est très répandue, par exemple dans Google Chrome (voir boîte de dialogue « À propos de Chrome »).
- Pour les bibliothèques : faire référence au `THIRD-PARTY-LICENSES.md` dans le fichier `README.md`.
- Dans le cadre du processus de révision du code, insister pour que les modifications apportées aux bibliothèques soient mises à jour dans `THIRD-PARTY-LICENSES.md`.

**Exemple de bloc de code THIRD-PARTY-LICENSES.md**

```
Ce projet utilise des OSS :

* Spring Boot, http://projects.spring.io/spring-boot/, avec licence [Apache-
2.0](http://www.apache.org/licenses/LICENSE-2.0)
* caniuse-db, https://github.com/Fyrd/caniuse, avec licence [CC-BY-
4.0](https://creativecommons.org/licenses/by/4.0/)
```

# F. Documentation relative au document Em002-2.3 *Liste de contrôle OSS Mise à disposition et publication*

## Publication avec publiccode.yml

Publiccode.yml est un standard de métadonnées destiné aux dépôts qui contiennent des logiciels achetés ou développés par des administrations publiques. Son but est de faciliter la recherche et la réutilisation de ces logiciels. Il s'agit d'un standard international reconnu et décrit publiquement sur le site https://yml.publiccode.tools/.
Il fournit des informations structurées qui servent de base pour des répertoires et des catalogues OSS. Un fichier publiccode.yml contient en général des informations telles que le titre et la description d'un logiciel (dans plusieurs langues le cas échéant), le niveau de développement, des données de contact, des indications sur la licence, etc. Des extensions spécifiques au pays sont prévues pour la Suisse.

Un fichier yml par logiciel doit être créé dans le répertoire principal du dépôt. Si la solution est composée de plusieurs dépôts (auxiliaires), un seul fichier est suffisant, étant donné qu'il décrit le cas d'utilisation de ladite solution.

# G. Autres sources et licences

La liste de contrôle, la documentation qui l'accompagne et les modèles correspondants sont basés en partie sur les sources suivantes :

- Google Open Source Docs de Google LLC, avec licence CC-BY-4.0[39]
- Producing Open Source Software, How to run a Successful Free Software Project von Karl Fogel, sous licence CC-BY-SA-4.0[40]
- GitHub Healthy Contributions[41]
- Standard for Pubilc Code[42]

[39] Google Open Source Docs – https://opensource.google.com/docs/

[40] Producing Open Source Software, How to run a Successful Free Software Project - https://producingoss.com/

[41] GitHub Healthy Contributions

[42] Standard for Public Code – https://standard.publiccode.net/